SOLVE MORE WITH FEWER STEPS: ASSIGNING CREDIT BY UNCOVERING DEPENDENCIES IN AGENTIC RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic reinforcement learning (RL) is a powerful technique for improving the multi-step reasoning of agents. The success of agentic RL largely relies on whether it can properly assign credit for final outcomes to intermediate steps, namely the credit assignment problem (CAP). However, solving the CAP remains highly challenging, as long reasoning trajectories often obscure the contribution of each step. To tackle this problem, we propose **CreditFlow**, a novel credit assignment method that directs agents to focus on critical steps, thus improving both reasoning performance and efficiency. Our key idea is to assign credit based on *underlying dependencies* (whether step j depends on the result of step i), rather than the *temporal order* (whether step j occurs after step i). The major challenge in developing CreditFlow is to efficiently distinguish underlying dependencies from the temporal adjacency in raw trajectories. To uncover the dependencies, CreditFlow introduces a rule-based provenance tracking mechanism to dynamically trace how later steps use results from earlier ones, *incurring negligible overhead* and *avoiding LLM-induced hallucinations*. Experiments on eight mainstream benchmarks show that CreditFlow outperforms the strongest baseline by **5 points** on average, while reducing the average number of inference steps by over **20%**.

1 INTRODUCTION

Large language model (LLM)-based agents (OpenAI, 2026; Anthropic, 2026; Qwen Team, 2026) have achieved remarkable success across diverse domains, including open-domain knowledge (Zhang et al., 2025a; Tao et al., 2026), mathematics (Chang et al., 2026; Mai et al., 2026), science and medicine (Ghareeb et al., 2026; Ferber et al., 2026), and general problem solving (Zhang et al., 2026a; Soni et al., 2026). A key driver behind these advances is their ability to use external tools and reason over multiple steps (Wei et al., 2026; Li, 2025; Cao et al., 2025). This interactive nature makes reinforcement learning (RL) particularly powerful for improving the multi-step reasoning of agents, as it directly optimizes sequential decisions from environmental feedback (Zhang et al., 2025b; Ke et al., 2025). Consequently, agentic RL has rapidly emerged as a major research frontier for advancing increasingly capable agents (Li et al., 2026b; Dong et al., 2026; Singh et al., 2025).

Credit assignment is pivotal to agentic RL, as it determines how delayed final outcomes translate into learning signals for intermediate steps (Pignatelli et al., 2023; Zhang, 2026). In multi-step reasoning, individual steps can contribute very differently to the final outcome: even a successful trajectory may contain redundant or weakly relevant steps, while different intermediate steps can contribute very differently to the final outcome. Proper credit assignment therefore must trace the delayed outcome back to the intermediate steps that actually shape it, so that agents receive accurate learning signals (Tan et al., 2026; Wang et al., 2025). However, as agents interleave reasoning, tool use, and environment feedback over long trajectories, these complex interactions increasingly obscure the contribution of each step, making accurate credit assignment highly challenging (Tan et al., 2026; Li et al., 2026a; Zhou et al., 2025).

Existing efforts have progressively moved from coarse trajectory-level supervision toward finer-grained credit assignment for intermediate steps, to better distinguish how different steps contribute along long reasoning trajectories. Outcome-level methods optimize all steps in a trajectory using its final outcome or trajectory-level advantage, providing simple and scalable supervision while leaving the different contributions of individual steps implicit (Shao et al., 2024; Jin et al., 2025; Li

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.

et al., 2026b). Recent studies therefore derive step-level learning signals from process or progress estimation, retrospective or role-based critics, verifiable intermediate feedback, and environment feedback (Wang et al., 2025; Choudhury, 2025; Tan et al., 2026; Zhang et al., 2026b; Yuan et al., 2026; Xu et al., 2026). Other methods estimate finer-grained credit from information-theoretic or counterfactual signals (Wang et al., 2026; Meng & Chen, 2026), or use trajectory and state-transition structures to measure the contribution of individual steps (Liu et al., 2026; Li et al., 2026a; Cheng et al., 2026). Despite these advances, existing methods still organize credit assignment around *temporally indexed steps or transitions*, estimating each step's contribution from its quality, progress, outcome variation, or structural position. This leaves a fundamental gap in current credit assignment: the **cross-step dependencies** that determine whether intermediate results are actually used downstream and ultimately contribute to the final outcome remain largely unmodeled.

In this paper, we propose **CreditFlow**, a novel credit assignment method that shifts the basis of credit assignment from *temporal order* to the *underlying dependencies across interaction steps*. Our key insight is that an intermediate step should receive credit according to whether its result is actually used by subsequent decisions and contributes to the final outcome, rather than simply because it occurs earlier in the trajectory. This simple principle is challenging to realize in practice, because such dependencies are implicit in raw interaction trajectories and can be easily conflated with temporal adjacency or mere information availability. Moreover, directly relying on LLMs to infer pairwise dependencies introduces substantial additional inference cost and may hallucinate unsupported relations, leading to erroneous credit paths. To overcome these challenges, CreditFlow introduces a rule-based provenance tracking mechanism that dynamically traces how later steps consume results produced by earlier ones. Then, it organizes these producer-consumer relations into an explicit dependency graph, incurring negligible overhead while avoiding LLM-induced hallucinations. Based on the uncovered dependency structure, CreditFlow propagates terminal feedback along valid dependency paths to construct dependency-aware step-level advantages, thereby preferentially reinforcing steps that truly support the final outcome.

Our Main Contributions. (1) We introduce a new perspective on credit assignment for agentic RL by attributing final outcomes according to cross-step dependencies, rather than temporal order alone. This perspective explicitly captures how information produced by earlier steps supports subsequent decisions and ultimately contributes to the final outcome. (2) We develop CreditFlow, which introduces a rule-based provenance tracking mechanism to efficiently uncover the dependencies from raw interaction trajectories and organize them into an explicit dependency graph. CreditFlow then propagates terminal feedback along valid dependency paths to derive dependency-aware step-level advantages, while requiring no additional LLM calls and integrating seamlessly with standard RL optimization. (3) We conduct extensive experiments on eight mainstream benchmarks spanning multi-hop question answering, scientific and medical reasoning, mathematics, and general agentic problem solving with Qwen2.5-7B-Instruct and Qwen3-8B. CreditFlow outperforms the strongest baseline by **5 points** on average, while reducing the average number of inference steps by over **20%**.

2 PRELIMINARIES

Multi-Step Agentic RL. We consider an LLM-based agent with policy π_θ that solves a task x through multiple interactions with external tools or environments. Before step t, the agent has an interaction history $h_{t-1} = (x, a_1, o_1, \ldots, a_{t-1}, o_{t-1})$, where a_i and o_i denote the action and observation at step i, respectively. At step t, the agent takes an action $a_t \sim \pi_\theta(\cdot \mid h_{t-1})$ and receives an observation o_t. The agent continues this interaction until it produces a final answer $y = o_T$. We denote the interaction trajectory as $\tau = (x, a_1, o_1, \ldots, a_T, o_T)$ and its terminal reward as $R(\tau) \in \mathbb{R}$.

Credit Assignment in Agentic RL. Credit assignment is challenging because a terminal reward $R(\tau)$ evaluates the trajectory as a whole, while different steps may contribute differently to the final answer. We use Group Relative Policy Optimization (GRPO) (Shao et al., 2024) as a representative example. For each input x, GRPO samples a group of G interaction trajectories $\{\tau^{(g)}\}_{g=1}^{G}$ with terminal rewards $\mathcal{R} = \{R^{(g)}\}_{g=1}^{G}$. It normalizes these rewards within the group to obtain the trajectory-level advantage $A_{\mathrm{tr}}^{(g)} = (R^{(g)} - \mathrm{mean}(\mathcal{R}))/(\mathrm{std}(\mathcal{R}) + \varepsilon)$, where ε is a small constant used to avoid a zero denominator. Standard outcome-level GRPO then uses the same $A_{\mathrm{tr}}^{(g)}$ to supervise all trainable outputs within $\tau^{(g)}$. Therefore, it distinguishes trajectories according to their final outcomes, but does not distinguish the contributions of different steps within the same trajectory. In this paper, we focus on this credit assignment stage, without changing the underlying optimization procedure.

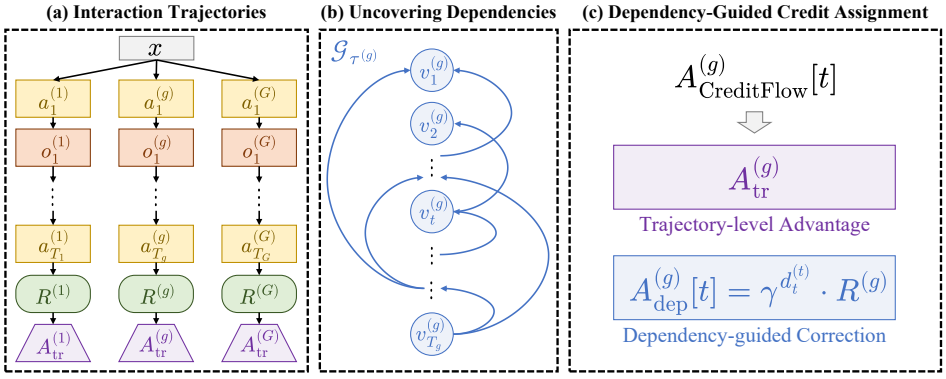


Figure 1: **Overview of CreditFlow.** (a) Agentic RL samples multiple interaction trajectories and computes trajectory-level advantages from their outcomes. (b) CreditFlow uncovers cross-step dependencies within each trajectory and organizes them into a dependency graph. (c) CreditFlow propagates terminal rewards along these dependencies to compute step-level corrections, which augment the original trajectory-level advantage for policy optimization. Appendix A shows concrete tool-use trajectories together with their dependency graphs.

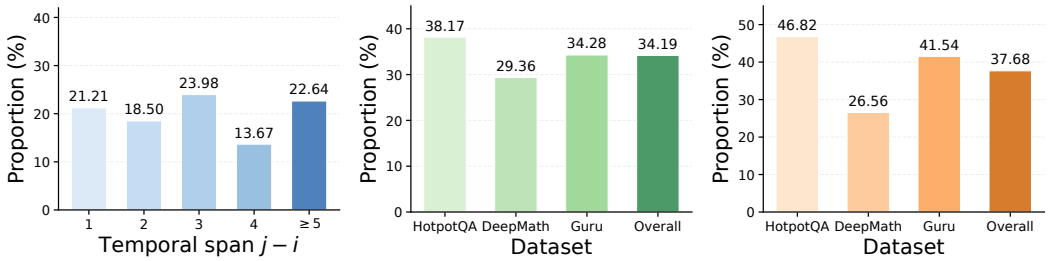
3 CreditFlow: Credit Assignment via Uncovering Dependencies

A useful way to view credit assignment in multi-step agentic RL is through two questions: *which steps actually support subsequent decisions and the final answer*, and *how terminal feedback should be assigned across these steps*. Existing methods increasingly provide finer-grained credit by estimating the contribution of individual steps, but typically do not explicitly model how information from one step supports later decisions. CreditFlow instead makes such cross-step dependencies explicit and uses them to guide credit assignment. Our method unfolds in three parts: **Section 3.1** first shows that temporal order poorly reflects how information flows across an interaction trajectory, motivating the dependency-guided credit assignment. **Section 3.2** then introduces a rule-based procedure to uncover cross-step dependencies and organizes them into a dependency graph. Finally, **Section 3.3** propagates terminal feedback along this graph to construct dependency-aware step-level advantages for policy optimization. **Figure 1** provides an overview of CreditFlow, while **Appendix A** shows concrete tool-use trajectories together with their dependency graphs.

3.1 Motivation: From Temporal Order to Cross-Step Dependencies

Key insight. Consider two steps $i < j$ in an interaction trajectory. Their temporal order only tells us that step i occurs before step j, but does not reveal whether step j actually uses the information obtained at step i. We instead focus on a stronger relation: step j *depends on* step i when the action a_j uses the observation o_i produced at step i. This relation directly captures how information flows across interaction steps. For example, although a trajectory may proceed through steps $1 \to 2 \to 3 \to 4 \to 5$, the final answer may rely only on information flowing through $1 \to 3 \to 5$, while the observations from steps 2 and 4 never affect subsequent decisions. Therefore, temporal proximity can assign credit to irrelevant steps while overlooking distant but essential ones. This motivates a simple principle behind CreditFlow: *we assign final credit according to cross-step dependencies rather than temporal order*.

Empirical motivation. We further examine how temporal order differs from the actual dependency structure in interaction trajectories. We analyze 256 interaction trajectories collected from the initial policy on HotpotQA, DeepMath-103K, and Guru-STEM, and record the dependencies among their intermediate steps. Figure 2 reveals three observations. **(1) Dependencies are frequently non-local.** For a dependency from step j to step i, we define its temporal span as $j - i$. As shown in Figure 2a, only 21.21% of dependency edges have a span of one, while 78.79% span more than one interaction step. Moreover, 22.64% span at least five steps. Thus, useful information often flows across distant interactions rather than only between adjacent steps. **(2) Temporal adjacency does not necessarily indicate dependency.** As shown in Figure 2b, only 34.1% of adjacent step pairs $(t, t+1)$ contain a dependency from the earlier observation to the later action. The ratios remain low across all three



(a) Temporal span of dependencies. (b) Dependencies among $(t, t+1)$. (c) Steps supporting the final answer.

Figure 2: **Temporal adjacency poorly reflects the dependency structure in interaction trajectories.** (a) Most dependencies span multiple interaction steps. (b) Only a minority of adjacent step pairs $(t, t+1)$ exhibit an actual dependency. (c) Only a subset of intermediate steps lie on dependency paths that reach the final answer.

datasets, ranging from 29% to 38%. Thus, temporal proximity alone cannot reliably identify how information flows through a trajectory. **(3) Only a subset of intermediate steps supports the final answer.** As shown in Figure 2c, only 37.68% of intermediate steps lie on dependency paths that eventually reach the final answer. Therefore, many steps occur along the interaction trajectory without directly supporting the final outcome. Together, these observations show that **temporal order provides a poor proxy for the dependency structure in interaction trajectories**. This motivates us to explicitly uncover cross-step dependencies before assigning credit. Figure 5 in Appendix A further illustrates how dependency structures can differ substantially from temporal execution order.

3.2 Uncovering Dependencies from Interaction Trajectories

Given the above motivation, the question is how to uncover the dependencies between steps from raw interaction trajectories. However, this is non-trivial for three reasons.

Challenge 1. Raw trajectories are highly redundant and noisy. Agent actions often mix reasoning, tool selection, arguments, and control instructions, while observations may contain lengthy retrieved passages, tool outputs, execution messages, and other incidental information. Such heterogeneous content makes it difficult to isolate the information that actually carries cross-step dependencies.

Challenge 2. Information availability does not imply dependency. A later action can access much of the preceding interaction history, but only a small subset of earlier observations may actually affect its decision. Therefore, temporal proximity or mere co-occurrence in the context cannot reliably reveal whether one step depends on another.

Challenge 3. Directly using LLMs to infer pairwise dependencies from raw trajectories would incur substantial additional inference cost and may hallucinate unsupported dependencies, which can further introduce erroneous credit paths. We provide empirical evidence for this concern in Section 4.3: replacing our rule-based construction with an external LLM to build the dependency graph consistently degrades performance.

To address these challenges, CreditFlow introduces a **rule-based provenance tracking mechanism** that dynamically traces how later decisions consume results produced by earlier steps, and organizes these provenance relations into a dependency graph. Specifically, each observation o_i is assigned a unique provenance identifier when it is generated. When the agent subsequently takes an action a_j, the interaction protocol records the identifiers of preceding observations whose results are actually used by a_j, yielding the provenance set $\mathrm{Use}(a_j)$. A deterministic parser and checker further validate every recorded reference against the current interaction history, ensuring that only existing and temporally valid results can instantiate dependencies.

Based on these validated provenance relations, we construct a dependency graph $\mathcal{G}_\tau = (\mathcal{V}_\tau, \mathcal{E}_\tau)$ for trajectory τ, where $\mathcal{V}_\tau = \{v_1, \ldots, v_T\}$. Each v_t represents interaction step t, consisting of action a_t and observation o_t. For any $i < j$, we introduce a directed dependency edge $(v_j, v_i) \in \mathcal{E}_\tau \iff o_i \in \mathrm{Use}(a_j)$, and analogously connect v_T to v_i when the final answer uses o_i. In this way, CreditFlow

recovers the dependency structure directly from explicit producer-consumer provenance during interaction, rather than inferring it post hoc from temporal proximity or raw trajectory semantics.

Discussion. We highlight two properties to better situate our approach.

- **Relation to existing graph-based credit assignment.** Graph structures have also been explored for credit assignment in multi-step agentic reasoning. For example, SALT (Li et al., 2026a) constructs trajectory graphs by identifying shared and divergent steps across multiple rollouts, while GraphGPO (Cheng et al., 2026) aggregates rollouts into a state-transition graph and estimates step-level credit according to progress toward the task goal. However, these precedents do not diminish the novelty of CreditFlow. Their graphs primarily organize trajectory or state-transition structures, whereas our graph explicitly represents *cross-step dependencies*: an edge specifies that a downstream action depends on an observation produced by an earlier step. This dependency semantics enables CreditFlow to trace how intermediate results contribute to downstream decisions and route credit accordingly.
- **Rule-based extraction is both efficient and reliable.** CreditFlow constructs the dependency graph directly from interaction records using deterministic rules, without additional LLM calls for dependency inference. This introduces negligible extraction overhead compared with pairwise LLM-based inference. More importantly, it avoids hallucinated dependencies that could create spurious credit paths and corrupt the resulting training signals.

3.3 Dependency-Guided Credit Assignment

After uncovering the underlying dependency structure, we use it for credit assignment and compute the corresponding step-level advantages. We take GRPO with binary rewards (1 for success and 0 for failure) as a representative example, while the same formulation naturally extends to other RL algorithms and reward settings. For each input x, suppose we sample a group of G interaction trajectories $\{\tau^{(g)}\}_{g=1}^{G}$ with terminal rewards $\mathcal{R} = \{R^{(g)}\}_{g=1}^{G}$. Then, the trajectory-level advantage is $A_{\text{tr}}^{(g)} = (R^{(g)} - \text{mean}(\mathcal{R}))/(\text{std}(\mathcal{R}) + \varepsilon)$. However, such trajectory-level advantages have two limitations. First, as noted in prior work, trajectory-level advantages overlook the heterogeneous contributions of different steps. Second, when all G trajectories are successful, $A_{\text{tr}}^{(g)}$ becomes zero and provides no further learning signal. However, even in this case, some steps contribute more directly to the final outcome than others and should still be preferentially reinforced. Distinguishing these steps can direct the agent to focus more on critical steps and avoid unnecessary interactions, thereby improving reasoning efficiency.

To this end, we introduce a dependency-aware correction term for each step. Given a trajectory $\tau^{(g)}$ and the corresponding dependency graph $\mathcal{G}_{\tau^{(g)}}$ with the node set $\mathcal{V}_{\tau}^{(g)} = (v_1^{(g)}, \ldots, v_{T_g}^{(g)})$, we denote the shortest distance from $v_{T_g}^{(g)}$ to $v_t^{(g)}$ by $d_t^{(g)}$, $t = 1, \ldots, T_g$. Note that $d_{T_g}^{(g)} = 0$ and $d_t^{(g)} = +\infty$ whenever there is no directed path from $v_{T_g}^{(g)}$ to $v_t^{(g)}$. Then, we assign each step t a correction term:

$$A_{\text{dep}}^{(g)}[t] = \gamma^{d_t^{(g)}} \cdot R^{(g)}, \tag{1}$$

where $\gamma \in (0, 1)$ is a decay factor. Finally, CreditFlow assigns the following advantage to step t:

$$A_{\text{CreditFlow}}^{(g)}[t] = A_{\text{tr}}^{(g)} + \lambda A_{\text{dep}}^{(g)}[t], \tag{2}$$

where λ is a hyperparameter. Then, we directly use $A_{\text{CreditFlow}}^{(g)}[t]$ to compute the policy loss and gradients for step t, without modifying the underlying optimization procedure.

Discussion. The hyperparameter λ balances the trajectory-level supervision and the dependency-aware step-level credit. When all trajectories in a group are successful, we have $A_{\text{tr}}^{(g)} = 0$, and $A_{\text{dep}}^{(g)}[t]$ dominates the resulting advantage. This allows CreditFlow to continue distinguishing and reinforcing critical steps even when the trajectory-level signal becomes useless. However, we still retain $A_{\text{tr}}^{(g)}$, because steps outside the dependency paths may still provide useful learning signals. Therefore, the two terms provide complementary supervision: $A_{\text{tr}}^{(g)}$ differentiates trajectories according to their overall outcomes, while $A_{\text{dep}}^{(g)}[t]$ differentiates steps according to their contributions within each trajectory. Moreover, λ provides a smooth interpolation between the trajectory-level learning and dependency-aware learning, with $\lambda = 0$ recovering the advantage in the original RL algorithm.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTINGS

Tasks and Datasets. We focus on enhancing the agents' overall capabilities across four domains—open domain knowledge, mathematics, science and medicine, and general problem solving. We construct a mixed dataset of 4,000 samples from HotpotQA (Yang et al., 2018), Guru (Cheng et al., 2025), and DeepMath-103K (He et al., 2025). For HotpotQA, we sample 500 easy, 1,000 medium, and 500 hard training samples. For the STEM split of Guru, we use the 800 problems with the highest pass rates under Qwen2.5-7B-Instruct as easy samples and randomly sample 200 from the remaining set as hard samples. For DeepMath-103K, we use its difficulty annotations to select the 800 easiest problems and randomly sample 200 harder ones. The training environment provides Python, Wikipedia search, and offline web search tools for program-aided reasoning and retrieval-based evidence gathering. Further details are provided in Appendix B.1.

Baselines. We compare CreditFlow with several representative baselines. For *agentic RL*, **Search-R1** (Jin et al., 2025) trains reasoning-and-search agents with outcome rewards over multi-turn retrieval trajectories; **AgentFlow** (Li et al., 2026b) optimizes the planner within a modular planner-executor agent system through in-the-flow RL; and **ARPO** (Dong et al., 2026) uses entropy-guided adaptive rollouts and advantage attribution to optimize multi-turn tool interactions. For *fine-grained and structure-aware RL*, **SALT** (Li et al., 2026a) constructs trajectory graphs to derive step-level advantages from outcome rewards; **TreeRL** (Hou et al., 2025) integrates on-policy tree search with intermediate supervision and branches from high-uncertainty reasoning steps; and **GraphGPO** (Cheng et al., 2026) aggregates rollout trajectories into a state-transition graph and assigns edge-level advantages according to progress toward the task goal. Further details are provided in Appendix B.2.

Benchmarks and Evaluation Protocol. We evaluate CreditFlow on benchmarks covering multi-hop question answering, scientific and medical reasoning, mathematical reasoning, and general agentic problem solving. Specifically, we use HotpotQA (Yang et al., 2018), MuSiQue (Trivedi et al., 2022), and 2Wiki (Ho et al., 2020) for multi-hop evidence retrieval and aggregation; GPQA-diamond (Rein et al., 2023) and MedQA (Jin et al., 2021) for domain-specific scientific and medical reasoning; AIME24 (Mathematical Association of America, 2024) and AMC23 (Mathematical Association of America, 2023) for mathematical reasoning; and GAIA (Mialon et al., 2023) for open-ended tool-use problem solving. We evaluate each method five times under the same evaluation setting and report the mean performance across the five runs. Further details are provided in Appendix B.3.

Training Details. We train Qwen2.5-7B-Instruct and Qwen3-8B using our implementation built upon verl (Sheng et al., 2025) and AgentFlow (Li et al., 2026b) frameworks, with engineering modifications to support our training pipeline. As verified in Section 4.4, the performance improvements do not arise from the underlying training framework or implementation choices, but from the proposed credit-assignment method itself. We train the models for one epoch on the dataset. For each query, we sample four trajectories with a decoding temperature of 1.0. We use a learning rate of 5×10^{-7}. The maximum model output length is 8K tokens, and each trajectory is limited to at most 8 interaction rounds. Further details are provided in Appendix B.4.

4.2 MAIN RESULTS

CreditFlow achieves consistent performance improvements across diverse reasoning tasks. Table 1 summarizes the overall performance of CreditFlow and baselines across diverse reasoning tasks. We draw the following conclusions. (1) CreditFlow consistently achieves the best overall performance across both Qwen2.5-7B-Instruct and Qwen3-8B, improving the average score from 43.4 to 48.6 and from 51.3 to 56.0 over AgentFlow, respectively. These improvements demonstrate the effectiveness of dependency-aware credit assignment in identifying and reinforcing intermediate decisions that contribute to final outcomes. (2) Among existing agentic RL approaches, AgentFlow achieves strong performance by optimizing agent planning and tool-use behaviors, while methods such as TreeRL, SALT, GraphGPO, and ARPO provide finer-grained supervision through tree search, trajectory or state-transition structures, and step-level advantage estimation. However, these methods primarily derive credit from trajectory structure, state transitions, branching behavior, or estimated step quality, rather than explicitly modeling whether a downstream decision *depends on* the result produced by an earlier step. In contrast, CreditFlow directly captures such cross-step dependencies and propagates outcome feedback along valid dependency paths, enabling credit assignment to

Table 1: **Performance and inference efficiency of CreditFlow and baselines.** Results are reported with Qwen2.5-7B-Instruct and Qwen3-8B. Each entry reports the task success score, while the last column reports the average inference steps. GPQA refers to the GPQA-diamond subset. Base and Search-R1 do not report average inference steps because they do not adhere to the same multi-step agent interaction protocol as other baselines. All performance scores are scaled to the range $[0, 100]$ and reported with one decimal place.

Methods	HotpotQA	MuSiQue	2Wiki	GPQA	MedQA	AIME24	AMC23	GAIA	*Avg.*	*Avg. Steps*
# Qwen2.5-7B-Instruct										
Base	15.6±2.1	4.4±1.8	23.4±2.2	30.3±2.0	60.8±2.4	10.0±1.9	35.0±2.3	3.1±1.6	*22.8*	–
Search-R1	45.8±1.9	18.0±1.8	31.5±2.0	34.5±1.9	62.0±2.0	14.0±1.8	45.0±2.1	8.5±1.7	*32.4*	–
TreeRL	45.8±2.0	18.2±1.9	38.5±2.1	35.6±2.0	62.5±2.1	15.0±1.8	48.0±2.2	9.8±1.7	*34.2*	*6.3*
SALT	46.5±1.9	18.8±1.8	40.2±2.0	36.8±1.9	64.0±2.0	16.7±1.8	50.0±2.1	11.5±1.7	*35.6*	*6.6*
GraphGPO	49.2±1.9	20.1±1.8	45.6±2.0	37.9±1.9	65.8±2.0	18.3±1.8	53.5±2.1	13.2±1.7	*38.0*	*6.5*
ARPO	50.0±1.8	20.8±1.8	47.0±1.9	38.5±1.9	66.5±1.9	20.0±1.8	54.9±2.0	14.0±1.7	*39.0*	*6.8*
AgentFlow	56.6±1.8	21.2±1.7	62.3±1.7	38.1±1.8	69.4±1.8	**23.3**±1.8	55.0±1.9	21.3±1.7	*43.4*	*6.9*
CreditFlow	**61.4**±1.6	**26.2**±1.5	**71.4**±1.5	**43.0**±1.6	**77.7**±1.5	**24.4**±1.7	**60.0**±1.7	**24.9**±1.5	★***48.6***	★***4.9***
# Qwen3-8B										
Base	22.4±2.0	10.8±1.8	16.2±2.1	38.4±2.0	66.0±2.3	16.7±1.9	45.0±2.2	5.5±1.7	*27.6*	–
Search-R1	50.5±1.8	21.0±1.7	38.5±1.9	39.0±1.8	70.0±1.9	23.3±1.8	52.5±2.0	13.8±1.7	*38.6*	–
TreeRL	51.0±1.9	21.0±1.8	42.0±2.0	39.0±1.9	67.0±2.0	23.3±1.9	52.5±2.1	13.5±1.7	*38.7*	*6.2*
SALT	54.0±1.8	21.8±1.8	43.5±1.9	38.5±1.9	68.5±2.0	25.0±1.9	55.0±2.0	15.0±1.7	*40.2*	*6.1*
GraphGPO	55.0±1.8	23.0±1.7	48.5±1.9	41.2±1.8	72.0±1.9	26.7±1.8	58.0±2.0	18.0±1.7	*42.8*	*5.9*
ARPO	56.0±1.8	23.8±1.7	50.0±1.9	42.0±1.8	73.5±1.9	28.3±1.8	60.0±2.0	19.5±1.6	*44.1*	*6.5*
AgentFlow	64.2±1.7	26.8±1.6	77.6±1.5	40.4±1.7	81.4±1.6	26.7±1.7	62.5±1.8	30.7±1.6	*51.3*	*6.4*
CreditFlow	**69.6**±1.5	**27.8**±1.4	**80.8**±1.3	**45.4**±1.5	**84.2**±1.4	**33.3**±1.6	**70.0**±1.5	**37.0**±1.4	★***56.0***	★***5.0***

reflect how intermediate results actually support subsequent decisions and the final outcome. (3) The consistent improvements of CreditFlow across different domains indicate that dependency-aware credit assignment provides a general mechanism for improving agentic reasoning rather than being limited to specific task categories.

CreditFlow improves reasoning efficiency with fewer inference steps. Table 1 further compares the inference efficiency of different methods. With Qwen2.5-7B-Instruct and Qwen3-8B, CreditFlow requires only 4.9 and 5.0 average inference steps, respectively, compared with 6.9 and 6.4 steps for AgentFlow. This corresponds to reductions of 29.0% and 21.9%, respectively, while achieving higher task success scores. These results suggest that dependency-aware credit assignment helps agents avoid redundant interactions and learn more efficient reasoning trajectories.

4.3 Understanding Dependency-Aware Credit Assignment

Dependency Graph Validation. To evaluate the reliability of the dependency graphs constructed by CreditFlow, we conduct a human validation study on randomly sampled interaction trajectories from the training set. Specifically, we randomly select 50 trajectories from each training source, resulting in 150 trajectories in total. Three trained PhD-level annotators independently verify whether the dependency relations extracted by CreditFlow correspond to actual information usage between intermediate steps. For precision evaluation, annotators examine all extracted dependency edges and determine whether the downstream action actually uses information from the upstream observation. For recall evaluation, we randomly sample an equal number of non-edge step pairs and ask annotators whether a dependency relation exists between them. The final annotations are obtained through majority voting among the three annotators.

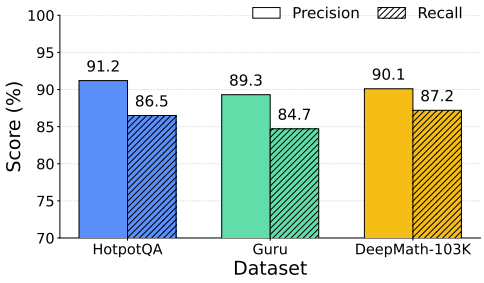


Figure 3: **Human validation of dependency graphs.** We evaluate the precision and recall of extracted dependency relations on three training datasets. Precision measures whether extracted edges correspond to valid dependency relations, while recall is estimated by examining randomly sampled candidate step pairs.

Table 2: **Effect of different dependency graph construction strategies.** We compare different strategies for constructing dependency graphs while keeping the same downstream credit assignment and policy optimization procedures. Random first constructs dependency graphs using CreditFlow and then randomly shuffles the two endpoint node IDs of every edge, preserving graph density while removing semantic dependency information. LLM-based construction infers dependency relations from interaction trajectories using Qwen3-8B and Qwen3.5-35B-A3B.

Method	HotpotQA	MuSiQue	2Wiki	GPQA	MedQA	AIME24	AMC23	GAIA	Avg
Random	66.6	26.0	77.2	41.4	78.8	23.3	62.5	29.1	50.6
LLM (8B)	68.1	26.8	78.0	43.0	80.1	26.7	67.5	30.2	52.6
LLM (35B)	69.4	27.6	79.4	45.9	81.6	30.0	72.5	31.4	54.7
CreditFlow	**69.6**	**27.8**	**80.8**	**45.4**	**84.2**	**33.3**	**70.0**	**37.0**	**56.0**

Figure 3 presents the validation results. CreditFlow achieves consistently high precision and recall across all three datasets, with precision above 89% and recall above 84%. These results demonstrate that the rule-based extraction procedure can reliably identify dependency relations in multi-step agent trajectories without requiring additional LLM inference.

Table 3: **Training overhead of different dependency graph construction strategies.** We report the average cost per training step. "No Graph" corresponds to standard GRPO without dependency graphs. "Tokens / Step" denotes the total token consumption, including additional graph construction tokens when applicable. "Graph Tokens / Step" denotes the additional tokens used for dependency graph construction.

Method	Time / Step (s)	Tokens / Step	Graph Tokens / Step
No Graph	941.7±12.4	8715K±84K	0K
Random	932.3±11.8	8652K±79K	0K
LLM (35B)	1042.1±14.6	10542K±96K	1990K±31K
CreditFlow	937.4±12.1	**8431K±76K**	0K

Effect and Cost of Dependency Graph Construction. Using Qwen3-8B, we compare CreditFlow with three alternatives: (1) ***No Graph***, i.e., standard GRPO without dependency-aware credit assignment; (2) ***Random***, which preserves graph density but randomly shuffles edge endpoints to destroy semantic dependencies; and (3) ***LLM-based construction***, which infers dependency relations from trajectories using Qwen3-8B or Qwen3.5-35B-A3B.

Tables 2 and 3 show that CreditFlow achieves the best performance, outperforming both *Random* and LLM-generated graphs. Although a larger LLM improves graph quality, it still underperforms our rule-based construction, supporting the reliability of provenance-based dependency extraction. Meanwhile, CreditFlow incurs negligible overhead relative to *No Graph*, whereas LLM-based construction increases training time by about 12%. It also reduces token consumption from 8715K to 8431K per training step, consistent with the improved reasoning efficiency of the trained agent.

4.4 Ablation Studies

To better understand the source of performance improvements and the robustness of CreditFlow, we conduct ablations on both framework design and hyperparameter configurations with Qwen3-8B. Specifically, we compare CreditFlow with our base framework using GRPO, where the framework is developed upon AgentFlow (Li et al., 2026b) with additional engineering modifications. This comparison examines whether the improvements mainly come from dependency-aware credit assignment rather than framework-level enhancements. We further evaluate the sensitivity of CreditFlow to the key hyperparameters λ and γ. In our experiments, λ is selected from $\{0.01, 0.02, 0.04\}$ and γ is selected from $\{0.75, 0.85, 0.95\}$, with the default setting of $\lambda = 0.02$ and $\gamma = 0.85$.

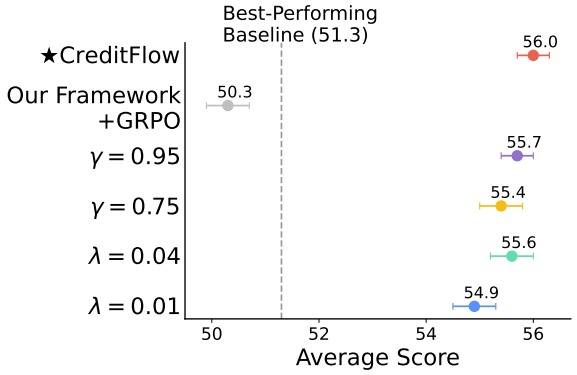


Figure 4: **Ablation study on framework design and hyperparameter sensitivity.** "Framework + GRPO" denotes our base framework built upon AgentFlow with engineering modifications but without dependency-aware credit assignment.

These ranges are not obtained through extensive hyperparameter search. Instead, we choose λ values that roughly maintain the dependency-aware advantage component $A^{(g)}_{\text{dep}}$ within 20%-50% of the overall advantage, balancing dependency-based and original credit signals. The selected γ values cover commonly used discount factors for credit propagation, representing different decay rates.

Figure 4 summarizes the results. We draw the following conclusions. (1) The performance improvement of CreditFlow mainly comes from dependency-aware credit assignment rather than framework-level modifications. Our base framework with GRPO achieves an average score of 50.3, which is below the best-performing baseline (i.e., AgentFlow) with a score of 51.3. In contrast, CreditFlow achieves an average score of 56.0, outperforming AgentFlow by 4.2 points. This demonstrates that explicitly modeling dependency relations provides substantial benefits beyond engineering improvements to the underlying framework. (2) CreditFlow remains effective under different hyperparameter configurations. Across different values of λ and γ, all variants achieve comparable performance to the default setting and consistently outperform AgentFlow. In particular, λ controls the relative contribution of dependency-aware advantages, where a smaller value may limit the influence of dependency information while a larger value may introduce excessive dependency signals. Similarly, γ controls the decay rate of propagated credits, balancing the contribution of short-range and long-range dependencies. The stable performance across these configurations indicates that CreditFlow does not rely on delicate hyperparameter tuning and can robustly benefit from dependency-aware credit assignment.

5 Related Work

We provide a concise overview here and defer a more detailed discussion to Appendix C.

Agentic Reinforcement Learning. Reinforcement learning has become a major paradigm for improving the multi-step reasoning and tool-use capabilities of LLM-based agents (Zhang et al., 2025b; Ke et al., 2025; Singh et al., 2025). While early approaches largely optimize complete trajectories using outcome-level supervision (Shao et al., 2024; Jin et al., 2025), recent methods increasingly target long-horizon agent interactions with improved exploration, planning, and finer-grained optimization (Li et al., 2026b; Dong et al., 2026). As interaction trajectories become longer and more complex, accurately translating delayed outcomes into learning signals for intermediate decisions has therefore become increasingly important.

Credit Assignment for Agentic RL. Recent studies move beyond uniform trajectory-level supervision by deriving finer-grained credit from process or progress estimation, retrospective critics, verifiable intermediate feedback, counterfactual or information-theoretic signals, and structural representations of trajectories (Wang et al., 2025; Choudhury, 2025; Tan et al., 2026; Zhang et al., 2026b; Yuan et al., 2026; Xu et al., 2026; Wang et al., 2026; Meng & Chen, 2026; Hou et al., 2025; Li et al., 2026a; Cheng et al., 2026). Our work follows this direction but introduces a different basis for credit assignment. Rather than estimating the quality of individual temporally indexed steps or organizing rollouts according to branching, trajectory, or state-transition structure, CreditFlow explicitly models **cross-step dependencies**: whether a downstream decision actually uses a result produced by an earlier step. This dependency semantics reveals how information flows through an interaction trajectory and enables terminal feedback to be propagated along the paths that genuinely support subsequent decisions and the final outcome. Moreover, CreditFlow uncovers these dependencies through rule-based provenance tracking, providing explicit and verifiable credit paths without additional LLM inference.

6 Conclusion

We present CreditFlow, a dependency-aware credit assignment method for agentic reinforcement learning. Instead of assigning credit primarily according to temporal order or isolated step quality, CreditFlow explicitly models how intermediate results are used by later decisions and propagates terminal feedback along these cross-step dependencies. To efficiently uncover such dependencies, we introduce a rule-based provenance tracking mechanism that constructs explicit dependency graphs with negligible additional overhead and without extra LLM inference. Experiments across diverse agentic reasoning benchmarks demonstrate that this dependency-based perspective consistently improves both reasoning performance and inference efficiency. Overall, our results suggest that explicitly modeling how information flows across interaction steps provides a promising direction for more accurate and efficient credit assignment in long-horizon agentic RL.

AI USE STATEMENT

We used generative AI tools to polish the writing and improve the readability of the manuscript. All AI-assisted edits were reviewed by the authors, who take full responsibility for the final content of the paper. We note that the LLM-based agents trained and evaluated in this work are themselves AI systems; such model use is intrinsic to the subject of this research and is distinct from the authors' use of generative AI in preparing the manuscript.

ETHICS STATEMENT

We adhere to the ICLR Code of Ethics. This work studies algorithmic credit assignment using established research datasets and offline corpora and does not collect personal or sensitive user data. Human involvement is limited to trained annotators who evaluate dependency relations in sampled model interaction trajectories for aggregate validation; these annotations are not used to make decisions about individuals. As with other general-purpose tool-using agents, improved agentic capabilities may introduce dual-use or reliability risks when deployed with unsafe tools or excessive autonomy. Practical deployment should therefore employ task-appropriate access control, logging, monitoring, and human or domain-expert oversight in high-stakes settings.

REPRODUCIBILITY STATEMENT

We support reproducibility by providing the complete formulation of CreditFlow and its dependency-graph construction in Section 3, together with the experimental settings in Section 4.1. Appendix B.1 describes the training data and offline tool environment, Appendix B.2 details the baselines, Appendix B.3 specifies the benchmarks and evaluation protocol, and Appendix B.4 provides the training configurations for CreditFlow. Section 4.3 further reports the dependency-graph validation and construction comparisons, while Section 4.4 provides the ablation and hyperparameter-sensitivity settings. Together, these sections document the data selection, tool access, model backbones, optimization and rollout configurations, evaluation protocol, and robustness analyses used in our experiments.

REFERENCES

Anthropic. The future of ai at work: Introducing cowork. https://www.anthropic.com/webinars/future-of-ai-at-work-introducing-cowork, January 2026. Accessed: 2026-09-17.

Pengfei Cao, Tianyi Men, Wencan Liu, Jingwen Zhang, Xuzhao Li, Xixun Lin, Dianbo Sui, Yanan Cao, Kang Liu, and Jun Zhao. Large language models for planning: A comprehensive and systematic survey. *arXiv preprint arXiv:2505.19683*, 2025.

Qikai Chang, Zhenrong Zhang, Pengfei Hu, Jun Du, Jiefeng Ma, Yicheng Pan, Jianshu Zhang, Quan Liu, and Gao Jianqing. Thor: Tool-integrated hierarchical optimization via rl for mathematical reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 21487–21510, 2026.

Xin Cheng, Shuo He, Lang Feng, HaiYang Xu, Ming Yan, Lei Feng, and Bo An. Beyond trajectory-level attribution: Graph-based credit assignment for agentic reinforcement learning. *arXiv preprint arXiv:2605.26684*, 2026.

Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Yonghao Zhuang, Nilabjo Dey, Yuheng Zha, Yi Gu, Kun Zhou, Yuqi Wang, Yuan Li, Richard Fan, Jianshu She, Chengqian Gao, Abulhair Saparov, Haonan Li, Taylor W. Killian, Mikhail Yurochkin, Zhengzhong Liu, Eric P. Xing, and Zhiting Hu. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective, 2025. URL https://arxiv.org/abs/2506.14965.

Sanjiban Choudhury. Process reward models for llm agents: Practical framework and directions. *arXiv preprint arXiv:2502.10325*, 2025.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. In *International Conference on Learning Representations*, volume 2026, pp. 16981–17017, 2026.

Dyke Ferber, Lars Hilgers, Christiane Höper, Benedict Kinny-Köster, Jan-Niklas Eckardt, Katharina Egger-Heidrich, Marius Bill, Martin MK Schneider, Jan Clusmann, Lejla Kadric, et al. Towards autonomous medical artificial intelligence agents. *Nature*, pp. 1–10, 2026.

Ali Essam Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J Szostkiewicz, Dmytro Shved, Gavin J Gyimesi, Jon M Laurent, Samantha M Wright, Muhammed T Razzak, et al. A multi-agent system for automating scientific discovery. *Nature*, pp. 1–3, 2026.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. 2025. URL https://arxiv.org/abs/2504.11456.

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

Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. Treerl: Llm reinforcement learning with on-policy tree search. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12355–12369, 2025.

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

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, Peifeng Wang, Silvio Savarese, et al. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *arXiv preprint arXiv:2504.09037*, 2025.

Jiazheng Li, Yawei Wang, Qiaojing Yan, Yijun Tian, Zhichao Xu, Huan Song, Panpan Xu, and Lin Lee Cheong. Salt: Step-level advantage assignment for long-horizon agents via trajectory graph. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 4709–4725, 2026a.

Xinzhe Li. A review of prominent paradigms for llm-based agents: Tool use, planning (including rag), and feedback learning. In *Proceedings of the 31st international conference on computational linguistics*, pp. 9760–9779, 2025.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. In *International Conference on Learning Representations (ICLR)*, 2026b.

Yuze Liu, Chaochao Lu, and Chao Yang. Shadow: Dynamic-aware credit assignment against long-horizon tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23935–23944, 2026.

Xinji Mai, Haotian Xu, Weinong Wang, Yingying Zhang, Wenqiang Zhang, et al. Agentic rl scaling law: Spontaneous code execution for mathematical problem solving. *Advances in Neural Information Processing Systems*, 38:7325–7340, 2026.

Mathematical Association of America. 2023 american mathematics competitions (amc). https://maa.org/subject/amc/, 2023. Accessed: 2026-05-07.

Mathematical Association of America. American invitational mathematics examination (aime). https://maa.org/maa-invitational-competitions/, 2024. Accessed: 2026-05-07.

Zibin Meng and Kani Chen. Craft: Counterfactual credit assignment from free sibling rollouts for self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2606.29476*, 2026.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*, 2023.

OpenAI. Chatgpt is now a partner for your most ambitious work. `https://openai.com/index/chatgpt-for-your-most-ambitious-work/`, July 2026. Accessed: 2026-09-17.

Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, Olivier Pietquin, and Laura Toni. A survey of temporal credit assignment in deep reinforcement learning. *arXiv preprint arXiv:2312.01072*, 2023.

Qwen Team. Qwen code: An open-source ai coding agent. `https://github.com/QwenLM/qwen-code`, 2026. Accessed: 2026-09-17.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2023.

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

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.

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.

Aditya Bharat Soni, Boxuan Li, Xingyao Wang, Valerie Chen, and Graham Neubig. Coding agents with multimodal browsing are generalist problem solvers. In *Findings of the Association for Computational Linguistics: EACL 2026*, pp. 6052–6069, 2026.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon llm agents. *arXiv preprint arXiv:2603.08754*, 2026.

Zhengwei Tao, Haiyang Shen, Baixuan Li, Wenbiao Yin, Jialong Wu, Kuan Li, Zhongwang Zhang, Huifeng Yin, Rui Ye, Yong Jiang, et al. Empowering efficiency and efficacy in webagent via enabling info-rich seeking. In *International Conference on Learning Representations*, volume 2026, pp. 133034–133055, 2026.

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

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, et al. Information gain-based policy optimization: A simple and effective approach for multi-turn search agents. In *International Conference on Learning Representations*, volume 2026, pp. 104502–104529, 2026.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. Spa-rl: Reinforcing llm agents via stepwise progress attribution. *arXiv preprint arXiv:2505.20732*, 2025.

Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Dongqi Fu, et al. Agentic reasoning for large language models. *arXiv preprint arXiv:2601.12538*, 2026.

Yuanda Xu, Zhengze Zhou, Hejian Sang, Xiaomin Li, Jiaxin Zhang, Xinchen Du, Sen Na, Zhipeng Wang, and Alborz Geramifard. Triage: Role-typed credit assignment for agentic reinforcement learning. *arXiv preprint arXiv:2606.32017*, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Huining Yuan, Zelai Xu, Huaijie Wang, Xiangmin Yi, Jiaxuan Gao, Xiao-Ping Zhang, Yu Wang, Chao Yu, and Yi Wu. Verifiable process rewards for agentic reasoning. *arXiv preprint arXiv:2605.10325*, 2026.

Chenchen Zhang. From reasoning to agentic: Credit assignment in reinforcement learning for large language models. *arXiv preprint arXiv:2604.09459*, 2026.

Ding-Chu Zhang, Yida Zhao, Jialong Wu, Liwen Zhang, Baixuan Li, Wenbiao Yin, Yong Jiang, Yu-Feng Li, Kewei Tu, Pengjun Xie, et al. Evolvesearch: An iterative self-evolving search agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 13134–13147, 2025a.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, et al. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*, 2025b.

Guibin Zhang, Haiyang Yu, Kaiming Yang, Bingli Wu, Fei Huang, Yongbin Li, and Shuicheng Yan. Evoroute: Experience-driven self-routing llm agent systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 38213–38225, 2026a.

Yaocheng Zhang, Haohuan Huang, Zijun Song, Zijie Zhao, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. Criticsearch: Fine-grained credit assignment for search agents via a retrospective critic. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 12272–12290, 2026b.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025.

A Illustrative Examples of Cross-Step Dependencies

Figure 5 provides two concrete examples of how cross-step dependencies differ from temporal execution order. The dashed arrows show the execution sequence, while the solid colored arrows show dependencies from downstream steps to earlier steps whose results they use. Therefore, temporally executed steps do not necessarily lie on the dependency paths that support the final answer.

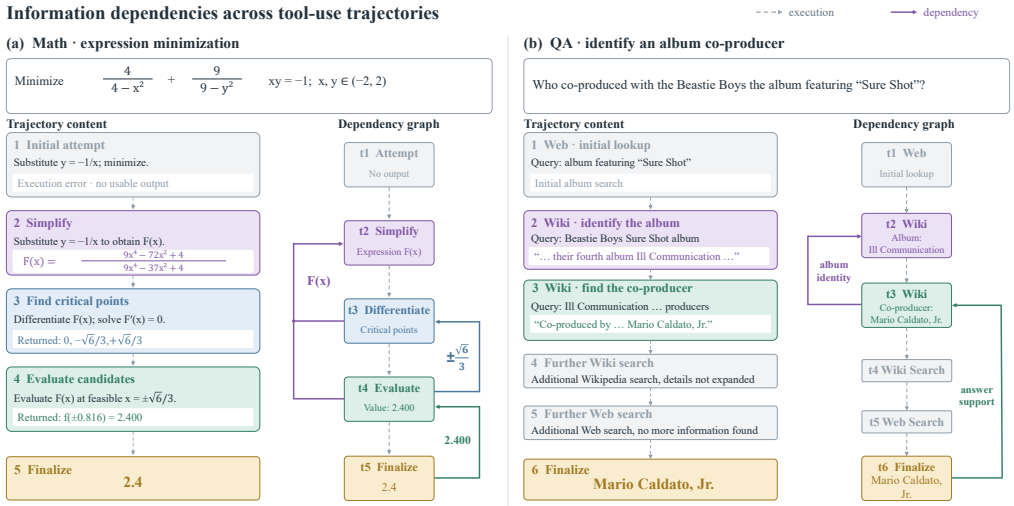


Figure 5: **Illustrative examples of cross-step dependencies in tool-use trajectories.** Dashed arrows show temporal execution order, while solid colored arrows show dependencies from downstream steps to earlier steps whose results they use. **(a)** In the math example, the initial attempt produces no usable result and does not support later steps. The final answer instead depends on the evaluated value, which relies on the critical points and the simplified expression. **(b)** In the QA example, the final answer depends on the retrieved co-producer, which in turn relies on identifying the album, while the other searches do not support the final answer.

B Experimental Details

B.1 Tasks and Datasets

We construct a mixed training set of 4,000 samples from HotpotQA (Yang et al., 2018), Guru (Cheng et al., 2025), and DeepMath-103K (He et al., 2025). These datasets cover multi-hop retrieval, STEM reasoning, and mathematical reasoning.

- HotpotQA is a Wikipedia-based multi-hop question answering dataset that requires models to retrieve and reason over multiple supporting documents. We sample 2,000 examples from its training split, including 500 easy, 1,000 medium, and 500 hard samples.
- Guru is a multi-domain reasoning dataset that covers areas such as mathematics, science, code, logic, simulation, and tabular reasoning. We use its STEM split. Following the difficulty measured by Qwen2.5-7B-Instruct, we select the 800 problems with the highest pass rates as easy samples and randomly sample 200 problems from the remaining set as hard samples.
- DeepMath-103K is a large-scale mathematical reasoning dataset with difficulty annotations. We select the 800 easiest problems according to these annotations and randomly sample another 200 problems from the remaining harder problems.

Tool Environment. During training, all three datasets use the same tool set: Python, Wikipedia search, and offline web search. Python supports program-aided reasoning, while the two search tools support retrieval-based evidence gathering. The offline web search tool retrieves from a fixed

corpus instead of accessing the open Internet during training. This setting prevents unrestricted online answer lookup while preserving multi-step retrieval and tool-use interactions.

B.2 Baselines and Implementations

We compare CreditFlow with representative methods covering outcome-based agentic RL, multi-turn agent optimization, and fine-grained or structure-aware credit assignment.

- **Search-R1** (Jin et al., 2025) trains an LLM to interleave reasoning with multiple search-engine queries through reinforcement learning. Rather than relying on supervised demonstrations of tool use, Search-R1 optimizes complete reasoning-and-retrieval trajectories using outcome-based rewards and masks retrieved contents from policy optimization. It therefore provides a representative outcome-supervised baseline for learning multi-turn retrieval behavior.
- **AgentFlow** (Li et al., 2026b) is an in-the-flow agentic RL framework that decomposes the agent system into specialized modules and directly optimizes the planner during multi-turn interactions. Its Flow-GRPO algorithm converts long-horizon agent optimization into turn-wise policy updates while using the verifiable trajectory-level outcome to supervise planner decisions across the interaction. AgentFlow serves as a strong agentic RL baseline that jointly captures planning and tool-use behaviors in a modular agent system.
- **ARPO** (Dong et al., 2026) is designed specifically for multi-turn tool-use reinforcement learning. Motivated by the observation that model uncertainty often increases after tool interactions, ARPO uses entropy-guided adaptive rollouts to allocate additional exploration to high-uncertainty interaction steps. It further introduces advantage attribution to capture differences among stepwise tool-use behaviors, providing finer-grained optimization than conventional trajectory-level RL.
- **TreeRL** (Hou et al., 2025) integrates on-policy tree search into reinforcement learning to improve both exploration and intermediate supervision. Instead of independently sampling complete reasoning chains, TreeRL branches from intermediate reasoning states and derives dense on-policy process supervision from the resulting search structure. To improve search efficiency, it preferentially branches from high-uncertainty intermediate steps rather than uniformly expanding the trajectory.
- **SALT** (Li et al., 2026a) is a lightweight step-level advantage assignment method for long-horizon agents. Given multiple completed trajectories sampled for the same prompt, SALT constructs a trajectory graph that captures shared and divergent trajectory structures and uses this graph together with outcome rewards to estimate the quality of individual steps. The resulting step-level advantages refine the original group-based RL signal without modifying the rollout procedure or requiring an additional critic model.
- **GraphGPO** (Cheng et al., 2026) performs graph-based credit assignment by aggregating multiple rollout trajectories into a unified state-transition graph. It estimates each state's progress toward the task goal using global information encoded in the graph and assigns an advantage to each transition according to how much it reduces the estimated distance to the goal. This enables edge-level credit assignment that distinguishes useful transitions even when they occur within trajectories with different final outcomes.

These baselines cover complementary approaches to improving agentic RL, including trajectory-level outcome optimization, adaptive exploration, intermediate supervision, and graph-based step-level credit assignment. In particular, SALT and GraphGPO explicitly exploit graph structures, making them important comparisons for CreditFlow. Their graphs primarily represent shared trajectory structures or state transitions across rollouts, whereas the dependency graph in CreditFlow has a different semantics: each edge explicitly denotes that a downstream action uses a result produced by an earlier interaction step. Accordingly, CreditFlow assigns credit according to the cross-step flow of intermediate results rather than trajectory topology or state-transition progress.

Implementation and Fairness. We implement each baseline following its standard training recipe and default configurations whenever applicable. For fair comparison, all trainable methods use the same backbone models, training data, tool environment, maximum interaction budget, and evaluation protocol. We keep method-specific components, such as rollout strategies, reward formulations,

credit-assignment mechanisms, and optimization configurations, consistent with their original designs rather than forcing a shared configuration across fundamentally different algorithms.

B.3 Benchmarks and Evaluation Protocol

We evaluate CreditFlow on eight benchmarks covering multi-hop question answering, scientific and medical reasoning, mathematical reasoning, and general agentic problem solving.

- For multi-hop question answering, we use HotpotQA (Yang et al., 2018), MuSiQue (Trivedi et al., 2022), and 2Wiki (Ho et al., 2020). HotpotQA requires reasoning over multiple supporting Wikipedia documents. MuSiQue contains multi-hop questions constructed to emphasize connected reasoning across multiple pieces of evidence. 2WikiMultiHopQA combines structured and unstructured knowledge and requires multi-step evidence aggregation.
- For scientific and medical reasoning, we use GPQA-diamond (Rein et al., 2023) and MedQA (Jin et al., 2021). GPQA-diamond contains graduate-level multiple-choice questions in biology, physics, and chemistry. MedQA contains multiple-choice questions collected from professional medical examinations.
- For mathematical reasoning, we use AIME24 (Mathematical Association of America, 2024) and AMC23 (Mathematical Association of America, 2023), which cover a broad range of competition mathematics problems.
- For general agentic problem solving, we use GAIA (Mialon et al., 2023). GAIA evaluates general-purpose assistants on tasks that require reasoning and tool use. We evaluate on its Level 1 and Level 2 tasks, which match the capabilities supported by our evaluation environment.

Evaluation Protocol. Following the evaluation practice in AgentFlow (Li et al., 2026b), we evaluate large benchmarks on fixed subsets sampled from their test sets. Specifically, we randomly sample 500 test examples from each of HotpotQA, MuSiQue, 2Wiki, and MedQA, and use the same sampled subset for all methods. We evaluate on the full GPQA-diamond, AIME24, and AMC23 benchmarks, and use all Level 1 and Level 2 tasks from GAIA. We report task success scores on a $[0, 100]$ scale for all benchmarks. During inference, we use stochastic decoding with temperature 0.7, top-$p = 0.8$, and top-$k = 20$. We allow at most 10 interaction rounds and set the maximum generation length of each model output to 8K tokens. For each benchmark, all methods use the same tool access, interaction budget, decoding configuration, and scoring protocol. We evaluate each method five times under the same evaluation setting and report the mean performance across the five runs.

B.4 Training Details

We train Qwen2.5-7B-Instruct and Qwen3-8B for one epoch using our implementation built upon verl (Sheng et al., 2025) and AgentFlow (Li et al., 2026b), with engineering modifications to support our training pipeline. For each training query, we sample four interaction trajectories with a decoding temperature of 1.0. We use a learning rate of 5×10^{-7}. Each trajectory contains at most 8 interaction rounds, and the maximum generation length of each model output is 8K tokens. All experiments are conducted on eight A100 GPUs.

C Related Work and Discussions

Agentic Reinforcement Learning. Reinforcement learning has recently emerged as an effective paradigm for improving the reasoning, planning, and tool-use capabilities of LLM-based agents (Zhang et al., 2025b; Ke et al., 2025; Singh et al., 2025). A common starting point is outcome-level optimization, where all decisions within a sampled trajectory are trained using supervision derived from the final task outcome. GRPO (Shao et al., 2024), for example, provides a simple critic-free framework based on relative rewards across groups of sampled responses, while Search-R1 (Jin et al., 2025) extends outcome-based RL to multi-turn reasoning-and-retrieval settings. Such methods are scalable and effective, but the final outcome alone does not distinguish the heterogeneous contributions of intermediate decisions. More recent work therefore develops optimization schemes tailored to long-horizon agentic interactions. AgentFlow (Li et al., 2026b) performs in-the-flow RL within a modular planner–executor architecture, directly optimizing planning decisions over

multi-step interactions. ARPO (Dong et al., 2026) combines entropy-guided adaptive rollout with advantage attribution to improve exploration and learning in multi-turn tool-use settings. These advances substantially strengthen agentic RL, while also making the credit assignment problem increasingly prominent: as trajectories interleave reasoning, tool invocation, and environmental feedback over many turns, the final reward becomes an increasingly coarse description of which intermediate decisions should be reinforced.

Fine-Grained Credit Assignment. A growing line of work addresses this limitation by constructing finer-grained supervision for intermediate steps. Process-based approaches estimate the quality or progress of individual decisions through process reward models or learned critics (Choudhury, 2025; Wang et al., 2025). Retrospective approaches use information available after the trajectory is completed to reassess earlier actions; for example, hindsight-based methods employ post-hoc critics to refine intermediate value or advantage estimates (Tan et al., 2026; Zhang et al., 2026b). Other methods exploit verifiable intermediate feedback or semantic roles to provide localized supervision (Yuan et al., 2026; Xu et al., 2026). These methods share the important goal of distinguishing useful and harmful decisions within the same trajectory, but their credit signals are primarily derived from the estimated quality, progress, correctness, role, or counterfactual contribution of individual temporally indexed steps. Another line of work derives credit through information-theoretic or counterfactual objectives (Wang et al., 2026; Meng & Chen, 2026). Rather than assigning the same outcome-level advantage to every step, these methods measure how particular intermediate decisions affect uncertainty or downstream outcomes. Although such approaches provide a finer view of contribution than trajectory-level supervision, they still primarily score individual decisions or transitions rather than explicitly representing which earlier results are consumed by which later decisions.

Structure-Aware and Graph-Based Credit Assignment. Several recent approaches further exploit non-linear structures to improve credit assignment. TreeRL (Hou et al., 2025) incorporates on-policy tree search and strategically branches from uncertain intermediate reasoning states, providing dense process supervision while improving exploration. SALT (Li et al., 2026a) constructs a trajectory graph from multiple rollouts of the same prompt and uses the resulting structure to estimate the quality of individual steps and derive step-level advantages. GraphGPO (Cheng et al., 2026) aggregates multiple rollout trajectories into a global state-transition graph and assigns edge-level advantages according to how transitions reduce the estimated distance to the task goal. Related structure-aware methods similarly exploit shared states, branching patterns, or trajectory topology to expose information unavailable from an isolated linear rollout (Liu et al., 2026). CreditFlow is closely related to this broader direction in that it also uses a graph to support finer-grained credit assignment, but the *semantics of the graph are fundamentally different*. Existing trajectory or state-transition graphs primarily encode how sampled rollouts branch, merge, revisit states, or progress toward a task goal. In contrast, an edge in the dependency graph of CreditFlow has an explicit producer–consumer meaning: it states that a downstream action *uses* an observation produced by an earlier interaction step. Consequently, two temporally adjacent steps need not be connected, while distant steps can be directly linked when information produced by one is consumed by the other. This distinction allows CreditFlow to model the actual flow of intermediate results within a trajectory rather than using structural proximity as a proxy for contribution. This dependency view also leads to a different credit-assignment mechanism. After uncovering cross-step dependencies, CreditFlow propagates terminal feedback only along valid dependency paths, so that intermediate decisions are reinforced according to whether their produced results support subsequent decisions and ultimately contribute to the final answer. Moreover, rather than invoking an additional LLM or learned critic to infer these relations, CreditFlow uses rule-based provenance tracking to recover them directly from interaction records. This construction introduces negligible additional inference overhead, produces explicit and verifiable dependency edges, and avoids hallucinated relations that could otherwise create spurious credit paths. Together, these properties distinguish CreditFlow from prior process-based, counterfactual, and graph-based credit-assignment methods while remaining complementary to their broader goal of providing more informative supervision for long-horizon agents.