EMTIR-GRPO: Efficient Multi-Tool Augmented Large Language Models via Reinforcement Learning

Shixin Jiang[1*], **Zhihao Zhu**[1*], **Jiafeng Liang**[1*], **Yang Wu**[1], **Ming Liu**[1,2†], **Bing Qin**[1,2]
[1]Harbin Institute of Technology, Harbin, China
[2]Peng Cheng Laboratory, Shenzhen, China
{sxjiang,zhzh,jfliang,mliu,qinb}@ir.hit.edu.cn

Abstract

Tool-integrated reasoning (TIR) enables large language models (LLMs) to invoke external tools for tasks beyond their internal capacity but often suffers from tool overuse. Existing approaches leverage imitation learning or reward shaping to improve efficiency, yet they mainly target single-tool scenarios and ignore the varying invocation costs across tools in multi-tool reasoning (MTIR). To address these gaps, we propose EMTIR-GRPO, a simple yet effective RL algorithm for cost-aware MTIR. Built upon GRPO, we introduce a composite reward considering format completeness, answer correctness, and tool efficiency. By incorporating a cost-aware coefficient with group optimal cost estimation, EMTIR-GRPO explicitly models heterogeneous tool costs and encourages more cost-effective tool-use strategies. Experiments on MTIR-QA and MTIR-TC demonstrate significant efficiency gains (e.g., Δ+10.9 on Tool-Star-7B and Δ+3.6 on ReCall-7B) while maintaining or even improving accuracy (e.g., 55.4 vs. 52.0 on Tool-Star-7B). Additional budget-constrained and tool-free evaluations further validate its effectiveness in maximizing cost-efficiency and reducing cognitive offloading.

1 Introduction

Recent advances in large language models (LLMs) have demonstrated impressive reasoning capabilities when fine-tuned via reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2024; DeepSeek-AI et al., 2025; Team et al., 2025). However, relying on language-only reasoning capabilities of LLMs is often insufficient for tasks that require interaction with external environments, such as accessing up-to-date domain knowledge or performing precise computations (Qu et al., 2025). To overcome these limitations, the tool-integrated reasoning (TIR) paradigm enables LLMs to interact with external tools—such as search engines, code interpreters, or domain-specific APIs—within a multi-step, feedback-driven loop to arrive at solutions (Paranjape et al., 2023; Qin et al., 2024).

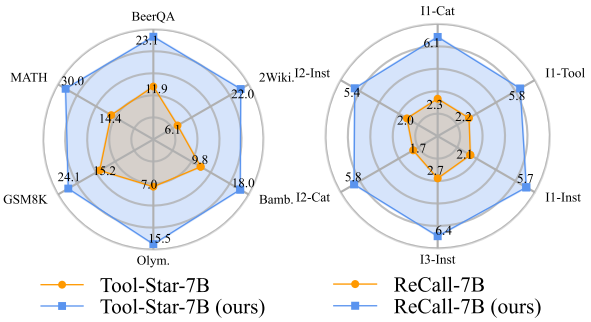

Figure 1: Overview of tool productivity improvements on MTIR-QA (Tool-Star) and MTIR-TC (Tool-Call).

While early efforts to enhance tool-use capabilities focused on in-context learning (ICL) (Yao et al., 2023; Paranjape et al., 2023) and supervised fine-tuning (SFT) (Qin et al., 2024; Schick et al., 2023), more recent work has extended RLVR to TIR (Jin et al., 2025; Li et al., 2025b; Qian et al., 2025a; Dong et al., 2025a). However, despite achieving strong performance and generalization, similar to the *overthinking* in pure text-based reasoning (Sui et al., 2025), current TIR methods that focus on answer correctness often encourage indiscriminate tool usage—*overuse*, which introduces critical challenges during both training and inference (Wang et al., 2025a). On the one hand, frequent and unnecessary tool calls lead to substantial computational and time overheads. On the other hand, excessive reliance on tools can hinder the development and utilization of the intrinsic reasoning ability, a phenomenon referred to as cognitive offloading (Wang et al., 2025b,a). Consequently, recent studies increasingly focus on the efficiency of tool-integrated reasoning in addition to performance.

Early efforts to improve tool-use efficiency re-

[*]Equal contribution
[†]Corresponding author

Findings of the Association for Computational Linguistics: ACL 2026, pages 27877–27894
July 2-7, 2026

lied on prompt engineering and SFT (Wang et al., 2025b; Shen et al., 2024; Qian et al., 2025b; Lyu et al., 2024), while more recent RLVR-based methods such as ActLess and Frugal (Wang et al., 2025a; Java et al., 2025) achieved efficiency gains with reward shaping. Despite their success, existing RL-based methods remain limited to single-tool reasoning and overlook cost-efficient strategies in multi-tool integrated reasoning (MTIR) (Dong et al., 2025a; Paranjape et al., 2023). Meanwhile, existing MTIR approaches (Shen et al., 2024; Qian et al., 2025b) often rely on complex workflows and annotated datasets, limiting their generality. Moreover, most methods (Song et al., 2025) estimate cost by the number of tool calls, ignoring the varying invocation costs of different tools.

To address the challenges of MTIR efficiency, we propose Efficient MTIR with Group Relative Policy Optimization (EMTIR-GRPO), a simple yet effective RL-based algorithm for cost-aware multi-tool reasoning. We first provide a formal definition of GRPO in the MTIR setting. Then, we design a composite reward mechanism that integrates three critical aspects: format completeness, answer correctness, and tool efficiency. We further estimate the trajectory cost of tool usage and further introduce group optimal cost estimation based on a set of sampled trajectories. Combining these elements yields a cost-effectiveness coefficient, which adaptively adjusts the answer reward and encourages the model to pursue more efficient tool-use strategies.

To validate the generality of our method (Singh et al., 2025), we evaluate EMTIR-GRPO in both MTIR-Question Answering (MTIR-QA) (Dong et al., 2025a) and MTIR-Task Completion (MTIR-TC) (Chen et al., 2025). As shown in Figure 1, experimental results demonstrate that our approach improves tool-use efficiency while maintaining the performance. And budget-constrained inference and no-tool reasoning experiments further highlight the ability of EMTIR-GRPO to maximize cost-effectiveness and mitigate cognitive offloading caused by over-reliance on tools. In summary, our key contributions are as follows:

- We propose EMTIR-GRPO, which applies reward shaping to encourage efficient tool use in MTIR while preserving task-solving capability.
- EMTIR-GRPO introduces a cost-aware learning mechanism that accounts for varying tool costs, enabling the construction of cost-efficient agents.
- Extensive experiments on MTIR-QA and MTIR-TC validate the effectiveness of EMTIR-GRPO, as well as its ability to maximize budget utilization and mitigate cognitive offloading.

2 Related Works

Tool Utilization for LLMs Teaching LLMs to use tools enables them to interact with external environments while overcoming several inherent limitations, such as no access to up-to-date or domain-specific knowledge and poor mathematical operation capabilities (Qu et al., 2025). Early TIR methods primarily rely on in-context learning (Yao et al., 2023; Paranjape et al., 2023; Qin et al., 2024) or SFT-based methods (Schick et al., 2023; Gou et al., 2024; Li et al., 2025a). However, these methods often show limited generalization to unseen tasks or tool configurations. Recent work explores training LLMs to explore strategies for appropriate tool use via RLVR (Jin et al., 2025; Li et al., 2025b; Dong et al., 2025a). Empirical results show that RLVR enables models to learn more robust and adaptive tool use strategies (Qian et al., 2025a; Chen et al., 2025).

Tool-Integrated Reasoning Efficiency Previous studies have primarily addressed the efficiency of TIR through prompt engineering and supervised fine-tuning (Wang et al., 2025b; Qian et al., 2025b; Lyu et al., 2024; Guan et al., 2025). For example, SelfDC (Wang et al., 2025b) introduced a prompt-based framework, which leverages self-confidence scores to decide whether external tools should be invoked. And recent works such as Actless (Wang et al., 2025a), FrugalRAG (Java et al., 2025), and InForage (Qian and Liu, 2025) incorporate efficiency factors into reward shaping within the RLVR paradigm, thereby improving generalization (Song et al., 2025). However, their designs are still limited to single-tool settings and only consider the number of tool calls. They ignore varying invocation costs among different tools and cannot be effectively generalized to multi-tool integrated reasoning (MTIR) scenarios, including both MTIR-QA (Dong et al., 2025a) and MTIR-TC (Chen et al., 2025).

3 Methodology

We introduce the MTIR process under the ReAct paradigm (Yao et al., 2023), where multiple tools are orchestrated for reasoning and execution. Then we formulate the cost-aware MTIR optimization problem and propose an RLVR-based optimization

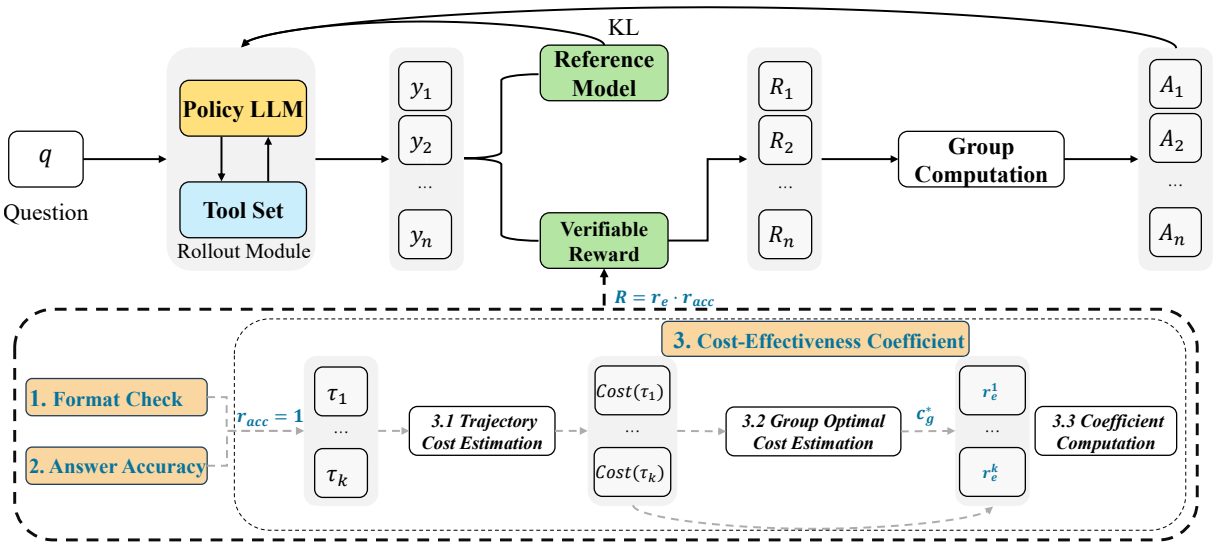

Figure 2: **Overview of the EMTIR-GRPO algorithm**. We extend GRPO with multi-tool integration in rollout and incorporates a composite verifiable reward mechanism with cost-effectiveness coefficients.

algorithm with reward shaping. We further instantiate this algorithm in two scenarios.

3.1 Problem Formulation

The *Multi-Tool Integrated Reasoning* (MTIR) is the process where the language model $\mathcal{M}$ collaborates with multiple external tools to carry out multi-step reasoning. Under the ReAct paradigm, given a question q and an environment $\mathcal{E}$ equipped with a tool set $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ containing n tools, the MTIR trajectory τ_k up at step k is denoted as:

$$\tau_k = (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k), \quad \mathcal{A}_i = \langle r_i, \mathcal{T}_i, o_i \rangle,$$

where $\mathcal{A}_i$ denotes the i-th intermediate reasoning step. Specifically, r_i is the natural language reasoning generated by $\mathcal{M}$ at step i, $\mathcal{T}_i \subseteq \mathcal{T}$ denotes the set of tools invoked at step i, and o_i denotes corresponding results from $\mathcal{E}$ after executing tools in $\mathcal{T}_i$. At each step $k+1$, the model M generates $\mathcal{A}_{k+1}$ conditioned on the trajectory history $(\mathcal{A}_1, \ldots, \mathcal{A}_k)$. This iterative generation continues until it reaches the predefined maximum interaction turns or produces a final answer a, at which point $\mathcal{A}_{k+1} = (r_{k+1})$ with $T_{k+1} = \emptyset$ and $o_{k+1} = \emptyset$.

We consider the efficiency of tool calls conditioned on the correctness of the final answer a with respect to the ground truth $\hat{a}$. During the MTIR process, each tool invocation incurs a non-negligible cost, and we denote the cumulative cost of a trajectory τ as $\mathrm{Cost}(\tau)$. Based on this, we formulate the *cost-aware MTIR optimization problem* as follows:

$$\arg\min_{\tau} \ \mathrm{Cost}(\tau) \quad \text{subject to} \quad \mathrm{Acc}(a, \hat{a}) = 1,$$

where $\mathrm{Acc}(a, \hat{a})$ represents the accuracy evaluation function. With this objective, the model is required not only to ensure accuracy but also to adopt a cost-aware strategy for tool usage. Instead of simply minimizing the number of tool calls (Wang et al., 2025a; Song et al., 2025), we explicitly account for diverse possible cost dimensions (e.g., computational latency, monetary expenditure, or resource consumption) (Zheng et al., 2024; Xu et al., 2025), which is crucial for MTIR involving multiple tools with heterogeneous costs (Wang et al., 2025c).

3.2 EMTIR-GRPO

Previous offline approaches such as SFT or DPO for efficient MTIR often overfit to the cost patterns and tool configurations in offline data (Lyu et al., 2025; Xu et al., 2025). As shown in Figure 2, we adopt the RLVR algorithm (Lambert et al., 2024) with multi-tool integration, which enables sample-efficient and effective online exploration. Moreover, we design a *composite reward mechanism* that explicitly guides the model toward cost-aware MTIR strategies (Wang et al., 2025c).

3.2.1 Group Relative Policy Optimization with Multi-tool Integration

We employ Group Relative Policy Optimization (GRPO) (Shao et al., 2024) as the RLVR algorithm with multi-tool integration. GRPO extends Proximal Policy Optimization (PPO) (Schulman et al., 2017) by eliminating the critic and instead estimating the baseline from a group of sampled responses. Specifically, for each input task $q \sim \mathcal{D}$, GRPO sam-

ples a set of responses $\{y_i\}_{i=1}^{G}$ from the existing policy π_{old}, where y_i consists of the multi-tool integrated reasoning trajectory τ_i and the final answer a_i. The current policy model π_θ is then optimized by maximizing the following objective:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{q\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\text{old}}(\cdot|q;\mathcal{T}))} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{\sum_{t=1}^{|y_i|} I(y_{i,t})} \sum_{t=1:I(y_{i,t})=1}^{|y_i|} \min\left(\frac{\pi_\theta(y_{i,t}|q,y_{i,<t};\mathcal{T})}{\pi_{\text{old}}(y_{i,t}|q,y_{i,<t};\mathcal{T})} A_i, \text{clip}\left(\frac{\pi_\theta(y_{i,t}|q,y_{i,<t};\mathcal{T})}{\pi_{\text{old}}(y_{i,t}|q,y_{i,<t};\mathcal{T})}, 1-\epsilon, 1+\epsilon \right) A_i \right) - \beta\mathbb{D}_{KL}\left[\pi_\theta||\pi_{\text{ref}}\right] \right],$$

where A_i denotes the trajectory advantage of the i-th rollout computed based on the rewards of rollouts within the group G. The hyperparameters ϵ and β control the clipping ratio and the penalty weight for the KL divergence between the current policy π_θ and the reference policy π_{ref}. Moreover, $I(y_{i,t})$ is the token loss masking operation such that $I(y_{i,t}) = 1$ if $y_{i,t}$ is an LLM-generated token and $I(y_{i,t}) = 0$ if $y_{i,t}$ is a token from tool executing results. This masking prevents unintended learning dynamics caused by optimization on observation tokens (Jin et al., 2025). The token masking is also applied to compute the KL divergence loss $\mathbb{D}_{\text{KL}}$.

3.2.2 Cost-Aware Reward Design

To achieve cost-aware MTIR optimization, we construct a response-level verifiable reward grounded in three factors: (1) format completeness, (2) answer correctness, and (3) tool efficiency.

Format Check. This component checks two main criteria: (1) the correct ordering of reasoning, tool calls, and observations; and (2) the final answer is properly enclosed within the designated tags. The format check guides the model to output in a consistent and parsable manner.

Answer Accuracy. This component evaluates whether the final result a matches the ground-truth $\hat{a}$. The answer accuracy reward r_{acc} incentivizes the model to solve the task correctly, ensuring that the primary objective of MTIR is met. A binary 0/1 reward is assigned for wrong and correct answers to combine with the cost-effectiveness coefficient.

Cost-Effectiveness Coefficient. We further introduce a tool cost-effectiveness coefficient r_e to encourage efficient tool use. This coefficient measures the efficiency of tool usage in the MTIR process, conditioned on the model producing correct and well-formatted outputs. It takes values in $(0, 1)$ to scale the r_{acc} accordingly with the calculation given in Section 3.2.3. Thus, the overall reward R for a response y is defined as follows:

$$R = \begin{cases} r_{\text{e}} \cdot r_{\text{acc}} & \text{If Good Format} \\ -1 & \text{Otherwise} \end{cases}, \; r_{\text{acc}} = \text{Acc}(a, \hat{a})$$

3.2.3 Cost-effectiveness Coefficient

Trajectory Cost Estimation. For a given question q and its associated tool set $\mathcal{T}$, according to our definition in Section 2.2, an MTIR trajectory τ sampled by the policy model contains the usage counts $\{n_1, n_2, \ldots, n_{|\mathcal{T}|}\}$ for each tool in $\mathcal{T}$. We define the tool usage cost of the trajectory τ as:

$$\text{Cost}(\tau) = \sum_{j=1}^{|\mathcal{T}|} n_j \cdot c_j, \; c_j = c_{j,tool} + c_{j,LLM} + T_j,$$

where $c_{j,tool}$ denotes the estimated cost of invoking tool t_j itself, $c_{j,LLM}$ denotes the estimated cost of using the LLM for tool invocation, and T_j denotes the estimated consumed time. Following Zheng et al. (2024), we assume that all costs are measurable in a unified unit. Since $c_{j,\text{tool}}$ may vary in practice, unlike prior work (Xu et al., 2025) that assumes fixed cost settings, we assign a randomly sampled invocation cost $c_j \in [1, 10]$ to every tool $t_j \in \mathcal{T}$ for each query q. This formulation reflects realistic cost variability and avoids overfitting.

Coefficient Computation. For the current policy π_θ, we assume that there exists a minimal tool invocation cost c^* required to solve a query q. To reasonably estimate the cost-effectiveness of the current trajectory τ, inspired by Wang et al. (2025a), we compare its invocation cost $\text{Cost}(\tau)$ with the optimal cost c^* for solving the query, and derive the adjustment coefficient r_e as:

$$r_e = \begin{cases} \cos\left(\frac{\text{Cost}(\tau)}{2\text{Cost}(\tau)+s} * \pi\right) & \text{if } \text{c}^* = 0 \\ \sin\left(\frac{\text{Cost}(\tau)}{\text{Cost}(\tau)+\text{c}^*} * \pi\right) & \text{otherwise} \end{cases}$$

where s is a smoothing constant that controls the decay rate of the reward when the c^* is zero. The r_e tends toward 1 as $\text{Cost}(\tau)$ approaches the optimal cost c^*, or as $\text{Cost}(\tau)$ decreases when $\text{c}^* = 0$.

And the cost-effectiveness coefficient reduces to a simple call-count effectiveness coefficient when all tool costs are equal or only one tool is invoked, highlighting the broad applicability of our method.

Group Optimal Cost Estimation. The value of c^* varies under different cost settings. As the intrinsic reasoning ability influences tool dependency, the optimal cost also differs across models and training stages. To estimate c^* for the current policy on a given query q, we collect trajectories from the current sampled group that satisfy $r_{acc} = 1$, forming the set $\text{Cost}(\tau_i), \ldots, \text{Cost}(\tau_k)$. We then define $c_g^* = \min\{\text{Cost}(\tau_i), \ldots, \text{Cost}(\tau_k)\}$ as the estimated optimal cost c^*. This estimation avoids additional manual annotation or strong supervision and dynamically adapts to current reasoning ability.

3.3 Instantiations of EMTIR-GRPO

To demonstrate the generality of EMTIR-GRPO, we instantiate it in two representative scenarios.

3.3.1 MTIR-QA

The multi-tool integrated question answering (MTIR-QA) aims to enhance LLMs with external tools to address two key limitations of purely text-based question answering (Trivedi et al., 2022; Cobbe et al., 2021): knowledge hallucination and inaccurate computation (Paranjape et al., 2023; Dong et al., 2025a). We employ a Python interpreter and a search engine to achieve this.

During rollouts, LLMs conduct internal reasoning within the `<think>...</think>` and invoke one of the two tools per interaction by placing executable code in the `<python>...</python>` or search queries in `<search>...</search>`. While many questions in MTIR-QA can be answered without tools, excessive tool invocation leads to cognitive offloading (Wang et al., 2025a). In MTIR-QA, we apply EMTIR-GRPO not only to improve tool efficiency but also to reduce overreliance on tools, thereby mitigating cognitive offloading.

3.3.2 MTIR-TC

The multi-tool integrated task completion (MTIR-TC) leverages domain-specific APIs to fulfill user requests (Guo et al., 2024), in contrast to MTIR-QA, where tools primarily assist reasoning (Singh et al., 2025). For each query q, we incorporate into the system prompt the descriptions of all related tools in function calling schemas (OpenAI, 2023; Chen et al., 2025), guiding the model to select and invoke the appropriate tools for task completion.

During rollouts, the internal reasoning is also enclosed in `<think>...</think>`, while tool invocations with names and parameters are specified in `<toll_call> ... </toll_call>`, where multiple tools may be invoked in one interaction. In MTIR-TC, pure text-based reasoning is insufficient, and LLMs must rely on external tools to accomplish the assigned tasks, which often leads to excessive API calls and high invocation costs (Zheng et al., 2024). Therefore, in MTIR-TC, we apply EMTIR-GRPO to not only improve tool usage efficiency but also maximize tool invocation benefits to save budget.

4 Experiment

4.1 Set up

Training Setting We implement our EMTIR-GRPO on three representative frameworks: Tool-Star (Dong et al., 2025a) and ARPO (Dong et al., 2025b) as MTIR-QA cases, and ReCall (Chen et al., 2025) as the MTIR-TC case. We use Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct as the backbone models for training. To ensure fair comparison, we reuse their released training data (Wang et al., 2025a). Inspired by Bai et al. (2025), we progressively decrease the maximum interaction turns to encourage broader exploration of tool-use behaviors in the early stage. Other hyperparameters follow the original training settings, with further details provided in Appendix D.1 and D.2.

Benchmark For MTIR-QA, we evaluate two categories of benchmarks: (1) knowledge-intensive benchmarks: Bamboogle (Press et al., 2023), 2WikiMultihopQA (Ho et al., 2020), Musique (Trivedi et al., 2022), and BeerQA (Qi et al., 2021); (2) mathematical benchmarks: AIME25 (aim), OlympiadBench-math (He et al., 2024), Math (Hendrycks et al., 2021), and GSM8K (Cobbe et al., 2021). We follow the test set split of Tool-Star (Dong et al., 2025a) for MTIR-QA. For MTIR-TC, we adopt Stable-ToolBench (Guo et al., 2024), a stable test set derived from ToolBench. In addition, we employ the MirrorAPI-Cache model (Guo et al., 2025) to locally simulate the required RapidAPI services.

Baselines For MTIR-QA, we consider: (1) Single-tool RL: CIR (Bai et al., 2025), ReSearch (Chen et al., 2025), and Search-R1 (Jin et al., 2025); (2) Base Models with MTIR prompt: Qwen2.5 (Yang et al., 2024); (3) Multi-tool RL: AutoTIR (Wei et al., 2025), ARPO, and Tool-

Table 1: Overall results of various baselines on the MTIR-QA tasks. "(ours)" denotes the application of our proposed EMTIR-GRPO during the RL stage. The top two results are highlighted in **bold** and underlined.

Method	Knowledge-Intensive Domain								Mathematical Domain								AVG	
	Bamb.		2Wiki.		MuSiQ.		BeerQA		AIME25		MATH		GSM8K		Olym.			
	Acc	TP	Acc	TP	Acc	TP	Acc	TP	Acc	TP	Acc	TP	Acc	TP	Acc	TP	Acc	TP
Models based on Qwen2.5-Instruct-3B																		
MTIR-Prompt	24.0	8.7	20	5.5	14.5	4.2	32.5	12.1	0	0	51.4	16.0	37	8.1	25	**36.3**	25.6	11.4
Search-R1	48.8	13.0	46	11.6	23.5	5.0	41	13.9	0	0	72.4	12.4	76.8	13.2	28	6.5	42.1	9.4
CIR	26.2	9.9	23	8.1	8.5	3.6	20.5	8.4	13.3	1.4	78.4	13.4	77.2	14.6	33.8	4.5	35.2	8.0
Tool-Star	52.8	13.0	52.5	12.8	24	4.9	46.5	12.7	16.7	3.0	82	16.2	84.6	16.6	36.4	6.7	49.4	10.7
ARPO	53.6	13.2	50.5	12.1	27.5	5.7	45	11.3	16.7	3.3	79.2	15.5	84	16.3	37.6	7.0	49.3	10.6
Tool-Star (ours)	**57.6**	**15.0**	50	13.0	26	6.0	49.5	**17.3**	**20**	**5.5**	**82.2**	17.2	83.8	16.7	37.6	8.3	**50.8**	12.4
ARPO (ours)	52	13.7	**53.5**	**13.6**	**28**	**6.2**	**50.5**	16.3	13.3	2.5	81.6	**26.8**	**85.2**	**34.1**	**39**	10.0	50.4	**15.4**
Models based on Qwen2.5-Instruct-7B																		
MTIR-Prompt	48	12.0	40.5	8.4	24.5	5.5	45	13.7	3.3	0.4	77.2	10.9	79.2	10.2	37.6	6.9	44.4	8.5
Search-R1	50.4	11.5	44.5	6.8	22	3.8	44	12.4	6.7	1.1	75.8	9.3	69.6	7.5	33.4	3.9	43.3	7.4
ReSearch	56	7.8	49	6.8	30.5	3.8	48.5	8.7	3.3	0.3	71.8	5.3	65.8	3.8	32.2	2.6	44.6	4.9
CIR	35.2	12.6	23.5	14.7	9	8.4	21	17.9	**23.3**	4.1	86.4	20.5	92	22.1	42.6	8.7	41.6	13.6
AutoTIR	53.6	11.7	48.5	8.1	30	5.2	**51.5**	15.4	20	1.1	85.6	12.7	89.8	15.4	42.2	3.0	52.3	8.6
Tool-Star	**60.8**	9.8	56	6.1	21.5	2.5	46.5	7.6	20	3.3	84.8	14.4	80.6	15.2	45.4	7.0	52.0	8.2
ARPO	60	15.9	57.5	14.6	**31.5**	6.9	**51.5**	16.7	16.7	3.0	87	17.2	92	18.1	47.8	9.2	**55.5**	12.7
Tool-Star (ours)	57.6	18.0	57.5	22.0	30	8.9	49	**27.0**	**23.3**	**7.4**	87.2	30.0	92	24.1	46.4	15.5	55.4	19.1
ARPO (ours)	59.2	**31.5**	**59**	**32.8**	27.5	**14.2**	47	25.8	**23.3**	**7.4**	**87.8**	**82.8**	**92.6**	**111.6**	47.4	**20.1**	**55.5**	**40.8**

Star. For MTIR-TC, we consider RL-based approaches, including ReCall (Chen et al., 2025) and ToolRL (Qian et al., 2025a). All baselines are based on the same backbone model[1].

Evaluation Setting and Metrics We employ greedy decoding for evaluation and no limit on the maximum interaction turns to fully examine the tool-use capability. We adopt LLM-based judging (Qwen2.5-72B-Instruct) to verify answer correctness for QA tasks and use GPT-4o to assess Final Answer Completeness (FAC) for StableToolBench (Guo et al., 2025). Additionally, we evaluate tool productivity (TP) to measure the effectiveness of tool use (Wang et al., 2025a). Formally, TP on a given benchmark B is defined as $\mathrm{TP}_B = \frac{\mathrm{ACC}_B}{\frac{1}{|\{\tau \in B\}|}\sum_{\tau \in B}\mathrm{Cost}(\tau)}$, where ACC denotes the score on B, and $\frac{1}{|\{\tau \in B\}|}\sum_{\tau \in B}\mathrm{Cost}(\tau)$ is the average cost across all trajectories evaluated on B. The metric reflects how efficiently the model converts tool usage into correct answers. Consistent with training, each tool in the evaluation is randomly assigned a cost from the range $[1, 10]$.

4.2 Main Results

Results on MTIR-QA Table 1 shows the results for MTIR-QA. Several key insights can be drawn:

- **Efficiency improvements:** Both ARPO and Tool-Star trained with EMTIR-GRPO exhibit substantial improvements in tool-use efficiency compared to their original versions. On the 3B backbone, ARPO (ours) and Tool-Star (ours) achieve average TP gains of Δ+4.8 and Δ+1.7, respectively. On the 7B backbone, Tool-Star (ours) improves by Δ+10.9 and ARPO (ours) by Δ+28.1.

- **Accuracy preservation:** Despite incorporating explicit constraints on tool usage, our models maintain comparable performance to the original settings and even surpass them. For example, on the 7B backbone, Tool-Star (ours) achieves a higher average accuracy (55.4 vs. 52.0) despite no additional DPO training (Dong et al., 2025a).

- **Scalability with backbone size:** Under the same EMTIR-GRPO training, the 7B models yield substantially larger relative TP improvements than the 3B models—132.9% vs. 15.9% on Tool-Star and 221.3% vs. 45.3% on ARPO. This suggests that larger models, with richer internal knowledge and stronger reasoning capabilities, can benefit more from EMTIR-GRPO.

- **Overall superiority:** Our models consistently outperform competing baselines in both performance and efficiency. On the 7B backbone, ARPO (ours) attains the highest average accuracy and exceeds task-specific single-tool methods such as Search-R1 and CIR, as well as more sophisticated RL-based approaches like AutoTIR, in TP across all benchmarks.

[1]For baselines lacking a 3B Instruct version or 7B Instruct version, we replicate them using the exact parameter settings from their open-source code.

Table 2: Results on the MTIR-TC benchmark **StableToolBench**. Symbols follow the same definitions as in Table 1. The benchmark includes six test scenarios: I1-Inst, I1-Tool, I1-Cat, I2-Inst, I2-Cat, and I3-Inst (Guo et al., 2024).

Method	I1-Inst		I1-Tool		I1-Cat		I2-Inst		I2-Cat		I3-Inst		AVG	
	FAC	TP	FAC	TP	FAC	TP	FAC	TP	FAC	TP	FAC	TP	FAC	TP
Models based on Qwen2.5-Instruct-3B														
MTIR-Prompt	2.4	0.1	1.27	0.1	2.61	0.1	1.9	0.1	2.4	0.1	0	0	1.8	0.1
ToolRL	42.8	3.7	**46.8**	4.3	44.4	4.2	**36.8**	2.9	35.2	2.7	**32.8**	2.7	39.8	3.4
ReCall	26.6	1.5	34.4	2.0	41.2	2.4	27.4	1.5	27.4	1.6	27.9	1.7	30.8	1.8
ReCall (ours)	**42.9**	**4.4**	41.7	**4.4**	**49.7**	**5.6**	**36.8**	**3.3**	**37.9**	**3.5**	**32.8**	**3.5**	**40.3**	**4.1**
Models based on Qwen2.5-Instruct-7B														
MTIR-Prompt	7.4	0.3	7.6	0.3	11.8	0.5	2.8	0.1	4.8	0.2	4.9	0.2	6.5	0.3
ToolRL	49.4	3.0	**57.7**	3.6	**60.8**	3.7	48.1	2.9	54.0	3.3	41.0	2.3	51.8	3.1
ReCall	46.2	2.1	48.5	2.2	53.6	2.3	50.9	2.0	47.6	1.7	49.2	2.7	49.3	2.2
ReCall (ours)	**55.2**	**5.7**	47.4	**5.3**	54.9	**6.1**	**54.7**	**5.4**	**58.1**	**5.8**	**50.8**	**6.4**	**53.5**	**5.8**

Results on MTIR-TC Table 2 reports the results on MTIR-FC tasks. We observe that ReCall (ours) consistently achieves higher tool-use efficiency than the original ReCall on both backbones, with average TP gains of Δ+2.3 on 3B and Δ+3.6 on 7B. In addition, it also improves average FAC by Δ+9.5 on 3B and Δ+4.2 on 7B. Moreover, even compared with ToolRL, which adopts more complex process-level rewards, our approach delivers superior performance and efficiency. The results across both MTIR-FC and MTIR-QA highlight the strong generalization ability and superiority of EMTIR-GRPO across different MTIR scenarios.

Table 3: Ablation studies on the MTIR-QA tasks.

Method	Bamb.		AIME25		Olym.	
	Acc	TP	Acc	TP	Acc	TP
Tool-Star-3B (ours)	**57.6**	**15.0**	**20**	**5.5**	**37.6**	**8.3**
- w/o cost	54.4	14.8	13.3	**5.5**	34	7.3
- w/o c^* estimate	51.2	13.5	10	2.3	36.6	7.1
- w/o coefficient	55.2	13.6	10	1.7	35.2	6.6

Table 4: Ablation studies on the MTIR-TC tasks.

Method	I1-Cat		I2-Inst		I2-Cat	
	Acc	TP	Acc	TP	Acc	TP
ReCall-3B (ours)	**49.7**	**5.6**	**36.8**	**3.3**	**37.9**	**3.5**
- w/o cost	35.9	3.5	29.8	2.5	31.1	2.7
- w/o c^* estimate	40.5	4.3	35.8	3.1	31.5	3.0
- w/o coefficient	43.1	2.8	34.7	2.1	36.8	2.3

5 Analysis

5.1 Ablation Study

We perform ablation studies on Tool-Star-3B (ours) and ReCall-3B (ours) to examine the impact of different components in EMTIR-GRPO. Specifically, *w/o cost* replaces the proposed trajectory cost with the number of tool calls; *w/o* c^* *estimation* discards the group-based minimum cost c_g^* as the optimal reference; and *w/o coefficient* removes the cost-effectiveness coefficient by setting $r_e = 1$.

As shown in Table 3 and Table 4, the ablation results yield three key findings: (1) **Impact of cost-awareness:** Compared with efficiency optimization solely on the number of calls, incorporating cost-aware coefficients better guides the model toward more efficient MTIR. This effect is especially evident in MTIR-TC, where tools are diverse and mandatory; (2) **Role of group-based optimal cost estimation:** Using the group-based minimum cost as the optimal reference, instead of a fixed value, allows us to dynamically capture the tool-use capability of the current model and adaptively adjust the training reward; (3) **Effect of the coefficient:** The cost-effectiveness coefficient improves tool usage efficiency by explicitly rewarding cost-sensitive strategies while maintaining overall performance.

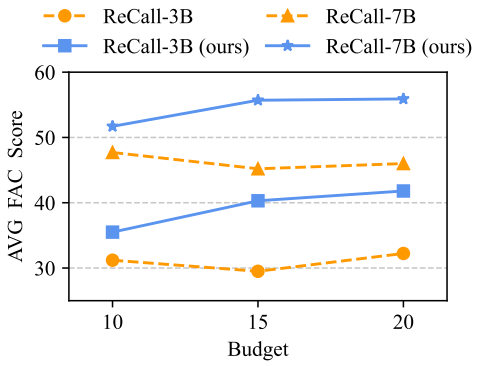

Figure 3: **Results of Budget-Constrained Inference.** We use the average FAC scores on StableToolBench to evaluate the performance under different budget.

5.2 Analyzing The Efficient Tool Use Behavior

Budget-Constrained Inference We design the budget-constrained MTIR experiment and conduct it in MTIR-FC, following Zheng et al. (2024). After the model consumes the predefined tool call budget, it can only output the result directly. This experiment evaluates the ability to maximize tool invocation benefits under a constrained budget. We set budget thresholds to 10, 15, and 20, using the randomly assigned tool costs described in Section 4.1. Additional details are in Appendix D.3.

As shown in Figure 3, models trained with EMTIR-GRPO consistently outperform the original ReCall models under different budget constraints, demonstrating that incorporating the efficiency coefficient improves cost-awareness and maximize budget utilization. Furthermore, while the performance of ReCall degrades with increasing budget, our models exhibit monotonic improvements with larger budgets, further demonstrating their ability to adaptively regulate tool usage.

Reason and Calling Token Efficiency In MTIR-QA, we further analyze the efficiency of each interaction step $\mathcal{A}_i$ by tracking the per-call token usage for tool invocations $\mathcal{T}_i$ and reasoning r_i. This provides a fine-grained view of token consumption in both reasoning and tool-calling processes.

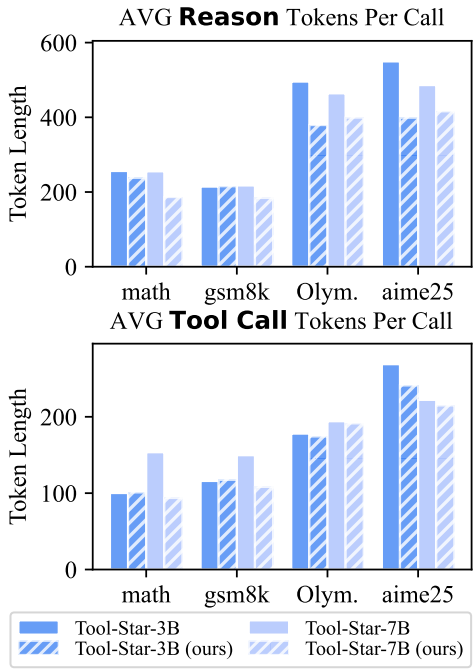

Figure 4: **Analysis of token efficiency.** Comparison of average token lengths per call on four math benchmarks.

As shown in Figure 4, our method yields fewer tokens for both reasoning and tool actions than the original Tool-Star. This demonstrates that EMTIR-GRPO not only reduces the overall cost of tool usage but also optimizes the efficiency of decision-making and execution for each tool invocation.

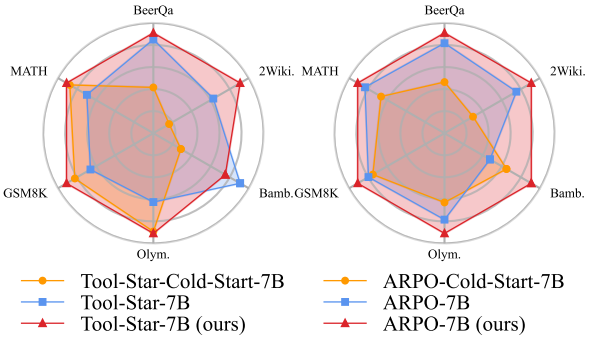

Figure 5: **Analysis of model intrinsic reasoning ability** without tool integration on ARPO and Tool-Star under Cold-Start, Origin RL, and EMTIR-GRPO.

5.3 Analyzing the Intrinsic Reasoning Ability

Previous studies have noted that tool learning may negatively affect the intrinsic reasoning ability (Wang et al., 2025a). To analyze the impact of the efficiency coefficient on the intrinsic reasoning capability, we conduct an evaluation in MTIR-QA tasks with tools disabled. Details in Appendix D.4.

As illustrated in Figure 5, under the no-tool integration condition—where the model must rely solely on its internal reasoning—our method seldom falls behind the cold-start trained Qwen2.5-7B-Instruct backbone and in most cases demonstrates stronger intrinsic reasoning ability than Tool-Star and ARPO. These results indicate that our method not only enhances cost-efficient tool utilization but also mitigates cognitive offloading, thereby enhancing the inherent reasoning ability of models.

6 Conclusion

In this paper, we propose EMTIR-GRPO, a reinforcement learning algorithm tailored to cost-aware multi-tool integrated reasoning (MTIR). Unlike prior approaches that overlook heterogeneous tool costs or rely solely on raw calling times of tool invocations, EMTIR-GRPO introduces a composite reward mechanism balancing answer correctness, format completeness, and tool efficiency. By incorporating the cost-effectiveness coefficient with group optimal cost estimation, our method adaptively guides models toward more efficient tool-use strategies. Extensive experiments across MTIR-QA and

MTIR-TC benchmarks demonstrate that EMTIR-GRPO substantially improves tool-use efficiency while preserving or even enhancing accuracy. Overall, EMTIR-GRPO provides a simple yet effective solution for developing robust, scalable, and cost-efficient multi-tool augmented LLM agents.

Limitations

Although EMTIR-GRPO significantly improves the efficiency of cost-aware tool invocation while maintaining accuracy on several representative MTIR-QA and MTIR-TC frameworks, our work still has limitations. First, we did not explore more sophisticated tool scenarios, including a wider variety of tools, more complex tools, or tools with overlapping functionality (e.g., calculators and code interpreters). Second, our cost modeling remains overly simplistic: we use random allocation to simulate the variability of tool call costs in the real world. A more reasonable cost estimation method is needed to achieve more realistic cost optimization, such as accounting for the number of tokens consumed by each tool call in LLMs. Moreover, we have not incorporated reasoning as a cognitive tool, and reasoning itself incurs substantial cost. Decoupling reasoning from tool invocations and integrating it into the tool set could enable a more comprehensive optimization of tool efficiency. Finally, due to computational constraints, we conducted experiments only on 3B and 7B models and did not validate larger models or additional frameworks. In future work, we plan to extend our algorithm to more complex agent tasks, explore more principled cost estimation, and integrate reasoning as a cognitive tool into the tool set.

Acknowledgments

The research in this article is supported by the National Science Foundation of China (U22B2059, 62276083), Key Research and Development Program of Heilongjiang Providence (2022ZX01A28).

References

AI-MO/aimo-validation-aime · Datasets at Hugging Face — huggingface.co. https://huggingface.co/datasets/AI-MO/aimo-validation-aime. [Accessed 16-02-2025].

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 12248–12267. Association for Computational Linguistics.

Fei Bai, Yingqian Min, Beichen Zhang, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen. 2025. Towards effective code-integrated reasoning. *CoRR*, abs/2505.24480.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. 2025. Research: Learning to reason with search for llms via reinforcement learning. *Preprint*, arXiv:2503.19470.

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*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *CoRR*.

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 81 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

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. *CoRR*, abs/2505.16410.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. 2025b. Agentic reinforced policy optimization. *CoRR*, abs/2507.19849.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. 2025. Deeprag: Thinking to retrieval step by step for large language models. *CoRR*, abs/2502.01142.

Zhicheng Guo, Sijie Cheng, Yuchen Niu, Hao Wang, Sicheng Zhou, Wenbing Huang, and Yang Liu. 2025. Stabletoolbench-mirrorapi: Modeling tool environments as mirrors of 7, 000+ real-world apis. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 5247–5270. Association for Computational Linguistics.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 11143–11156. Association for Computational Linguistics.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3828–3850. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

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, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. 2025. Reinforce++: An efficient rlhf algorithm with robustness to both prompt and reward models. *Preprint*, arXiv:2501.03262.

Abhinav Java, Srivathsan Koundinyan, Nagarajan Natarajan, and Amit Sharma. 2025. Frugalrag: Learning to retrieve and reason for multi-hop QA. *CoRR*, abs/2507.07634.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2024. Tülu 3: Pushing frontiers in open language model post-training. *CoRR*, abs/2411.15124.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. 2024. RLAIF vs. RLHF: scaling reinforcement learning from human feedback with AI feedback. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Chengpeng Li, Mingfeng Xue, Zhenru Zhang, Jiaxi Yang, Beichen Zhang, Xiang Wang, Bowen Yu, Binyuan Hui, Junyang Lin, and Dayiheng Liu. 2025a. START: self-taught reasoner with tools. *CoRR*, abs/2503.04625.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. 2025b. Torl: Scaling tool-integrated RL. *CoRR*, abs/2503.23383.

Bohan Lyu, Yadi Cao, Duncan Watson-Parris, Leon Bergen, Taylor Berg-Kirkpatrick, and Rose Yu. 2024. Adapting while learning: Grounding llms for scientific problems with intelligent tool usage adaptation. *CoRR*, abs/2411.00412.

Bohan Lyu, Yadi Cao, Duncan Watson-Parris, Leon Bergen, Taylor Berg-Kirkpatrick, and Rose Yu. 2025. Adapting while learning: Grounding LLMs for scientific problems with tool usage adaptation. In *Forty-second International Conference on Machine Learning*.

OpenAI. 2023. Function calling and other api updates.

Long Ouyang, Jeffrey 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 F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Bhargavi Paranjape, Scott M. Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Túlio Ribeiro. 2023. ART: automatic multi-step reasoning and tool-use for large language models. *CoRR*, abs/2303.09014.

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, Singapore, December 6-10, 2023*, pages 5687–5711. Association for Computational Linguistics.

Peng Qi, Haejun Lee, Tg Sido, and Christopher D. Manning. 2021. Answering open-domain questions of varying reasoning steps from text. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3599–3614. Association for Computational Linguistics.

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

Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiusi Chen, Avirup Sil, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025b. SMART: self-aware agent for tool overuse mitigation. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 4604–4621. Association for Computational Linguistics.

Hongjin Qian and Zheng Liu. 2025. Scent of knowledge: Optimizing search-enhanced reasoning with information foraging. *CoRR*, abs/2505.09316.

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, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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 Comput. Sci.*, 19(8):198343.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. KDD '20.

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. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

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

Yuanhao Shen, Xiaodan Zhu, and Lei Chen. 2024. SMARTCAL: an approach to self-aware tool-use evaluation and calibration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: EMNLP 2024 - Industry Track, Miami, Florida, USA, November 12-16, 2024*, pages 774–789. Association for Computational Linguistics.

Wenxuan Shi, Haochen Tan, Chuqiao Kuang, Xiaoguang Li, Xiaozhe Ren, Chen Zhang, Hanting Chen, Yasheng Wang, Lifeng Shang, Fisher Yu, and Yunhe Wang. 2025. Pangu deepdiver: Adaptive search intensity scaling via open-web reinforcement learning. *CoRR*, abs/2505.24332.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning. *CoRR*, abs/2505.01441.

Huatong Song, Jinhao Jiang, Wenqing Tian, Zhipeng Chen, Yuhuan Wu, Jiahao Zhao, Yingqian Min, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher++: Incentivizing the dynamic knowledge acquisition of llms via reinforcement learning. *CoRR*, abs/2505.17005.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Trans. Mach. Learn. Res.*, 2025.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, and 75 others. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *CoRR*, abs/2501.12599.

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. *Preprint*, arXiv:2504.14870.

Hongru Wang, Boyang Xue, Baohang Zhou, Tianhua Zhang, Cunxiang Wang, Huimin Wang, Guanhua Chen, and Kam-Fai Wong. 2025b. Self-dc: When to reason and when to act? self divide-and-conquer for compositional unknown questions. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 6510–6525. Association for Computational Linguistics.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.

Ningning Wang, Xavier Hu, Pai Liu, He Zhu, Yue Hou, Heyuan Huang, Shengyu Zhang, Jian Yang, Jiaheng Liu, Ge Zhang, Changwang Zhang, Jun Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. 2025c. Efficient agents: Building effective agents while reducing cost. *Preprint*, arXiv:2508.02694.

Yifan Wei, Xiaoyan Yu, Yixuan Weng, Tengfei Pan, Angsheng Li, and Li Du. 2025. Autotir: Autonomous tools integrated reasoning via reinforcement learning. *CoRR*, abs/2507.21836.

Hongshen Xu, Zihan Wang, Zichen Zhu, Lei Pan, Xingyu Chen, Lu Chen, and Kai Yu. 2025. Alignment for efficient tool calling of large language models. *CoRR*, abs/2503.06708.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yuanhang Zheng, Peng Li, Ming Yan, Ji Zhang, Fei Huang, and Yang Liu. 2024. Budget-constrained tool learning with planning. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 9039–9052. Association for Computational Linguistics.

Richard Zhuang*, Trung Vu*, Alex Dimakis, and Maheswaran Sathiamoorthy. 2025. Improving multi-turn tool use with reinforcement learning. https://www.bespokelabs.ai/blog/improving-multi-turn-tool-use-with-reinforcement-learning. Accessed: 2025-04-17.

A Related Works

Reinforcement Learning with Verifiable Rewards Reinforcement Learning with Verifiable Rewards (RLVR) is a novel training approach designed to enhance language models in tasks with verifiable outcomes, such as math and coding (Lambert et al., 2024; DeepSeek-AI et al., 2025; Team et al., 2025). Unlike previous Reinforcement Learning from Human Feedback (RLHF), which relies on a trained reward model (Ouyang et al., 2022; Lee et al., 2024), RLVR instead employs rule-based verification functions, such as exact answer matching, to generate reward signals. This approach simplifies the reward mechanism while maintaining strong alignment with the task's inherent correctness criteria. The approach has led to significant progress, exemplified by models such as DeepSeek-R1 (DeepSeek-AI et al., 2025) and Kim1.5 (Team et al., 2025). which achieve SOTA performance in reasoning tasks. Furthermore, the development of robust policy optimization algorithms, such as PPO (Schulman et al., 2017), Reinforce++ (Hu et al., 2025), RLOO (Ahmadian et al., 2024) and GRPO (DeepSeek-AI et al., 2025), has played a key role in RLVR's success.

B Benchmarks

B.1 MTIR-QA Benchmarks

- **Bamboogle** (Press et al., 2023) is a challenging multi-hop question-answering dataset designed to prevent models from using the reasoning shortcuts found in other benchmarks. It features hand-crafted questions requiring up to four reasoning hops, effectively testing a model's ability to perform genuine compositional reasoning across multiple facts.

- **2WikiMultihopQA** (Ho et al., 2020) is a large-scale dataset for multi-hop question answering that tests complex reasoning. It uniquely requires models to integrate information by combining structured knowledge from Wikidata with unstructured text from Wikipedia.

- **Musique** (Trivedi et al., 2022) is a challenging multi-hop question-answering dataset designed to evaluate deep reasoning. Its primary difficulty comes from dependency-based questions, where each inference step requires information retrieved from the previous one. This structure makes Musique a rigorous benchmark for testing a model's ability to perform true sequential logical reasoning rather than simple information retrieval.

- **BeerQA** (Qi et al., 2021) is a benchmark for evaluating open-domain question answering systems on questions with unknown reasoning complexity. It combines single-hop questions from SQuAD Open, two-hop questions from HotpotQA, and new questions requiring three

or more steps, all mapped to a unified Wikipedia corpus. This design prevents models from using stylistic shortcuts and creates a robust test for answering questions of varying and arbitrary reasoning depths.

- **AIME25** (aim) consists of 30 challenging math problems. It is directly composed of the real questions from the American Invitational Mathematics Examination (AIME I & II) newly released in February 2025. AIME25's knowledge areas are extremely wide. It deeply covers core mathematical sections such as algebra, geometry, number theory, and combinatorial mathematics. This characteristic enables the AIME25 dataset to effectively distinguish the mathematical reasoning abilities of different models.

- **OlympiadBench-math** (He et al., 2024) is a curated subset of the comprehensive OlympiadBench benchmark, specifically designed for a rigorous evaluation of advanced mathematical reasoning. The dataset comprises open-ended mathematics problems sourced from elite international competitions and deliberately excludes the collection's proof-based problems. This focus on open-ended questions ensures that a model's performance can be assessed through objective and straightforward verification of its final answers, making it a formidable test of deep reasoning and creative problem-solving skills.

- **MATH** (Hendrycks et al., 2021) is a significant academic dataset. It is designed to test and enhance models' mathematical reasoning skills. It covers a wide range of mathematical fields, including abstract algebra, calculus, and discrete mathematics. The dataset divides training data into three levels, which helps effectively evaluate model performance at different stages.

- **GSM8K** (Cobbe et al., 2021) is a widely adopted benchmark designed to evaluate the multi-step mathematical reasoning of large language models. Introduced by OpenAI, the dataset consists of high-quality, grade-school level math word problems that each require 2 to 8 sequential steps to solve. The problems are grounded in basic arithmetic operations, making GSM8K an essential tool for assessing a model's foundational capabilities in logical deduction and numerical computation.

B.2 MTIR-TC Benchmarks

- **StableToolBench** (Guo et al., 2024) is a large-scale benchmark designed to address the critical issues of instability and irreproducibility in evaluating the tool-learning capabilities of Large Language Models. Evolving from its predecessor, ToolBench, it introduces a novel virtual API server that combines a comprehensive caching system with LLM-powered API simulators to guarantee consistent and available tool responses, mitigating failures from real-world online APIs. These components make StableToolBench an essential and reliable resource for assessing an LLM's ability to utilize diverse, real-world tools in a consistent and comparable manner over time.

C Baselines

C.1 MTIR-QA Baselines

- **CIR** (Bai et al., 2025) is a post-training framework designed to teach models to effectively and efficiently integrate a Code Interpreter for complex mathematical reasoning. Unlike methods that simply prompt for tool use, CIR strategically inserts instructive hints at key points in the reasoning process. This approach guides the model on when to offload complex calculations to the interpreter, mitigating inefficiencies like delayed computation and redundant verification. Ultimately, this framework enhances a model's ability to seamlessly combine natural language reasoning with precise code execution, improving both accuracy and token efficiency.

- **Search-R1** (Jin et al., 2025) is a method that utilizes reinforcement learning to train a model from scratch, enabling it to actively call a search engine during its reasoning process. The model learns to dynamically generate necessary search queries, retrieve relevant information, and then integrate these findings into its reasoning chain to construct a solution. In our experiments, we adhere to the original prompting strategy to ensure a faithful evaluation of its capabilities.

- **Tool-Star** (Dong et al., 2025a) is a reinforcement learning framework that utilizes a two-stage training paradigm, consisting of a cold-start fine-tuning phase and a multi-tool self-critic RL stage. This framework allows the model to autonomously invoke and coordinate multiple external tools, such as a search engine and a

code interpreter, during a single reasoning process. This setup expands the model's capabilities beyond single-tool settings, enabling it to solve complex problems that require both dynamic information retrieval and precise computation.

- **ARPO** (Dong et al., 2025b) is an agentic reinforcement learning algorithm designed to address the limitations of traditional trajectory-level methods in training multi-turn, tool-using agents. The algorithm features an entropy-based adaptive rollout mechanism, which allows the model to dynamically intensify exploration by branching its reasoning paths at high-uncertainty steps that typically follow tool interactions. This approach enhances the model's ability to learn effective stepwise tool-use strategies.

- **AutoTIR** (Wei et al., 2025) AutoTIR is a reinforcement learning framework designed to address the challenge of rigid, predefined tool-use patterns that can degrade the model's core language competence. The framework uses a hybrid reward mechanism that allows the model to autonomously decide whether to invoke a tool and which specific tool to use based on the task context. It enhances the model's ability to learn a flexible and generalizable tool-use policy, improving overall performance across knowledge-intensive, mathematical, and general language modeling domains.

C.2 MTIR-TC Baselines

- **ReCall** (Chen et al., 2025) is a reinforcement learning framework designed to train LLMs to effectively reason with tool calls, evolving from its predecessor, ReSearch, to handle more generalized, multi-tool scenarios. The framework addresses the scarcity of varied and verifiable data for complex, multi-step tool-use tasks, enabling the model to learn planning and decision-making in a wide range of environments. Ultimately, this approach enhances the model's ability to generalize across arbitrary tools while maintaining strong performance in specialized domains.

- **ToolRL** (Qian et al., 2025a) is an approach that applies reinforcement learning to enhance general-purpose, multi-tool use in LLMs by focusing on a principled and fine-grained reward design. This method utilizes a detailed reward structure that provides feedback on multiple components, including the accuracy of the output format, tool names, parameter names, and parameter values. This approach enhances the model's ability to learn robust and generalizable tool-use strategies, significantly improving its performance in complex and unfamiliar scenarios.

D Implementation Details

D.1 MTIR-QA Training

Details of Tools (1) **Python Interpreter:** Code snippets generated by the language model are executed in a sandboxed environment, returning either the execution result or error messages based on correctness. Our Python interpreter implementation follows the design of ToRA (Gou et al., 2024), ensuring secure and accurate execution; (2) **Search Engine:** Following Jin et al. (2025), we use Wikipedia[2] as the retrieval corpus and E5 (Wang et al., 2024) as the retriever during training to reduce training cost. And following Dong et al. (2025a) and Shi et al. (2025), we use LangSearch[3] as the web search engine for mathematical benchmarks during evaluation.

Supervised Fine-Tuning Following ARPO (Dong et al., 2025b) and Tool-Star (Dong et al., 2025a), we conduct supervised fine-tuning of the Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct models using the Llama Factory framework with a learning rate of 7×10^{-6}. We employ DeepSpeed ZeRO-3 (Rasley et al., 2020) and FlashAttention2 (Dao, 2023) for optimization. The models are trained for 3 epochs with a batch size of 128, a weight decay of 0.1, and BF16 mixed precision, with a maximum input length of 4096 tokens. And we reuse the cold-start datasets from Dong et al. (2025b) and Dong et al. (2025a) as training data for ARPO (ours) and Tool-Star (our), respectively.

Reinforcement Learning In the reinforcement learning phase, we build upon the baseline implementations of Tool-Star and ARPO, with modifications to the reward function to realize our EMTIR-GRPO algorithm. The learning rate for the policy LLM is set to 1×10^{-6}, with 8 rollouts per sample for Tool-Star and 16 rollouts for ARPO. The total training batch size is 128, and the mini-batch size is 16. The maximum output length is set to 4096 to ensure completeness of exploration. Following CIR (Bai et al., 2025), the maximum number of tool

[2] https://archive.org/details/enwiki-2018122
[3] https://langsearch.com/

interaction rounds during training progressively decreases from 4 to 2. To stabilize training, we set the KL divergence coefficient in GRPO to 0. The reinforcement learning stage runs for 2 epochs, using the same training data as ARPO and Tool-Star. All experiments are conducted on 8 NVIDIA A800 GPUs. Notably, for Tool-Star, we omit the self-critic DPO training phase. The prompt template for training and main results is illustrated in Figure 8.

Details of Cost-Aware Reward Implementation We apply a format check based on the ordering of these XML-style tags described in Section 3.3.1. To avoid conflicts between the F1 reward and the cost-effectiveness coefficient (Dong et al., 2025a), we adopt exact match evaluation for knowledge-intensive questions and math verification for computational questions, both yielding binary 0/1 rewards. During the training, the model may solve problems purely through text-based reasoning without invoking tools, which triggers the smoothing mechanism described in Section 3.2.2. To further amplify the advantage differences among correct trajectories with varying costs within the same group, we define the smooth constant s as the product of the maximum number of interaction turns and the cost of a search calling for knowledge-intensive questions and a Python calling for mathematical questions.

D.2 MTIR-TC Training

Details of Tools (1) **Synthetic Environment for Training:** Following ReCall (Chen et al., 2025), we construct a synthetic environment in Python, which typically consists of a synthetic database (e.g., a Python dictionary) together with a set of executable functions[4]; (2) **Simulation Environment for Evaluation:** For stable evaluation, we employ the MirrorAPI-Cache (Guo et al., 2025) to simulate the RapidAPI[5] environment.

Reinforcement Learning Similar to ReCall (Chen et al., 2025), we adopt a *zero RL training* paradigm without cold-start initialization. Our method builds upon the baseline implementations of ReCall, with necessary modifications to realize the proposed EMTIR-GRPO algorithm. Since ReCall does not provide specific hyperparameter configurations, we follow ARPO to set the corresponding values. The learning rate for the policy LLM is set to 1×10^{-6}, with 8 rollouts per sample. The total training batch size is 128, and the mini-batch size is 16. The maximum output length is set to 4096 tokens. During training, the maximum number of tool interaction rounds is progressively reduced from 5 to 3. The KL-divergence coefficient in GRPO is set to 1×10^{-3}, following Zhuang* et al. (2025). The reinforcement learning stage runs for 1 epoch, using the same training data as ReCall. All experiments are conducted on 8 NVIDIA A800 GPUs. The prompt template for training and main results is illustrated in Figure 9.

Details of Cost-Aware Reward Implementation We also apply a format check based on the ordering of these XML-style tags described in Section 3.3.2. Since the ground-truth answers for task completion are often difficult to evaluate by exact match, we adopt partially exact match evaluation to obtain binary rewards to avoid the problem of sparse reward. Moreover, we define the corresponding smooth constant s as the product of the maximum number of interaction turns and the maximum cost among those tools in the system prompt.

D.3 Budget-Constrained Inference

Under the MTIR-TC task setting, we specify the budget and the cost of each tool directly in the prompt, and return the remaining budget in the observation after every tool invocation. When the cost of a tool call exceeds the remaining budget, the execution is terminated and directly returns the string "The cost of this API exceeds your remaining budget of [left_budget]." as the execution result. The corresponding system prompt is shown in Figure 10. This reasoning setup allows us to analyze the model's ability to maximize task performance under constrained budgets.

D.4 No-Tool Integrated Inference

Under the MTIR-QA task setting, we explicitly prohibit the model from using tools through prompt instructions. However, since RL-trained models may exhibit degraded instruction-following ability, tool calls can still occur. To address this, we append a "no tool available" response directly, forcing the model to rely on its own reasoning capabilities. The corresponding system prompt is illustrated in Figure 11.

[4] https://attractive-almandine-935.notion.site/ReCall
[5] https://rapidapi.com/

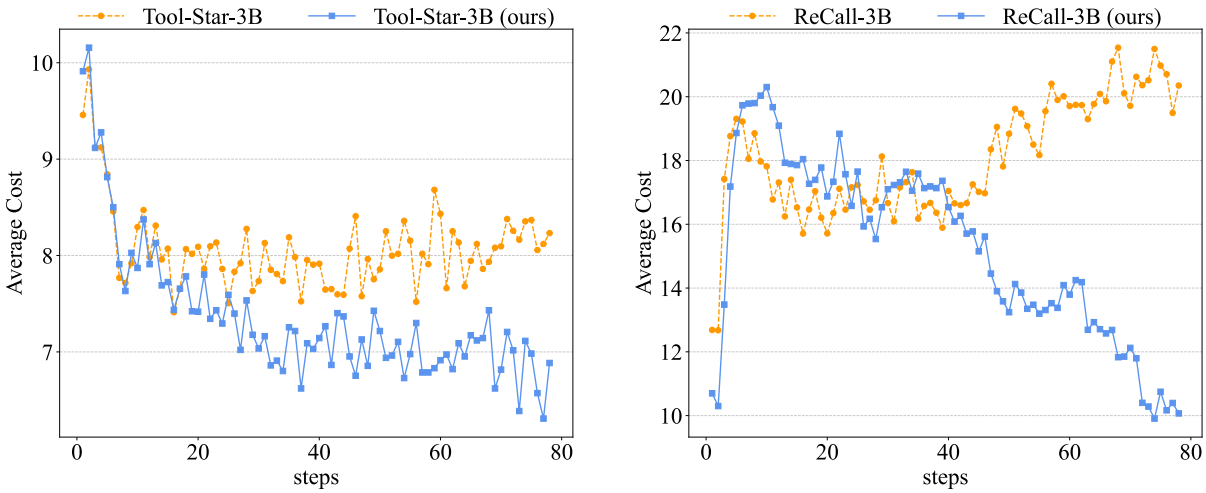

Figure 6: **RL Training Curves.** Comparison of average cost during training.

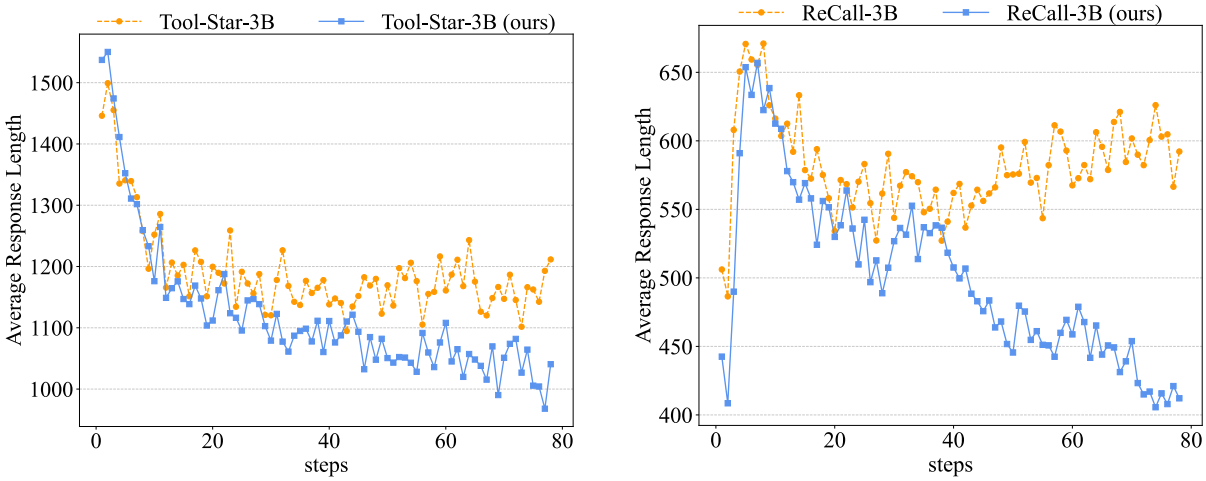

Figure 7: **RL Training Curves.** Comparison of average response length during training.

E Training Curves

As illustrated in Figure 6 and Figure 7, our approach demonstrates not only more pronounced but also more stable reductions in both tool call cost and response length over the course of training, when compared to Tool-Star and ReCall. This consistent downward trend indicates that our method is able to progressively optimize tool usage efficiency while simultaneously shortening responses, ultimately leading to a more cost-effective and resource-efficient training process.

Instruction for MTIR-QA

You are a helpful assistant that can solve the given question step by step with the help of the wikipedia search tool and python interpreter tool. Given a question, you need to first think about the reasoning process in the mind and then provide the answer. During thinking, you can invoke the wikipedia search tool to search and python interpreter tool to calculate the math problem for fact information about specific topics if needed. You should make every tool call count and obtain useful results, considering that each Python interpreter call costs [python_cost] and each search call costs [search_cost]. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags respectively, and the search query and result are enclosed within <search> </search> and <result> </result> tags respectively. After receiving the search or python result, you should continue your reasoning process begin with <think>. For example, <think> This is the reasoning process. </think> <search> search query here </search> <result> search result here </result> <think> This is the reasoning process. </think> <python> python code here </python> <result> python interpreter result here </result> <think> This is the reasoning process. </think> <answer> The final answer is

$$\boxed{answer\ here}$$

</answer>. In the last part of the answer, the final exact answer is enclosed within \boxed{} with latex format.

Figure 8: The System Prompt for MTIR-QA training and main evaluation.

Instruction for MTIR-TC

In this environment you have access to a set of tools you can use to assist with the user query. You may perform multiple rounds of function calls. In each round, you can call one or more functions.
Here are available functions in JSONSchema format: ‘ ‘ ‘json {func_schemas} ‘ ‘ ‘
In your response, you need to first think about the reasoning process in the mind and then conduct function calling to get the information or perform the actions if needed. The reasoning process and function calling are enclosed within <think> </think> and <tool_call> </tool_call> tags. The results of the function calls will be given back to you after execution, and you can continue to call functions until you get the final answer for the user's question. You should make every function call count and obtain useful results. The costs of the function calls are as follows:
{cost_sentence}
Finally, if you have got the answer, enclose it within
boxed with latex format and do not continue to call functions, i.e., <think> Based on the response from the function call, I get the weather information. </think> The weather in Beijing on 2025-04-01 is

$$\boxed{20C}.$$

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>

Figure 9: The System Prompt for MTIR-TC training and main evaluation.

Instruction for Budget-Constrained Inference

In this environment you have access to a set of tools you can use to assist with the user query. You may perform multiple rounds of function calls. In each round, you can call one or more functions.
Here are available functions in JSONSchema format: ‘ ‘ ‘json {func_schemas} ‘ ‘ ‘
In your response, you need to first think about the reasoning process in the mind and then conduct function calling to get the information or perform the actions if needed. The reasoning process and function calling are enclosed within <think> </think> and <tool_call> </tool_call> tags. The results of the function calls will be given back to you after execution, and you can continue to call functions until you get the final answer for the user's question. You should make every function call count and obtain useful results. You have a total budget of [total_budget]. The costs of the function calls are as follows:
{cost_sentence}
Finally, if you have got the answer, enclose it within
boxed with latex format and do not continue to call functions, i.e., <think> Based on the response from the function call, I get the weather information. </think> The weather in Beijing on 2025-04-01 is

$$\boxed{20C}.$$

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>

Figure 10: The System Prompt for Budget-Constrained evaluation.

Instruction for MTIR-TC

You are a helpful assistant that can solve the given question step by step based on your own knowledge without using tools. Given a question, you need to first think about the reasoning process in the mind and then provide the answer. During thinking, You must not invoke the Wikipedia search tool for factual information and use Python code execution for calculation. The reasoning process is enclosed within <think> and </think>, and the answer is enclosed within <answer> and </answer> tags. For example, <think> This is the reasoning process. </think><answer> The final answer is

$$\boxed{answer\ here}$$

</answer>. In the last part of the answer, the final exact answer is enclosed within \boxed{} with latex format.

Figure 11: The System Prompt for No-Tool evaluation.