EFFICIENT AGENTIC REINFORCEMENT LEARNING WITH TOOL-AWARE CREDIT ASSIGNMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic reinforcement learning (RL) equips large language models (LLMs) with tool-use capabilities that substantially improve reasoning on complex tasks. However, incorporating external tool feedback perturbs the model's predicted token distribution, introducing distribution shifts that destabilize policy optimization. To address this issue, we propose **TACT**, an efficient framework that stabilizes policy learning through tool-aware credit assignment. Specifically, to quantify a tool's impact on the reasoning process, **TACT** performs counterfactual reasoning by masking tool observation tokens. By computing the information gain between the original trajectory and its counterfactual counterpart, we establish a rigorous token-level metric for credit allocation. Guided by this causal perspective, **TACT** hierarchically refines the optimization process. At the trajectory level, executions with failed outcomes are filtered out to prevent erroneous feedback from exacerbating distribution shifts. At the token level for remaining trajectories, the information gain serves as the criterion for shaping token-level advantages, allocating higher credit strictly to tokens where tool interaction meaningfully advances the final outcome. By aligning policy updates with validated tool contributions, **TACT** mitigates the error interference from failed trajectories while reinforcing the effect of tokens with higher information gain. Extensive experiments on 9 benchmarks spanning mathematical reasoning, code generation, and function calling, across 3 model scales, demonstrate the superiority of **TACT** over existing methods.

1 INTRODUCTION

Agentic reinforcement learning (RL) has emerged as a powerful post-training paradigm for enhancing the multi-step reasoning capabilities of large language models (LLMs) (Zhang et al., 2026; Singh et al., 2025; Jiang et al., 2025; Li et al., 2025b; Lin et al., 2025; Zheng et al., 2026; Cheng et al., 2025). By enabling models to interact with external modules like code interpreters and search engines, these policies achieve notable performance gains (Dong et al., 2025a; Xu et al., 2026; Dong et al., 2026; Feng et al., 2025a; Bai et al., 2025; Wang et al., 2025a). However, integrating external execution directly into the training loop introduces a critical bottleneck: policy optimization often becomes highly unstable (Mai et al., 2025; Wang et al., 2025b; Yu et al., 2025b).

This instability originates from the structural disruption caused by tool interactions. When external execution outcomes are injected into the context, they abruptly alter the underlying token patterns and trajectory dynamics (Xue et al., 2026; Mai et al., 2025; Qian et al., 2025). If a tool returns noisy, malformed, or failed feedback, the resulting distribution shift heavily distorts the optimization landscape, leading to brittle policy updates (Shang et al., 2025; Bai et al., 2025). As illustrated in Figure 1, when a tool call fails, the tool-integrated reasoning (TIR) model is exposed to noisy external observations that severely disrupt the subsequent reasoning process, leading to failed responses. Moreover, the training curves show that as these tool-induced distribution shifts accumulate during training, policy optimization gradually collapses in the later stages. Thus, the core limitation lies in insulating the policy from the distribution shifts of invalid tool interactions.

Existing works attempt to address this challenge through various trajectory-level and step-level interventions. One prominent line of research focuses on rollout data curation and reward shaping, such as filtering trajectories with malformed formats or assigning global rewards based purely on final task success (Xue et al., 2026; Yu et al., 2025b; Wang et al., 2025b; Qian et al., 2025). Another

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

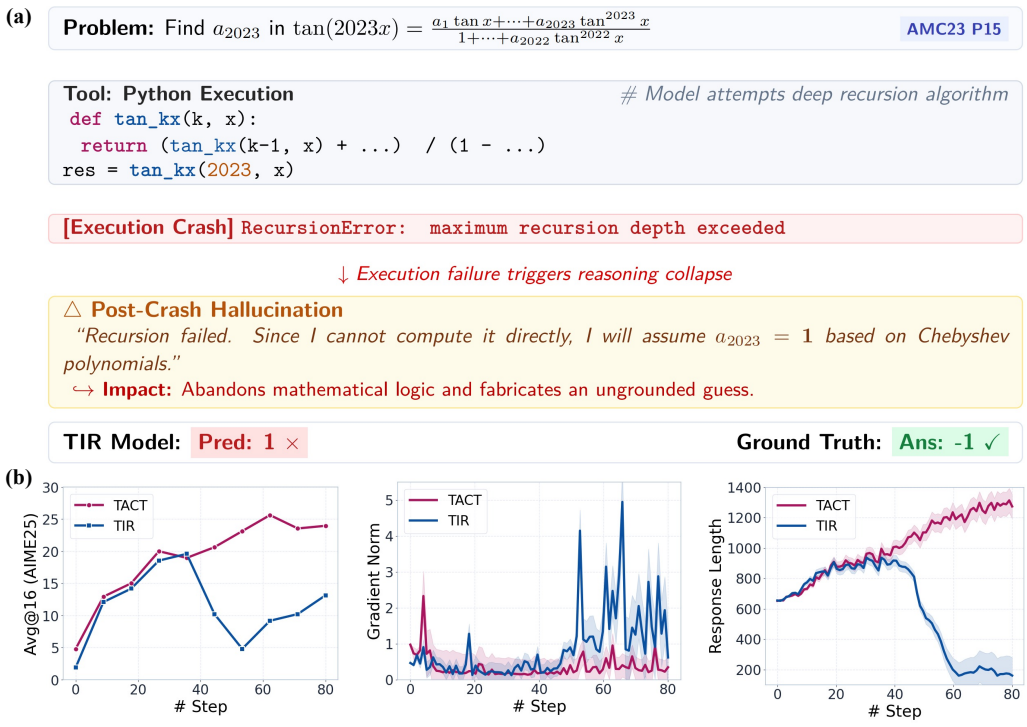


Figure 1: Example of TIR reasoning and comparison with **TACT**. (a) A failed tool call can severely mislead subsequent reasoning, and (b) the distribution shift accumulates over training, eventually collapsing the TIR model. **TACT** remains stable under the same environment.

line explores advanced search and sampling strategies to enhance overall reasoning paths (Bai et al., 2025; Dong et al., 2025a; 2026; Xu et al., 2026). However, these methods do not account for specific tool execution outcomes during optimization. These approaches treat entire trajectories uniformly, regardless of whether a tool returned an error, yielded nothing, or provided critical information. *Without a mechanism to assign credit based on these distinct execution contexts, the tool's actual contribution fails to effectively influence the policy updates.*

To address this limitation, we propose **TACT**, an efficient framework for stable policy learning with tool-aware credit assignment. The key insight is to directly tie policy updates to the verified utility of tool executions, rather than relying solely on uniform trajectory-level rewards. To break the uniform credit assignment bottleneck, **TACT** isolates the tool's contribution on the reasoning process by asking a pivotal question: how would the model's reasoning trajectory evolve in the absence of tool intervention? Specifically, **TACT** establishes a counterfactual reference trajectory by masking the tool observation tokens, the external execution results injected into the context. By evaluating the model's next-token predictive distributions with and without the tool feedback, we calculate the information gain across the reasoning steps. This metric quantifies how much the tool reduces the reasoning uncertainty, providing a rigorous token-level measure of the tool contribution without requiring auxiliary reward models.

Guided by this causal perspective, **TACT** executes tool-aware credit assignment through a hierarchical optimization scheme. *At the trajectory level*, it filters out rollouts with failed tool execution or rollout groups with uniformly incorrect answers prior to backpropagation, preventing invalid feedback from inducing harmful distribution shifts. *At the token level for remaining rollouts*, **TACT** leverages the computed information gain to reshape advantages. Rather than applying uniform updates, it assigns higher credit strictly to reasoning tokens driven by tool feedback, while unaffected tokens receive baseline updates. By aligning policy updates with the actual utility of tool feedback, **TACT** resolves the optimization instability inherent in agentic RL. **TACT** preserves exploratory behavior, encouraging the model to confidently invoke external tools, while insulating the policy from the disruptive distribution shifts triggered by uninformative or erroneous tool responses. Consequently, **TACT** improves both the efficiency and stability of policy optimization in tool-augmented reasoning tasks. The contributions of this work are summarized as follows:

- We propose **TACT**, an efficient agentic RL framework that stabilizes policy optimization through tool-aware credit assignment. By employing counterfactual reasoning to mask tool execution results, **TACT** calculates the information gain to quantify the contribution of tool interactions.
- **TACT** applies a hierarchical credit assignment mechanism guided by the information gain. At the trajectory level, **TACT** filters out failed trajectories to prevent distribution shifts. At the token level, it shapes advantages to assign higher credit to reasoning steps advanced by tool feedback, resolving the bottleneck of uniform reward allocation.
- We conduct extensive experiments on 7 mathematical reasoning benchmarks across 3 model scales, demonstrating the superiority of **TACT** over existing agentic RL methods. We further evaluate on LiveCodeBench and BFCL to verify its generalization to code generation and tool-use capability.

2 Related Work

Agentic RL has significantly expanded the capabilities of LLMs, evolving them from passive text generators to active agents capable of multi-turn environmental interaction and tool utilization (Shinn et al., 2023; Mai et al., 2025; Dong et al., 2025b; Chen et al., 2026; Zong et al., 2026). Recent research has increasingly shifted from heuristic prompt engineering towards end-to-end policy optimization driven by external environmental feedback (Li et al., 2025a; Luo et al., 2025). On the system and architectural front, frameworks such as VERLTOOL (Jiang et al., 2025), ARTIST (Singh et al., 2025), TORL (Li et al., 2025b), and ReTool (Feng et al., 2025a) have co-designed distributed RL infrastructures to seamlessly integrate code execution modules into the training loop, thereby enhancing sample efficiency. Concurrently, on the algorithmic front, researchers have sought to stabilize tool-integrated training. Approaches like RAGEN (Wang et al., 2025b) and ToolRL (Qian et al., 2025) explore targeted reward shaping, while SimpleTIR (Xue et al., 2026) mitigates gradient instability by filtering out void execution turns. Furthermore, methods like GRPO-RoC (Shang et al., 2025) address environmental noise through advanced resampling-on-correct strategies. GiGPO (Feng et al., 2025b) introduces a two-level structure for estimating relative advantage.

Despite these advancements, existing methods fail to explicitly account for tool execution outcomes. Data curation approaches often filter based on surface formats rather than actual tool success (Yu et al., 2025b), while step-level strategies rely on model-internal signals, ignoring how external feedback reshapes subsequent reasoning (Bai et al., 2025; Dong et al., 2025a). Consequently, standard optimization struggles with uniform credit assignment. To address this gap, **TACT** introduces a tool-aware credit assignment framework based on information gain. By quantifying the impact of tool feedback via counterfactual reasoning, we perform trajectory-level filtering of failed executions and token-level advantage shaping, directly aligning policy updates with tool contributions.

3 Preliminaries

GRPO. **TACT** is built upon group relative policy optimization (GRPO) as its foundational optimization algorithm. GRPO stabilizes policy updates by normalizing advantage estimates over groups of sampled responses, eliminating the need for a separate value network. For each input query $x \sim \mathcal{D}$, the rollout policy $\pi_{\theta_{\text{old}}}$ generates a group of responses $\{o^i\}_{i=1}^G$ with corresponding rewards $\{r^i\}_{i=1}^G$. The advantage for the i-th response is estimated as:

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

The learning objective of GRPO is defined as:

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\theta) =& \mathbb{E}_{x\sim\mathcal{D},\,\{o^i\}_{i=1}^G\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o^i|}\sum_{t=1}^{|o^i|}\Big(\min\big(\rho_t^i(\theta)\hat{A}^i,\ \text{clip}(\rho_t^i(\theta),\\ & 1-\epsilon,\ 1+\epsilon)\hat{A}^i\big) - \beta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})\Big)\Bigg],\end{aligned} \tag{2}$$

where $\rho_t^i(\theta) = \frac{\pi_\theta(o_t^i|x,o_{<t}^i)}{\pi_{\theta_{\text{old}}}(o_t^i|x,o_{<t}^i)}$ is the probability ratio, ϵ is the clipping hyperparameter, and β controls the strength of the KL divergence penalty against the reference model π_{ref}.

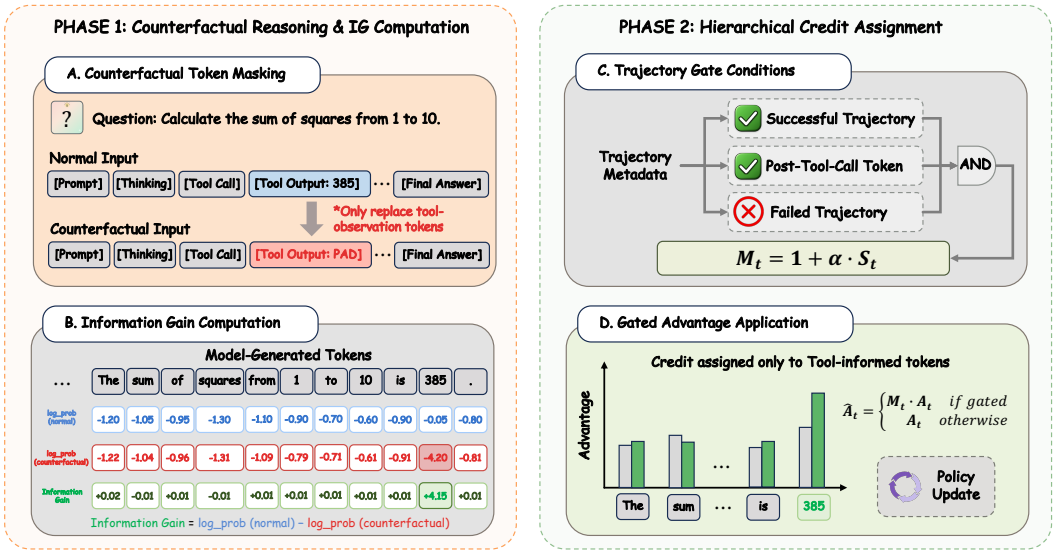


Figure 2: Overall framework of **TACT** with two learning phases. Phase 1: counterfactual reasoning and IG computation. Phase 2: hierarchical credit assignment guided by IG.

Agentic RL. Unlike standard LLM optimization over pure text generation, agentic RL interleaves language generation with tool interactions. In this work, the model can invoke a *code interpreter* to execute programs and obtain feedback. Let $x \sim \mathcal{D}$ denote an input query and π_θ denote the policy model. The overall response o consists of a reasoning trajectory $\mathcal{R} = \{s_t\}_{t=1}^{|\mathcal{R}|}$ and a final answer $y = \{y_t\}_{t=1}^{|y|}$. The trajectory $\mathcal{R}$ contains both model-generated tokens and environment-returned tokens. Let $\mathcal{I}_m \subseteq \{1, \ldots, |\mathcal{R}|\}$ denote the indices of tokens generated by the policy model. The generation probability of the model's outputs within the overall response $o = (\mathcal{R}, y)$ is strictly formulated as:

$$P_\theta(o \mid x) = \underbrace{\prod_{t \in \mathcal{I}_m} P_\theta(s_t \mid s_{<t}, x)}_{\text{Agentic Reasoning}} \cdot \underbrace{\prod_{t=1}^{|y|} P_\theta(y_t \mid y_{<t}, \mathcal{R}, x)}_{\text{Answer Generation}}. \tag{3}$$

Following GRPO-based optimization, the policy is trained to maximize the expected reward of the overall trajectory and final answer.

4 Approach

As illustrated in Figure 2, **TACT** dynamically shifts optimization from uniform trajectory rewards to token-level, tool-aware advantage shaping through a two-phase learning process. First, by establishing a counterfactual reasoning process, we compute the information gain to meticulously isolate the intrinsic contribution of tool feedback from the model's predictive dynamics. Second, guided by this metric, we execute a hierarchical advantage shaping mechanism that synergistically filters out failed trajectories and selectively amplifies the learning signals for high-utility reasoning steps.

4.1 Counterfactual Reasoning

To accurately assign credit in tool-integrated reasoning, it is essential to isolate the exact contribution of external tool observations from the model's intrinsic generation capabilities. We achieve this by introducing counterfactual reasoning, which answers a pivotal question: how would the model's predictive distribution change if the external tool feedback were entirely absent? By establishing this hypothetical baseline, we can precisely quantify the utility of external knowledge, thereby resolving the bottleneck of conventional optimization that treats all tokens uniformly.

Recall from Section 3 that given an input query x, the model generates an overall response sequence $o = \{o_t\}_{t=1}^{|o|}$. This sequence interleaves the model's intrinsically generated tokens with the environ-

ment's returned tool execution results within the reasoning trajectory $\mathcal{R}$. Let $\mathcal{I}_{\text{obs}} \subset \{1, \ldots, |o|\}$ denote the set of positional indices corresponding specifically to the tokens of these returned execution results. We construct a *counterfactual response sequence* o^{cf} by masking out this external feedback. The t-th token of the counterfactual sequence, denoted as o_t^{cf}, is formally defined as:

$$o_t^{\text{cf}} = \begin{cases} \texttt{[MASK]} & \text{if } t \in \mathcal{I}_{\text{obs}}, \\ o_t & \text{otherwise.} \end{cases} \tag{4}$$

By conditioning on the counterfactual prefix $o_{<t}^{\text{cf}}$ instead of the original prefix $o_{<t}$ during rollout, we compute the counterfactual predictive distribution $\log \pi_\theta(o_t \mid o_{<t}^{\text{cf}}, x)$ for each generated token. Note that this masking only replaces token identities at $t \in \mathcal{I}_{\text{obs}}$, leaving the sequence length and token positions unchanged, only the predicted probabilities shift, so no trajectory length mismatch arises when computing the information gain.

4.2 Information Gain Computation

Building upon the counterfactual reasoning, we compute the token-level information gain to quantitatively evaluate the extent to which each token's generation relies on the tool observation. Rather than relying on sparse environmental rewards or heuristic-based shaping, IG bridges the gap between tool execution and internal policy learning. It provides a mathematically rigorous proxy for tool utility by explicitly measuring the reduction in the policy model's predictive uncertainty when external result is introduced. For each generated token o_t, the IG is formally defined as the log-probability difference between the actual prediction and its counterfactual baseline:

$$\text{IG}_t = \log \pi_\theta(o_t \mid o_{<t}, x) - \log \pi_\theta(o_t \mid o_{<t}^{\text{cf}}, x). \tag{5}$$

From an information-theoretic and causal inference perspective (Miller et al., 2025; Raman et al., 2023; Pearl, 2009; Kong et al., 2026), IG_t effectively isolates the effect of the external environment on the model's reasoning state. A highly positive IG_t implies that the generation of o_t is strongly conditioned on the tool's output, such as accurately extracting specific calculated values or parsing execution logs, indicating that the tool successfully injected critical task-solving knowledge. Conversely, an IG_t near zero suggests the token belongs to generic linguistic structures that are generated independently of the tool. Furthermore, a negative IG_t implies that the external feedback is noisy, malformed, or confusing, actively degrading the model's confidence in generation.

To integrate this metric into the RL objective as a stable advantage multiplier, we transform it into a strictly non-negative and bounded signal. We apply a ReLU activation followed by a value clamping mechanism:

$$S_t = \text{clip}(\text{ReLU}(\text{IG}_t),\, 0,\, S_{\max}), \tag{6}$$

which guarantees stability through two mechanisms. First, ReLU acts as an asymmetric filter, zeroing out negative IG values so the policy is not penalized for recovering from noisy tool feedback. Second, $S_{\max}$ strictly bounds the credit multiplier, mitigating gradient explosion risks caused by extreme distributional shifts. This bounded signal distills complex tool interactions into a robust token-level metric. Also, we provide detailed computational burden analysis of IG computation in Appendix D.6.

4.3 Hierarchical Credit Assignment

Guided by the bounded IG signal S_t, **TACT** executes tool-aware credit assignment through a hierarchical optimization. Specifically, it first filters out structurally invalid rollouts at the macro-level to prevent noisy gradient updates, and then dynamically scales the token-level advantages of successful rollouts based on S_t. This two-tiered approach guarantees that learning signals are selectively amplified exclusively for high-utility reasoning steps within structurally sound trajectories.

Trajectory-Level Filtering. In agentic RL, unreliable trajectories introduce noisy gradients that can severely disrupt policy optimization. Specifically, we identify two sources of unreliable feedback: (1) *failed tool execution*, where the tool call returns an error, times out, or fails to execute due to malformed invocation, and (2) *uninformative rollout groups*, where all responses within a group yield an incorrect final answer, providing no meaningful reward signal for advantage estimation. Amplifying the advantages of such trajectories via S_t would inadvertently reinforce these unreliable

outcomes. Therefore, we define a trajectory validity indicator $\mathbb{1}[\text{valid}(o)]$. A rollout sequence o is defined as valid ($\mathbb{1}[\text{valid}(o)] = 1$) if and only if its tool executions complete successfully and it belongs to a rollout group containing at least one correct final answer. Any trajectory failing these criteria ($\mathbb{1}[\text{valid}(o)] = 0$) is excluded from IG-guided advantage shaping.

Token-Level Gated Advantage Shaping. Building upon the trajectory-level filtering, we next perform fine-grained advantage shaping at the token level. To achieve this, we convert the bounded IG signal S_t into a multiplicative scaling factor:

$$M_t = 1 + \alpha \cdot S_t, \tag{7}$$

where α acts as a scaling coefficient. This factor ensures that the baseline advantage is strictly amplified ($M_t \geq 1$) for qualifying tokens. To prevent the introduction of noise at inconsequential positions or the amplification of flawed reasoning paths, M_t is applied to the base advantage A_t (computed via Eq. 1) exclusively when a rigorous triple-gating condition G_t is satisfied:

$$G_t = \mathbb{1}[\text{valid}(o)] \cdot \mathbb{1}[\text{success}(o)] \cdot \mathbb{1}[t \in \mathcal{I}_{\text{post}}]. \tag{8}$$

This triple-gating mechanism enforces rigorous constraints, ensuring that the IG-based multiplier is activated for reasoning steps that are structurally valid, empirically successful, and causally dependent on tool feedback. First, the integrity filter ($\mathbb{1}[\text{valid}(o)]$) guarantees the rollout is free from execution formatting collapse, providing a structurally reliable foundation. Second, the success filter ($\mathbb{1}[\text{success}(o)]$) restricts the S_t bonus exclusively to rollouts that ultimately yield positive environmental rewards (e.g., producing the correct final answer). This actively prevents the policy from confidently amplifying tool-use patterns that lead to incorrect conclusions. Finally, the causal post-tool filter ($\mathbb{1}[t \in \mathcal{I}_{\text{post}}]$) strictly enforces temporal causality. Because tokens generated before the environment returns the tool observation cannot possibly be influenced by it, applying an IG-based bonus to these preceding tokens would be a mathematical fallacy. Ultimately, the reshaped tool-aware advantage $\hat{A}_t^i$ is defined as:

$$\hat{A}_t^i = \begin{cases} M_t \cdot A_t^i & \text{if } G_t = 1, \\ A_t^i & \text{otherwise.} \end{cases} \tag{9}$$

Note that the *otherwise* branch retains the original advantage rather than zeroing it out, since these tokens have already passed trajectory-level filtering and remain valid reasoning steps. Moreover, as rollout groups with entirely incorrect answers are filtered, so the M_t consistently amplifies credit rather than magnifying penalization. By applying $\hat{A}_t^i$ in the GRPO objective (Eq. 2), **TACT** dynamically aligns policy updates with the verified utility of tool interactions. Reasoning steps that actively leverage external tool outputs to advance the solution receive magnified gradients, effectively resolving the training instability and credit assignment bottleneck inherent in agentic RL. We provide hyperparameter analysis in Appendix D and practical implementation details of **TACT** in Appendix G.

5 Experiments

Our experiments are designed to answer the following questions: (i) How does **TACT** compare against existing agentic RL approaches across diverse reasoning tasks? (ii) How effectively does the IG-guided hierarchical credit assignment stabilize training and improve LLM reasoning performance? (iii) How do individual components and hyperparameters contribute to the overall effectiveness of **TACT**? (iv) How does **TACT** perform in code generation and tool-use tasks?

5.1 Experimental Setup

To comprehensively validate **TACT** across different model scales and data regimes, we select GRPO as our base optimization algorithm and conduct all training within the VeRL framework (Sheng et al., 2025). For small-scale validation, we train Qwen3-4B-Base on the Math3-5 dataset (Zeng et al., 2025). To evaluate scalability, we train Qwen3-8B-Base and Qwen2.5-7B on the larger DAPO-MATH-17K dataset (Yu et al., 2025a). We compare **TACT** against standard TIR, and SimpleTIR (Xue et al., 2026). Extensive evaluation is conducted on seven mathematical reasoning benchmarks: AIME 2024 & 2025 (Codeforces), AMC 2023 (amc, 2023), MATH500 (Hendrycks et al., 2021), OlympiadBench (He et al., 2024), Hmmt25 (Balunović et al., 2025), and Minerva (Lewkowycz et al., 2022), code generation benchmark LiveCodeBench-v5 (Jain et al., 2025), and function calling benchmark BFCL-v4 (Patil et al., 2025).

Table 1: Performance comparison on seven benchmarks under Qwen3-4B-Base, Qwen2.5-7B, and Qwen3-8B-Base base models on *Avg@16*, which denotes the average accuracy over 16 evaluations per benchmark. We bold the best scores and underline the sub-optimal results. Δ means the difference between **TACT** and sub-optimal results.

Method	AIME24	AIME25	MATH500	Olympiad	AMC23	Hmmt25	Minerva
Qwen3-4B-Base							
Base model	6.25	6.67	40.78	22.54	27.66	7.08	8.87
TIR	12.08	6.25	58.59	29.90	35.78	11.67	18.52
SimpleTIR	27.50	23.54	81.13	46.71	**64.53**	19.79	23.81
TACT	**35.00**	**25.21**	**84.59**	**51.05**	63.59	**21.25**	**25.16**
Δ	+7.50	+1.67	+3.46	+4.34	-0.94	+1.46	+1.35
Qwen2.5-7B							
Base model	5.00	3.96	31.99	13.80	21.56	6.67	5.00
TIR	10.63	10.42	42.11	20.88	28.91	11.04	7.95
SimpleTIR	33.33	21.04	77.51	42.95	68.75	16.66	21.23
TACT	**33.75**	**25.00**	**79.33**	**46.65**	**70.47**	**18.75**	**23.64**
Δ	+0.42	+3.96	+1.82	+3.70	+1.72	+2.09	+2.41
Qwen3-8B-Base							
Base model	7.50	7.08	42.26	22.38	25.16	7.29	8.96
TIR	6.88	6.46	29.73	16.91	24.22	3.75	8.36
SimpleTIR	43.33	29.79	86.36	55.75	**77.19**	25.42	25.97
TACT	**45.00**	**30.40**	**87.14**	**56.66**	76.41	**27.29**	**27.05**
Δ	+1.67	+0.61	+0.78	+0.91	-0.78	+1.87	+1.08

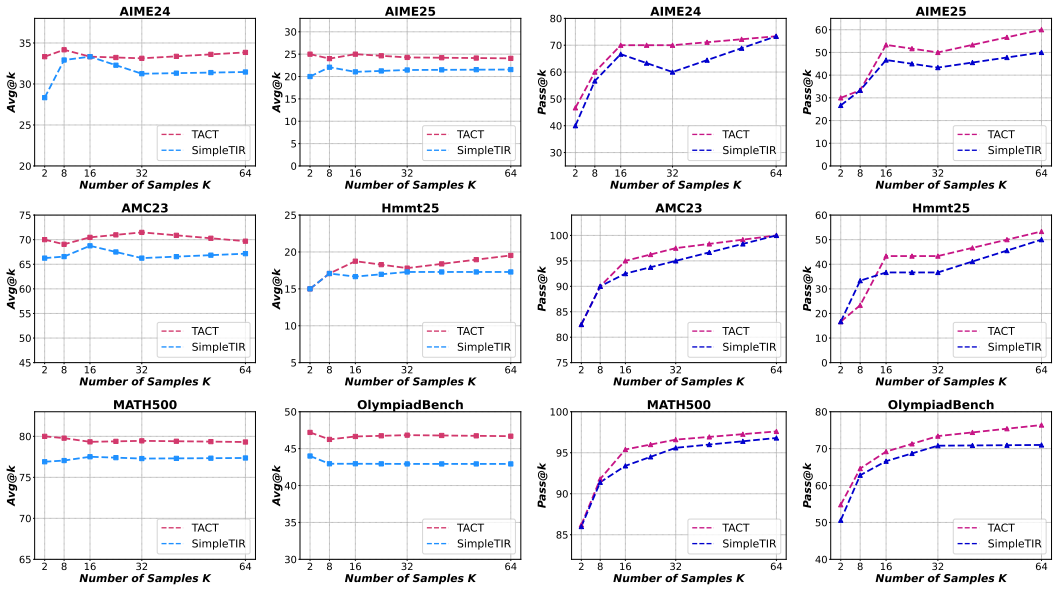


Figure 3: Asymptotic performance of *Pass@K* and *Avg@K* curves of **TACT** compared with SimpleTIR in 6 benchmarks with the increase of number of samples K under Qwen2.5-7B base model.

We evaluate performance using a comprehensive set of metrics. To assess accuracy, we report *Pass@K* (the proportion of problems where at least one of K sampled outputs is correct), *Avg@K* (the average accuracy across K samples), and *Len@K* (average response length). To evaluate tool-use efficiency, we quantify *Tool Call* (average tool invocations per response), and *Code Line* (average lines of code generated per rollout). More experimental details are provided in Appendix C.

5.2 Main Results

Overall Performance. Table 1 summarizes the comparison between **TACT** and two baselines across seven challenging reasoning benchmarks under three base models. **TACT** consistently outperforms

Table 2: Performance comparison on seven benchmarks on *Len@16*. We bold the best results. Δ means the difference between the results of **TACT** and SimpleTIR results.

	Method	AIME24	AIME25	AMC23	MATH500	OlympiadBench	Hmmt25	Minerva
Qwen3-4B-Base	**SimpleTIR**	4774.04	4683.71	3298.90	2205.36	3454.78	4622.45	2583.22
	TACT	**6351.49**	**5986.07**	**4506.41**	**2698.33**	**4588.86**	**5979.58**	**2633.03**
	Δ	+1577.45	+1302.36	+1207.51	+492.97	+1134.08	+1357.13	+49.81
Qwen2.5-7B	**SimpleTIR**	1911.89	1996.40	1414.06	1083.44	1561.74	1999.15	1394.50
	TACT	**2043.85**	**2147.46**	**1422.56**	**1091.55**	**1562.79**	**1977.95**	**1387.62**
	Δ	+131.96	+151.06	+8.50	+8.11	+1.05	-21.20	-6.88
Qwen3-8B-Base	**SimpleTIR**	4479.81	4568.94	2974.41	1976.81	2974.41	4157.43	1696.66
	TACT	**4734.47**	**4632.16**	**3063.42**	**2070.64**	**3323.72**	**4395.38**	**1799.78**
	Δ	+254.66	+63.22	+89.01	+93.83	+349.31	+237.95	+103.12

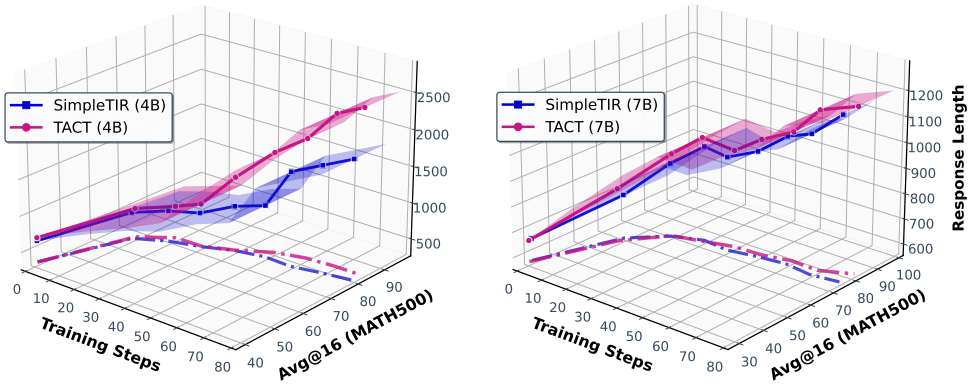


Figure 4: Learning curves of average accuracy (*Avg@16*) and response length during training for **TACT** and SimpleTIR under Qwen3-4B-Base and Qwen2.5-7B. The x-axis denotes training steps, the y-axis denotes average accuracy, and the z-axis denotes response length.

SimpleTIR across nearly all benchmarks and model scales, demonstrating the effectiveness of tool-aware credit assignment for stabilizing agentic RL optimization. Under Qwen3-4B-Base, **TACT** improves the average *Avg@16* accuracy from 41.00% to 43.69%, with particularly large gains on AIME24 ($+7.50$), OlympiadBench ($+4.34$), and MATH500 ($+3.46$). Similar improvements are observed on Qwen2.5-7B, where **TACT** raises the average accuracy from 40.21% to 42.51%, with consistent gains across all seven benchmarks. On the larger Qwen3-8B-Base, **TACT** achieves the best performance on six of seven benchmarks. Overall, these results demonstrate that explicitly modeling tool contribution enables more effective utilization of external execution feedback compared with uniform trajectory-level optimization.

Reasoning Performance. As shown in Table 2, **TACT** consistently achieves longer reasoning trajectories across all three base models, with the sole exception of Qwen2.5-7B on Hmmt25 and Minerva, where the reasoning length is slightly lower than the baseline. These results demonstrate that **TACT** encourages the model to engage in more extensive reasoning rather than terminating prematurely, which in turn contributes to its improved reasoning performance.

5.3 Property Analysis

Sample Efficiency. Figure 4 further illustrates the training dynamics of **TACT** and SimpleTIR in terms of reasoning accuracy and response length. Compared with SimpleTIR, **TACT** achieves higher *Avg@16* accuracy throughout training while maintaining effective exploration with longer reasoning trajectories. This indicates that the proposed information-gain-guided credit assignment does not simply encourage additional generation, but selectively reinforces reasoning steps that benefit from successful tool interactions. By filtering unreliable rollouts and assigning larger optimization signals

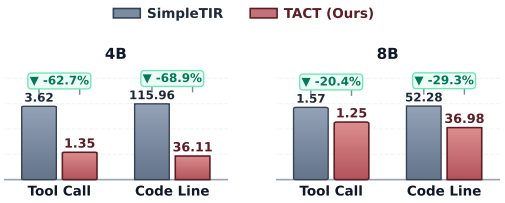


Figure 5: Performance comparison on AIME24 benchmark under 4B and 8B base models.

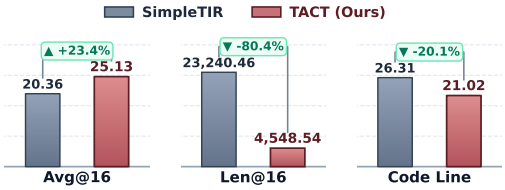


Figure 6: Generalization performance on the LiveCodeBench-v5 of code generation.

to tokens causally influenced by tool feedback, **TACT** improves both reasoning effectiveness and training stability in tool-integrated reinforcement learning.

Tool-Integrated Performance. To analyze the efficiency of tool-integrated reasoning, we compare the number of tool calls and code lines on AIME24 under 4B and 8B base models, as shown in Figure 5. We observe that **TACT** achieves better reasoning performance with fewer tool calls and shorter code, indicating more efficient tool utilization. Notably, SimpleTIR exhibits substantially longer code on the weaker 4B base model, whereas **TACT** maintains consistent efficiency across both model scales. This further validates the efficiency of **TACT** in tool-integrated reasoning.

Generalization. As shown in Figure 6, we evaluate the generalization performance of **TACT** in Qwen2.5-7B base model on LiveCodeBench-v5 for code generation, a benchmark with a distribution distinct from training data. **TACT** achieves a higher performance upper bound while requiring shorter reasoning length and code lines, whereas SimpleTIR relies on substantially useless longer reasoning. These results demonstrate that **TACT** generalizes effectively to code generation while maintaining concise and efficient reasoning performance.

Tool-Use Capability. To assess **TACT**'s tool-use execution capability, we evaluate on BFCL-v4 (Patil et al., 2025), which covers multi-step Python function-calling tasks with diverse interaction patterns (Figure 7). **TACT** outperforms SimpleTIR, with notable gains on Multiple AST and Parallel AST, including a substantial 19.0 point improvement on Parallel AST, demonstrating its effectiveness in handling complex, parallel tool interactions. **TACT** also achieves a slightly higher success rate on Irrelevance Detection. These results confirm that **TACT** can improve the tool-use capability of LLMs agent in executing multi-step Python tool calls. ***Refer to Appendix D for more experimental results.***

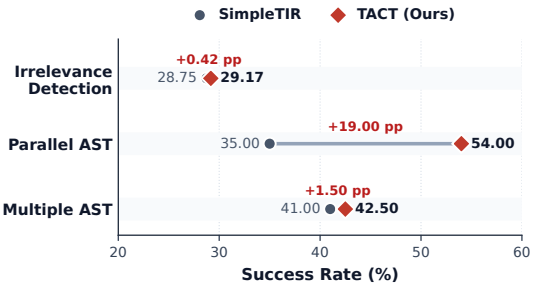


Figure 7: Performance on BFCL-v4 for multi-step Python function calls under Qwen2.5-7B.

6 Conclusion and Limitation

We propose **TACT**, a framework for tool-aware credit assignment. By constructing counterfactual trajectories, **TACT** measures each token's tool contribution via information gain, without auxiliary reward models. Guided by this signal, **TACT** filters unreliable rollouts and reshapes token-level advantages to reward reasoning steps that genuinely benefit from tool feedback. Experiments on 9 benchmarks across 3 model scales show that **TACT** improves both reasoning performance and training stability, while generalizing well to code generation and function calling tasks.

Limitation. **TACT** does not consider distinguishing different tool types that may exhibit distinct feedback structures and noise patterns. The IG computation may not fully capture tool-specific characteristics, potentially limiting the precision of credit assignment in multi-tool settings. Extending **TACT** to account for heterogeneous tool behaviors is a promising direction for future work.

AI USE STATEMENT

In this work, we used AI tools to check and polish grammar in the manuscript text. No content, ideas, or experimental results were generated by AI tools.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable data, or sensitive applications, and we do not identify any ethical concerns associated with this research. All benchmarks and models used in our experiments are publicly available and used in accordance with licenses.

REPRODUCIBILITY STATEMENT

The implementation and experimental details are shown in Appendix B, C and H. We will provide the source code of **TACT** after acceptance.

REFERENCES

American mathematics competitions - amc. 2023. URL https://maa.org/.

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

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, February 2025. URL https://matharena.ai/.

Yifei Chen, Guanting Dong, and Zhicheng Dou. Et-agent: Incentivizing effective tool-integrated reasoning agent via behavior calibration. *arXiv preprint arXiv:2601.06860*, 2026.

Mingyue Cheng, Shuo Yu, Daoyu Wang, Qingchuan Li, Xiaoyu Tao, Jie Ouyang, Yucong Luo, Yitong Zhou, Qi Liu, and Enhong Chen. Agent-r1: A unified and modular framework for agentic reinforcement learning. *arXiv preprint arXiv:2511.14460*, 2025.

MAA Codeforces. American invitational mathematics examination-aime 2024, 2024.

Guanting Dong, Licheng Bao, Zhongyuan Wang, Kangzhi Zhao, Xiaoxi Li, Jiajie Jin, Jinghan Yang, Hangyu Mao, Fuzheng Zhang, Kun Gai, et al. Agentic entropy-balanced policy optimization. *arXiv preprint arXiv:2510.14545*, 2025a.

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

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. Agentic reinforced policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TX4k7BF6aO.

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

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=QXEhBMNrCW.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. 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)*, pp. 3828–3850, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pp. 58791–58831, 2025.

Dongfu Jiang, Yi Lu, Zhuofeng Li, Zhiheng Lyu, Ping Nie, Haozhe Wang, Alex Su, Hui Chen, Kai Zou, Chao Du, et al. Verltool: Towards holistic agentic reinforcement learning with tool use. *arXiv preprint arXiv:2509.01055*, 2025.

Fanqi Kong, Jiayi Zhang, Mingyi Deng, Chenglin Wu, Yuyu Luo, and Bang Liu. Infopo: Information-driven policy optimization for user-centric agents. *arXiv preprint arXiv:2603.00656*, 2026.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

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

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

Bingqian Lin, Yunshuang Nie, Ziming Wei, Jiaqi Chen, Shikui Ma, Jianhua Han, Hang Xu, Xiaojun Chang, and Xiaodan Liang. Navcot: Boosting llm-based vision-and-language navigation via learning disentangled reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(7):5945–5957, 2025. doi: 10.1109/TPAMI.2025.3554559.

Haipeng Luo, Huawen Feng, Qingfeng Sun, Can Xu, Kai Zheng, Yufei Wang, Tao Yang, Han Hu, and Yansong Tang. Agentmath: Empowering mathematical reasoning for large language models via tool-augmented agent. *arXiv preprint arXiv:2512.20745*, 2025.

Xinji Mai, Haotian Xu, Xing W, Weinong Wang, Yingying Zhang, and Wenqiang Zhang. Agentic RL scaling law: Spontaneous code execution for mathematical problem solving. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=kXieirlPjF.

Moritz Miller, Bernhard Schölkopf, and Siyuan Guo. Counterfactual reasoning: an analysis of in-context emergence. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

Judea Pearl. *Causality*. Cambridge university press, 2009.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru WANG, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=eOLdGbXT6t.

Chirag Raman, Alec Nonnemaker, Amelia Villegas-Morcillo, Hayley Hung, and Marco Loog. Why did this model forecast this future? information-theoretic saliency for counterfactual explanations of probabilistic regression models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Ning Shang, Yifei Liu, Yi Zhu, Li Lyna Zhang, Weijiang Xu, Xinyu Guan, Buze Zhang, Bingcheng Dong, Xudong Zhou, Bowen Zhang, et al. rstar2-agent: Agentic reasoning technical report. *arXiv preprint arXiv:2508.20722*, 2025.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*, 2025.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. Otc: Optimal tool calls via reinforcement learning. *arXiv e-prints*, pp. arXiv–2504, 2025a.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.

Ran Xu, Jingjing Chen, Jiayu Ye, Yu Wu, Jun Yan, Carl Yang, and Hongkun Yu. Incentivizing agentic reasoning in LLM judges via tool-integrated reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=AXNRILww9c.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun MA, and Bo An. SimpleTIR: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=EplNy91Xqh.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025a.

Zhaochen Yu, Ling Yang, Jiaru Zou, Shuicheng Yan, and Mengdi Wang. Demystifying reinforcement learning in agentic reasoning. *arXiv preprint arXiv:2510.11701*, 2025b.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhong-Zhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita Velez, Yue Liao, Hongru WANG, Mengyue Yang, Heng Ji, Jun Wang, Shuicheng YAN, Philip Torr, and LEI BAI. The landscape of agentic reinforcement learning for LLMs: A survey. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=RY19y2RI1O. Survey Certification.

Jie Zheng, Dusit Niyato, Ruichen Zhang, Jiacheng Wang, Jiangtian Nie, Hongyang Du, Jiawen Kang, Haijun Zhang, Abbas Jamalipour, and Dong In Kim. Advanced deep reinforcement learning for agentic ai and their applications in wireless network. *IEEE Transactions on Cognitive Communications and Networking*, 2026.

Zefang Zong, Dingwei Chen, Yang Li, Qi Yi, Bo Zhou, Chengming Li, Bo Qian, Peng Chen, and Jie Jiang. At^{2}po: Agentic turn-based policy optimization via tree search, 2026. URL https://arxiv.org/abs/2601.04767.

SUPPLEMENTARY MATERIAL

TABLE OF CONTENTS

A Broader Impact **14**

B Details of TACT **14**

B.1 Details of Trajectory Filtering 14

B.2 Details of Policy Optimization 15

C Experimental Setup **15**

C.1 Experimental Setup 15

C.2 Benchmarks 16

D More Results **16**

D.1 More Comparison 16

D.2 Reasoning Performance 17

D.3 Training Performance 19

D.4 Tool-Integrated Performance 19

D.5 Hyperparameter Sensitivity 20

D.6 Time Cost Analysis 20

D.7 Ablation Analysis 20

D.8 Function Calling Capability 21

E Case Study **22**

F Experimental Platforms and Licenses **22**

F.1 Platforms 22

F.2 Licenses 22

G Practical Implementations **25**

H Pseudo code of TACT during training **27**

A BROADER IMPACT

Beyond its empirical gains, **TACT** answers a broader question in agentic RL: how can policy optimization remain stable and efficient when part of the training signal originates outside the model's own control? By grounding credit assignment in a counterfactual calculation of tool-use influence, our approach offers one answer, separate what the model would have done on its own from what the environment actually changed, and let that difference guide the policy learning. We expect this perspective in **TACT** to be useful well beyond code execution settings, wherever agents must reason over information they did not generate themselves, including web search, API calls, or interaction with other diverse agents.

More broadly, the underlying idea behind **TACT** is not tied to any specific tool or task: whenever a training signal mixes reliable and unreliable components, distinguishing them before optimization, rather than treating the entire signal uniformly, can prevent unstable or wasted updates. This principle becomes increasingly important as RL post-training pipelines grow more complex, incorporating feedback from diverse and imperfect sources such as external tools, retrieved documents, or other learned models. In such settings, naively aggregating all signals into a single scalar reward obscures the underlying cause of success or failure, making it difficult for the policy to learn which of its own decisions were actually responsible for the observed outcome. This suggests a general lesson for RL post-training system design beyond LLM agents: precise attribution of why an outcome occurred is often as valuable as the outcome itself, and can be leveraged to guide learning more efficiently. We believe that as agentic systems increasingly rely on heterogeneous and imperfect external feedback, causal attribution mechanisms of this kind will become an essential component of stable and sample-efficient policy optimization, rather than an optional refinement.

Practically, training agentic systems that are both stable and sample-efficient lowers the cost of developing capable tool-using models, making this line of research more accessible outside large industrial labs. At the same time, models that fail less erratically when tools misbehave are safer to deploy in settings where tool outputs directly affect downstream actions. We see **TACT** as one step toward agentic training methods that are not only performant, but also transparent about which parts of a model's behavior are genuinely earned through reasoning versus inherited from its environment. Our counterfactual formulation treats all tool calls uniformly and has so far been validated primarily on code-execution tools. Tool modalities with different feedback structures, such as search or multi-agent communication, may require adapting the masking and information-gain estimation accordingly. We view extending this causal attribution framework to a broader range of tool types and interaction patterns as an important next step toward more general and trustworthy agentic training.

B DETAILS OF **TACT**

B.1 DETAILS OF TRAJECTORY FILTERING

To enforce temporal causality and prevent noise amplification, we strictly construct masks for both token positions and trajectory rollouts. Specifically, we define a post-tool mask $m_{\text{post},t} \in \{0, 1\}$ that identifies model-generated reasoning tokens produced in turns immediately following external tool feedback, while explicitly excluding the tool observation tokens ($\mathcal{I}_{\text{obs}}$). This mask is computed by tracking the position of each tool-call boundary within the rollout sequence: once a tool response is injected into the context, all subsequently generated tokens up to the next tool call (or the end of the sequence) are marked as eligible for IG-based advantage reshaping, whereas tokens preceding the first tool call, as well as the injected observation tokens themselves, are permanently excluded from this mechanism, since they cannot causally depend on tool feedback that has not yet occurred.

At the trajectory level, we apply two complementary filtering criteria to ensure that only reliable rollouts contribute to policy updates. First, rollouts exhibiting structural failures—such as formatting errors, unparseable code blocks, or execution timeouts during tool invocation—are excluded from loss computation entirely, as these reflect pathological generation behavior rather than genuine reasoning quality. Second, under the GRPO framework, rollout groups in which all trajectories yield identical rewards are assigned $\mathbb{1}[\text{valid}(o)] = 0$, since the resulting zero-variance advantage estimate (Eq. 1) provides no useful gradient signal and would otherwise be discarded implicitly by the optimization objective; we make this exclusion explicit to avoid wasted forward passes on uninformative groups. By default (scope = success), the IG multiplier is further restricted exclusively to successful rollouts

with positive baseline advantages, ensuring that the credit-amplification mechanism reinforces correct reasoning patterns rather than inadvertently strengthening spurious or incorrect trajectories that happen to contain high-IG tokens.

To compute $\text{IG}_t = \log \pi_\theta(o_t \mid o_{<t}, x) - \log \pi_\theta(o_t \mid o^{\text{cf}}_{<t}, x)$ efficiently, we replace all tool observation tokens in the input sequence with `pad_token_id` to form the counterfactual sequence o^{cf}, as described in Section 4.1. This substitution preserves the sequence length and positional encoding, ensuring that the counterfactual forward pass remains structurally aligned with the original trajectory at every token position. A non-differentiable forward pass (`torch.no_grad()`) is pre-executed on o^{cf} prior to the main policy gradient step, avoiding FSDP autograd state conflicts and memory overhead; the resulting counterfactual log-probabilities are cached and reused during advantage computation, incurring only a single additional forward pass per rollout batch rather than per-token recomputation. The raw IG signal is then clamped to form the advantage scaling factor

$$M_t = 1 + \alpha \cdot \text{clip}(\text{ReLU}(\text{IG}_t), 0, S_{\max}), \tag{10}$$

where we set $\alpha = 0.01$ and $S_{\max} = 3.0$ across all experiments unless otherwise specified; we find this configuration to strike a stable balance between amplifying genuinely tool-driven tokens and avoiding excessive advantage inflation that could destabilize training.

B.2 Details of Policy Optimization

For a prompt x and a group of G sampled rollouts $\{o^{(1)}, \ldots, o^{(G)}\}$ with trajectory rewards $R_i \in \{0, 1\}$, the base advantage A_i is estimated via GRPO group standardization: $A_i = (R_i - \bar{R})/(\sigma_R + \epsilon_{\text{adv}})$, where $\bar{R}$ and σ_R denote the mean and standard deviation of rewards within the group, and ϵ_{adv} is a small constant added for numerical stability when all rollouts in a group receive identical rewards. Token-level base advantages are initialized as $A^i_t = A_i$ for every position t in trajectory $o^{(i)}$, ensuring that all tokens within a trajectory initially share the same group-relative credit before token-level reshaping is applied. When the triple-gating condition $G_t = 1$ is satisfied, the reshaped advantage is $\hat{A}^i_t = M_t \cdot A^i_t$; otherwise, A^i_t is passed through unmodified, as described in Section 4.3.

The policy network π_θ is optimized by minimizing the total loss $\mathcal{L}_{\text{total}}(\theta) = \mathcal{L}_{\text{CLIP}}(\theta) - c_{\text{ent}}\mathcal{L}_{\text{entropy}}(\theta) + c_{\text{kl}}\mathcal{L}_{\text{KL}}(\theta)$. Here, $\mathcal{L}_{\text{CLIP}}$ is the standard PPO clipped surrogate objective evaluated with shaped advantages $\hat{A}_t$ in place of the original GRPO advantage in Eq. 2, allowing the tool-aware credit assignment mechanism to be seamlessly integrated into standard policy gradient training without modifying the underlying clipping or optimization procedure. $\mathcal{L}_{\text{entropy}}$ regularizes policy exploration by encouraging a non-degenerate output distribution, mitigating premature convergence to overly deterministic tool-invocation patterns. $\mathcal{L}_{\text{KL}}$ penalizes token-level KL divergence between π_θ and the reference policy π_{ref}, constraining policy drift and further stabilizing training in the presence of tool-induced distribution shift. The coefficients c_{ent} and c_{kl} balance these regularization terms against the primary clipped objective.

For system-level efficiency, training is scaled using PyTorch FSDP for distributed parameter sharding, Ulysses Sequence Parallelism to handle long tool-augmented reasoning trajectories, and FlashAttention with unpadded variable-length execution to avoid wasted computation on padding tokens introduced by heterogeneous trajectory lengths across the rollout group. Optimization uses AdamW with a maximum gradient norm clip of 1.0 to prevent large updates from the amplified advantages $\hat{A}^i_t$ from destabilizing training.

C Experimental Setup

C.1 Experimental Setup

To ensure reproducibility, we provide a detailed description of the training and evaluation hyperparameters in Table 3 and Table 4. Parameters not explicitly listed follow the default configurations of the VeRL (Sheng et al., 2025) framework. Hyperparameters specific to individual baselines are configured strictly according to their original papers, as described in Section 5.

As shown in Table 3, we conduct agentic RL training on three base models across different scales: Qwen3-4B-Base, Qwen2.5-7B, and Qwen3-8B-Base. For training data, we employ the **Math3-5**

dataset from SimpleRL (Zeng et al., 2025) (8K samples) for Qwen3-4B-Base, and the **DAPO-MATH-17K** dataset (Yu et al., 2025a) for both Qwen2.5-7B and Qwen3-8B-Base. ***All baseline methods are reproduced under identical configurations with TACT in the VeRL platform.*** Instead of epoch-based scheduling, all models are uniformly trained for a fixed budget of 80 steps to ensure comparable training progression and optimization dynamics. The rollout group size G is set to 16 for all models, with a minimum valid rollout threshold of 4 per prompt to guarantee sufficient contrastive signals for advantage estimation. Rejection sampling is enabled with an oversampling multiplier of 2.0 to compensate for rollouts filtered out by trajectory-level validation. The clipping ratio bounds are set to $\epsilon_{\text{high}} = 0.28$ and $\epsilon_{\text{low}} = 0.20$, alongside a dual-clip ratio $c = 3.0$. Tool-output tokens and void turns are masked during policy gradient computation. For **TACT**, the information gain scaling factor α is set to 0.1 for Qwen3-4B-Base and 0.01 for both Qwen2.5-7B and Qwen3-8B-Base, with the upper clamping bound $S_{\max}$ set to 3.0 across all models. All training experiments are carried out on four NVIDIA A100 GPUs.

C.2 Benchmarks

In our experiments, we conduct extensive validation on seven mathematical reasoning benchmarks alongside code generation and agentic function calling benchmarks to comprehensively assess reasoning performance, coding proficiency, and agentic tool-use capability:

- **AIME 2024 & 2025** (Codeforces): Collections of 30 problems each from the American Invitational Mathematics Examination 2024 and 2025, featuring challenging multi-step competition mathematics problems.
- **AMC 2023** (amc, 2023): A curated set of 40 problems from the 2023 American Mathematics Competitions (AMC 12).
- **MATH500** (Hendrycks et al., 2021): A 500-problem subset from the MATH benchmark covering seven core mathematical subjects at competition mathematics level.
- **OlympiadBench** (He et al., 2024): An Olympiad-level benchmark designed for advanced mathematical reasoning. We evaluate on the `OE_TO_maths_en_COMP` subset, which contains 674 open-ended, text-only English mathematics competition problems.
- **Minerva** (Lewkowycz et al., 2022): A benchmark consisting of 272 university- and graduate-level STEM problems sourced from MIT OpenCourseWare courses.
- **HMMT 25** (Balunović et al., 2025): A challenging dataset collected from the February 2025 Harvard-MIT Mathematics Tournament, serving as a rigorous, contamination-free benchmark for complex Olympiad-level mathematical reasoning.
- **LiveCodeBench-v5** (Jain et al., 2025): A holistic and contamination-free evaluation benchmark for code generation. Version 5 consists of algorithmic problems collected from competitive programming platforms (such as LeetCode, AtCoder, and Codeforces), assessing models on unseen, real-world coding challenges.
- **BFCL-v4** (Patil et al., 2025): The Berkeley Function-Calling Leaderboard (v4) evaluation framework. In our experiments, we specifically evaluate on the Non-Live and Live subsets covering multi-step function-calling tasks with diverse interaction patterns (e.g., Multiple AST, Parallel AST, and Irrelevance Detection) to comprehensively assess agentic tool-use and execution capabilities.

D More Results

D.1 More Comparison

Table 5 reports $Pass@16$ results, which reflect the upper bound of reasoning capability by measuring whether at least one of 16 sampled outputs passes verification. **TACT** consistently raises this upper bound over SimpleTIR across nearly all benchmarks and model scales, indicating that our tool-aware credit assignment not only improves average accuracy but also expands the model's capacity to reach correct solutions under sufficient sampling. Under Qwen3-4B-Base, **TACT** achieves particularly striking gains, most notably a $+23.33$ improvement on AIME24 and a $+10.00$ improvement on Hmmt25, while reaching a perfect 100.00 on AMC23. On Qwen2.5-7B, **TACT** delivers consistent improvements across all seven benchmarks, with gains exceeding 5 points on AIME25, AMC23, and HMMT25. On the larger Qwen3-8B-Base, **TACT** maintains its advantage on five benchmarks and matches SimpleTIR on HMMT25, with only a marginal 0.20-point gap on OlympiadBench—well within the variance expected from Pass@16 evaluation. Overall, these results suggest that as model

Table 3: Training hyperparameters across model scales.

Hyperparameter	Qwen3-4B-Base	Qwen2.5-7B	Qwen3-8B-Base
Training dataset	SimpleRL-Math3-5 (8K)	DAPO-MATH-17K	DAPO-MATH-17K
Total training steps	80	80	80
GRPO rollout size G	16	16	16
Min. valid rollouts	4	4	4
Oversampling multiplier	2.0	2.0	2.0
Sampling temperature	1.0	1.0	1.0
Learning rate	1e-6	1e-6	1e-6
Max prompt length	8000	8000	8000
Max response length	4000	4000	4000
Max observation length	256	256	256
Max agent turns	5	5	5
PPO mini-batch size	128	128	128
Clip ratio high ϵ_{high}	0.28	0.28	0.28
Clip ratio low ϵ_{low}	0.20	0.20	0.20
Dual-clip ratio c	3.0	3.0	3.0
KL loss coefficient	0	0	0
IG scaling factor α (**TACT**)	0.1	0.01	0.01
IG clamp bound $S_{\max}$ (**TACT**)	3.0	3.0	3.0

Table 4: Evaluation hyperparameters across model scales.

Hyperparameter	Qwen3-4B-Base	Qwen2.5-7B	Qwen3-8B-Base
Sampling temperature	1.0	1.0	1.0
Top-p	0.7	0.7	0.7
Max prompt length	10000	36000	30000
Max response length	30000	12000	10000
Max agent turns	10	10	10
Evaluation rollout size K	16	16	16

capacity increases, both methods approach a shared performance ceiling on certain benchmarks, while **TACT** continues to deliver the more substantial gains on the more challenging datasets such as AIME24 and AIME25, highlighting its effectiveness in unlocking latent reasoning capability rather than merely stabilizing existing behavior. Together with the $Avg@16$ results in Table 1, these results demonstrate that the proposed tool-aware credit assignment framework improve the reasoning performance and does not narrow exploration capability. Instead, **TACT** keeps rollouts from falling into tool-use failures that are difficult to recover.

D.2 Reasoning Performance

We evaluate the reasoning performance of the learned policies by analyzing the performance scaling behavior with respect to the number of sampled responses K. Specifically, we compare **TACT** with SimpleTIR using the *Avg@K* and *Pass@K* metrics across six mathematical reasoning benchmarks under the Qwen3-4B-Base model. The results are shown in Figure 8.

The *Pass@K* metric measures whether at least one correct solution can be obtained from K sampled responses, reflecting the upper-bound performance under a given sampling budget. As K increases, **TACT** consistently achieves higher Pass@K values and reaches its performance plateau with fewer samples compared with SimpleTIR. For example, on AIME24, **TACT** reaches an 80.0% pass rate at approximately $K = 16$, while SimpleTIR requires a larger sampling budget to approach a similar performance level. *Avg@K* evaluates the average quality of sampled responses and therefore reflects the overall reliability of the policy distribution. Across different benchmarks, **TACT** maintains higher

Table 5: Performance comparison on seven datasets under Qwen3-4B-Base, Qwen2.5-7B, and Qwen3-8B-Base base models on *Pass@16*, which represents at least one of 16 sampled model outputs passes verification. We bold the best scores and underline the sub-optimal results. Δ means the difference between **TACT** and sub-optimal results.

Method	AIME24	AIME25	MATH500	Olympiad	AMC23	Hmmt25	Minerva
Qwen3-4B-Base							
Base model	40.00	33.33	90.60	65.40	90.00	26.67	33.09
TIR	36.67	30.00	89.40	64.80	90.00	26.67	37.87
SimpleTIR	56.67	53.33	95.00	72.80	97.50	43.33	40.44
TACT	**80.00**	**53.33**	**96.80**	**75.60**	**100.00**	**53.33**	**45.22**
Δ	+23.33	+0.00	+1.80	+2.80	+2.50	+10.00	+4.78
Qwen2.5-7B							
Base model	36.67	26.67	88.00	55.00	80.00	26.67	36.67
TIR	40.00	20.00	79.80	49.80	70.00	23.33	30.51
SimpleTIR	66.67	46.67	93.40	66.60	92.50	36.67	39.34
TACT	**70.00**	**53.33**	**95.40**	**69.20**	**97.50**	**43.33**	**41.91**
Δ	+3.33	+6.66	+2.00	+2.60	+5.00	+6.66	+2.57
Qwen3-8B-Base							
Base model	50.00	30.00	92.20	69.60	85.00	26.67	36.03
TIR	40.00	26.67	79.80	50.40	80.00	16.67	35.66
SimpleTIR	73.33	56.67	98.00	**76.60**	97.50	**56.67**	43.01
TACT	**76.67**	**60.00**	**98.40**	76.40	**100.00**	**56.67**	**43.38**
Δ	+3.34	+3.33	+0.40	-0.20	+2.50	+0.00	+0.37

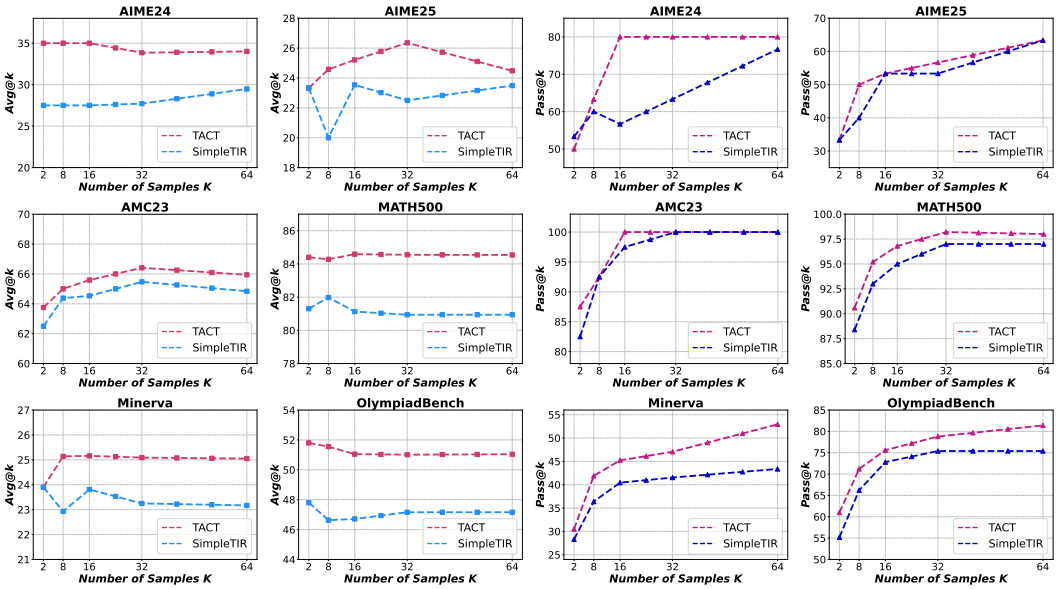


Figure 8: Asymptotic performance of *Pass@K* and *Avg@K* curves of **TACT** compared with SimpleTIR in 6 benchmarks with the increase of number of samples K under Qwen3-4B-Base base model.

Avg@K curves as the sampling budget increases, indicating that the generated responses are more consistently effective. These results demonstrate that the proposed tool-aware credit assignment framework improves the quality of the learned policy distribution rather than relying solely on increased sampling for obtaining correct solutions.

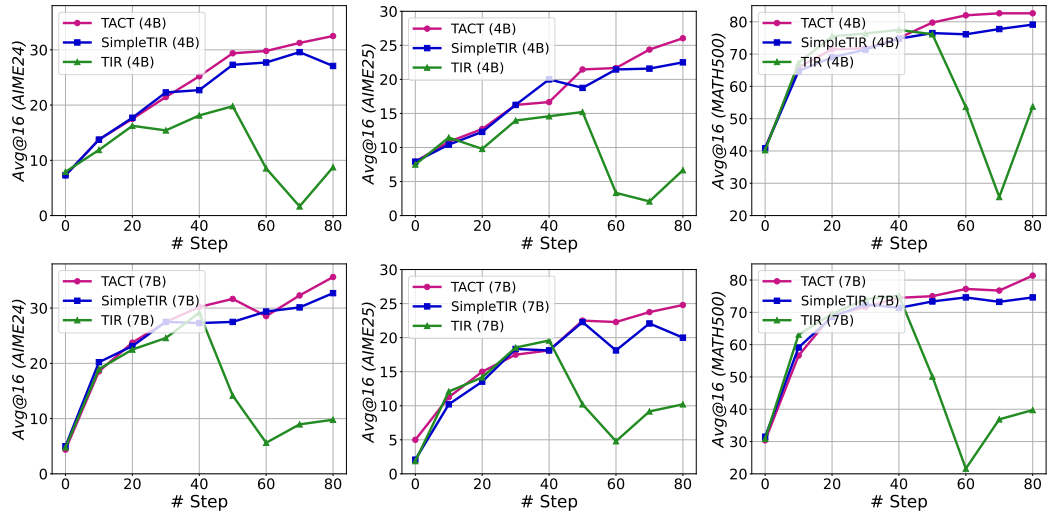


Figure 9: Learning performance of **TACT** compared with two baselines in three benchmarks under Qwen3-4B-Base and Qwen2.5-7B base models.

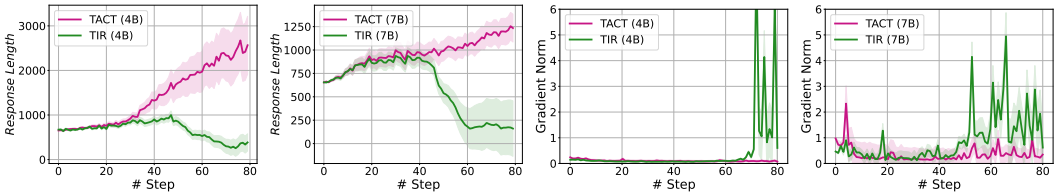


Figure 10: Evolution of response length and gradient norm during training for **TACT** and standard TIR under Qwen3-4B-Base and Qwen2.5-7B.

D.3 Training Performance

The training dynamics of **TACT** become clear when comparing its optimization behavior against SimpleTIR and standard TIR on AIME24, AIME25, and MATH500 (Figure 9). Tracking the *Avg@16* performance on Qwen3-4B-Base and Qwen2.5-7B reveals that **TACT** steadily improves and achieves the highest final accuracy. Standard TIR, by contrast, suffers a severe performance collapse in later stages despite initial gains, while SimpleTIR avoids this collapse but learns noticeably slower. This comparison confirms that tool-aware credit assignment successfully sustains effective learning signals throughout the training process.

Figure 10 further explains this stability through the lens of response lengths and gradient norms. Standard TIR typically peaks early in reasoning length before degenerating into truncated responses, whereas **TACT** steadily extends its reasoning trajectories, protecting the model's capacity for multi-step deduction. From an optimization standpoint, the gradient norm curves reveal why TIR struggles: it suffers from abrupt gradient spikes late in training (especially on the 4B model), pointing to severe interference from noisy tool feedback. **TACT** effectively flattens these disruptive spikes. By filtering unreliable rollouts and gating token-level updates based on verified tool utility, it guarantees a consistently smooth optimization landscape.

D.4 Tool-Integrated Performance

To evaluate tool utilization efficiency, we track the number of tool invocations (*Tool Call*) and the length of generated code (*Code Line*) across six mathematical benchmarks in Figure 12 and Figure 13. SimpleTIR frequently relies on excessive tool queries and generates overly long code blocks. This redundancy is especially severe under the 4B model (Figure 12), where the average code length

easily exceeds 70 lines on datasets like AIME25 and HMMT25, proving that uniform trajectory-level optimization fails to restrict uninformative tool usage. **TACT** directly resolves this inefficiency. By scaling learning signals according to actual information gain, it dramatically compresses code length by 43.1% to 68.6% and reduces tool calls by up to 61.0% on the 4B model. This highly compact interaction pattern seamlessly transfers to the stronger 8B model (Figure 13), where **TACT** consistently stabilizes at 1.0 to 1.5 tool calls per problem. By grounding policy updates in actual tool utility, **TACT** teaches the model to invoke tools only when truly needed. **TACT** achieves higher reasoning accuracy while significantly reducing tool invocation overhead.

D.5 Hyperparameter Sensitivity

To assess hyperparameter sensitivity, we evaluate the scaling coefficient α and the clipping threshold $S_{\max}$ on the Qwen2.5-7B model across seven benchmarks (Figure 14). The framework demonstrates strong robustness across configurations, demonstrating that the balance between credit amplification and optimization stability is maintained. Specifically, the default $\alpha = 0.01$ achieves the optimal trade-off; increasing it to 0.1 over-amplifies the IG signal and introduces training variance, whereas reducing it to 0.001 dilutes the intended guidance. Furthermore, **TACT** is relatively insensitive to the clipping threshold, as relaxing $S_{\max}$ from 3 to 10 causes only marginal performance shifts. Nevertheless, the tighter bound of $S_{\max} = 3$ remains preferable for strictly capping extreme IG values and ensuring stable policy updates.

D.6 Time Cost Analysis

To evaluate the time cost introduced by our proposed IG calculation, we quantify the per-step execution wall-clock time with and without IG computation under identical hardware configurations. As illustrated in Figure 11, the baseline training loop without IG consumes 384.36 seconds per learning step. Incorporating the IG metric incurs a marginal overhead of only 16.52 seconds, resulting in a single-step execution time of 400.88 seconds. This corresponds to a negligible 4.30% relative computational overhead, with the IG calculation accounting for a mere 4.12% of the total step time. These empirical results demonstrate that our proposed method integrates seamless information-theoretic guidance with minimal computational throughput penalty, preserving high scalability for practical deployments.

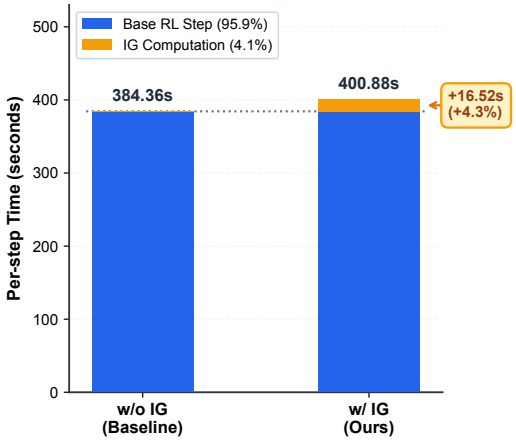


Figure 11: Time cost of one step training with and without IG computation.

D.7 Ablation Analysis

To further investigate the effect of the gating scope in IG-guided advantage shaping, we compare three variants of **TACT**: (1) **w/ Scope: *failed***, where the IG-based scaling is applied to all trajectories including failed tool executions and incorrect rollout groups; (2) **w/ Scope: *all***, where the scaling mechanism is applied to all valid tool-use trajectories without filtering based on the final outcome; and (3) **TACT (*success*)**, the default setting that only applies IG-based shaping to successful trajectories. As shown in Table 6, **TACT** achieves the highest average performance of 42.51%, outperforming both variants across most benchmarks. The **w/ Scope: *all*** variant obtains an average accuracy of 41.62%, which is higher than the failed-trajectory variant (41.16%), indicating that the IG signal can provide useful guidance when applied to reliable tool-use trajectories. However, the remaining gap compared with **TACT** demonstrates that filtering based on final task success is still important for selecting more effective optimization signals.

The performance degradation of **w/ Scope: *failed*** further shows the potential impact of unreliable feedback. Although this variant achieves competitive results on several benchmarks, including AIME24 and MATH500, applying IG-based scaling to failed trajectories introduces additional

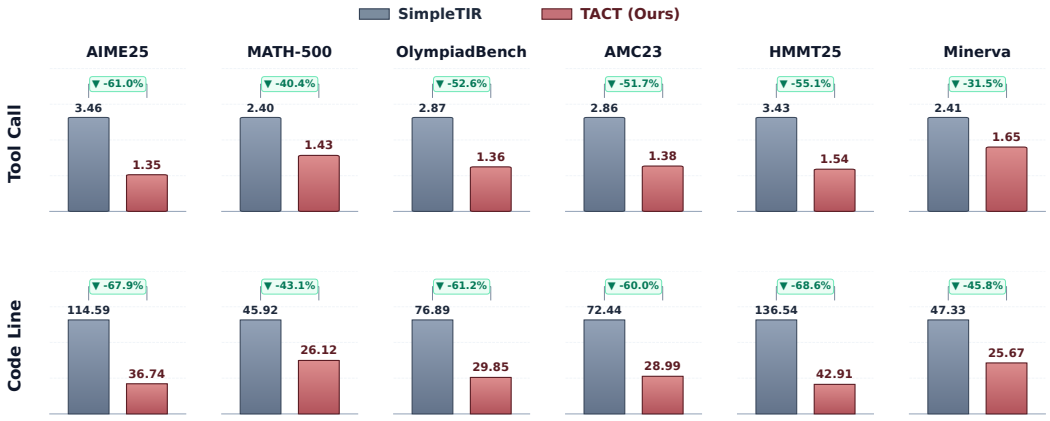


Figure 12: Performance of SimpleTIR and **TACT** of Tool Call and Code Line across remaining 6 mathematical reasoning benchmarks in Qwen3-4B-Base base model.

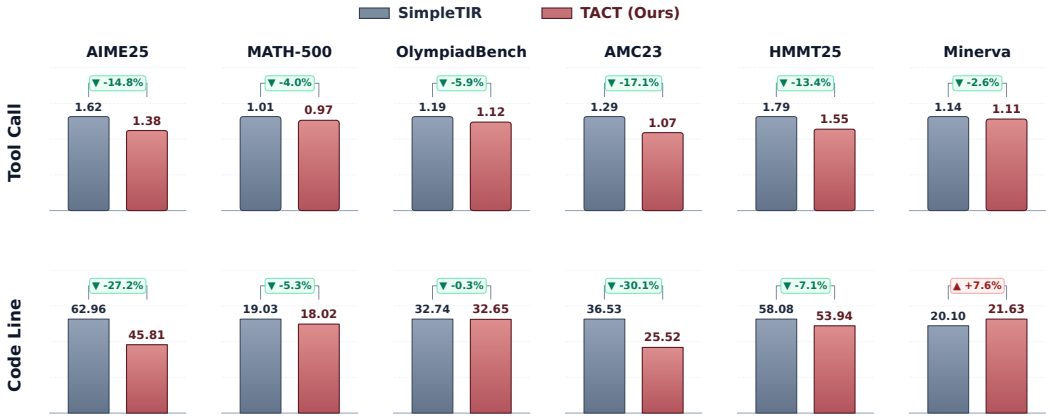


Figure 13: Performance of SimpleTIR and **TACT** of Tool Call and Code Line across remaining 6 mathematical reasoning benchmarks in Qwen3-8B-Base base model.

learning signals from unsuccessful tool interactions. These signals may partially interfere with policy optimization and reduce the overall effectiveness of credit assignment. Overall, the ablation results verify that trajectory-level filtering and token-level IG shaping are complementary components: the former removes unreliable updates, while the latter assigns larger credit to reasoning steps that benefit from tool feedback.

D.8 Function Calling Capability

To evaluate generalization to API-based tasks, we evaluate **TACT** on the BFCL-v4 benchmark across both Non-Live and Live tasks. The benchmark categorizes queries by interaction complexity, ranging from single (*Simple*), sequential (*Multiple*), and concurrent (*Parallel*) API calls, to mixed multi-layered dependencies (*PM*). It also evaluates tool selection accuracy (*Relevance*) and the ability to abstain from hallucinated calls (*Irrelevance*). On the static Non-Live split (Figure 15), **TACT** consistently outpaces the SimpleTIR baseline, particularly in structurally complex tasks. Most notably, it achieves a 19.0-point absolute surge in the Parallel AST category, proving that explicitly isolating tool contribution via information gain prevents the optimization interference typically caused by overlapping API calls. Concurrently, a higher success rate in Irrelevance Detection confirms that trajectory-level filtering successfully suppresses unneeded tool invocations.

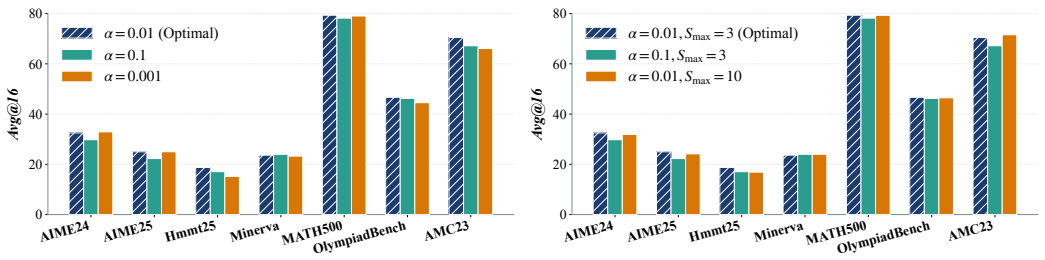


Figure 14: Hyperparameter analysis of α and S_{max} for **TACT** of *Avg@16* in seven benchmarks under Qwen2.5-7B base model.

Table 6: Ablation study on the gating scope of IG-Gate across seven benchmarks on *Avg@16* under Qwen2.5-7B base model. We bold the best results and underline the sub-optimal results.

Method	AIME24	AIME25	MATH500	Olympiad	AMC23	Hmmt25	Minerva	Avg.
Base model	5.00	3.96	31.99	13.80	21.56	6.67	5.00	12.57
TIR	10.63	10.42	42.11	20.88	28.91	11.04	7.95	18.85
w/ Scope: *failed*	**33.75**	22.29	<u>79.28</u>	<u>46.10</u>	68.44	15.21	<u>23.02</u>	41.16
w/ Scope: *all*	<u>32.50</u>	<u>23.96</u>	78.19	45.35	**71.88**	<u>16.88</u>	22.56	<u>41.62</u>
TACT (*success*)	**33.75**	**25.00**	**79.33**	**46.65**	<u>70.47</u>	**18.75**	**23.64**	**42.51**

Figure 16 extends this evaluation to the Live tasks, exposing the policy to the noise and unpredictability of real-time API execution. Here, **TACT**'s advantage peaks in the challenging *PM* category. While standard trajectory-level rewards often collapse under compounding errors when managing mixed concurrent and sequential dependencies, **TACT** remains highly stable. By dynamically gating credit assignment based solely on verified tool utility, the framework naturally resists external distribution shifts. These consistent gains across both static and dynamic environments confirm that **TACT** effortlessly scales to real-world function calling without relying on manual heuristic reward shaping.

E Case Study

As shown in Table 7, this case study highlights our method's superiority in prioritizing mathematical insight over blind computational brute-force, effectively avoiding the "tool-use trap" commonly observed in LLM-based reasoning. Faced with a complex non-linear system, the baseline model prematurely offloads the unsimplified equations to symbolic and numerical solvers. This leads to a degradation spiral: symbolic solvers fail to find explicit solutions, and numerical optimizers yield floating-point approximations that cannot reliably resolve the exact coprime integers required by the problem. In contrast, our method demonstrates a robust "reason-first, verify-later" paradigm. It strategically identifies the underlying geometric symmetry ($a = b$) to analytically reduce the constraints into a tractable cubic equation before invoking any external tools. By reserving code execution strictly for final fraction arithmetic and verification, our approach circumvents the automated solving bottleneck and efficiently converges to the exact rational solution.

F Experimental Platforms and Licenses

F.1 Platforms

All experiments of this approach are implemented on two Intel Xeon Platinum 8480+ CPUs and four NVIDIA A100 GPUs.

F.2 Licenses

In our implementation, we have utilized the following open-source libraries, each governed by its respective license agreements:

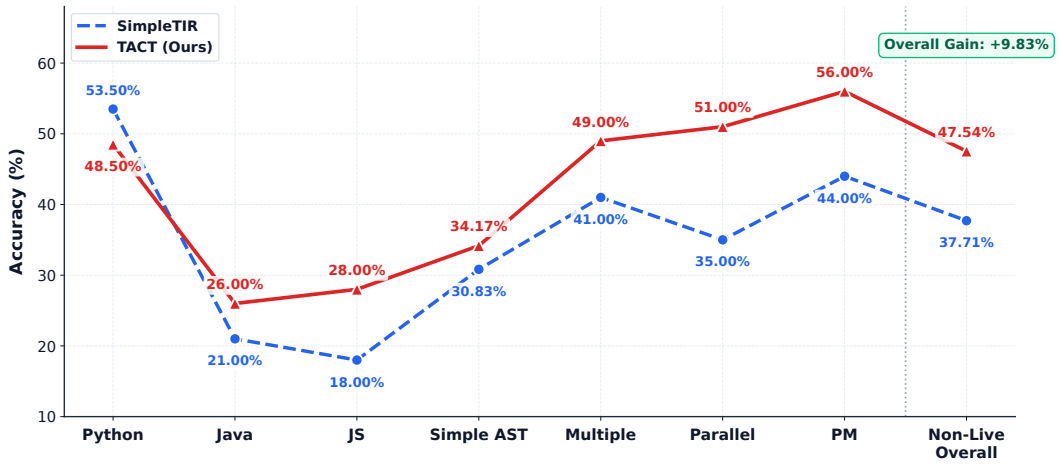


Figure 15: Performance of SimpleTIR and **TACT** in the Non-Live benchmark of BFCL-v4. The evaluation covers various programming languages (Python, Java, JS) and diverse interaction complexities. Specifically, **Simple AST** evaluates single API calls; **Multiple** evaluates sequential API calls; **Parallel** requires executing multiple APIs concurrently; and **PM** (Parallel Multiple) represents complex mixed dependencies requiring both concurrent and sequential API executions.

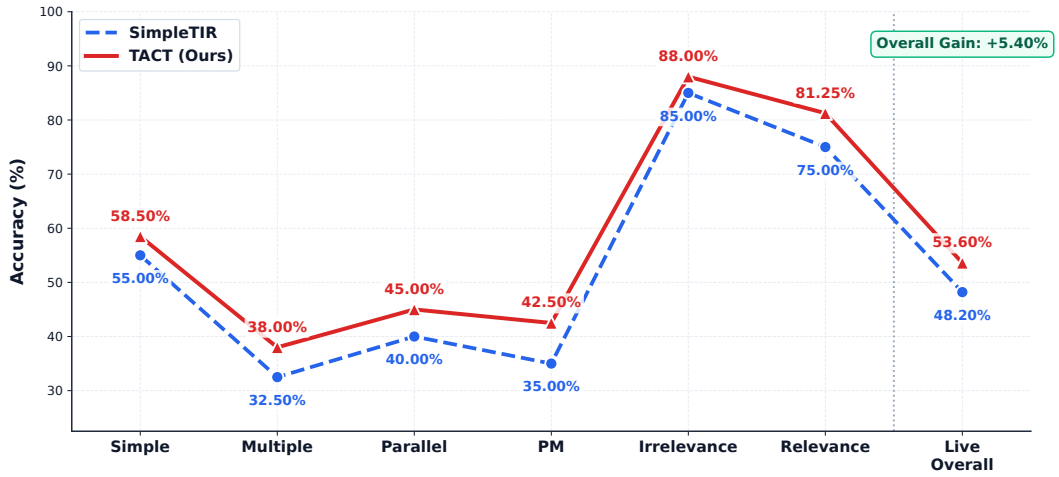


Figure 16: Performance comparison on the BFCL-v4 Live benchmark across varying interaction complexities. Similar to the Non-Live split, task complexities range from **Simple** (single API) to **Multiple** (sequential APIs), **Parallel** (concurrent APIs), and **PM** (mixed concurrent and sequential APIs). Additionally, **Irrelevance** measures the policy's ability to abstain from hallucinated tool calls when no external intervention is needed, while **Relevance** assesses the accuracy of tool selection.

Table 7: Case Study on AIME Problem. **TACT** demonstrates robust mathematical insight by utilizing algebraic symmetry to bypass numerical intractability and yield an exact rational fraction. SimpleTIR, conversely, falls into an endless loop of numerical optimization attempts.

Problem: Let $\mathcal{B}$ be the set of rectangular boxes with surface area 54 and volume 23. Let r be the radius of the smallest sphere that can contain each of the rectangular boxes that are elements of $\mathcal{B}$. The value of r^2 can be written as $\frac{p}{q}$, where p and q are relatively prime positive integers. Find $p+q$.
Ground Truth: 721

SimpleTIR (Baseline)

To find the radius r of the circumscribed sphere for dimensions a, b, c, we use $r^2 = \frac{a^2+b^2+c^2}{4}$. Given $2(ab+bc+ca) = 54$ and $abc = 23$, we formulate a non-linear system to solve for the dimensions symbolically.

```
import sympy as sp
a, b, c = sp.symbols('a b c', real=
    True, positive=True)
eq1 = sp.Eq(2*(a*b + b*c + c*a),
    54)
eq2 = sp.Eq(a*b*c, 23)
solutions = sp.solve((eq1, eq2), (a
    , b, c), dict=True)
```

Code Execution Result: KeyError: c
The symbolic solver fails to find explicit solutions. We adjust our strategy to use numerical optimization...

```
import numpy as np
from scipy.optimize import minimize
def objective(v):
    a, b, c = v
    return abs(2*(a*b+b*c+c*a)-54)
    + abs(a*b*c-23)
result = minimize(objective, [1, 1,
     23])
print(result.x[0]**2 + result.x
    [1]**2 + result.x[2]**2)
```

Code Execution Result: 41.06249999998738
The numerical result $r^2 \approx 10.2656$ cannot be converted reliably... We switch back to analytical methods...

```
L = a**2 + b**2 + c**2 + l1*(2*(a*b
    +b*c+c*a)-54) + l2*(a*b*c-23)
grad_L = [sp.diff(L, v) for v in (a
    , b, c, l1, l2)]
solutions = sp.solve(grad_L, (a, b,
     c, l1, l2), dict=True)
```

Code Execution Result: SyntaxError: invalid syntax
The non-linear system remains intractable... The model enters a loop of retrying different numerical solvers...

Final Answer: WRONG (Failed to resolve)

TACT (Ours)

To minimize the space diagonal $d = \sqrt{a^2+b^2+c^2}$ under the constraints $ab+bc+ca = 27$ and $abc = 23$, the minimum occurs when the box is as close to a cube as possible. We assume symmetry where two dimensions are equal: $a = b$.

Substituting $a = b$ into the volume constraint gives:

$$a^2 c = 23 \implies c = \frac{23}{a^2}$$

Substituting c into the surface area constraint yields:

$$a^2 + 2a\left(\frac{23}{a^2}\right) = 27 \implies a^3 - 27a + 46 = 0$$

We solve this cubic equation by testing rational roots (factors of 46). Testing $a = 2$:

$$2^3 - 27(2) + 46 = 8 - 54 + 46 = 0$$

Thus, $a = 2$ is a valid root. The dimensions are $a = 2, b = 2, c = 5.75$.

We calculate the exact value of r^2:

$$r^2 = \frac{2^2+2^2+(5.75)^2}{4} = \frac{4+4+33.0625}{4} = \frac{41.0625}{4}$$

We convert 41.0625 to an exact fraction: $41.0625 = \frac{657}{16}$.
Therefore, $r^2 = \frac{657/16}{4} = \frac{657}{64}$.
We verify if 657 and 64 are coprime and confirm the final fraction result via Python.

```
from fractions import Fraction
a, b = 2, 2
c = 23 / (a**2)
sum_of_squares = a**2 + b**2 + c**2
r_sq = Fraction(sum_of_squares).
    limit_denominator() / 4
print(r_sq.numerator + r_sq.
    denominator)
```

Code Execution Result: 721
The algebraic symmetry reduction leads directly to the exact rational solution, successfully bypassing the complex non-linear system that trapped the baseline.

Final Answer: 721

- **VeRL**: Apache License 2.0
- **Ray**: Apache License 2.0
- **vLLM**: Apache License 2.0
- **Weights & Biases**: MIT License
- **FlashAttention**: BSD 3-Clause License
- **TensorDict**: MIT License

G PRACTICAL IMPLEMENTATIONS

The detailed practical implementation of **TACT** is summarized in Algorithm 1. The optimization process proceeds through four phases per training iteration: **Phase 1: Rollout and Trajectory-Level Filtering.** Given a batch of sampled queries x, the policy model π_θ generates N rollout trajectories per query by interacting with the tool environment, establishing the base group-relative advantages A_t. At the macro level, to eliminate noise from malformed executions or uninformative prompt groups, we strictly evaluate each trajectory o via structural validity $\mathbb{1}[\text{valid}(o)]$ and task success $\mathbb{1}[\text{success}(o)]$. Filtering out invalid rollouts prior to computation prevents unnecessary counterfactual evaluations. **Phase 2: Counterfactual Reasoning and IG Computation.** For trajectories passing macro-level filtering, we isolate the impact of tool feedback by constructing a counterfactual response sequence o^{cf}, replacing tool observation tokens with `[PAD]`. A gradient-free counterfactual forward pass computes log-probabilities under o^{cf}, allowing us to measure the token-level Information Gain (IG_t) as the reduction in predictive uncertainty. To avoid gradient instability, IG_t is passed through a ReLU activation and value clamping to yield a bounded signal S_t, from which we derive the scaling factor $M_t = 1 + \alpha \cdot S_t$. **Phase 3: Triple-Gated Advantage Shaping.** At the micro level, to ensure the credit bonus is assigned exclusively to causally dependent and high-utility reasoning steps, we enforce a triple-gating condition $G_t = \mathbb{1}[\text{valid}(o)] \cdot \mathbb{1}[\text{success}(o)] \cdot \mathbb{1}[t \in \mathcal{I}_{\text{post}}] \cdot \mathbb{1}[A_t > 0]$. When $G_t = 1$, the base advantage is dynamically amplified ($\hat{A}_t = M_t \cdot A_t$); otherwise, the baseline advantage is preserved ($\hat{A}_t = A_t$), ensuring stability for standard valid steps. **Phase 4: Policy Optimization.** Finally, the reshaped advantages $\hat{A}_t$ are passed into the policy gradient optimization, aligning parameter updates with the verified utility of tool-assisted reasoning steps.

Algorithm 1 Tool-Aware Credit Assignment (**TACT**)

Input: Policy model π_θ, query dataset $\mathcal{D}$, rollout group size N, IG scaling factor α, maximum clipping threshold $S_{\max}$, learning rate η.

for each PPO update iteration **do**
  Sample a batch of queries $\{x\}$ from dataset $\mathcal{D}$
  // Phase 1: Rollout & Trajectory-Level Filtering
  for each query x **do**
    Sample N response trajectories $\{o^{(i)}\}_{i=1}^{N} \sim \pi_\theta(\cdot \mid x)$ interacting with environment
    Compute base token advantages $\{A_t^{(i)}\}$ via group-relative baseline
    for each trajectory $o^{(i)}$ **do**
      Evaluate trajectory validity indicator $\mathbb{1}[\text{valid}(o^{(i)})]$ (valid execution syntax & group has ≥ 1 correct answer)
      Evaluate trajectory success indicator $\mathbb{1}[\text{success}(o^{(i)})]$ (score > 0)
    end for
  end for
  // Phase 2: Counterfactual Reasoning & IG Signal Computation
  for each query x and each trajectory $o = \{o_t\}_{t=1}^{|o|}$ **do**
    if $\mathbb{1}[\text{valid}(o)] == 1$ **and** $\mathbb{1}[\text{success}(o)] == 1$ **then**
      Locate tool observation token positions $\mathcal{I}_{\text{obs}} \subset \{1, \ldots, |o|\}$ (info_mask $== 0$)
      Construct counterfactual sequence o^{cf} by replacing tool observations with `[PAD]`:
        $o_t^{\text{cf}} \leftarrow$ `[PAD]` if $t \in \mathcal{I}_{\text{obs}}$ else o_t
      Compute forward log-probs $\log \pi_{\theta_{\text{old}}}(o_t \mid o_{<t}, x)$ and counterfactual log-probs $\log \pi_{\theta_{\text{old}}}(o_t \mid o_{<t}^{\text{cf}}, x)$
      for each token position $t = 1, \ldots, |o|$ **do**
        Compute Information Gain: $\text{IG}_t \leftarrow \log \pi_{\theta_{\text{old}}}(o_t \mid o_{<t}, x) - \log \pi_{\theta_{\text{old}}}(o_t \mid o_{<t}^{\text{cf}}, x)$
        Compute bounded IG signal: $S_t \leftarrow \text{clip}(\text{ReLU}(\text{IG}_t),\ 0,\ S_{\max})$
        Compute advantage multiplier: $M_t \leftarrow 1 + \alpha \cdot S_t$
      end for
    end if
  end for
  // Phase 3: Triple-Gated Advantage Shaping
  for each query x and each trajectory $o = \{o_t\}_{t=1}^{|o|}$ **do**
    for each token position $t = 1, \ldots, |o|$ **do**
      Evaluate triple-gating condition:
        $G_t \leftarrow \mathbb{1}[\text{valid}(o)] \cdot \mathbb{1}[\text{success}(o)] \cdot \mathbb{1}[t \in \mathcal{I}_{\text{post}}] \cdot \mathbb{1}[A_t > 0]$
      if $G_t == 1$ **then**
        Reshape token advantage: $\hat{A}_t \leftarrow M_t \cdot A_t$
      else
        Retain base advantage: $\hat{A}_t \leftarrow A_t$
      end if
    end for
  end for
  // Phase 4: Policy Optimization
  Construct loss mask $\mathcal{M}_t \leftarrow \mathbb{1}[\text{valid}(o)] \cdot \text{loss_mask}_t$
  Compute policy loss $\mathcal{L}_{\text{GRPO}}(\theta)$ using reshaped advantages $\{\hat{A}_t\}$ and loss mask $\mathcal{M}_t$
  Update policy parameters: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{GRPO}}(\theta)$
end for

H PSEUDO CODE OF **TACT** DURING TRAINING

```
def train_tact_step(self, batch):
    # === Batch Tensors & Masks ===
    input_ids      = batch["input_ids"].to(self.device)
    responses      = batch["responses"].to(self.device)
    old_log_probs  = batch["old_log_probs"].to(self.device)
    advantages     = batch["advantages"].to(self.device)
    loss_mask      = batch["loss_mask"].to(self.device)
    info_mask      = batch["info_mask"].to(self.device)
    resp_len       = responses.size(1)
    post_tool_mask = batch["post_tool_mask"][:, -resp_len:].to(self.
    device)
    scores         = batch["token_level_scores"].to(self.device)

    # === Trajectory-Level Validity Filtering ===
    seq_scores   = scores.sum(dim=-1)
    valid_mask   = (seq_scores > 0).unsqueeze(-1)

    # === Counterfactual Sequence & Forward Pass ===
    cf_input_ids = input_ids.clone()
    obs_mask     = (info_mask == 0)
    cf_input_ids[obs_mask] = self.pad_token_id

    with torch.no_grad():
        cf_logits    = self.actor(cf_input_ids)[:, -resp_len-1:-1]
        cf_log_probs = logprobs_from_logits(cf_logits, responses)

    # === Bounded Information Gain (IG) Signal ===
    ig        = old_log_probs - cf_log_probs
    S_t       = torch.relu(ig)
    M_t       = 1.0 + self.alpha * torch.clamp(S_t, 0.0, self.s_max)

    # === Triple-Gated Advantage Shaping ===
    pos_adv_mask = (advantages > 0)
    gating_mask  = valid_mask & post_tool_mask.bool() & pos_adv_mask

    shaped_advantages = advantages.clone()
    shaped_advantages[gating_mask] *= M_t[gating_mask]

    # === Policy Forward & Optimization ===
    logits     = self.actor(input_ids)[:, -resp_len-1:-1]
    log_probs  = logprobs_from_logits(logits, responses)
    ratio      = torch.exp(log_probs - old_log_probs)
    surr1      = ratio * shaped_advantages
    surr2      = torch.clamp(ratio, 1.0 - self.clip_eps, 1.0 + self.
    clip_eps) * shaped_advantages
    policy_loss = -torch.min(surr1, surr2)

    # Combined Loss Masking (Trajectory & Token-Level)
    final_loss_mask = loss_mask * valid_mask
    loss       = (policy_loss * final_loss_mask).sum() /
    final_loss_mask.sum()

    # === Optimizer Update ===
    self.optimizer.zero_grad()
    loss.backward()
    torch.nn.utils.clip_grad_norm_(self.actor.parameters(), self.
    max_grad_norm)
    self.optimizer.step()

    return loss.item()
```