HiDiffTIR: Hierarchical Difficulty-Aware Policy Optimization for Multi-Turn Tool-Integrated Reasoning

Yucan Guo[1,2,3]*, Xiaohan Wang[4]*, Miao Su[1,2,3], Saiping Guan[1,2,3]†, Zhongni Hou[4], Jiajun Chai[4], Wei Lin[4], Guojun Yin[4], Xiaolong Jin[1,2,3]†, Jiafeng Guo[1,2,3], Xueqi Cheng[1,2,3],
[1]State Key Laboratory of AI Safety
[2]Institute of Computing Technology, Chinese Academy of Sciences
[3]School of Computer Science and Technology, University of Chinese Academy of Sciences
[4]Meituan
{guoyucan23z, guansaiping, jinxiaolong}@ict.ac.cn

Abstract

Tool-Integrated Reasoning (TIR) is a fundamental capability for LLM agents to solve complex tasks by interacting with external tools iteratively. Reinforcement Learning (RL) has become the dominant paradigm for enabling this capability. However, existing approaches typically assign uniform trajectory-level advantages and treat all correct tool calls equally, ignoring the varying difficulty and learning value across trajectories and reasoning steps. This can lead to imprecise learning signals that do not adequately distinguish between trivial and challenging tool-use patterns. To address this limitation, we propose HiDiffTIR, a **Hi**erarchical **Diff**iculty-aware policy optimization framework for multi-turn **TIR**. HiDiffTIR performs difficulty-aware credit assignment at both trajectory and turn levels, enabling the policy to focus on more informative trajectories and harder reasoning steps. Notably, this fine-grained optimization is achieved without additional supervision, relying solely on group-level statistics derived from standard RL rollouts. Extensive experiments on three tool-using benchmarks demonstrate that HiDiffTIR consistently improves multi-turn TIR performance and tool invocation accuracy over strong RL baselines, highlighting the necessity of difficulty-aware credit assignment for effective policy optimization in tool-integrated LLM agents. [1]

1 Introduction

Large Language Models (LLMs) have demonstrated strong reasoning capabilities across a wide range of tasks (OpenAI et al., 2023; DeepSeek-AI et al., 2025; Yang et al., 2025), leading to the emergence of LLM agents that can plan, act, and interact with external environments (Huang et al., 2024; Wang et al., 2024; Luo et al., 2025). A central capability of such agents is effective tool use, where they invoke external tools to solve diverse tasks (Masterman et al., 2024; Xu et al., 2025). Tool-Integrated Reasoning (TIR) formalizes this process by enabling LLM agents to iteratively interact with external tools (Gou et al., 2024; Qian et al., 2025), such as APIs (Patil et al., 2024; Qin et al., 2024), search engines (Li et al., 2025a; Jin et al., 2025), and code interpreters (Gou et al., 2024; Li et al., 2025b). This paradigm extends LLM agents beyond their static parametric knowledge and their known limitations in domains such as precise numerical computation (Frieder et al., 2023), using specific tools to solve complex multi-turn tasks that are otherwise challenging for purely inner knowledge-based reasoning (Lin and Xu, 2025; Qu et al., 2025).

Reinforcement learning (RL) has emerged as the prevailing paradigm for training LLMs to perform TIR (Qian et al., 2025; Xue et al., 2025). By employing RL algorithms like Group Relative Policy Optimization (GRPO) (Shao et al., 2024), models learn to interleave reasoning, tool invocation, and answer generation in an end-to-end manner. Most existing TIR methods adopt trajectory-level optimization, which has evolved through two primary stages. Early methods rely on outcome-based rewards (Li et al., 2025b; Dong et al., 2025), assessing only the final correctness of a reasoning chain, and thus suffer from reward sparsity. To provide denser feedback, recent works have shifted toward turn-level verification (Qian et al., 2025; Ye et al., 2025b), where the correctness of each individual tool-call is evaluated and subsequently aggregated to form the trajectory-level reward. Despite this progress, these methods typically assign uniform advantages across all steps within a trajectory, implicitly treating each tool call as equally informative for learning. This coarse-grained credit assignment can obscure the contribution of individ-

*Co-first authors.
†Corresponding authors.
[1]The code is publicly available at https://github.com/YucanGuo/HiDiffTIR.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 6325–6342
October 24-29, 2026 ©2026 Association for Computational Linguistics

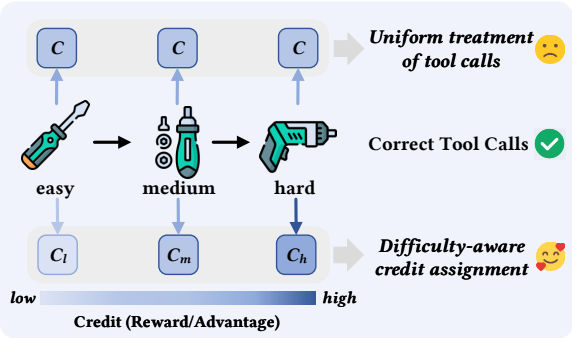


Figure 1: Comparison of credit assignment manners in TIR. Existing methods treat all correct tool calls uniformly, while HIDIFFTIR performs difficulty-aware credit assignment.

ual reasoning steps, especially in multi-turn settings where errors accumulate and only a subset of decisions are critical to success.

To address this limitation, more recent works have explored finer-grained credit assignment by incorporating turn-level signals and combining them with trajectory-level objectives (Li et al., 2026; Qu et al., 2026). These methods typically perform trajectory-turn fusion, where trajectory-level rewards are used in their original form, and turn-level signals are either directly applied or heuristically adjusted and propagated across steps. However, they treat all correct tool calls equivalently, without explicitly modeling the varying difficulty of different reasoning steps. As a result, the learning signal remains insensitive to the relative importance and challenge of different tool-use patterns.

In this paper, we revisit credit assignment in TIR from a difficulty-aware perspective. As shown in Figure 1, we argue that not all trajectories or tool-calling steps contribute equally to policy improvement. Some trajectories are more informative due to their relative difficulty, and some reasoning steps are inherently harder and thus more valuable for learning. Motivated by this, we propose HIDIFFTIR, a **Hi**erarchical **Diff**iculty-aware policy optimization framework for **TIR** that performs fine-grained credit assignment without requiring additional supervision. HIDIFFTIR operates at two levels. At the trajectory level, it differentiates rollouts based on their relative difficulty, enabling the policy to prioritize more informative training signals. At the turn level, it further refines credit assignment by redistributing advantages across reasoning steps according to their difficulty, allowing the model to focus on harder tool-calling decisions. Importantly, the judgment of tool-using difficulty relies solely on group-level statistics derived from standard RL rollouts and does not require external supervision such as LLM-based judges or rubrics. Our contributions are summarized as follows:

- We propose HIDIFFTIR, a hierarchical policy optimization framework that performs difficulty-aware credit assignment at both the trajectory and turn levels.
- We introduce a supervision-free method for fine-grained policy optimization that relies solely on statistics from standard RL rollouts.
- Extensive experiments demonstrate that HIDIFFTIR improves tool-use accuracy and multi-turn reasoning performance across multiple benchmarks.

2 Related Work

2.1 Tool-Integrated Reasoning

TIR has become a central paradigm for equipping LLM agents with external tools to solve complex tasks. Early work primarily relies on Supervised Fine-Tuning (SFT), where models are trained on solution annotations from strong LLMs that interleave reasoning steps and tool invocations (Schick et al., 2023; Tang et al., 2023; Patil et al., 2024; Qin et al., 2024). While effective in establishing basic tool-use capabilities, recent research has increasingly shifted toward RL to further improve policy learning through interaction (Xue et al., 2025; Li et al., 2025b; Singh et al., 2025). Within RL-based TIR, methods have evolved from trajectory-level optimization with outcome-based rewards (Li et al., 2025b; Dong et al., 2025) to incorporating process-based verification signals, such as tool name, parameter name, and argument values, to provide denser supervision (Qian et al., 2025; Ye et al., 2025b). More recent approaches further integrate trajectory-level and turn-level signals to provide individual advantage for each turn. For instance, DeepAgent (Li et al., 2026) combines outcome rewards with turn-level tool call rewards, while MatchTIR (Qu et al., 2026) further propagates turn-level rewards across steps using discounted accumulation. Despite these advances, existing methods generally treat all correct tool calls as equally informative, without explicitly modeling the varying difficulty of different trajectories or reasoning steps.

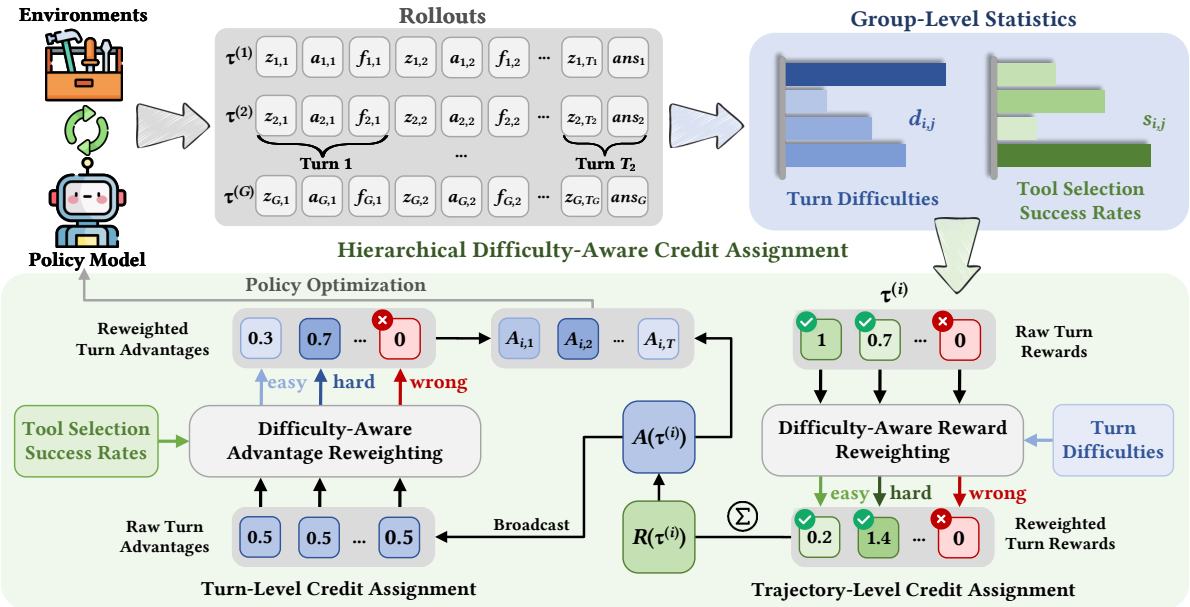


Figure 2: Illustration of the HiDiffTIR framework. HiDiffTIR implements a hierarchical difficulty-aware credit assignment mechanism: (a) trajectory-level, where raw turn rewards are reweighted based on turn difficulties to compute a difficulty-aware trajectory advantage, and (b) turn-level, where the trajectory advantage is redistributed to yield fine-grained reweighted turn advantages for policy optimization.

2.2 Reinforcement Learning for LLM Reasoning

RL has been widely adopted to improve the reasoning capabilities of LLMs beyond SFT. A prominent line of work is Reinforcement Learning from Human Feedback (RLHF), where models are optimized using preference signals collected from human annotators (Christiano et al., 2017; Ouyang et al., 2022). Recent work has demonstrated the effectiveness of Reinforcement Learning from Verifiable Rewards (RLVR), where models are optimized using automatically computed correctness signals (Shao et al., 2024; DeepSeek-AI et al., 2025; Yang et al., 2025). To improve reward quality, a line of studies also concentrates on enhancing credit assignment in LLM reasoning through process-oriented supervision (Uesato et al., 2022; Lightman et al., 2023). Such approaches typically rely on additional supervision signals, including process reward models (Zhang et al., 2025; Wang et al., 2025), LLM-based judges guided by evaluation rubrics (Gunjal et al., 2025; Huang et al., 2025), to assess intermediate reasoning steps and provide more fine-grained feedback. In parallel, there is growing interest in difficulty-aware RL (Zhang and Zuo, 2025; Gao et al., 2026; Heckel et al., 2026), where training emphasizes more challenging samples to improve model performance, with sample difficulty estimated based on outcome-level signals. However, these methods are largely designed for general reasoning tasks, making them insufficient for jointly modeling process-level feedback and difficulty in TIR.

3 HiDiffTIR

3.1 Problem Formulation

Given a user query q and a set of available tools $\mathcal{T} = \{t_1, \ldots, t_n\}$, the goal of a TIR agent π_θ is to solve the query through a multi-turn reasoning and tool use interaction process $\tau = \{(z_1, a_1, f_1), \ldots, (z_{T-1}, a_{T-1}, f_{T-1}), (z_T, ans)\}$, where T denotes the total number of turns. At each intermediate turn $i < T$, the agent generates a reasoning step z_i and a set of tool actions a_i, where each action specifies a selected tool from $\mathcal{T}$ along with its arguments. The environment executes these tool calls and returns feedback f_i. At the final turn, the agent produces a reasoning step z_T followed by the final answer ans without invoking any tool. The objective of a TIR agent is to learn a policy that generates trajectories leading to correct final answers by effectively interleaving reasoning and tool use.

3.2 Overall Framework

HiDiffTIR is a hierarchical policy optimization framework for TIR that introduces difficulty-aware credit assignment at two levels. The core idea is

to distinguish the relative learning importance of different trajectories and turns, rather than treating them uniformly. As shown in Figure 2, at the trajectory level, HiDiffTIR distinguishes trajectories according to the relative difficulty of all constituent tool calls (§3.3). At the turn level, it further refines credit assignment across reasoning steps, considering both tool-selection difficulty and tool-using difficulty (§3.4). These two components together enable fine-grained optimization of the policy model (§3.5).

3.3 Trajectory-Level Difficulty-Aware Credit Assignment

We now introduce trajectory-level difficulty-aware credit assignment, which extends turn-level rewards based on tool-use correctness. Specifically, we first compute the similarity between predicted and ground-truth tool calls to derive turn-level raw rewards. Next, we estimate the relative difficulty of each turn and reweight the raw rewards accordingly. Finally, we normalize these rewards to construct a difficulty-aware trajectory-level advantage.

Raw Reward Computation. Following previous works (Qian et al., 2025; Qu et al., 2026), we compute the raw reward for each turn in a trajectory using tool-level correctness. Given a trajectory $\tau^{(i)}$ with T turns, let $C_i = \{c_{i1}, \ldots, c_{im}\}$ denote the predicted tool calls and $C_i^* = \{c_{i1}^*, \ldots, c_{in}^*\}$ the ground-truth calls. A similarity matrix $M_i \in \mathbb{R}^{m \times n}$ is defined as

$$M_i[u, v] = \text{sim}(c_{iu}, c_{iv}^*), \tag{1}$$

where $u \in \{1, \cdots, m\}$, $v \in \{1, \cdots, n\}$, and the similarity function $\text{sim}(\cdot, \cdot) \in [0, 1]$ measures agreement on tool name, parameter names, and parameter values. Detailed computation of the similarity function is provided in Appendix A.3.

A maximum-weight bipartite matching π_i^* is then obtained via the Hungarian algorithm (Kuhn, 1955, 1956):

$$\pi_i^* = \arg\max_{\pi \in \Pi} \sum_{(u,v) \in \pi} M_i[u, v], \tag{2}$$

where Π denotes all one-to-one matchings between C_i and C_i^*.

Let $C_{i,j} \subseteq C_i$ denote the subset of tool calls issued at turn j. The raw reward for turn j is defined as

$$r_{i,j} = \frac{1}{|C_{i,j}|} \sum_{c_{iu} \in C_{i,j}} \mathbb{I}[(u, v) \in \pi_i^*] \cdot M_i[u, v], \tag{3}$$

where unmatched tool calls contribute zero. These turn-level rewards serve as the initial supervision signal for subsequent trajectory-level optimization.

Difficulty-Aware Reward Reweighting. To model the relative difficulty of different tool-use behaviors, group-level statistics are estimated over trajectories sampled for the same query q. Let $\mathcal{G}(q) = \{\tau^{(1)}, \cdots, \tau^{(G)}\}$ denote the group of trajectories generated for q. For each ground-truth tool call c^*, we compute its average reward across the group:

$$\bar{r}(c^*) = \frac{1}{|\mathcal{G}(q)|} \sum_{\tau^{(k)} \in \mathcal{G}(q)} r^{(k)}(c^*), \tag{4}$$

where $r^{(k)}(c^*)$ denotes the reward assigned to the tool call in trajectory $\tau^{(k)}$ that is matched to c^* via bipartite matching.

For each turn j in trajectory $\tau^{(i)}$, let $C_{i,j}^*$ denote the set of ground-truth tool calls associated with that turn. We define the turn-level average reward as

$$\bar{r}_{i,j} = \frac{1}{|C_{i,j}^*|} \sum_{c^* \in C_{i,j}^*} \bar{r}(c^*). \tag{5}$$

The difficulty of the turn is then defined as

$$d_{i,j} = 1 - \bar{r}_{i,j}, \tag{6}$$

and used to reweight the raw reward:

$$\tilde{r}_{i,j} = d_{i,j} \cdot r_{i,j}. \tag{7}$$

This weighting emphasizes turns associated with harder tool-use decisions while down-weighting consistently easy ones.

Trajectory-Level Score Construction. To ensure stable optimization, the reweighted rewards are normalized within each group. A min-max normalization is first applied:

$$\hat{r}_{i,j} = \frac{\tilde{r}_{i,j} - \min_{\tau \in \mathcal{G}(q)} \tilde{r}_{\tau,j}}{\max_{\tau \in \mathcal{G}(q)} \tilde{r}_{\tau,j} - \min_{\tau \in \mathcal{G}(q)} \tilde{r}_{\tau,j}}, \tag{8}$$

followed by rescaling to preserve the total reward mass:

$$\tilde{r}_{i,j}^{\text{norm}} = \hat{r}_{i,j} \cdot \frac{\sum_{\tau \in \mathcal{G}(q)} \tilde{r}_{\tau,j}}{\sum_{\tau \in \mathcal{G}(q)} \hat{r}_{\tau,j}}. \tag{9}$$

The trajectory-level score is then obtained by aggregating normalized turn rewards:

$$R(\tau^{(i)}) = \sum_{j=1}^{T_i} \tilde{r}_{i,j}^{\text{norm}}, \tag{10}$$

and converted into a group-relative advantage:

$$A(\tau^{(i)}) = \frac{R(\tau^{(i)}) - \mu_q}{\sigma_q}, \tag{11}$$

where μ_q and σ_q denote the mean and standard deviation of trajectory scores within $\mathcal{G}(q)$. This formulation yields a difficulty-aware trajectory-level objective that emphasizes relatively harder yet successful trajectories while maintaining stable comparisons within each group.

3.4 Turn-Level Difficulty-Aware Credit Assignment

While trajectory-level optimization captures the overall difficulty of a rollout, it does not distinguish which reasoning steps are more challenging within the trajectory. To further refine credit assignment, we first estimate the relative difficulty of each turn based on group-level statistics, and then redistribute the trajectory-level advantage across individual turns based on these turn-level difficulties.

Group-Level Tool Selection Success Rate. For each ground-truth tool call c^*, we compute its tool selection success rate over the group $\mathcal{G}(q)$:

$$s(c^*) = \frac{1}{|\mathcal{G}(q)| + \lambda} \sum_{\tau^{(k)} \in \mathcal{G}(q)} \mathbb{I}\big(r^{(k)}(c^*) > 0\big) + \lambda, \tag{12}$$

where $\mathbb{I}(\cdot)$ is an indicator function, and λ is a smoothing constant. Since a positive reward indicates that the correct tool is selected, $s(c^*)$ effectively measures how frequently each tool call is correctly invoked across trajectories.

After that, we aggregate the tool selection success rates into a turn-level statistic for each turn j in trajectory $\tau^{(i)}$:

$$s_{i,j} = \min_{c^* \in C^*_{i,j}} s(c^*), \tag{13}$$

which emphasizes the most challenging tool call within the turn.

Difficulty-Aware Advantage Reweighting. Based on the estimated tool selection success rates, we construct turn-level weights that emphasize more difficult decisions. For trajectories with positive advantage, we assign higher weights to turns with lower success rates while incorporating the overall tool-using quality:

$$\tilde{w}_{i,j} = \frac{r_{i,j}}{\sqrt{s_{i,j}}}. \tag{14}$$

This formulation suppresses incorrect turns while prioritizing harder correct decisions.

For trajectories with negative advantage, we do not apply difficulty-based reweighting and instead use uniform weights, ensuring stable penalization of unsuccessful rollouts. The weights are then normalized within each trajectory:

$$\hat{w}_{i,j} = \frac{\tilde{w}_{i,j}}{\frac{1}{T_i}\sum_{k=1}^{T_i} \tilde{w}_{i,k}}. \tag{15}$$

Finally, the trajectory-level advantage is redistributed across turns as

$$A_{i,j} = A(\tau^{(i)}) \cdot (\alpha + (1-\alpha)\hat{w}_{i,j}), \tag{16}$$

where $\alpha \in [0,1]$ controls the balance between trajectory- and turn-level difficulty-aware credit assignment, preserving the global optimization signal while enabling fine-grained emphasis on harder reasoning turns.

3.5 Policy Optimization

The final advantage assigned to each token is constructed by combining trajectory-level and turn-level signals in a hierarchical manner. The turn-level advantages $A_{i,j}$ are assigned to all tokens within the corresponding turn, resulting in token-level advantages $\tilde{A}_{i,t}$. Using the integrated advantage $\tilde{A}_{i,t}$, we optimize the policy under the GRPO algorithm. Given a batch of queries $q \sim \mathcal{D}$ and sampled trajectories $\{\tau_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$, the objective is defined as

$$\mathcal{J}(\theta) = \mathbb{E}_{q,\{\tau_i\}} \frac{1}{G} \sum_{i=1}^{G} \frac{1}{|\tau_i|} \sum_{t=1}^{|\tau_i|} \Big[\min \Big(w_{i,t}\tilde{A}_{i,t}, \text{clip}(w_{i,t}, 1-\epsilon, 1+\epsilon)\tilde{A}_{i,t} \Big) - \beta \mathbb{D}_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}) \Big], \tag{17}$$

where $w_{i,t} = \frac{\pi_\theta(y_{i,t}|y_{i,<t},q)}{\pi_{\theta_{\text{old}}}(y_{i,t}|y_{i,<t},q)}$ denotes the token-level importance sampling ratio, and β controls the strength of KL regularization. This objective maintains the standard GRPO optimization structure while incorporating difficulty-aware hierarchical credit assignment, enabling more effective learning for multi-turn TIR.

4 Experiments

In this section, we evaluate HIDIFFTIR through comparative experiments (§4.2), ablation study (§4.3), and tool-use accuracy analysis (§4.4), with additional results detailed in Appendix B.

Table 1: Performance comparison between HiDiffTIR and the baselines on in-domain and out-of-domain benchmarks across Qwen3-4B and Qwen3-8B backbones. The best results within each backbone group are indicated in bold, while the underlined values represent the second-best results.

Methods	In-Domain (ID)				Out-of-Domain (OOD)					
	FTRL				BFCL				ToolHop	Avg.
	Solve-P	Solve-R	Solve-F1	**Avg.**	Base	MF	MP	LC	AC	
Qwen3-4B										
Vanilla	30.78	29.65	25.85	28.76	41.50	31.00	26.50	27.50	31.63	31.63
GRPO	31.13	32.83	30.67	31.54	45.00	37.50	26.50	29.50	37.25	35.15
ToolRL	28.26	28.32	23.78	26.79	32.50	31.00	22.50	20.00	30.28	27.26
FTRL	34.10	34.07	31.54	33.24	43.00	35.50	31.00	28.00	38.63	35.23
MatchTIR (OT)	31.79	37.52	32.60	33.97	50.00	40.50	26.50	35.00	41.95	38.79
MatchTIR (KM)	32.39	39.70	34.21	35.43	50.50	**47.00**	28.50	36.50	42.55	41.01
HiDiffTIR	**35.41**	**44.74**	**38.29**	**39.48**	**51.00**	39.50	**31.50**	**37.00**	**49.95**	**41.79**
Qwen3-8B										
Vanilla	28.08	36.55	29.74	31.46	47.50	46.00	37.50	34.50	42.21	41.54
GRPO	31.59	39.75	32.54	34.63	52.50	45.50	34.50	36.50	40.64	41.93
ToolRL	25.57	35.31	26.72	29.20	41.00	39.50	31.50	25.00	32.93	33.99
FTRL	32.32	38.87	32.85	34.68	51.50	45.00	35.50	34.00	36.72	40.54
MatchTIR (OT)	33.61	42.56	33.61	36.59	55.50	**52.00**	38.50	36.00	45.80	45.56
MatchTIR (KM)	36.33	44.18	37.33	39.28	60.00	49.00	39.00	40.50	46.16	46.93
HiDiffTIR	**36.90**	**44.82**	**39.04**	**40.25**	**61.50**	49.00	**40.00**	**42.00**	**51.26**	**48.75**

4.1 Experimental Setting

Training Dataset. We conduct training on the FTRL dataset (Ye et al., 2025b), which comprises 2,215 training data generated via an automated environment construction pipeline. By executing all tools locally as code, it circumvents the unreliability of online APIs, providing stable and verifiable tool-use training.

Evaluation Datasets. We evaluate HiDiffTIR on three widely used TIR benchmarks: (1) **FTRL** (Ye et al., 2025b), the held-out test dataset generated by the FTRL pipeline; (2) **ToolHop** (Ye et al., 2025a), a query-driven benchmark specifically designed to evaluate multi-hop tool reasoning; and (3) the **Berkeley Function Call Leaderboard (BFCL)** (Patil et al., 2025), a comprehensive and executable function call benchmark that evaluates function-calling ability across diverse APIs and parameter configurations.

Baselines. We compare HiDiffTIR against several strong TIR baselines: (1) **Vanilla**, which directly performs TIR using the backbone Qwen3 (Yang et al., 2025) model without RL training; (2) **GRPO** (Shao et al., 2024), the standard GRPO framework, which computes advantages solely based on the outcome reward; (3) **ToolRL** (Qian et al., 2025), an RL-based TIR training framework that introduces process-based tool-call verification rewards; (4) **FTRL** (Ye et al., 2025b), a feedback-driven training framework using a verifiable reward mechanism based on precision and completeness; and (5) **MatchTIR** (Qu et al., 2026), a recent fine-grained credit assignment framework that derives turn-level rewards with two credit assignment manners: **MatchTIR (KM)**, which formulates the assignment as a bipartite matching problem using the Kuhn-Munkres algorithm (Kuhn, 1955), and **MatchTIR (OT)**, which smooths the assignment through optimal transport (Cuturi, 2013).

Implementation Details. We conduct experiments using Qwen3-4B and Qwen3-8B (Yang et al., 2025) as the backbone models. All RL training is implemented based on the verl framework (Sheng et al., 2025), with vLLM (Kwon et al., 2023) accelerating the rollout generation. For GRPO settings, we sample a group size of 16 responses per prompt at a temperature of 1.0. The policy model is trained for 5 epochs using a global batch size of 256, restricting the multi-turn reasoning process to a maximum of 10 turns per trajectory. We set the

Table 2: Ablation study on credit assignment using Qwen3-4B.

Methods	FTRL				BFCL				ToolHop	Avg.
	Solve-P	Solve-R	Solve-F1	**Avg.**	Base	MF	MP	LC	AC	
HiDiffTIR	**35.41**	**44.74**	**38.29**	**39.48**	51.00	39.50	**31.50**	**37.00**	49.95	**41.79**
w/o trajectory-level	33.65	40.69	35.77	36.70	49.50	**40.00**	30.00	32.50	51.36	40.67
w/o turn-level	33.43	42.45	36.13	37.34	**51.50**	38.50	29.00	34.50	**51.96**	41.09

smoothing constant λ to 1.0 and the fusion coefficient α to 0.7. All experiments are executed on a single node equipped with 8 NVIDIA H20 GPUs. Additional implementation details are provided in Appendix A.

4.2 Main Results

The main experimental results are presented in Table 1. On the basis of these results, we make the following observations.

HiDiffTIR consistently achieves the best overall performance across different evaluation benchmarks. HiDiffTIR outperforms all baselines on the average score for both Qwen3-4B and Qwen3-8B. Compared with existing RL-based TIR methods, our method demonstrates greater improvements in both in-domain and out-of-domain settings. In particular, our method consistently surpasses recent strong baselines such as FTRL and MatchTIR, showing that incorporating hierarchical difficulty-aware optimization provides complementary benefits beyond existing process-based and turn-level reward modeling strategies.

The effectiveness of HiDiffTIR generalizes across different backbone sizes. Although larger backbones generally achieve stronger overall performance, HiDiffTIR consistently improves upon the corresponding baselines under both the 4B and 8B settings. Notably, the performance gains on the in-domain FTRL benchmark are more pronounced for Qwen3-4B, suggesting that smaller models benefit more from fine-grained difficulty-aware optimization. This observation indicates that improved credit assignment can partially compensate for weaker intrinsic reasoning and tool-use capabilities in smaller models. Meanwhile, the consistent improvements on Qwen3-8B demonstrate the robustness and scalability of our framework.

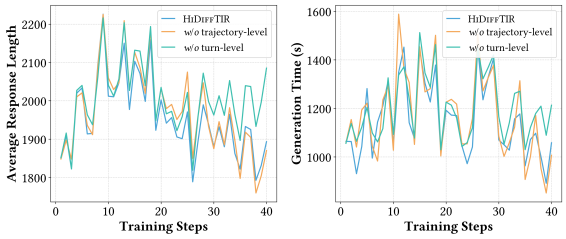
HiDiffTIR brings larger improvements on datasets with a larger number of tools. As shown in Table 5, datasets such as FTRL and ToolHop contain substantially larger numbers of tools compared with BFCL. Correspondingly, HiDiffTIR achieves more evident improvements on these benchmarks, which require reasoning over thousands of candidate tools. In contrast, on datasets with relatively fewer tools, such as BFCL, the performance improvements are comparatively modest, while stronger backbone models already obtain noticeable benefits from scaling alone. These findings highlight that the advantages of difficulty-aware optimization become increasingly important as the complexity of the tool environment grows.

Current RL-based TIR methods still struggle with missing-information scenarios. Although most methods achieve clear improvements on the BFCL benchmark, the gains on the Missing Functions (MF) and Missing Parameters (MP) subsets remain modest, as they require the model to recognize when the currently available tools or user-provided information are insufficient. Existing RL-based TIR training mainly emphasizes successful tool execution and task completion, which may bias the policy toward aggressively invoking tools rather than learning insufficiency detection. Specifically, because our difficulty-aware reweighting amplifies successful complex tool sequences, it inadvertently reinforces this aggressive calling bias in sparse tool settings. Consequently, while HiDiffTIR significantly outperforms most baselines on the MF subset, it slightly trails MatchTIR. This observation suggests an important future direction for TIR, i.e., integrating self-awareness of capability boundaries into RL optimization.

4.3 Ablation Study

We further conduct ablation studies on Qwen3-4B to evaluate the contribution of each component in HiDiffTIR.

Effectiveness of Hierarchical Components. As shown in Table 2, removing either trajectory-level or turn-level difficulty-aware credit assignment leads to consistent performance degradation compared with the full model, demonstrating that both components contribute to the final performance. In particular, removing trajectory-level optimization



(a) Average Response Length (b) Generation Time

Figure 3: Training dynamics regarding the average response length and generation time of different variants.

Table 3: Comparison of reasoning efficiency at the final training step.

Methods	Avg. Resp. Len.	Gen. Time (s)
HiDiffTIR	1,893	1,058
w/o trajectory-level	1,870	1,007
w/o turn-level	2,085	1,213

results in a larger overall drop on FTRL, indicating that distinguishing the relative difficulty of different trajectories provides an important optimization signal for RL training. Furthermore, performance variations on Out-Of-Domain (OOD) datasets like BFCL and ToolHop are less pronounced when omitting specific components. Since our difficulty signals are derived dynamically during training, they naturally target bottlenecks within the training distribution. Thus, these components are highly effective for seen tools but have a more constrained impact on completely unseen tools in OOD scenarios, although the full HiDiffTIR model still achieves the best overall average performance.

Efficiency of Hierarchical Components. The training dynamics of reasoning efficiency are visualized in Figure 3, with convergence values summarized in Table 3. We observe that HiDiffTIR achieves higher reasoning compactness compared to the *w/o* turn-level variant, which relies solely on trajectory-level credit assignment. Specifically, at the final training step, HiDiffTIR reduces the average response length from 2,085 to 1,893 tokens, resulting in a 12.8% decrease in generation latency. This suggests that fine-grained turn-level reweighting effectively identifies critical reasoning steps and suppresses redundant tool calls. Although the *w/o* trajectory-level variant exhibits slightly lower latency, its performance is substantially inferior to the full model, confirming that our hierarchical design achieves an optimal trade-off between reasoning accuracy and computational efficiency.

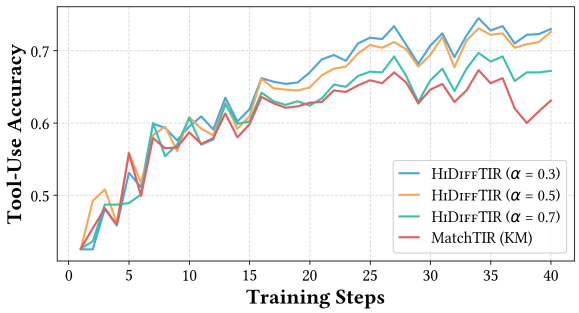


Figure 4: Training dynamics regarding tool-use accuracy of different models.

Table 4: Performance of models on FTRL dataset under different fusion coefficients.

Methods	FTRL			
	Solve-P	Solve-R	Solve-F1	Avg.
HiDiffTIR ($\alpha = 0.3$)	32.72	37.70	33.67	34.70
HiDiffTIR ($\alpha = 0.5$)	33.96	40.33	35.79	36.69
HiDiffTIR ($\alpha = 0.7$)	**35.41**	**44.74**	**38.29**	**39.48**

4.4 Tool-Use Accuracy Analysis

In this section, we analyze how difficulty-aware credit assignment influences the tool-use accuracy.

Impact on Tool Selection Precision. As illustrated in Figure 4, all variants of HiDiffTIR consistently outperform the competitive baseline MatchTIR (KM) in tool-use accuracy throughout the training process. This convergence to higher accuracy confirms that integrating difficulty-aware credit assignment provides a clearer gradient signal, effectively guiding the policy to master correct tool invocation patterns. Specifically, variants with smaller fusion coefficients ($\alpha = 0.3$ and $\alpha = 0.5$) exhibit higher peak tool accuracy during training compared to $\alpha = 0.7$. This trend occurs because a lower α assigns a larger proportional weight to the turn-level difficulty-aware credit assignment, forcing the policy to aggressively optimize high-difficulty tool calls within training rollouts.

Sensitivity Analysis of Fusion Coefficient. Despite the higher training accuracy at lower α values, Table 4 reveals that $\alpha = 0.7$ achieves the highest average score on the FTRL test set. This counter-intuitive phenomenon highlights the cooperative dynamics between the two credit assignment levels. A smaller α over-emphasizes turn-level difficulty-aware credit assignment, rigidly rewarding only ground-truth tool invocations and suppressing intermediate turns with necessary trial-and-error ex-

ploration. In complex TIR, such exploration is not meaningless but is indispensable for the agent to gather feedback and learn from failures. Conversely, setting $\alpha = 0.7$ provides a well-suited balance, leveraging trajectory-level credit assignment to safeguard the value of the exploratory sequence while utilizing turn-level credit assignment to resolve specific tool-use bottlenecks.

5 Conclusions

In this paper, we propose HiDiffTIR, a hierarchical policy optimization framework for TIR that performs difficulty-aware credit assignment at both the trajectory and turn levels. By modeling the relative difficulty of trajectories and redistributing advantages across reasoning steps, our method provides fine-grained learning signals that better capture the heterogeneous importance of tool-calling decisions, without requiring additional supervision. Extensive experiments on multiple tool-using benchmarks demonstrate that HiDiffTIR consistently improves multi-turn TIR performance and tool invocation accuracy, suggesting that incorporating difficulty-aware signals is a promising direction for improving the training of tool-using LLM agents.

Limitations

Despite the strong empirical results, our proposed HiDiffTIR has several limitations. Primarily, owing to limited computational resources, our empirical evaluations were restricted to moderately sized open-source models, specifically Qwen3-4B and Qwen3-8B. Although the core algorithmic design of our hierarchical difficulty-aware policy optimization framework is inherently model-agnostic, its behavioral dynamics and scalability when integrated into LLMs with larger parameters have not yet been fully validated. Additionally, because our difficulty-aware signals are derived dynamically from online group-level rollout statistics, the framework implicitly assumes that the base model possesses a foundational level of instruction-following capability. In exceptionally intricate tasks where the initial policy completely fails to produce any valid tool invocations, the empirical difficulty estimation might suffer from extreme reward sparsity, potentially necessitating a supervised warm-up phase to kickstart effective reinforcement learning exploration. Finally, the difficulty estimation mechanism in our framework serves as a heuristic. It captures empirical learning difficulty from the perspective of operational bottlenecks for the policy during RL rollouts, rather than providing an absolute measure of static semantic complexity.

Acknowledgments

This work is partially funded by the National Key Research and Development Program of China under Grant No. 2024YFC3308200, the Strategic Priority Research Program of the CAS under Grant No. XDB0680102, the National Natural Science Foundation of China under Grants 62306299, 62441229 and 62406308, and the Innovation Funding of ICT, CAS under Grant No. E561010.

References

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Marco Cuturi. 2013. Sinkhorn distances: lightspeed computation of optimal transport. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 22922300, Red Hook, NY, USA. Curran Associates Inc.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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

Simon Frieder, Luca Pinchetti, Chevalier Chevalier, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. 2023. Mathematical capabilities of chatgpt. *Advances in neural information processing systems*, 36:27699–27744.

Haowen Gao, zhenyu zhang, Liang Pang, Fangda Guo, douhongjian, Guannan Lv, ShaoGuo Liu, Tingting Gao, Huawei Shen, and Xueqi Cheng. 2026. DIVA-GRPO: Enhancing multimodal reasoning through difficulty-adaptive variant advantage. In *The Fourteenth International Conference on Learning Representations*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, yelong shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu

Chen. 2024. ToRA: A tool-integrated reasoning agent for mathematical problem solving. In *The Twelfth International Conference on Learning Representations*.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean M. Hendryx. 2025. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *NeurIPS 2025 Workshop on Efficient Reasoning*.

Reinhard Heckel, Mahdi Soltanolkotabi, and Christos Thramboulidis. 2026. Asymmetric prompt weighting for reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2602.11128*.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*.

Zenan Huang, Yihong Zhuang, Guoshan Lu, Zeyu Qin, Haokai Xu, Tianyu Zhao, Ru Peng, Jiaqi Hu, Zhanming Shen, Xiaomeng Hu, and 1 others. 2025. Reinforcement learning with rubric anchors. *arXiv preprint arXiv:2508.12790*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*.

H. W. Kuhn. 1955. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

H. W. Kuhn. 1956. Variants of the hungarian method for assignment problems. *Naval Research Logistics Quarterly*, 3(4):253–258.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025a. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5420–5438.

Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, and Zhicheng Dou. 2026. Deepagent: A general reasoning agent with scalable toolsets. In *Proceedings of the ACM Web Conference 2026*, WWW '26, page 22192230, New York, NY, USA. Association for Computing Machinery.

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

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*.

Heng Lin and Zhongwen Xu. 2025. Understanding tool-integrated reasoning. *arXiv preprint arXiv:2508.19201*.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiao Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Meng Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, and 7 others. 2025. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*.

Tula Masterman, Sandi Besen, Mason Sawtell, and Alex Chao. 2024. The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *arXiv preprint arXiv:2404.11584*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565.

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*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang,

Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*.

Changle Qu, Sunhao Dai, Hengyi Cai, Jun Xu, Shuaiqiang Wang, and Dawei Yin. 2026. Matchtir: Fine-grained supervision for tool-integrated reasoning via bipartite matching. *arXiv preprint arXiv:2601.10712*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.

Joykirat Singh, Yash Pandya, Pranav Vajreshwari, Raghav Magazine, and Akshay Nambi. 2025. Agentic reasoning and tool integration for LLMs via reinforcement learning. In *First Workshop on Foundations of Reasoning in Language Models*.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024. A survey on large language model based autonomous agents. *Front. Comput. Sci.*, 18(6).

Weiyun Wang, Zhangwei Gao, Lianjie Chen, Zhe Chen, Jinguo Zhu, Xiangyu Zhao, Yangzhou Liu, Yue Cao, Shenglong Ye, Xizhou Zhu, Lewei Lu, Haodong Duan, Yu Qiao, Jifeng Dai, and Wenhai Wang. 2025. Visualprm: An effective process reward model for multimodal reasoning. *arXiv preprint arXiv:2503.10291*.

Weikai Xu, Chengrui Huang, Shen Gao, and Shuo Shang. 2025. Llm-based agents for tool learning: A survey. *Data Science and Engineering*, pages 1–31.

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. *arXiv preprint arXiv:2509.02479*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Junjie Ye, Zhengyin Du, Xuesong Yao, Weijian Lin, Yufei Xu, Zehui Chen, Zaiyuan Wang, Sining Zhu, Zhiheng Xi, Siyu Yuan, Tao Gui, Qi Zhang, Xuanjing Huang, and Jiecao Chen. 2025a. ToolHop: A query-driven benchmark for evaluating large language models in multi-hop tool use. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2995–3021, Vienna, Austria. Association for Computational Linguistics.

Junjie Ye, Changhao Jiang, Zhengyin Du, Yufei Xu, Xuesong Yao, Zhiheng Xi, Xiaoran Fan, Qi Zhang, Xuanjing Huang, and Jiecao Chen. 2025b. Feedback-driven tool-use improvements in large language models via automated build environments. *arXiv preprint arXiv:2508.08791*.

Jixiao Zhang and Chunsheng Zuo. 2025. Grpo-lead: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5642–5665.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 10495–10516.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. 2023. Pytorch fsdp: Experiences on scaling fully sharded data parallel. *Proc. VLDB Endow.*, 16(12):38483860.

A Implementation Details

A.1 Datasets and Evaluation Metrics

Datasets. The statistics of the training and evaluation datasets are shown in Table 5.

Table 5: Dataset statistics

Dataset	# of queries	# of tools
FTRL (train)	2,215	4,433
FTRL (test)	200	1,109
ToolHop	995	3,912
BFCL (multi-turn)	800	84

Evaluation Metrics. We evaluate model performance using metrics of each benchmark.

- **FTRL** (Ye et al., 2025b). Let N_{gen} denote the total number of tool calls generated by the model, N_{gt} denote the total number of required tool calls in the ground-truth trajectory, and N_{match} denote the number of successfully matched tool calls. The metrics of FTRL are defined as follows:
  - **Solve-P** measures the precision of tool invocations.
  $$\text{Solve-P} = \begin{cases} \frac{N_{match}}{N_{gen}} & \text{if } N_{gen} > 0 \\ 1.0 & \text{if } N_{gen} = 0 \end{cases} \tag{18}$$
  - **Solve-R** evaluates the completeness of the task execution.
  $$\text{Solve-R} = \frac{N_{match}}{N_{gt}} \tag{19}$$
  - **Solve-F1** is the harmonic mean of Solve-P and Solve-R.
  $$\text{Solve-F1} = \frac{2 \cdot N_{match}}{N_{gen} + N_{gt}} \tag{20}$$
- **ToolHop** (Ye et al., 2025a). The metric of ToolHop is **Answer Correctness (AC)**. Assume that for a test instance, the standard answer is a and the model's final response is o. The performance is evaluated as:
$$\text{AC} = \begin{cases} 1 & \text{if } a \in o \\ 0 & \text{otherwise} \end{cases}. \tag{21}$$
- **BFCL** (Patil et al., 2025). In BFCL V3, model performance is evaluated across several categories, including Base Multi-Turn (**Base**), Missing Functions (**MF**), Missing Parameters (**MP**), and Long-Context Multi-Turn (**LC**).

A.2 Baselines

We compare HiDiffTIR with following baselines:

- **Vanilla**. We deploy the backbone Qwen3 (Yang et al., 2025) model using standard generation configurations without any further RL training. This serves as the lower bound, demonstrating the inherent instruction-following and tool-use capabilities of the model.
- **GRPO** (Shao et al., 2024). GRPO serves as the standard trajectory-level RL baseline. During training, for a given prompt, it generates a group of G trajectories, evaluates the final outcome of each trajectory to assign a reward. The advantages are computed by normalizing these rewards within the group and are subsequently broadcast uniformly to all tokens in the trajectory.
- **ToolRL** (Qian et al., 2025). This framework introduces process-based supervision to TIR. Rather than relying solely on outcome rewards, ToolRL incorporates intermediate verification rewards such as the correctness of tool names, parameter names, and parameter values. These intermediate rewards are aggregated into a single trajectory reward, and the resulting advantage is then shared across the entire trajectory during policy optimization.
- **FTRL** (Ye et al., 2025b). The method accompanying the FTRL dataset. It employs a verifiable reward mechanism that evaluates the precision of tool invocations and the completeness of task execution. These factors are combined to calculate a total reward, which is used to derive a single advantage value broadcast to all tokens in the sequence.
- **MatchTIR** (Qu et al., 2026). The state-of-the-art fine-grained credit assignment TIR framework that derives dense, turn-level advantages by aligning generated trajectories with ground-truth tool-use sequences. We evaluate two implementations of its alignment module: (1) **MatchTIR (KM)** formulates the alignment as a hard bipartite matching problem. It uses the Kuhn-Munkres (KM) algorithm (Kuhn, 1955) to find the optimal one-to-one mapping between generated and ground-truth tool calls based on semantic similarity, distributing rewards strictly to matched pairs. (2)

Table 6: Training prompt template for the policy model.

Training Prompt Template for the Policy Model

system
Tools
You may call one or more functions to assist with the user query.
You are provided with function signatures within <tools></tools> XML tags:
<tools>
{Tool Descriptions}
</tools>
For each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>

user
Please call given tools to answer the question. Please note that all your information must be obtained by calling tools and not by answering the question directly.
If the call fails, you need to try to correct it and continue until you arrive at an answer. Only output the final answer (in words, numbers or phrase) inside the <answer></answer> tag, without any explanations or extra information.
Question: {question}

assistant

MatchTIR (OT) employs Optimal Transport (OT) (Cuturi, 2013) with Sinkhorn solver to create a soft alignment matrix. This allows for a probabilistic distribution of rewards across multiple generated turns that share semantic overlap with the ground truth.

A.3 Tool Call Similarity Computation

For reward computation, a matching matrix $M_i \in \mathbb{R}^{m \times n}$ is constructed for each trajectory $\tau^{(i)}$, where each entry $M_i[u, v]$ measures the alignment between a predicted tool call $c_{iu} \in C_i$ and a ground-truth tool call $c^*_{iv} \in C^*_i$. The similarity score $M_i[u, v]$ consists of three components below, following previous works (Qian et al., 2025; Qu et al., 2026).

Tool Name Matching. Let c^{name}_{iu} and $c^{\text{name}*}_{iv}$ denote the tool names of the predicted and ground-truth calls. The tool name score is

$$s_{\text{name}} = \mathbb{I}(c^{\text{name}}_{iu} = c^{\text{name}*}_{iv}) \in \{0, 1\}, \tag{22}$$

where $\mathbb{I}[\cdot]$ is an indicator function.

Parameter Name Matching. If the tool names match, the parameter name similarity is computed as the Jaccard similarity over the sets of parameter names $P_{c_{iu}}$ and $P_{c^*_{iv}}$:

$$s_{\text{param}} = \frac{|P_{c_{iu}} \cap P_{c^*_{iv}}|}{|P_{c_{iu}} \cup P_{c^*_{iv}}|} \in [0, 1]. \tag{23}$$

Parameter Value Matching. Finally, the correctness of parameter values is assessed for each ground-truth parameter $k \in P_{c^*_{iv}}$:

$$s_{\text{value}} = \sum_{k \in P_{c^*_{iv}}} \mathbb{I}(c_{iu}[k] = c^*_{iv}[k]) \in [0, |P_{c^*_{iv}}|]. \tag{24}$$

The three components are combined and normalized to produce the final similarity score:

$$M_i[u, v] = s_{\text{name}} \cdot \frac{s_{\text{name}} + s_{\text{param}} + s_{\text{value}}}{2 + |P_{c^*_{iv}}|} \in [0, 1]. \tag{25}$$

Table 7: Performance comparison of HiDiffTIR on Qwen3-4B with different group sizes G.

Methods	FTRL				BFCL				ToolHop	Avg.
	Solve-P	Solve-R	Solve-F1	**Avg.**	Base	MF	MP	LC	AC	
HiDiffTIR (G=4)	30.24	36.93	31.93	33.03	49.50	41.00	26.00	34.50	49.95	40.19
HiDiffTIR (G=8)	32.90	40.65	34.74	36.10	47.50	37.50	30.00	32.50	**51.36**	39.77
HiDiffTIR (G=16)	**35.41**	**44.74**	**38.29**	**39.48**	**51.00**	**39.50**	**31.50**	**37.00**	49.95	**41.79**

A.4 Training Prompt Template

The training prompt template for the policy model is shown in Table 6.

A.5 Training Details

To facilitate reproducibility, we provide the complete set of hyperparameters and infrastructure configurations used for training HiDiffTIR.

Infrastructure and Hardware. All models are trained on a single compute node equipped with 8 NVIDIA H20 GPUs. We leverage Fully Sharded Data Parallel (FSDP) (Zhao et al., 2023) to distribute the policy model weights, avoiding parameter and optimizer offloading to maximize throughput. For rollout generation, we utilize vLLM (Kwon et al., 2023) with a GPU memory utilization threshold of 0.7.

Data and Generation Settings. We format the prompts using the standard Qwen3 system style and explicitly enable the generation of reasoning trajectories. To handle long trajectories, we set the maximum model length to 32,768 tokens and the maximum response length to 8,192 tokens. During the rollout phase, we sample 16 responses per prompt ($G = 16$) with a generation temperature of 1.0 and a maximum turn of 10.

Optimization and Loss Formulation. The policy is optimized over 5 total epochs. We use a global training batch size of 256 prompts, broken down into mini-batch sizes of 32. The actor learning rate is fixed at 10^{-6} without a critic warmup phase. The entropy coefficient is set to 0.001, and the KL penalty coefficient is set to 0.001.

B Additional Experiments

B.1 Group Size Analysis

The group size G is a critical hyperparameter in GRPO-based frameworks, as it determines the sample size used to estimate the baseline and normalize rewards within each group. In HiDiffTIR, G further influences the robustness of the empirical difficulty estimation used for both trajectory-level and turn-level credit assignment. To investigate this impact, we conduct experiments on Qwen3-4B with $\alpha = 0.7$ across three different group sizes: 4, 8, and 16. The experimental results are summarized in Table 7. We observe distinct performance trends between the In-Domain (ID) and Out-Of-Domain (OOD) metrics.

For the ID FTRL dataset, performance consistently improves as the group size increases. The ID average score rises steadily from 33.03 ($G = 4$) to 39.48 ($G = 16$). This trend indicates that our hierarchical difficulty-aware mechanisms benefit from larger group sizes, which provide more statistically robust estimations of group-level statistics within each training iteration. Accurately quantifying the relative difficulty of trajectories and turns allows the model to better identify and optimize informative reasoning steps in a difficulty-aware manner, thus improving ID task success rates.

Conversely, for OOD scenarios, the relationship is not strictly monotonic. While the optimal OOD average score of 41.79 is achieved at $G = 16$, $G = 8$ actually shows a slight regression (39.77) compared to $G = 4$ (40.19). This volatility stems from the intrinsic mechanism of our difficulty-aware framework. By design, HiDiffTIR utilizes group rollouts to intensively optimize the policy toward mastering seen tools, particularly focusing on resolving high-difficulty tool-calling bottlenecks. While this targeted credit assignment significantly enhances the model's proficiency and precision on seen tool-use patterns, it can lead to behavioral variances when the model encounters entirely unseen tools in OOD environments, as their underlying API structures and difficulty characteristics differ from the training set. Nevertheless, $G = 16$ still yields a well-suited balance, achieving the highest performance on both in-domain tasks and overall OOD generalization.

B.2 Case Study

To intuitively demonstrate the superiority of HIDIFFTIR, we present a qualitative case study involving a complex 5-hop reasoning query. As shown in Table 8, the user query requires identifying a specific local handicraft through a chain of five implicit entities. We compare our approach against the untrained Qwen3-4B backbone and MatchTIR (KM), a strong baseline built on the same backbone, with results reported in Tables 9-11, respectively. As illustrated in the interaction trajectories, HIDIFFTIR exhibits the following distinct advantages.

Mitigation of Parameter Hallucination and Error Loops. Both baseline models exhibit a tendency to hallucinate tool parameters during the initial reasoning phase. As shown in Table 10 and Table 11, when calling the initial `famous_peak_identifier` tool, both Qwen3-4B and MatchTIR incorrectly append unnecessary geographic constraints, hallucinating "Alps" or "Himalayas" instead of passing the required empty arguments. While MatchTIR manages to correct itself after wasting two turns, the untrained Qwen3-4B model lacks this reflective capability, entering a fatal trial-and-error loop that continuously guesses random regions until it exhausts the maximum turn limit. In contrast, HIDIFFTIR completely suppresses these hallucinated priors, immediately generating the precise arguments required to navigate the tool logic.

Calibration of Overconfidence and Adaptation to Environmental Feedback. A further striking observation from the trajectories is the backbone model's overconfidence in its parametric knowledge. When facing consecutive empty responses from the tools, the untrained Qwen3-4B exhibits severe resistance to self-correction. Instead of rectifying its over-specified parameters, the model stubbornly suspects that the tool itself is flawed (e.g., explicitly assuming in Turn 6 and Turn 9 that "the tool's data" or "the tool's database" is incomplete). This overconfidence traps the agent, preventing it from utilizing negative environmental feedback. While the RL-trained MatchTIR largely mitigates this extreme overconfidence and demonstrates the capacity to adjust based on environmental feedback, it still exhibits a lingering over-reliance on its internal parametric knowledge, frequently defaulting to hallucinated assumptions during exploration. Moving beyond this behavior, HIDIFFTIR effectively neutralizes this stubbornness, teaching the LLM agent to respect and adapt to strict tool constraints rather than blindly persisting with its hallucinated assumptions.

Enhanced Reasoning Efficiency and Minimal Redundancy. Beyond hallucination and overconfidence, baseline models also struggle with inefficient and redundant exploration. As observed in the trajectories, whenever the untrained Qwen3-4B faces an obstacle, it continuously appends arbitrary parameters (e.g., `include_protected` in Turn 8 and `environmental_certification` in Turn 9) in futile attempts to force a result. Even the RL-trained MatchTIR, despite eventually reaching the correct answer, exhibits similar inefficiency by repeatedly over-specifying non-essential arguments (e.g., guessing `radius` in Turn 4 and extra boolean constraints in Turn 7). In contrast, HIDIFFTIR learns a highly compact reasoning policy with minimal redundancy. In this specific case, it executes the entire task flawlessly in exactly five optimal tool-calling turns, achieving absolute zero redundancy. This highlights the core advantage of our hierarchical difficulty-aware policy optimization framework. By accurately amplifying the learning signals for high-difficulty bottleneck steps, the framework implicitly penalizes superfluous exploratory actions, yielding a much more efficient and precise reasoning trajectory compared to uniform or heuristic-based credit assignments.

Table 8: The user query and ground truth answer of the case study.

User Query: What is the local handicraft of the town where the woodworker who carves sculptures using the wood from the forest near the mountain with a famous peak?

Ground Truth Tool Calls:

1. `{"name": "famous_peak_identifier", "arguments": {}}`
2. `{"name": "nearby_forest_locator", "arguments": {"landmark": "Mount Everest"}}`
3. `{"name": "wood_sourcing_locator", "arguments": {"forest_name": "Langtang National Park Forest", "product_type": "sculptures"}}`
4. `{"name": "individual_location_finder", "arguments": {"individual_name": "Ramesh Thapa"}}`
5. `{"name": "local_handicraft_identifier", "arguments": {"location": "Pokhara"}}`

Ground Truth Answer: Wooden masks

Table 9: Interaction trajectory of HiDiffTIR. The model conducts precise TIR by successfully chaining five sequential tool calls without any hallucinations or redundant trial-and-error.

HiDiffTIR

[Turn 1]

<think> Okay, let's tackle this question step by step ... So, the plan is to first find the mountain with a famous peak. Then, determine the forest near that mountain. Next, find the woodworker who uses that forest's wood to carve sculptures. Then, locate the town where this woodworker is based, and finally, identify the local handicraft of that town ... Starting with the mountain with a famous peak. The tool 'famous_peak_identifier' can help here ... </think>

<tool_call> `{"name": "famous_peak_identifier", "arguments": {}}` </tool_call>

<tool_response> Mount Everest - The highest mountain in the world, known for its climbing fame. </tool_response>

[Turn 2]

<think> Okay, so the first step was to find a mountain with a famous peak, and the response was Mount Everest. Now, the next part of the question is to find a forest near this mountain. The tool to use here is nearby_forest_locator ... </think>

<tool_call> `{"name": "nearby_forest_locator", "arguments": {"landmark": "Mount Everest"}}` </tool_call>

<tool_response> `{"forest": "Langtang National Park Forest"}` </tool_response>

[Turn 3]

<think> Okay, so the forest near Mount Everest is Langtang National Park Forest. Now the next step is to find a woodworker who carves sculptures using wood from this forest. The relevant tool here is wood_sourcing_locator ... </think>

<tool_call> `{"name": "wood_sourcing_locator", "arguments": {"forest_name": "Langtang National Park Forest", "product_type": "sculptures"}}` </tool_call>

<tool_response> Ramesh Thapa is a woodworker who sources wood from Langtang National Park Forest. </tool_response>

[Turn 4]

<think> Okay, now I know the woodworker is Ramesh Thapa. The next step is to find out where he's based. The tool to use here is individual_location_finder ... </think>

<tool_call> `{"name": "individual_location_finder", "arguments": {"individual_name": "Ramesh Thapa"}}` </tool_call>

<tool_response> `{"location": "Pokhara"}` </tool_response>

[Turn 5]

<think> Okay, now I know Ramesh Thapa is based in Pokhara. The next step is to find the local handicraft of Pokhara. The tool to use here is local_handicraft_identifier ... </think>

<tool_call> `{"name": "local_handicraft_identifier", "arguments": {"location": "Pokhara"}}` </tool_call>

<tool_response> Wooden masks </tool_response>

[Turn 6]

<think> ... All the tool calls led to the correct answers, so the final answer should be wooden masks. </think>

The local handicraft of the town where the woodworker who carves sculptures using wood from the forest near the mountain with a famous peak (Mount Everest) is **wooden masks**.

Table 10: Interaction trajectory of Qwen3-4B. The model hallucinates tool parameters, enters a trial-and-error loop, and ultimately fails to complete the reasoning chain within the turn limit.

Qwen3-4B

[Turn 1] ***[Issue: Parameter Hallucination & Redundant Over-specification]***

<think> Okay, let's tackle this question step by step ... So, the plan is to first find the mountain with a famous peak ... Let's assume the mountain is in a region like the Alps ... </think>

<tool_call> {"name": "famous_peak_identifier", "arguments": {"region": "Alps", "fame_type": "climbing"}} </tool_call>

<tool_response> No famous peaks found matching the criteria. </tool_response>

[Turn 2] ***[Issue: Parameter Hallucination & Redundant Over-specification]***

<think> Okay, the first tool call didn't find any famous peaks in the Alps with a climbing reputation. Maybe I need to adjust the parameters ... So, I'll call the famous_peak_identifier again with region set to 'Himalayas' and fame_type as 'scenic'. </think>

<tool_call> {"name": "famous_peak_identifier", "arguments": {"region": "Himalayas", "fame_type": "scenic"}} </tool_call>

<tool_response> No famous peaks found matching the criteria. </tool_response>

[Turns 3-5] (*Repeated failed attempts to find the peak*) ***[Issue: Trapped in Error Loop]***

[Turn 6] ***[Issue: Parameter Hallucination & Redundant Over-specification & Overconfidence]***

<think> Okay, after several attempts to find a famous peak using different regions and fame types, it seems the tool isn't returning any results. Maybe the issue is with the parameters or the tool's data ... Let's try using a specific mountain name, like Mount Fuji, and check if it's in the Himalayas ... </think>

<tool_call> {"name": "famous_peak_identifier", "arguments": {"region": "Japan", "fame_type": "cultural", "historical_event": "first_ascent"}} </tool_call>

<tool_response> Mount Fuji - A cultural icon of Japan. </tool_response>

[Turn 7] ***[Issue: Redundant Over-specification]***

<think> ... The next step is to find the forest near Mount Fuji. Using the 'nearby_forest_locator' tool with the landmark as Mount Fuji and a reasonable radius, say 50 km ... </think>

<tool_call> {"name": "nearby_forest_locator", "arguments": {"landmark": "Mount Fuji", "region": "Japan", "radius": 50}} </tool_call>

<tool_response> {"forest": "No specific forest found near the given landmark." } </tool_response>

[Turn 8] ***[Issue: Parameter Hallucination & Redundant Over-specification]***

<think> ... Alternatively, perhaps the forest is part of a protected area or has a different name ... Let me try adjusting the radius or region ... So, next step: call 'nearby_forest_locator' again with ... a larger radius, say 100 km, to cover more area. </think>

<tool_call> {"name": "nearby_forest_locator", "arguments": {"landmark": "Mount Fuji", "region": "Japan", "radius": 100, "include_protected": true}} </tool_call>

<tool_response> {"forest": "No specific forest found near the given landmark." } </tool_response>

[Turn 9] ***[Issue: Parameter Hallucination & Redundant Over-specification & Overconfidence]***

<think> ... Maybe the issue is that the landmark name isn't precise enough, or the tool's database doesn't have entries for forests near Mount Fuji ... Let me try a different approach ... Let me use the 'wood_sourcing_locator' tool with the forest name as "Mount Fuji Forest" and product type as "sculptures" ... </think>

<tool_call> {"name": "wood_sourcing_locator", "arguments": {"forest_name": "Mount Fuji Forest", "product_type": "sculptures", "region": "Japan", "environmental_certification": "FSC"}} </tool_call>

<tool_response> No specific woodworker information available for the given forest. </tool_response>

[Turn 10] ***[Issue: Parameter Hallucination & Redundant Over-specification]***

<think> ... Now, since the tools aren't yielding specific results, maybe the approach needs adjustment ... Let's try using the 'local_handicraft_identifier' tool with Japan as the location, assuming the handicraft is related to woodcarving ... </think>

<tool_call> {"name": "local_handicraft_identifier", "arguments": {"location": "Japan", "craft_type": "woodwork", "cultural_significance": true}} </tool_call>

<tool_response> Information on local handicrafts is not available for the specified location. </tool_response>

[Generation Terminated: Maximum turn limit (10) reached without providing a final answer.]

Table 11: Interaction trajectory of MatchTIR (KM). Although the model successfully reasons out the correct target answer, it exhibits redundancy in the early reasoning stages due to over-specification of geographic constraints.

MatchTIR (KM)

[Turn 1] ***[Issue: Parameter Hallucination & Redundant Over-specification]***

<think> Okay, let's tackle this question step by step ... So, the first thing I need to do is find out which mountain has a famous peak. The tool "famous_peak_identifier" can help with that ... Let's assume the mountain is a well-known one, maybe in a region like the Alps or Himalayas ... </think>

<tool_call> {"name": "famous_peak_identifier", "arguments": {"region": "Alps", "fame_type": "climbing"}} </tool_call>

<tool_response> No famous peaks found matching the criteria. </tool_response>

[Turn 2] ***[Issue: Parameter Hallucination & Redundant Over-specification]***

<think> Okay, the first tool call didn't return any results. Maybe the mountain isn't in the Alps. ... Let's adjust the region to "Himalayas" and see if that works ... Wait, the tool's parameters allow for optional region, so maybe I should specify a region where famous peaks are common ... </think>

<tool_call> {"name": "famous_peak_identifier", "arguments": {"region": "Himalayas", "fame_type": "scenic"}} </tool_call>

<tool_response> No famous peaks found matching the criteria. </tool_response>

[Turn 3] ***[Issue: Redundant Over-specification]***

<think> Okay, maybe the issue is with the region. Let's try a different approach. Instead of specifying a region, just use the fame_type as "climbing" and see if the tool can find a mountain with a famous peak ... </think>

<tool_call> {"name": "famous_peak_identifier", "arguments": {"fame_type": "climbing"}} </tool_call>

<tool_response> Mount Everest - The highest mountain in the world, known for its climbing fame. </tool_response>

[Turn 4] ***[Issue: Redundant Over-specification]***

<think> ... The tool returned Mount Everest. Now, the next part of the question is to find the forest near this mountain ... I need to use the 'nearby_forest_locator' tool ... I should set the landmark to "Mount Everest" and maybe specify a radius to search around it. The default radius is 50 km, which should be sufficient ... </think>

<tool_call> {"name": "nearby_forest_locator", "arguments": {"landmark": "Mount Everest", "radius": 50}} </tool_call>

<tool_response> {"forest": "Langtang National Park Forest"} </tool_response>

[Turn 5]

<think> Okay, so the forest near Mount Everest is Langtang National Park Forest. Now the next step is to find the woodworker who carves sculptures using wood from this forest ... So I'll call the wood_sourcing_locator with forest_name set to Langtang National Park Forest and product_type as sculptures. </think>

<tool_call> {"name": "wood_sourcing_locator", "arguments": {"forest_name": "Langtang National Park Forest", "product_type": "sculptures"}} </tool_call>

<tool_response> Ramesh Thapa is a woodworker who sources wood from Langtang National Park Forest. </tool_response>

[Turn 6]

<think> ... Ramesh Thapa is the woodworker who sources wood from Langtang National Park Forest. The next step is to find out where he is based ... The tool that can help with that is the 'individual_location_finder' ... The required parameter is individual_name, so I'll set that to "Ramesh Thapa" and maybe include the country as Nepal to narrow it down ... </think>

<tool_call> {"name": "individual_location_finder", "arguments": {"individual_name": "Ramesh Thapa", "country": "Nepal"}} </tool_call>

<tool_response> {"location": "Pokhara"} </tool_response>

[Turn 7] ***[Issue: Redundant Over-specification]***

<think> Okay, so now I know that Ramesh Thapa is based in Pokhara. The next step is to find the local handicraft of Pokhara... Let's call the function with location: Pokhara. </think>

<tool_call> {"name": "local_handicraft_identifier", "arguments": {"location": "Pokhara", "cultural_significance": true, "materials_used": true}} </tool_call>

<tool_response> Wooden masks </tool_response>

[Turn 8]

<think> Okay, so the final answer is that the local handicraft in Pokhara is wooden masks ... All steps seem to connect correctly. The answer should be wooden masks. </think>

The local handicraft of the town where the woodworker who carves sculptures using wood from the forest near the mountain with a famous peak is **wooden masks**.