

RLVMR: Reinforcement Learning with Verifiable **Meta-Reasoning** Rewards for Robust Long-Horizon Agents

Zijing Zhang, Ziyang Chen*, Mingxiao Li, Zhaopeng Tu*, and Xiaolong Li

Hunyuan AI Digital Human, Tencent

<https://github.com/Tencent/DigitalHuman/tree/main/RLVMR>

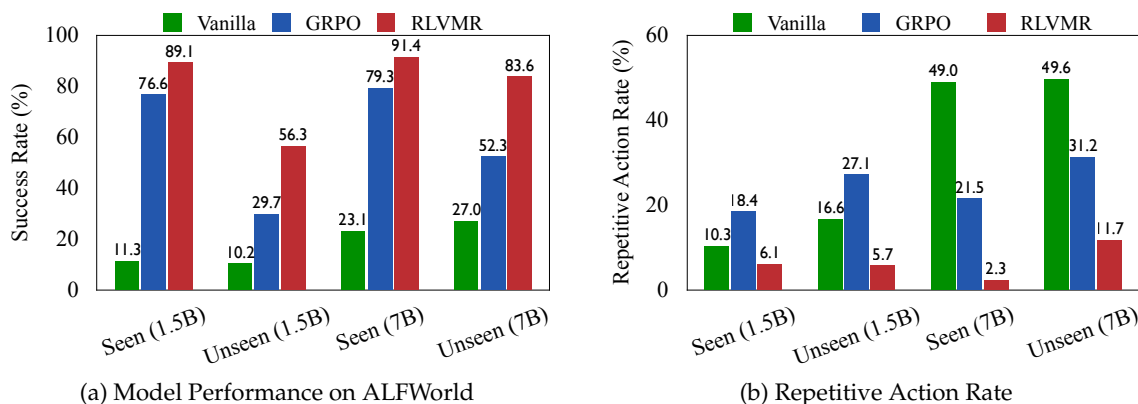


Figure 1: Reinforcement learning with outcome-only rewards (e.g., GRPO) improves performance over vanilla models but fosters inefficient exploration, characterized by high rates of repetitive actions that hinder generalization to unseen tasks. In contrast, our proposed RLVMR significantly improves success rates and generalization by directly mitigating this inefficient exploration.

Abstract

The development of autonomous agents for complex, long-horizon tasks is a central goal in AI. However, dominant training paradigms face a critical limitation: reinforcement learning (RL) methods that optimize solely for final task success often reinforce flawed or inefficient reasoning paths, a problem we term **inefficient exploration**. This leads to agents that are brittle and fail to generalize, as they learn to find solutions without learning *how* to reason coherently. To address this, we introduce **RLVMR**, a novel framework that integrates dense, process-level supervision into end-to-end RL by rewarding verifiable, meta-reasoning behaviors. RLVMR equips an agent to explicitly tag its cognitive steps—such as planning, exploration, and reflection—and provides programmatic, rule-based rewards for actions that contribute to effective problem-solving. These process-centric rewards are combined with the final outcome signal and optimized using a critic-free policy gradient method. On the challenging ALFWorld and ScienceWorld benchmarks, RLVMR achieves new state-of-the-art results, with our 7B model reaching an 83.6% success rate on the most difficult unseen task split. Our analysis confirms these gains stem from improved reasoning quality, including significant reductions in redundant actions and enhanced error recovery, leading to more robust, efficient, and interpretable agents.

*Correspondence to: Zhaopeng Tu <zptu@tencent.com> and Ziyang Chen <willzychen@tencent.com>.

1 Introduction

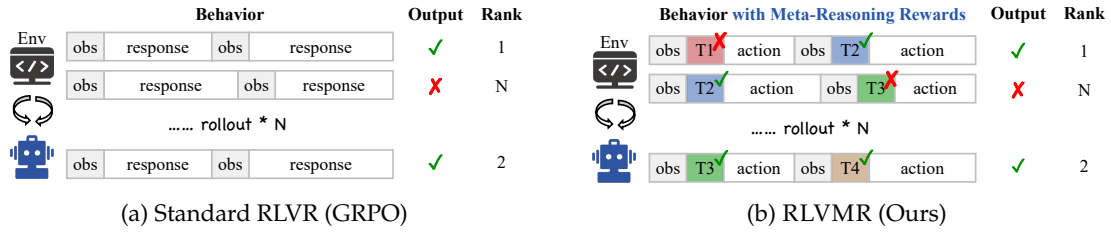


Figure 2: Comparison of LLM agent RL training paradigms: (a) Standard RL with outcome-only rewards (e.g., GRPO) inadvertently reinforces trajectories with inefficient or illogical intermediate reasoning steps. (b) Our RLVMR approach provides dense, verifiable rewards for beneficial meta-reasoning behaviors (e.g., T1-T4), directly shaping a more robust and coherent reasoning process.

The quest to build autonomous agents capable of solving complex, long-horizon tasks has gained significant momentum with the rise of Large Language Models (LLMs) (Zeng et al., 2024; Wang et al., 2022; Bai et al., 2024). However, dominant training paradigms face a fundamental trade-off. On one hand, Supervised Fine-Tuning (SFT) on expert trajectories can teach agents efficient behaviors, but these policies are often brittle and fail to generalize to novel situations (Chu et al., 2025). On the other hand, Reinforcement Learning (RL) from environmental feedback encourages exploration and can lead to better generalization, but it typically optimizes for a single, sparse reward signal: final task success.

This reliance on outcome-only rewards raises a critical, yet underexplored question: **Are agents learning to reason coherently, or are they just finding brittle shortcuts to success?** Our work investigates a pervasive issue we term **inefficient exploration**, where agents are rewarded for successful outcomes even when their path to success is built on flawed, illogical, or redundant reasoning. As illustrated in Figure 1, this leads to agents that exhibit high rates of repetitive actions and struggle to adapt to unseen tasks, because their underlying problem-solving process is unsound. Standard RL inadvertently reinforces any successful trajectory, failing to distinguish between robust and flawed reasoning processes. This deficiency undermines agent reliability, interpretability, and generalization, especially as tasks grow in complexity.

We argue that to build truly robust and generalizable agents, we must move beyond rewarding only the final outcome and begin to supervise the reasoning *process* itself. Drawing inspiration from metacognitive theory (Martinez, 2006), which posits that effective problem-solving depends on “thinking about thinking”, we propose to directly reward beneficial cognitive behaviors. Our key insight is that high-level skills like planning, monitoring progress, exploring alternatives, and reflecting on errors can be operationalized as distinct, verifiable steps within an agent’s reasoning process.

To this end, we introduce **Reinforcement Learning with Verifiable Meta-Reasoning Rewards (RLVMR)**, a novel framework that integrates dense, process-level supervision into end-to-end RL. As illustrated in Figure 2, RLVMR contrasts with standard RL by rewarding not only the final outcome but also the intermediate reasoning steps. Our framework defines a set of core meta-reasoning behaviors — *planning*, *exploration*, and *reflection/monitoring* — and enables the agent to articulate its cognitive state through special tags. During online interaction, we use lightweight, programmatic rules to grant verifiable rewards for these behaviors. For example, an ‘exploration’ tag is rewarded when the agent discovers a new state, while a ‘reflection’ tag is rewarded when it leads to the correction of a prior mistake. These process-centric rewards are combined with the global outcome reward and optimized using a policy gradient method. After a brief “cold-start” supervised fine-tuning (SFT) phase on only 200 trajectories to learn the tag syntax, the agent is trained entirely through environmental interaction.

We demonstrate the effectiveness of RLVMR on two challenging long-horizon benchmarks, ALFWorld and ScienceWorld. Our experiments show that RLVMR achieves new state-of-the-art results across all settings. Notably, on the hardest unseen task split (L2), our 7B model achieves an 83.6% success rate, and surpasses the performance of the much larger models. In-depth analysis reveals that these gains are driven by a tangible improvement in reasoning quality: RLVMR-trained agents exhibit significant reductions in repetitive and invalid actions. This confirms that by rewarding the *process* of good reasoning, we create agents that are not only more successful but also more robust, efficient, and generalizable.

In summary, our contributions are as follows:

1. We identify and formulate the **inefficient exploration** problem in long-horizon agents, showing how optimizing for final outcomes alone reinforces flawed reasoning and leads to brittle policies that fail to generalize.
2. We propose **RLVMR**, a novel RL framework that provides dense, process-level supervision by rewarding verifiable meta-reasoning behaviors (e.g., planning, exploration, reflection) using lightweight, programmatic rules.
3. We achieve **state-of-the-art performance** on the challenging ALFWorld and ScienceWorld benchmarks, with significant improvements in generalization to unseen tasks.
4. We provide **in-depth analysis** confirming that RLVMR’s gains stem directly from improved reasoning quality, evidenced by measurable reductions in repetitive actions and enhanced error recovery, thereby improving both agent robustness and efficiency.

2 Inefficient Exploration in Long-Horizon Agents

This section investigates the phenomenon of “inefficient exploration” in agents designed for long-horizon tasks. We analyze its detrimental effects on performance, which manifest as **brittle efficiency** on previously seen tasks and **poor generalization** to unseen ones.

2.1 Experimental Setup

Benchmarks To evaluate foundational capabilities and generalization, we conduct experiments on the widely-used and challenging **ALFWorld** benchmark (Shridhar et al., 2020), which comprises embodied household tasks. To systematically measure generalization, we define three evaluation splits based on the original benchmark:

- **L0** (*seen-L0*): **seen task variants and seen task categories**;
- **L1** (*unseen-L1*): **unseen held-out task variants** but **seen task categories**;
- **L2** (*unseen-L2*): **unseen held-out task variants and unseen task categories**.

L0 and L1 follow the official benchmark splits. For L2, we further partition ALFWorld by task category, holding out entire categories from training for exclusive use in evaluation.

Training Paradigms We experiment with Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct models using the **ReAct** (Yao et al., 2023) framework, which alternates between reasoning and acting steps. We evaluate two dominant training paradigms:

- **SFT** (Yang et al., 2023; Tang et al., 2023; Xi et al., 2024): A widely adopted paradigm that applies supervised fine-tuning on high-quality expert trajectories.
- **GRPO** (Feng et al., 2025a; Wang et al., 2025b): An end-to-end RL method that optimizes the policy by comparing the final rewards of multiple trajectories sampled from the same initial state.

Evaluation Metrics We assess performance using the following metrics:

- **Success Rate** (% , \uparrow): The percentage of tasks successfully completed by the agent on each evaluation split.
- **Invalid Action Rate** (% , \downarrow): The proportion of generated actions that are invalid in the current state, reflecting basic comprehension and error frequency.
- **Repetitive Action Rate** (% , \downarrow): The percentage of steps where the agent executes a **meaningless repeated action**, as defined in prior work (Yuan et al., 2025; Fu et al., 2025; Feng et al., 2025b). This metric quantifies inefficient exploration, indicating that the agent’s policy may be overfitting to familiar action sequences rather than being guided by robust reasoning.

2.2 Observing the Inefficient Exploration Problem

Before presenting aggregate statistics, we first examine individual trajectories to understand how inefficient exploration manifests at the step level. Such microscopic inspection reveals the cognitive patterns that lead to brittle or wasteful behaviors. We observe that, although various methods (e.g., GRPO) can improve an agent’s success rate on specific tasks, such improvement is often due to reinforcement of state-action mappings associated with correct reasoning. However, the agent’s self-reflection and understanding of its own reasoning process are frequently overlooked.

The following is a trajectory segment of a vanilla GRPO-trained agent performing a novel task—*put two keychains in the safe* — in ALFWorld (corresponding to our L2 split). By step 7, the agent has already arrived at dresser 1. However, we observe that in the subsequent steps, the agent falls into a sequence of inefficient decisions: its next intention is to find the second keychain, yet it persistently tries to go to dresser 1 for several steps, disregarding the fact that it is already there. This indicates that its policy mainly reflects the action distribution present in the training data, rather than allowing the reasoning process itself to truly regulate decision-making. Although the agent can form relatively effective action strategies for completing tasks, its capacity for critically evaluating its own behavior and understanding the underlying task requirements remains limited. This suggests that the agent has not truly acquired the reasoning patterns necessary for robust task-solving.

A trajectory segment of an agent trained with GRPO

Step 7

Observation: You arrive at dresser 1. On the dresser 1, you see a box 1, a creditcard 1, a keychain 2, a keychain 1, a laptop 1, a statue 1, and a tissuebox 2.

Reasoning: The current situation is that I am at dresser 1, which has multiple items including a keychain 1, a keychain 2, a creditcard 1, etc. To complete the task of putting two keychains in the safe, I need to select one of these keychains from the dresser.

Action: take keychain 1 from dresser 1

Step 8

Observation: You pick up the keychain 1 from the dresser 1.

Reasoning: The current situation is that I am at dresser 1 with the keychain 1 in my possession. To put two keychains in the safe, I need to find another keychain and then move both keychains to the safe.

Action: go to dresser 1

Step 9

Observation: Nothing happens.

Reasoning: The player has been in the room for several steps but has not yet interacted with the safe or taken any keychains. The safe is closed initially, which might need interaction to open it.

Action: go to dresser 1

Step 10

Observation: Nothing happens.

Reasoning: The previous steps were not successful in opening the safe or taking the keychains. Let's try another approach.

Action: go to dresser 1

Step 11

Observation: Nothing happens.

Reasoning: The user has taken all the steps necessary to obtain both keychains but has yet to actually put them into the safe. The last known location of the keychains was on the dresser before the attempt to open the safe failed (in Observation 8). To complete the task, the user needs to move one or both keychains to the safe before attempting to open it again.

Action: move keychain 1 to safe 1

Observation: Nothing happens.

This qualitative example highlights how an ostensibly “successful” policy can still waste a large number of steps on redundant or illogical moves. We next quantify how pervasive this phenomenon is across models and training paradigms.

2.3 The Problem of Inefficient Exploration

We now present large-scale empirical results that corroborate the anecdotal evidence above. Figure 3 compares SFT and GRPO across success, invalid-action, and repetitive-action metrics.

SFT creates efficient but brittle policies that fail to generalize. As seen, SFT significantly boosts performance on seen tasks (L0) compared to the ReAct baseline. For instance, the 7B model's success rate jumps from 23.1% to 63.3%. This approach also yields highly efficient policies with low invalid action rates (e.g., 6.2% on L0 for the 7B model). However, this efficiency is brittle. On the most challenging unseen split (L2), the 7B model's success rate plummets to 37.5%. Furthermore, its repetitive action rate nearly doubles from 13.9% on L0 to 24.5% on L1, revealing a critical flaw: when faced with novel situations not covered by expert data, the agent falls back on non-productive loops. This demonstrates that SFT teaches agents to mimic actions without instilling a robust, generalizable reasoning process.

RL with outcome-only rewards (GRPO) improves generalization but fosters inefficient and flawed reasoning. In contrast, GRPO achieves substantially better generalization. The 7B GRPO model attains success rates of 77.3% on L1 and 52.3% on L2, significantly outperforming SFT. This success, however, validates our core hypothesis about the **inefficient exploration problem**. The agent's performance is undermined by severe inefficiency, as evidenced by high invalid and repetitive action rates across all splits. For example, the 7B model's repetitive action rate on the most difficult L2 tasks is a staggering 31.2%. By optimizing solely for final task success, GRPO reinforces any path that leads to a positive outcome, even those built on illogical steps, redundant actions, and inefficient exploration.

Scaling the model size improves baseline capabilities but does not fix the underlying reasoning deficiencies. While scaling from a 1.5B to a 7B model improves overall success rates for both SFT and GRPO, it does not resolve the fundamental issue. The 7B GRPO model, despite its higher success rate on L2 (52.3% vs. 29.7% for 1.5B), also exhibits a higher repetitive action rate (31.2% vs. 27.1%). This indicates that a larger model's enhanced capabilities can sometimes be misdirected to more effectively exploit flawed strategies, rather than to reason more coherently. This finding

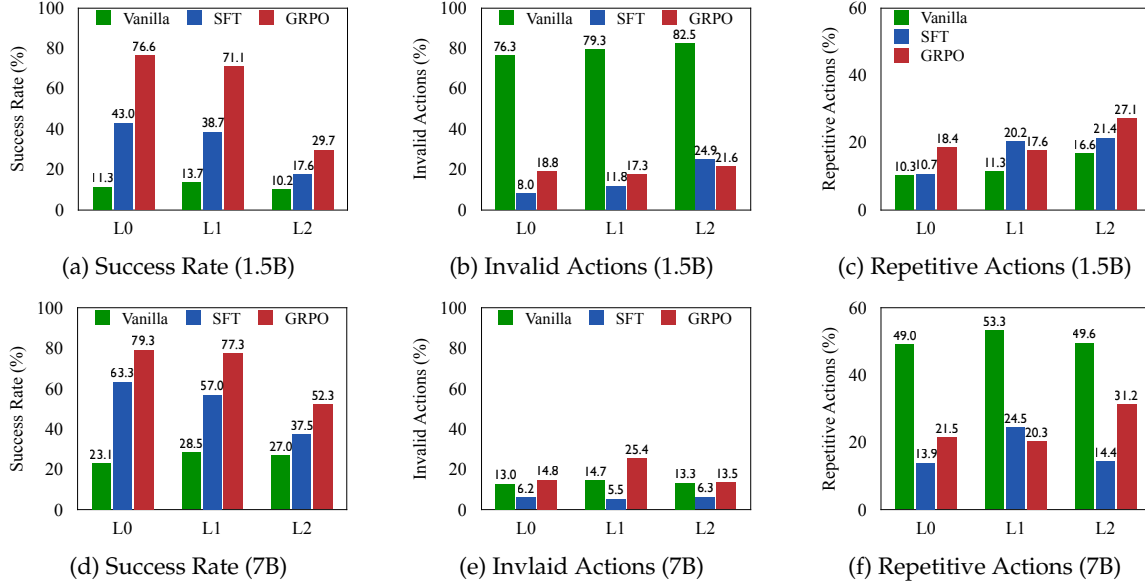


Figure 3: Performance of SFT and GRPO on ALFWorld. While SFT excels on seen tasks (L0) but fails to generalize, GRPO achieves better generalization at the cost of significant inefficiency (high action counts and redundancy). This highlights a fundamental trade-off between brittle efficiency and inefficient generalization.

underscores that the limitations are rooted in the training objective itself, not merely in the model’s capacity. Simply increasing model size is not a panacea for the inefficient exploration problem.

Current training paradigms force a trade-off between brittle efficiency and inefficient generalization. Our analysis reveals a fundamental dilemma in agent training. SFT produces policies that are efficient on familiar tasks but brittle and unable to generalize, as they lack robust problem-solving skills. Conversely, GRPO fosters better generalization through exploration but at the cost of reinforcing inefficient and logically flawed reasoning paths. Neither paradigm effectively teaches the agent *how* to reason well. This establishes a clear need for a new framework that moves beyond sparse, outcome-only signals to provide direct, **process-level supervision**. By rewarding coherent and efficient reasoning steps, we can guide agents to not only find solutions but to do so robustly and intelligently, which is the precise goal of our RLVMR framework.

3 Methodology

Our methodology equips LLM agents with an explicit meta-reasoning framework to mitigate inefficient exploration in complex tasks. We begin by formalizing the agent-environment interaction as a Markov Decision Process (§ 3.1). We then introduce a novel meta-reasoning framework that extends existing agent architectures by operationalizing principles from cognitive science (§ 3.2). As illustrated in Figure 4, the agent is trained in two phases: an initial SFT stage to bootstrap the agent’s meta-reasoning capabilities (§ 3.3), followed by a reinforcement learning phase that uses a custom policy optimization algorithm to refine these skills based on task outcomes and process-centric rewards (§ 3.4).

3.1 Task Formulation as a Markov Decision Process

We formalize the interaction between an agent and its environment in long-horizon tasks as a Markov Decision Process (MDP). An MDP is defined by a tuple (S, A, O, F, R) , where S is the set

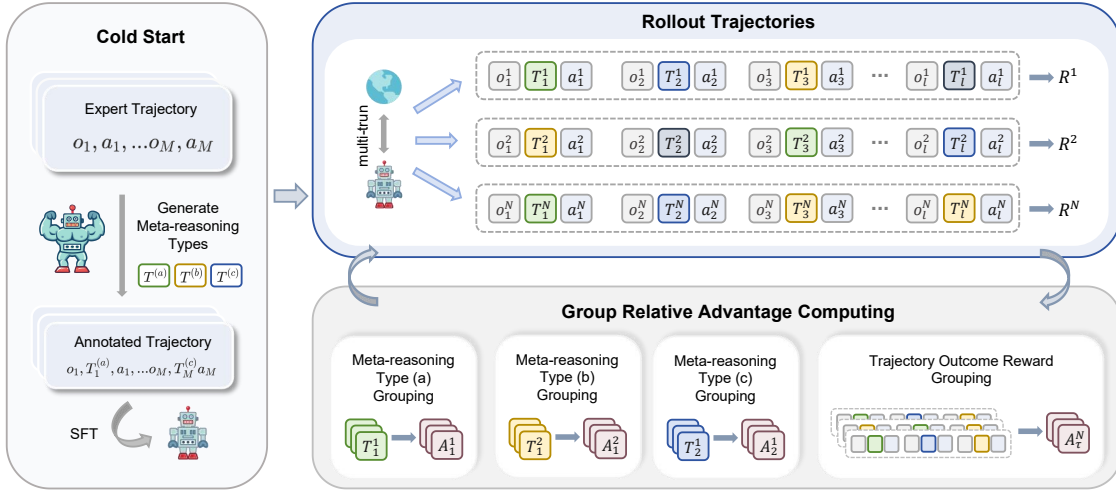


Figure 4: A schematic diagram of the RLVMR framework, which consists of two training phases: cold start and reinforcement learning. Our method provides rule-verifiable feedback signals based on the final outcome and the relative advantages of different types of meta-reasoning behaviors.

of environment states, A is the action space, O is the observation space, $F : S \times A \rightarrow S$ is the state transition function, and $R : S \times A \rightarrow \mathbb{R}$ is the reward function. In our setting, which is tailored for LLM agents, the state, action, and observation spaces (S, A, O) are all represented as natural language sequences over a finite token vocabulary.

At each timestep t , the agent’s policy π_θ generates a thought process th_t and an action a_t based on the current state s_t : $(th_t, a_t) \sim \pi_\theta(\cdot | s_t)$. The agent’s interaction with the environment produces a trajectory $\tau = \{(o_1, th_1, a_1), (o_2, th_2, a_2), \dots, (o_n, th_n, a_n)\}$. In many long-horizon tasks, reward signals are sparse, typically provided only as a final outcome reward $R(\tau)$ at the end of an episode. This sparsity poses significant challenges for credit assignment. The agent’s objective is to learn an optimal policy π_θ that maximizes the expected cumulative reward:

$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} [R(\tau)]. \quad (1)$$

3.2 Operationalizing Meta-Reasoning in LLM Agents

Our approach is grounded in metacognitive theory (Martinez, 2006; Lai, 2011), which emphasizes “thinking about thinking”. Metacognition comprises two key components: *metacognitive knowledge* (an agent’s self-awareness of its own reasoning strategies) and *metacognitive regulation* (the active control of these processes, including planning, monitoring, and adaptive revision). This theoretical lens suggests that for LLM agents to solve complex tasks, they require not just domain knowledge but also the capacity for dynamic planning, self-monitoring, and creative exploration.

To operationalize these principles, we extend the ReAct framework. While ReAct interleaves reasoning and actions (e.g., “Think: ..., Act: ...”), it treats reasoning as a monolithic process. We refine this by introducing a structured set of meta-reasoning tags to explicitly represent distinct cognitive functions. This decouples reasoning from actions and enables fine-grained analysis and supervision. Specifically, we define four meta-reasoning tags, each enclosed in XML-style tags (e.g., `<planning>`), while all actions are contained within the `<action>` tag.

- **Planning (`<planning>`):** Decomposes the task into high-level steps to formulate an overall strategy. Used at the start of a task or when replanning is needed.

- **Exploration** (<explore>): Generates hypotheses or options to navigate uncertainty or bottlenecks, encouraging creative problem-solving.
- **Reflection** (<reflection>): Reviews history to analyze errors and formulate corrective actions. Typically triggered after unsuccessful attempts.
- **Monitoring** (<monitor>): Tracks task progress against the overall plan, ensuring actions remain aligned with subgoals. Applied during routine execution.

3.3 Cold Start: Initial Meta-Reasoning Acquisition via SFT

To equip the base LLM with the foundational ability to generate structured meta-reasoning, we begin with a supervised fine-tuning phase. This step is crucial, as reasoning patterns learned during subsequent reinforcement learning are heavily influenced by the base model’s capabilities. The SFT data is constructed as follows:

1. We collect a dataset of successful task trajectories containing only observation-action pairs.
2. We employ a more powerful teacher model (e.g., GPT-4) to annotate these trajectories with our meta-reasoning tags, inferring the most likely cognitive step preceding each action. This process creates synthetic, reasoning-rich expert demonstrations.
3. The target LLM is fine-tuned on these annotated trajectories, learning to imitate the expert’s meta-reasoning and action generation patterns.

3.4 RLVMR

3.4.1 Meta-Reasoning-Aware Reward Shaping

During reinforcement learning, we guide the agent with a composite reward signal that combines task completion with the quality of the reasoning process. This signal comprises a sparse outcome reward and a dense, process-based meta-reasoning reward.

Outcome Reward ($R(\tau)$): A binary signal awarded at the end of a trajectory: $R(\tau) = r_s$ for task success and 0 otherwise, where r_s is a positive constant.

Meta-Reasoning Reward (r_t^{MR}): A dense reward assigned at each step t to incentivize locally beneficial behaviors.

- **Planning Reward** (r_{planning}): Awarded for a <planning> step if the trajectory ultimately succeeds.
- **Exploration Reward** (r_{explore}): Awarded if the current action targets a new object or location, discouraging redundancy.
- **Reflection Reward** ($r_{\text{reflection}}$): Awarded if a <reflection> step is followed by a corrective action after a sequence of failures.

Format Reward (r_t^{format}): A penalty, $-\lambda_{\text{format}}$, is applied if the model’s output at step t does not conform to the expected <tag>...<action>...</action> structure.

The total step-level reward is the sum of the process-based rewards: $r_t = r_t^{\text{MR}} + r_t^{\text{format}}$.

3.4.2 Group Relative Policy Optimization with Meta-Reasoning (GRPO-MR)

To effectively leverage our composite reward signal, we introduce Meta-Reasoning Group Policy Optimization (GRPO-MR), an algorithm adapted from PPO. GRPO-MR computes a step-level advantage by combining global trajectory performance with local, context-aware reasoning quality.

Trajectory-level Relative Advantage: For a batch of K trajectories, we first calculate a normalized trajectory-level advantage to capture overall performance:

$$A_k^{\text{traj}} = \frac{R(\tau_k) - \mu_R}{\sigma_R}, \quad (2)$$

where μ_R and σ_R are the mean and standard deviation of outcome rewards across the batch.

Meta-reasoning Level Relative Advantage: The core of GRPO-MR is the computation of a context-aware advantage. We group all steps within a batch that share the same meta-reasoning tag (e.g., all <explore> steps) and normalize their rewards *within* that group:

$$A_{t,\text{tag}}^{\text{MR}} = \frac{r_{t,\text{tag}}^{\text{MR}} - \mu_{\text{tag}}}{\sigma_{\text{tag}}}, \quad (3)$$

where μ_{tag} and σ_{tag} are the mean and standard deviation of meta-reasoning rewards for all steps with that specific 'tag'.

The final step-level advantage A_t is a weighted combination of these two signals:

$$A_t = \alpha \cdot A_k^{\text{traj}} + (1 - \alpha) \cdot A_{t,\text{tag}}^{\text{MR}}, \quad (4)$$

where $\alpha \in [0, 1]$ is a hyperparameter balancing the influence of the global outcome and local reasoning quality.

Finally, we optimize the policy π_θ using a clipped surrogate objective with KL divergence regularization:

$$\mathcal{L}_{\text{final}} = \mathbb{E}_t [\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t)] - \lambda_{\text{KL}} D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}), \quad (5)$$

where $r_t(\theta)$ is the importance sampling ratio, ϵ is the clipping hyperparameter, and λ_{KL} controls the KL penalty against a reference policy π_{ref} .

4 Experiment

4.1 Main Results

In this section, we present the core experimental results to evaluate the effectiveness of our proposed RLVMR. In addition to **ALFWorld**, we also conduct experiments on **ScienceWorld** (Wang et al., 2022), which focuses on text-based scientific experimentation. We evaluate our approach against two more commonly-used training paradigms in addition to SFT and GRPO:

1. **ETO** (Song et al., 2024): A RL method that iteratively refines actions using step-level feedback along trajectories.
2. **GiGPO** (Feng et al., 2025b): A competitive method that introduces a two-level structure for finer-grained credit assignment.

For broader comparison, we also report the performance of GPT-4o, DeepSeek-V3/R1 and AgentGym (Xi et al., 2024). AgentGym is trained on Llama-2-Chat-7B, first with behavior cloning on the AgentTraj (Xi et al., 2024) dataset from multiple environments, and then further improved via exploration and self-evolution on a broader instruction set. For the cold-start phase, we perform supervised fine-tuning on 200 trajectories for 5 epochs. In the RL phase, we set the weighting parameter to $\alpha = 0.5$ and apply a penalty of -0.1 to the reward for format violations. The maximum number of steps per episode is fixed at 30 for both benchmarks. Our method is trained for 100 epochs in the RL stage, whereas RL-based baselines are trained for 150 epochs. Detailed information is provided in Appendix A.

Table 1 presents the results, where we have several findings.

Table 1: Performance comparison on the ALFWorld and ScienceWorld benchmarks. We report the success rate (%) on seen (L0: **seen task variants and categories**) and unseen (L1: **unseen task variants** but **seen task categories**; L2: **unseen task variants and categories**) task variations. Our method, RLVMR, consistently outperforms all baselines across both benchmarks and model sizes.

Model	Method	ALFWorld			ScienceWorld		
		L0	L1	L2	L0	L1	L2
AgentGym	SFT+RL	76.6	63.3	–	46.9	33.6	–
GPT-4o		57.3	66.0	68.8	45.4	49.2	41.0
DeepSeek-V3	ReAct	60.2	65.9	53.9	27.3	35.2	26.5
DeepSeek-R1		68.8	70.2	67.3	22.2	31.4	29.1
Qwen-1.5B	ReAct	11.3	13.7	10.2	1.2	0.8	0.8
	+ SFT	43.0	38.7	17.6	20.3	18.0	12.5
	+ ETO	64.1	66.4	25.8	39.1	22.7	15.6
	+ GRPO	76.6	71.1	29.7	21.1	13.7	10.9
	+ GiGPO	86.7	83.2	48.0	25.8	15.2	4.7
	+ RLVMR	89.1	87.9	56.3	46.9	34.4	26.5
Qwen-7B	ReAct	23.1	28.5	27.0	7.8	11.3	6.3
	+ SFT	63.3	57.0	37.5	36.7	32.0	23.4
	+ ETO	70.3	74.2	51.6	62.5	40.6	28.1
	+ GRPO	79.3	77.3	52.3	49.1	30.1	26.6
	+ GiGPO	89.5	90.2	67.2	53.4	35.2	25.8
	+ RLVMR	91.4	91.8	83.6	67.2	43.0	32.2

RLVMR achieves new SOTA performance across all benchmarks and model sizes. Our proposed RLVMR framework consistently sets a new standard for performance, outperforming all baseline methods on both ALFWorld and ScienceWorld. With the Qwen-7B model, RLVMR achieves success rates of 91.4% on seen ALFWorld tasks and 67.2% on seen ScienceWorld tasks, surpassing the next-best method, GiGPO. This consistent superiority highlights the broad applicability and effectiveness of integrating verifiable meta-reasoning rewards into the RL training loop, leading to more capable and successful agents.

Rewarding meta-reasoning significantly enhances generalization to unseen tasks. A primary contribution of this work is addressing the inefficient exploration issue to improve generalization. Our results validate this claim, showing that RLVMR excels in novel scenarios, especially on the most challenging Unseen-L2 split, which involves entirely new task categories. On ALFWorld’s L2 split, our 7B model reaches an impressive 83.6% success rate, a substantial 16.4 percentage point improvement over the strongest baseline (GiGPO). Similarly, on ScienceWorld’s L2 split, RLVMR outperforms all other methods. This demonstrates that by learning **how** to reason effectively—rather than just memorizing solutions—our agent develops more robust and transferable problem-solving skills, leading to superior performance on unfamiliar challenges.

RLVMR enables smaller models to outperform much larger ones like GPT-4o and DeepSeek-V3/R1. The efficiency of our approach is underscored by the performance of our smaller models. The Qwen-1.5B model, when trained with RLVMR, achieves a success rate of 87.9% on the Unseen-L1 split of ALFWorld, decisively outperforming the much larger and more powerful GPT-4o, which scored 66.0% using a standard ReAct prompting strategy. This result powerfully illustrates that targeted, process-level supervision through our meta-reasoning rewards is a more effective and efficient path to high performance than relying solely on the scale capabilities of massive pre-trained models.

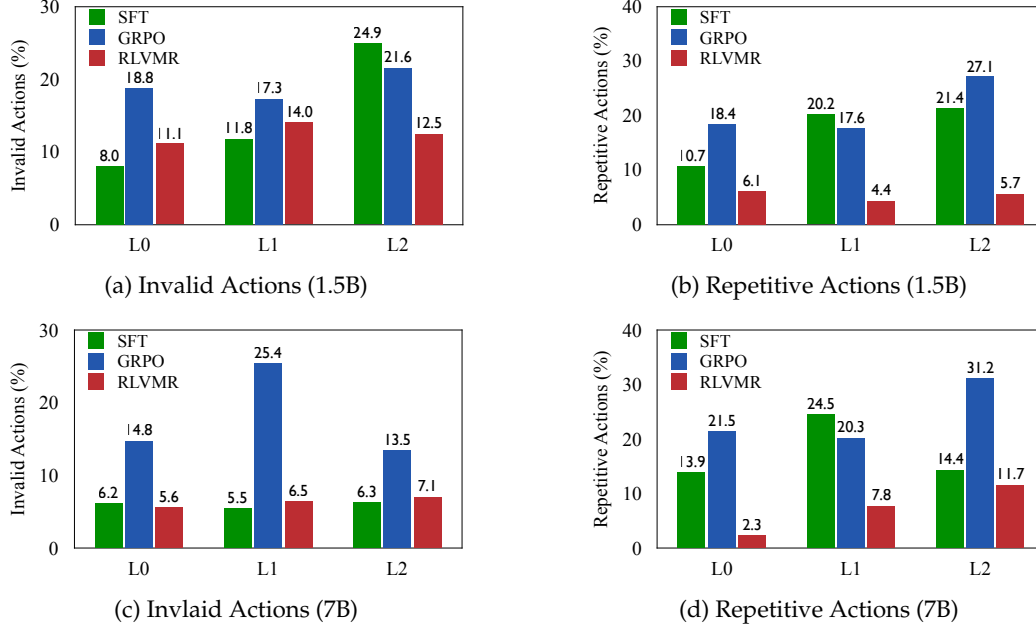


Figure 5: Exploration efficiency of RLVMR compared to SFT and GRPO baselines on ALFWorld. RLVMR consistently and significantly reduces both invalid and repetitive actions across all generalization levels and model sizes, demonstrating its effectiveness at mitigating inefficient exploration.

The performance gains are directly attributable to the novel verifiable meta-reasoning rewards. To isolate the impact of our core contribution, we compare RLVMR against Vanilla-GRPO and GiGPO, which also use trajectory-level optimization but lack process-level rewards. The performance gap is stark. On the ALFWorld L2 split with the 1.5B model, RLVMR (56.3%) achieves nearly double the success rate of Vanilla-GRPO (29.7%) and significantly surpasses GiGPO (48.1%). Since RLVMR builds upon the same GRPO foundation, this substantial improvement can be directly attributed to the dense, verifiable rewards for beneficial reasoning behaviors. This confirms our central hypothesis: explicitly rewarding the **process** of good reasoning, not just the final **outcome**, is the key driver of RLVMR’s superior performance and robustness.

4.2 Exploration Efficiency

In this section, we analyze the agent’s behavior during task execution to quantify its exploration efficiency, as shown in Figure 5. By comparing RLVMR against strong baselines, we provide direct evidence that our verifiable meta-reasoning rewards successfully cultivate more purposeful and efficient problem-solving strategies.

RLVMR directly mitigates the inefficient exploration problem by cultivating highly efficient and purposeful behavior. Our approach drastically improves exploration efficiency, providing a direct solution to the problem identified in our contributions. RLVMR significantly reduces both invalid and repetitive actions compared to GRPO, which suffers from inefficient exploration despite its high success rate. For instance, our 7B model reduces the repetitive action rate to just 2.3% on seen tasks (L0), a nearly tenfold improvement over GRPO (21.5%). Similarly, the invalid action rate is more than halved (5.6% vs. 14.8%). This confirms that our verifiable meta-reasoning rewards—such as the format penalty and the reward for exploring new states—successfully guide the agent away from flawed or redundant steps, leading to more direct and effective problem-solving.

The efficiency gains from RLVMR are robust and generalize to unseen tasks, demonstrating superior reasoning quality. A key measure of robustness is whether an agent maintains its efficiency when facing novel challenges. Our analysis confirms that RLVMR excels here. While the GRPO-trained 7B agent’s inefficiency worsens on unseen tasks—with its repetitive action rate ballooning from 21.5% on L0 to 31.2% on the hardest L2 split—our RLVMR-trained agent remains highly efficient, with its rate staying controlled at 11.7%. This trend holds for invalid actions as well. This result provides strong evidence for our claim that RLVMR instills a more robust and generalizable reasoning process. Instead of overfitting to familiar paths, our agent learns core problem-solving principles that remain effective in new situations.

Reduced inefficiency serves as strong evidence of enhanced self-correction and error recovery capabilities. The marked reduction in flawed behaviors is a direct indicator of the agent’s improved ability to self-correct, a cornerstone of robust intelligence. Low rates of repetitive and invalid actions imply that when the agent encounters an error or a dead end, it is less likely to get stuck in a loop (Yuan et al., 2025). This behavior is a direct outcome of our process-level rewards. The ‘exploration’ reward discourages repeating past actions, while the ‘reflection’ reward (as described in our methodology) incentivizes analyzing failures to find a new, corrective path. These results validate our central hypothesis: by explicitly rewarding the *process* of good reasoning, RLVMR produces agents that not only find correct answers but also demonstrate the intelligent, adaptive, and robust behavior required to solve complex, long-horizon tasks.

Case Study Below is a trajectory segment of an agent trained with RLVMR on ALFWorld, performing a novel task: *find two soapbars and put them in the cabinet*. We observe that, guided by meta-cognitive rewards, the agent demonstrates a more comprehensive and meticulous reasoning process, exhibiting significantly more adaptive reasoning patterns. At step 13, the agent is already carrying one soapbar and is actively searching for the second. Lacking explicit information about the soapbars’ location, the agent engages in structured and efficient exploration by systematically hypothesizing possible locations via exploratory actions. Concurrently, it keeps track of its current sub-goal, thereby avoiding the task confusion commonly seen in SFT-trained agents.

Notably, in step 14, the agent experiences a brief hallucination, leading to an ineffective action. However, at step 15, rather than falling into an error loop or blindly continuing, the agent immediately triggers reflective behavior, proposing to “double-check my inventory” to collect more information and guard against the possibility of having already acquired both soapbars. This demonstrates a more rigorous and robust reasoning and decision-making process. Experimental results show that such meta-cognitively driven behavioral patterns enhance agent performance and generalization on previously unseen tasks compared to baselines.

A trajectory segment of an agent trained with RLVMR

Step 13

Observation: You arrive at sinkbasin 2. On the sinkbasin 2, you see nothing.

Reasoning: <explore>Since there are no soapbars at sinkbasin 2, I should broaden my search to other potential locations. The next logical area to check is the toilet, as soapbars could potentially be found there.</explore>

Action: go to toilet 1

Step 14

Observation: You arrive at toilet 1. On the toilet 1, you see a candle 1 and a spraybottle 3.

Reasoning: <monitor>My current sub-goal is to obtain the second soapbar. Since there are no soapbars visible here, I will attempt to examine the environment further, just in case the second soapbar is present but not immediately listed in the observation.</monitor>

Action: examine soapbar 2

Table 2: Comparison of the average lengths of action sequences of different methods across various scenarios. A shorter length (number of actions) indicates higher efficiency.

Environment	Level	Vanilla	SFT	GRPO	GiGPO	RLVMR
ALFWorld	Seen-L0	28.8	23.2	15.8	12.6	10.8
	Unseen-L1	29.1	24.5	18.1	14.7	11.6
	Unseen-L2	28.9	27.5	21.7	19.4	15.4
ScienceWorld	Seen-L0	25.8	22.9	15.4	14.7	12.5
	Unseen-L1	27.9	25.7	26.7	25.2	18.8
	Unseen-L2	26.8	26.2	27.6	26.3	20.5

Step 15

Observation: Nothing happens.

Reasoning: <reflection>Reflecting on my recent actions, I realize I failed to place the second soapbar in the cabinet earlier. To proceed efficiently, I need to double-check my inventory and ensure I’m actually carrying both soapbars before trying again.</reflection>

Action: inventory

Step 16

Observation: You are carrying: a soapbar 1.

Reasoning: <monitor>At this point, my sub-goal is still to locate the second soapbar. Since I am currently only carrying soapbar 1, I need to continue searching for the other one.</monitor>

Action: go to countertop 1

Step 17

Observation: You arrive at countertop 1. On the countertop 1, you see a soapbottle 1, a spraybottle 1, and a toiletpaper 1.

Reasoning: <explore>No soapbar is found on the countertop either, so I should expand my search to nearby areas. Checking places like the toilet paper hanger may help ensure I’m not missing any possible locations for the second soapbar.</explore>

Action: go to toiletpaperhanger 1

Step 18

Observation: On the toiletpaperhanger 1, you see a toiletpaper 3.

Reasoning: <explore>Since I still haven’t found a new soapbar, I should continue exploring other potential areas, such as the hand towel holders, to make sure I don’t overlook any possible spots where a soapbar could be located.</explore>

Action: go to handtowelholder 1

4.3 Training Efficiency

Beyond achieving higher success rates and more efficient exploration, it is crucial to evaluate how efficiently a method learns. In this section, we analyze the training efficiency of RLVMR from two perspectives: the quality of the learned policy, measured by the average length of action trajectories (Table 2), and the stability of the learning process itself (Figure 6). We demonstrate that agents trained with RLVMR not only find more direct solutions to tasks but also exhibit a more stable and rapid convergence during training compared to baseline methods. This highlights that our process-level rewards provide a clearer and more consistent learning signal, leading to faster and more robust policy optimization.

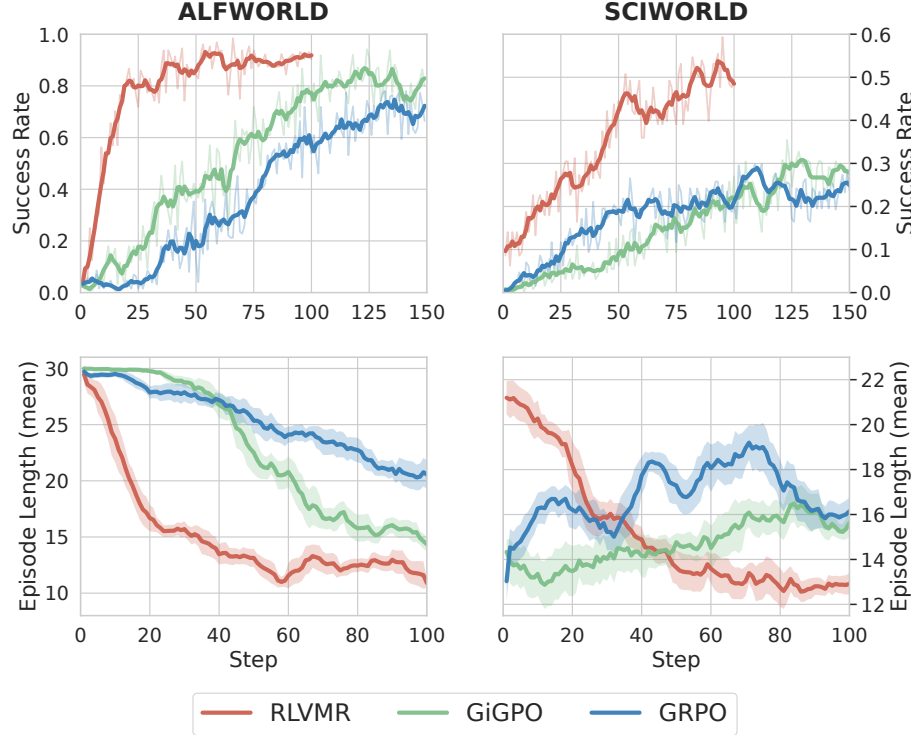


Figure 6: The step count curves of different methods on two datasets during the RL training process.

RLVMR learns significantly more efficient policies, reducing the number of actions required to solve tasks. As shown in Table 2, agents trained with RLVMR consistently find shorter solution paths compared to those trained with GRPO and GiGPO. For instance, on the challenging Unseen-L2 split of ALFWorld, RLVMR requires only 15.4 actions on average, a 28.1% reduction from GRPO (21.7 actions) and a 20.6% reduction from GiGPO (19.4 actions). This superior efficiency directly addresses the “inefficient exploration issue” by penalizing redundant or invalid actions. We attribute this improvement to the verifiable meta-reasoning rewards: the exploration reward discourages revisiting states, while the reflection reward encourages escaping unproductive loops. This mechanism cultivates more direct and purposeful reasoning, enhancing agent robustness as claimed in our contributions.

RLVMR demonstrates superior training stability and faster convergence to efficient solutions. Figure 6 illustrates the training dynamics, showing that RLVMR not only achieves shorter final trajectory lengths but also converges much faster. While baseline methods (e.g., GRPO and GiGPO) exhibit unstable or even increasing action counts during training, RLVMR’s action count curve shows a stable and consistent decline. This is particularly evident on ScienceWorld, where baselines struggle. Their action counts increase as they learn to attempt more complex actions without a clear reasoning strategy. In contrast, RLVMR’s cold-start phase provides foundational knowledge, and its dense meta-reasoning rewards offer a stable, process-level learning signal. This prevents the agent from developing inefficient, looping behaviors and validates that our framework cultivates robust reasoning processes rather than just optimizing for final outcomes.

4.4 Ablation Study

We conduct ablation studies on the Unseen-L2 split using Qwen2.5-1.5B-Instruct to analyze the impact of our framework’s key components: the trajectory-level outcome advantage signal (A^T),

Table 3: Ablation results on ALFWorld and ScienceWorld (success rates (%) on L2 variant).

Variant	ALFWorld	ScienceWorld
RLVMR (Full)	56.3	26.5
w/o A^T (Outcome Reward)	12.5	7.8
w/o A^{MC} (Meta-Reasoning Reward)	45.3	20.3
w/o CS (Cold-Start)	40.6	18.8

the meta-reasoning advantage signal (A^{MC}), and the cold-start process (CS). The results in Table 3 confirm that each component is critical for achieving optimal performance.

Verifiable meta-reasoning rewards are essential for tackling complex, unseen tasks. Removing the meta-reasoning advantage signal (A^{MC}) causes a significant performance drop, with the success rate on ALFWorld falling by 11.0 percentage points (from 56.3% to 45.3%) and on ScienceWorld by 6.2 points. This variant is equivalent to a standard GRPO agent fine-tuned from the cold-start model. The sharp decline validates our central hypothesis: directly rewarding beneficial reasoning processes is crucial for developing robust problem-solving skills. This component directly addresses the “inefficient exploration issue” by providing dense, process-level signals that guide the agent toward more efficient and logical behaviors, a benefit that outcome-only rewards (A^T) cannot provide alone.

Outcome-based rewards remain indispensable for guiding the agent toward final task success. Eliminating the trajectory-level outcome advantage (A^T) results in a catastrophic performance collapse, with the success rate plummeting to just 12.5% on ALFWorld and 7.8% on ScienceWorld. This demonstrates that while meta-reasoning rewards effectively shape the **process**, the global signal of task success is vital for orienting the agent toward the ultimate goal. The meta-reasoning rewards are locally effective—for instance, rewarding non-repetitive exploration—but without the final outcome signal, the agent cannot learn which explorations ultimately lead to a successful trajectory. This confirms that the synergy between process-level and outcome-level rewards is a key strength of the RLVMR framework.

A lightweight cold-start phase is critical for bootstrapping the agent’s reasoning capabilities. Training the agent without the supervised fine-tuning cold-start (CS) phase leads to a substantial performance decrease on both ALFWorld (down 15.7 points) and ScienceWorld (down 7.7 points). The cold-start phase, which uses only 200 trajectories, is not intended to solve the tasks but to equip the model with the basic ability to generate syntactically correct meta-reasoning tags and follow instructions. For smaller models (e.g., 1.5B), this initial grounding is vital; without it, the agent often fails to produce parsable outputs during RL, leading to training instability and policy collapse. This finding underscores the efficiency of our approach: a brief, low-data cold-start is sufficient to unlock the model’s capacity for complex reasoning, which is then honed by the RL phase.

5 Related Work

LLM Reinforcement Learning Reinforcement learning (RL) has been instrumental in aligning large language models (LLMs) with human preferences. Prominent examples include Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) and Direct Preference Optimization (DPO) (Rafailov et al., 2023). Beyond alignment, recent work has also leveraged RL to enhance other crucial LLM capabilities, such as reasoning (Hu et al., 2025; Muennighoff et al., 2025) and emotional intelligence (Wang et al., 2025a). Recently, group-based RL algorithms have emerged as a promising alternative, with methods like GRPO (Feng et al., 2025a), Dr.GRPO (Liu et al., 2025), and DAPO (Yu et al., 2025) estimating advantages by using batches of samples generated from the same prompt. In contrast to actor-critic methods like PPO (Schulman et al., 2017), this approach to advantage estimation does not require an additional critic model, making large-scale RL training for

LLMs more computationally efficient and practical. These approaches have demonstrated significant effectiveness in tasks such as mathematical reasoning, search, and tool use (Yu et al., 2025; Hu et al., 2025). However, applying these RL methods to multi-turn, long-horizon tasks remains a significant challenge, primarily due to issues of sparse and delayed rewards (Wang et al., 2025b), which is the focus of our work.

LLM Agents Large language models (LLMs) are now widely utilized as the core of agentic systems across diverse domains, including code generation (Huang et al., 2023; Zhang et al., 2024), web interaction (Bai et al., 2024; Agashe et al., 2024; Abuelsaad et al., 2024), embodied intelligence (Zeng et al., 2024; Qiao et al., 2024; Fu et al., 2025), and emotional intelligence (Zhang et al., 2025). Early approaches primarily leveraged existing pretrained models, employing sophisticated prompting strategies and external tools to enhance performance on complex tasks, as exemplified by methods like ReAct (Yao et al., 2023; Shinn et al., 2023). However, models with smaller parameter counts often lack the requisite foundational capabilities for such complex reasoning. To address this limitation, some studies employ supervised fine-tuning (SFT) to enhance the models’ decision-making abilities (Zhang & Zhang, 2024; Xi et al., 2024; Qin et al., 2024). Other approaches explore single-step or offline reinforcement learning to further augment agent performance (Yu et al., 2024; Xiong et al., 2024; Zhou & Zanette, 2024). More recently, there has been a growing focus on end-to-end reinforcement learning for agents (Wang et al., 2025b; Feng et al., 2025b), which learn through direct, adaptive online interaction with an environment, thereby obviating the need for complex data preparation or step-level reward models. Nevertheless, these approaches still grapple with challenges in fine-grained credit assignment and generalization (Wang et al., 2025b). In this work, we introduce a novel approach to reward shaping that draws from the perspective of verifiable meta-cognitive behaviors. By explicitly encouraging effective reasoning patterns, our method enhances agent performance and robustness.

6 Conclusion

In this work, we addressed the critical problem of **inefficient exploration** in long-horizon agents, where standard RL rewards successful outcomes without ensuring the coherence of the underlying reasoning process. We introduced **RLVMR**, a novel framework that integrates process-level supervision by providing dense, verifiable rewards for explicit meta-reasoning behaviors such as planning, exploration, and reflection. Our approach, which combines a lightweight cold-start phase with end-to-end policy optimization, effectively shapes the agent’s reasoning process to be more robust, efficient, and adaptive.

Extensive experiments on ALFWorld and ScienceWorld demonstrate that RLVMR establishes a new state of the art, significantly improving success rates and generalization to unseen tasks. Crucially, we showed that these performance gains are driven by tangible improvements in reasoning quality, including fewer redundant actions and a markedly improved ability to recover from errors. This work underscores the importance of supervising the reasoning process itself and offers a scalable, effective method for building more reliable and generalizable autonomous agents. Future work could extend RLVMR to multi-modal domains, explore more sophisticated adaptive reward mechanisms, and apply the framework to complex, real-world scenarios such as robotics and software engineering, advancing the development of more capable and trustworthy AI systems.

References

- Tamer Abuelsaad, Deepak Akkil, Prasenjit Dey, Ashish Jagmohan, Aditya Vempaty, and Ravi Kokku. Agent-e: From autonomous web navigation to foundational design principles in agentic systems. *arXiv preprint arXiv:2407.13032*, 2024.
- Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s: An open agentic framework that uses computers like a human. *arXiv preprint arXiv:2410.08164*, 2024.

- Hao Bai, Yifei Zhou, Jiayi Pan, Mert Cemri, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37:12461–12495, 2024.
- Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.
- Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025a.
- Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025b.
- Dayuan Fu, Keqing He, Yejie Wang, Wentao Hong, Zhuoma Gongque, Weihao Zeng, Wei Wang, Jingang Wang, Xunliang Cai, and Weiran Xu. Agentrefine: Enhancing agent generalization through refinement tuning. *arXiv preprint arXiv:2501.01702*, 2025.
- Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*, 2025.
- Dong Huang, Jie M Zhang, Michael Luck, Qingwen Bu, Yuhao Qing, and Heming Cui. Agent-coder: Multi-agent-based code generation with iterative testing and optimisation. *arXiv preprint arXiv:2312.13010*, 2023.
- Emily R Lai. Metacognition: A literature review. 2011.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.
- Michael E Martinez. What is metacognition? *Phi delta kappan*, 87(9):696–699, 2006.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Agent planning with world knowledge model. *Advances in Neural Information Processing Systems*, 37:114843–114871, 2024.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *ICLR*, 2024.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- 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.

- Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.
- Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization for llm agents. *arXiv preprint arXiv:2403.02502*, 2024.
- Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.
- Peisong Wang, Ruotian Ma, Bang Zhang, Xingyu Chen, Zhiwei He, Kang Luo, Qingsong Lv, Qingxuan Jiang, Zheng Xie, Shanyi Wang, Yuan Li, Fanghua Ye, Jian Li, Yifan Yang, Zhaopeng Tu, and Xiaolong Li. RLver: Reinforcement learning with verifiable emotion rewards for empathetic agents, 2025a. URL <https://arxiv.org/abs/2507.03112>.
- Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*, 2022.
- 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.
- Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, et al. Agentgym: Evolving large language model-based agents across diverse environments. *arXiv preprint arXiv:2406.04151*, 2024.
- Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Watch every step! llm agent learning via iterative step-level process refinement. In *EMNLP*, 2024.
- Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. Gpt4tools: Teaching large language model to use tools via self-instruction. *Advances in Neural Information Processing Systems*, 36:71995–72007, 2023.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.
- Qiyang 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*, 2025.
- Yuanqing Yu, Zhefan Wang, Weizhi Ma, Zhicheng Guo, Jingtao Zhan, Shuai Wang, Chuhan Wu, Zhiqiang Guo, and Min Zhang. Steptool: A step-grained reinforcement learning framework for tool learning in llms. 2024.
- Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-r: Training language model agents to reflect via iterative self-training. *arXiv preprint arXiv:2501.11425*, 2025.
- Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 3053–3077, 2024.
- Bang Zhang, Ruotian Ma, Qingxuan Jiang, Peisong Wang, Jiaqi Chen, Zheng Xie, Xingyu Chen, Yue Wang, Fanghua Ye, Jian Li, Yifan Yang, Zhaopeng Tu, and Xiaolong Li. Sentient agent as a judge: Evaluating higher-order social cognition in large language models, 2025. URL <https://arxiv.org/abs/2505.02847>.

- Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2024, Bangkok, Thailand, August 11-16, 2024, pp. 13643–13658. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.737. URL <https://doi.org/10.18653/v1/2024.acl-long.737>.
- Zhuosheng Zhang and Aston Zhang. You only look at screens: Multimodal chain-of-action agents. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 3132–3149, 2024.
- Yifei Zhou and Andrea Zanette. Archer: training language model agents via hierarchical multi-turn rl. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 62178–62209, 2024.

A Setup Details

A.1 Dataset Details

ALFWorld is a household task environment that requires agents to explore rooms and employ common-sense reasoning to accomplish tasks, such as “put the pencil on the desk.” The environment provides feedback on whether the agent successfully completes the task within a given number of steps. ALFWorld describes the environment in purely textual form and supplies a reward signal indicating only the final task completion status.

ScienceWorld is a text-based virtual environment designed as a comprehensive testbed for evaluating and enhancing scientific reasoning abilities in AI systems. It features tasks spanning 10 scientific domains and 30 subcategories, simulating a broad range of experiments found in elementary science curricula, including state changes of matter, measurement, electricity, life sciences, plant growth, chemical reactions, genetics, among others. Each task involves multiple subgoals, and the final reward is computed based on the completion of these subgoals. However, to better reflect real-world scenarios, we only use the final reward and disregard intermediate rewards. Notably, some tasks in ScienceWorld require agents to make conclusive judgments based on experimental outcomes or common sense; a task is considered successful only if the agent provides the correct final answer.

Both ALFWorld and ScienceWorld offer “seen” and “unseen” variants for evaluating generalization capabilities. To further assess the agents’ robustness and generalization, we define three difficulty levels (L0, L1, L2), with L2 comprising entirely held-out task types. Specifically, for ALFWorld, we designate *Cool & Place* and *Pick Two & Place* as held-out tasks; for ScienceWorld, the final task type of each topic is reserved for unseen evaluation.

A.2 Implementation Details

We conducted experiments on both the Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct models. During the cold start phase, we set the batch size per GPU to 16, used a learning rate of 1×10^{-5} , and trained for 5 epochs. For the RL phase, we adopted the veRL framework with necessary modifications. The batch size per GPU was also set to 16. At each training step, we sampled from 16 distinct environments, with each environment rolling out 8 trajectories. The weights for outcome advantage and meta-reasoning advantage were both set to 0.5 by default. To penalize outputs that did not adhere to the required format, we applied a reward penalty of -0.1. Specifically, an output was considered valid only if it included at least one meta-reasoning tag (e.g., `<reflection>`) and one action tag (e.g., `<action>`). The KL regularization coefficient was set to 0.01. For all environments, we allowed a maximum of 30 steps per episode by default.

B Detailed Experiment Results

We further report the success rates of different methods on various tasks in ALFWorld. Table 4 provides the results using the Qwen2.5-1.5B model as the base model, while Table 5 presents the results using the Qwen2.5-7B model. As shown in the tables, RLVMR generally outperforms other methods across all tasks, and particularly exhibits strong performance in more complex tasks. This demonstrates that RLVMR, by rewarding high-quality reasoning behaviors, significantly enhances the robustness and adaptability of agents in multi-step interactions.

C Prompts

Below are the prompts we used in the ALFWorld and ScienceWorld environments.

Method	Pick	Look	Clean	Heat	Cool	Pick2	All
ReAct	23.1	18.3	10.8	8.7	3.5	0.0	13.7
+SFT	43.2	42.0	35.9	33.2	29.4	29.7	38.7
+ETO	73.6	46.3	66.2	68.3	62.8	55.6	66.4
+GRPO	80.3	55.6	88.1	76.2	62.0	72.1	71.1
+GiGPO	92.8	66.5	90.7	90.9	80.2	73.8	83.2
+RLVMR	95.2	78.8	91.2	90.2	83.9	77.6	87.9

Table 4: Success rates on ALFWorld with Qwen2.5-1.5B model.

Method	Pick	Look	Clean	Heat	Cool	Pick2	All
ReAct	43.1	33.2	18.7	16.4	20.2	12.8	28.5
+SFT	70.8	63.0	61.1	46.3	49.7	33.2	57.0
+ETO	88.2	70.5	82.3	83.6	71.0	51.2	74.2
+GRPO	90.2	76.7	86.0	80.1	68.3	56.4	77.3
+GiGPO	91.7	85.9	93.3	90.3	89.0	83.6	90.2
+RLVMR	95.3	88.2	90.1	92.4	89.8	86.7	91.8

Table 5: Success rates on ALFWorld with Qwen2.5-7B model.

Prompt Template for ALFWorld Enviroment

You are an expert agent operating in the **ALFRED Embodied Environment**. Your task is to: {task_description}

Prior to this step, you have already taken {step_count} **step(s).**

Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}

You are now at step {current_step} and your current observation is: {current_observation}

Your admissible actions of the current situation are: {admissible_actions}.

Your previous overall plan is: {planning}. Please strictly adhere to your plan.

Now it's your turn to take an action, following these steps:

1. **First, reason using ONLY ONE tag pair and express your reasoning in one concise, brief sentence:**

- **<planning>** Plan or replan the entire task by breaking it down into high-level steps. Focus on outlining the full sequence required to complete the overall task, not just the immediate next action. Use this at the beginning of complex tasks or whenever the previous plan is incorrect or insufficient. It is necessary to list all the points separately. eg, step 1: xxx, step 2: xxx, step 3: xxx, etc.
- **<explore>** When results are unexpected or information is lacking, use current observations to think outside the box and list as many possible locations, items, or actions as possible. Use this approach when facing obstacles that require creative and innovative thinking.
- **<reflection>** Analyze the reasons for errors in task execution and correct them by exploring alternative approaches. 'No known action matches that input.' indicates the action is invalid. This is typically used when several consecutive actions yield no substantial progress.
- **<monitor>** Continuously track the current progress and history of reasoning and execution throughout the task. Recall the current subgoal and consider the

next concrete action, ensuring agent alignment with the overall plan. Typically used when task outcomes are as expected and no other mode of reasoning is required.

2. After your reasoning, you *MUST* select and present an admissible action for the current step within `<action> ... </action>` tags.

Specify the next action the agent should take to progress toward the task goal, following these guidelines:

(a) **Object and Receptacle References:** Use specific identifiers:

- [obj id] for objects (e.g., apple 1).
- [recep id] for receptacles (e.g., countertop 1).

(b) **Action Validity:** Follow the exact format below. Any deviation renders the action invalid:

- Valid actions: go to [recep id], take [obj id] from [recep id], put [obj id] in/on [recep id], open/close [recep id], use [obj id], heat/cool/clean [obj id] with [recep id].

Prompt Template for ScienceWorld Environment

You are an expert agent operating in the **ScienceWorld** environment, which is a text-based virtual environment centered around accomplishing tasks from the elementary science curriculum.

Your current task is: {task_description}

Prior to this step, you have already taken {step_count} **step(s).**

Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}

You are now at step {current_step} and your current observation is: {current_observation}

Here are the actions you may take:

- {"action": "open OBJ", "description": "open a container"}
- {"action": "close OBJ", "description": "close a container"}
- {"action": "activate OBJ", "description": "activate a device"}
- {"action": "deactivate OBJ", "description": "deactivate a device"}
- {"action": "connect OBJ to OBJ", "description": "connect electrical components"}
- {"action": "disconnect OBJ", "description": "disconnect electrical components"}
- {"action": "use OBJ [on OBJ]", "description": "use a device/item"}
- {"action": "look around", "description": "describe the current room"}
- {"action": "look at OBJ", "description": "describe an object in detail"}
- {"action": "look in OBJ", "description": "describe a container's contents"}
- {"action": "read OBJ", "description": "read a note or book"}
- {"action": "move OBJ to OBJ", "description": "move an object to a container"}

- {"action": "pick up OBJ", "description": "move an object to the inventory"}
- {"action": "put down OBJ", "description": "drop an inventory item"}
- {"action": "pour OBJ into OBJ", "description": "pour a liquid into a container"}
- {"action": "dunk OBJ into OBJ", "description": "dunk a container into a liquid"}
- {"action": "mix OBJ", "description": "chemically mix a container"}
- {"action": "go to LOC", "description": "move to a new location"}
- {"action": "eat OBJ", "description": "eat a food"}
- {"action": "flush OBJ", "description": "flush a toilet"}
- {"action": "focus on OBJ", "description": "signal intent on a task object"}
- {"action": "wait", "description": "take no action for 10 iterations"}
- {"action": "wait1", "description": "take no action for 1 iteration"}
- {"action": "task", "description": "describe current task"}
- {"action": "inventory", "description": "list your inventory"}

Your previous overall plan is: {planning}.

Please strictly adhere to your plan.

Now it's your turn to take an action, following these steps:

1. **First, reason using *ONLY ONE* tag pair and express your reasoning in one concise, brief sentence:**

- <planning>
Plan or replan the entire task by breaking it down into high-level steps. Focus on outlining the full sequence required to complete the overall task, not just the immediate next action.
Use this at the beginning of complex tasks or whenever the previous plan is incorrect or insufficient.
It is necessary to list all the points separately. eg, step 1: xxx, step 2: xxx, step 3: xxx, etc.
- <explore>
When results are unexpected or information is lacking, use current observations to think outside the box and list as many possible locations, items, or actions as possible.
Use this approach when facing obstacles that require creative and innovative thinking.
- <reflection>
Analyze the reasons for errors in task execution and correct them by exploring alternative approaches. 'No known action matches that input.' indicates the action is invalid.
This is typically used when several consecutive actions yield no substantial progress.
- <monitor>
Continuously track the current progress and history of reasoning and execution

throughout the task. Recall the current subgoal and consider the next concrete action, ensuring agent alignment with the overall plan.

Typically used when task outcomes are as expected and no other mode of reasoning is required.

2. **After your reasoning, you *MUST* select and present an appropriate action for the current step within `<action> ...</action>` tags.**