

SPARK: Strategic Policy-Aware Exploration via Dynamic Branching for Long-Horizon Agentic Learning

Jinyang Wu^{1*}, Shuo Yang^{1*}, Yuhao Shen²,
Shuai Zhang¹, Zhengqi Wen¹, Jianhua Tao^{1†}

¹Tsinghua University ²Zhejiang University
{wu-jy23, shuo-yan20}@mails.tsinghua.edu.cn

Abstract

Reinforcement learning has empowered large language models to act as intelligent agents, yet training them for long-horizon tasks remains challenging due to the scarcity of high-quality trajectories, especially under limited resources. Existing methods typically scale up rollout sizes and indiscriminately allocate computational resources among intermediate steps. Such attempts inherently waste substantial computation budget on trivial steps while failing to guarantee sample quality. To address this, we propose **SPARK** (Strategic Policy-Aware exploration via **Key**-state dynamic branching), a novel framework that selectively branches at critical decision states for resource-efficient exploration. Our key insight is to activate adaptive branching exploration at critical decision points to probe promising trajectories, thereby achieving precise resource allocation that prioritizes sampling quality over blind coverage. This design leverages the agent’s intrinsic decision-making signals to reduce dependence on human priors, enabling the agent to autonomously expand exploration and achieve stronger generalization. Experiments across diverse tasks (e.g., embodied planning), demonstrate that SPARK achieves superior success rates with significantly fewer training samples, exhibiting robust generalization even in unseen scenarios. Our code and checkpoints are available at <https://github.com/jinyangwu/SPARK>.

1 Introduction

Reinforcement Learning (RL) has proven effective in enhancing Large Language Model (LLM) reasoning capabilities, with notable success in domains such as mathematics (Guo et al., 2025; OpenAI, 2024b) and coding (Zhou et al., 2025). This has led to a growing consensus that RL holds the key to the next frontier (Zhang et al., 2025b; Yue et al., 2025): *agentic AI*—systems

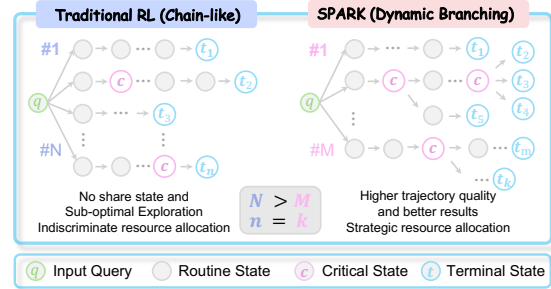


Figure 1: **Paradigm comparison: uniform vs. strategic exploration.** Standard RL (Left) wastes budget via uniform, independent sampling. In contrast, SPARK (Right) employs dynamic branching at critical junctures for precise resource allocation, yielding higher-quality trajectories under the similar computational budget.

capable of autonomously navigating dynamic environments and executing complex, long-horizon instructions (Zhang et al., 2025a; Wu et al., 2026b).

However, applying RL to agentic tasks faces a fundamental bottleneck: *the scarcity of high-quality trajectories under resource constraints*. Unlike math problems where solutions are self-contained and directly verifiable (Zhang et al., 2025b), agentic tasks require navigating vast state spaces where a single misstep can derail a long sequence (Feng et al., 2025; Zhang et al., 2025e). Consequently, successful trajectories are sparse, making it difficult for the policy to learn effectively (MoonshotAI, 2025; Wu et al., 2025).

Existing methods attempt to mitigate this challenge by expanding exploration budget and employing extensive search-based techniques during RL training (Xing et al., 2025; Ji et al., 2025). Yet, these approaches suffer from *indiscriminate resource allocation*: they distribute computational budget uniformly across all steps (Figure 1), rather than targeting decision points where additional exploration is most valuable. Consider an embodied agent tasked with preparing breakfast (Salimpour et al., 2025; Liang et al., 2025): uniform explo-

* Equal Contribution.

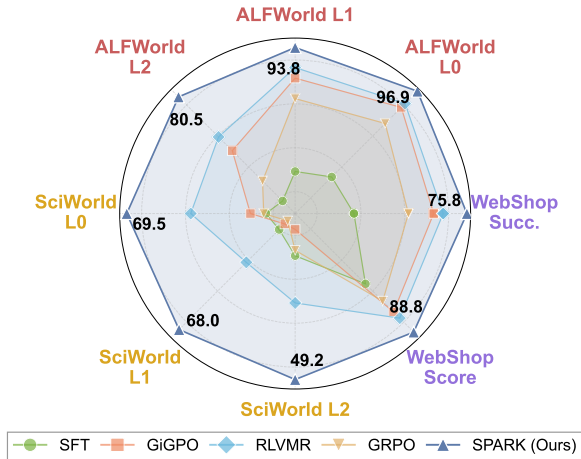


Figure 2: **Multi-benchmark performance comparison.** SPARK outperforms all baselines across ALFWorld (L0-L2), ScienceWorld (L0-L2), and WebShop tasks, achieving +73.5% average improvement.

ration wastes substantial resources on trivial steps like “opening the fridge door” while allocating insufficient budget to pivotal states like “choosing alternative ingredients when the intended ones are missing.” By prioritizing exhaustive coverage over sampling quality, such methods fail to yield high-value trajectories under resource constraints, resulting in inefficient exploration and unstable training.

We argue that effective exploration should be triggered at appropriate decision points. Rather than blind search, an intelligent agent should autonomously identify critical junctures at which embarking on additional exploration branches yields the greatest potential benefits. Building on this principle, we introduce **SPARK** (Strategic Policy-Aware expLoRation via **Key**-state dynamic branching), a novel framework that enables autonomous strategic exploration via dynamic branching at critical intermediate states, which we refer to as SPARK points. Our key insight is to leverage the agent’s intrinsic decision-making signals to govern the exploration topology. When encountering states with high epistemic uncertainty or semantic ambiguity, the agent selectively initiates additional exploration; otherwise, it proceeds linearly through routine decisions. This achieves precise resource allocation that prioritizes sampling quality over exhaustive coverage while minimizing reliance on human priors, thereby ensuring robust generalization. Extensive experiments show that SPARK significantly enhances performance, efficiency, and out-of-domain generalization over powerful baselines. Our contributions are three-fold:

- **Autonomous Strategic Exploration:** We propose an agentic RL framework that enables agents to autonomously identify intermediate states where additional exploration is needed, relying on intrinsic decision-making signals rather than handcrafted heuristics, thereby enabling precise and efficient resource allocation.
- **Adaptive Dynamic Branching:** By dynamically triggering branching at critical states, we achieve superior sample quality under constrained budget, generating more informative training trajectories than blind exploration.
- **Compelling Strong Empirical Validation:** Experiments (Figure 2) on challenging tasks (e.g., embodied planning in ALFWorld) demonstrate that SPARK achieves superior success rates with higher sample efficiency, exhibiting robust generalization even in unseen scenarios.

2 Methodology

We first formulate the agentic task as a partially observable decision process (§2.1), then introduce our core mechanism: *Dynamic Branching Exploration* that selectively allocates exploration budget based on intrinsic decision-making signals (§2.2). Figure 3 provides the overview of SPARK.

2.1 Problem Formulation

We formulate long-horizon agentic tasks as a Partially Observable Markov Decision Process (POMDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \gamma)$, where \mathcal{S} is the latent state space, \mathcal{A} is the action space, \mathcal{O} is the observation space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is discount factor.

At each timestep t , the environment is in a hidden state $s_t \in \mathcal{S}$. The agent receives observation $o_t \in \mathcal{O}$ (e.g., textual feedback), and maintains an interaction history $h_t = (o_0, z_0, a_0, \dots, o_t)$, where z_i denotes the reasoning trace and $a_i \in \mathcal{A}$ is the action at step i . This history serves as the belief state proxy. Then, the agent (policy model π_θ) generates a reasoning trace z_t followed by an action a_t :

$$(z_t, a_t) \sim \pi_\theta(\cdot \mid h_t). \quad (1)$$

Upon executing action a_t , the environment transitions to s_{t+1} according to $\mathcal{T}(s_{t+1} \mid s_t, a_t)$ and provides observation o_{t+1} . We represent a single

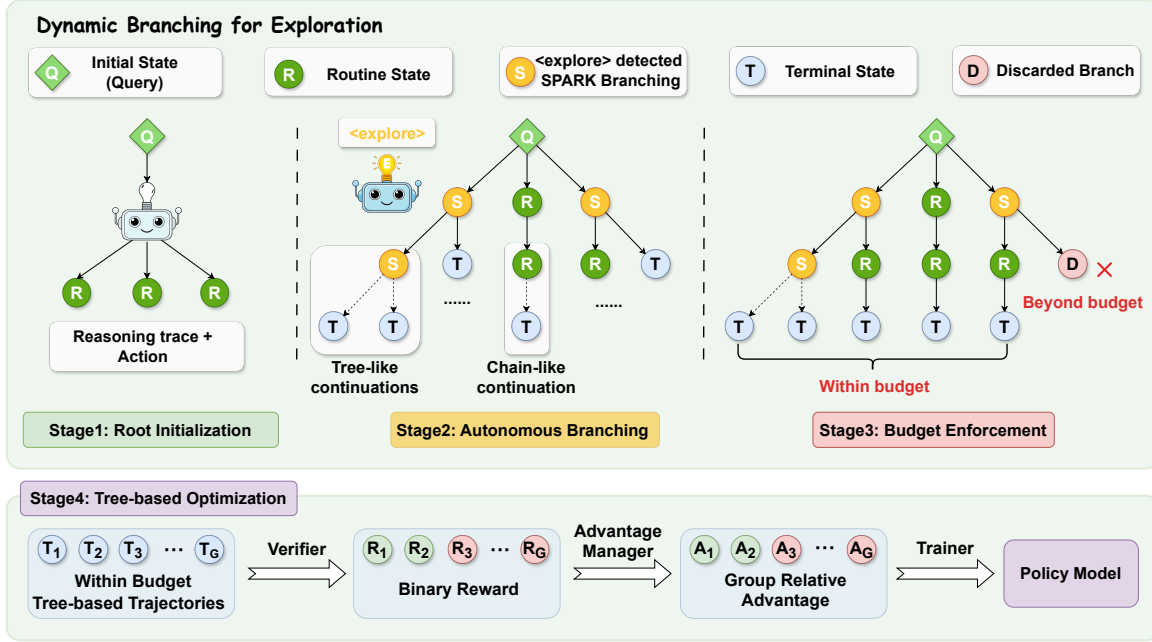


Figure 3: **Overview of SPARK framework.** SPARK performs dynamic branching exploration via: (1) *Root Initialization*: diverse starting trajectories; (2) *Autonomous Branching*: selective expansion at high-uncertainty states using intrinsic <explore> signals; (3) *Budget Enforcement*: constraining tree growth within computational limits. The resulting trajectory trees are then used for *Tree-Based Policy Optimization*.

step as the triplet $\tau_t = (o_t, z_t, a_t)$. An episode continues until the agent achieves the goal or reaches the maximum horizon K , yielding a trajectory $\tau = (\tau_0, \tau_1, \dots, \tau_T)$ where $T \leq K$.

For most agentic benchmarks (Singh et al., 2025), the reward function provides sparse terminal feedback: $R(\tau) \in \{0, 1\}$ indicating task success or failure. Our objective is to find an optimal policy π^* that maximizes the expected return:

$$J(\pi) = \mathbb{E}_{\tau \sim \pi}[R(\tau)]. \quad (2)$$

2.2 Dynamic Branching Exploration

Uniform exploration across all steps is highly inefficient in long-horizon RL (Wang et al., 2020). Trivial steps (e.g., open the door in front of a fridge) require minimal search, while critical junctures (e.g., choosing alternative ingredients when the intended ones are missing) benefit substantially from exploring multiple alternatives. Standard methods allocate resources uniformly, wasting computation on routine actions while under-exploring pivotal decisions. We address this through *dynamic branching*: a mechanism that constructs **hierarchical tree-structured exploration paths** by selectively allocating rollout budget at **pivotal decision points (SPARK points)**, guided by the agent’s intrinsic decision-making signals. Supervised fine-tuning

(SFT) is first employed to endow the model with the preliminary capability to generate such intrinsic signals, followed by RL optimization.

Let N denote the total rollout budget (maximum leaf nodes) and K denote the maximum episode length. Rather than sampling N independent trajectories, SPARK builds a trajectory forest that branches at states where additional exploration is valuable. Specifically, we operate in three stages:

Stage 1: Root Initialization. At initial state s_0 , task ambiguity is typically highest. To ensure exploration diversity, we create M parallel trajectory roots by sampling distinct reasoning-action pairs:

$$\{(z_0^{(i)}, a_0^{(i)})\}_{i=1}^M \sim \pi_\theta(\cdot | h_0), \quad (3)$$

where $M < N$ (typically $M = 3$ to 5). This establishes M independent trajectory trees.

Stage 2: Autonomous Branching. For each active trajectory i at step t , the agent generates a reasoning trace $z_t^{(i)}$ that internally deliberates whether the current decision requires additional exploration. We employ a branching criterion $\mathcal{B}(\cdot)$ that analyzes $z_t^{(i)}$ to determine exploration intensity:

$$b_t^{(i)} = \mathcal{B}(z_t^{(i)}), \quad (4)$$

Algorithm 1 SPARK: Strategic Policy-Aware Exploration via Key-State Dynamic Branching

Input: Policy π_θ ; task dataset \mathcal{Q} ; rollout budget N ; branching factor B ; initial roots M

Output: Optimized policy π_θ^*

```
1: for iteration = 1, 2, . . . do
2:    $\mathcal{D} \leftarrow \emptyset$ 
3:   for task  $q \sim \mathcal{Q}$  ▷ Dynamic Branching Exploration
4:     Initialize  $M$  parallel trajectory roots from  $q$ 
5:     for each active trajectory  $i$  at step  $t$  do
6:       Generate reasoning trace and action  $(z_t^{(i)}, a_t^{(i)}) \sim \pi_\theta(\cdot | h_t^{(i)})$ 
7:       Compute branching factor  $b_t^{(i)} \leftarrow \mathcal{B}(z_t^{(i)})$  upon triggering <explore> in  $z_t^{(i)}$ 
8:       Get effective branching factor  $b_{t,\text{eff}}^{(i)}$  under budget constraint  $N$  via Eq. (7)
9:       Prepare context for  $b_{t,\text{eff}}^{(i)}$  continuations and update active trajectory mask
10:    end for
11:    Collect completed trajectory group  $G$  into  $\mathcal{D}$ 
12:  end for
13:  Update  $\theta$  using GRPO objective with collected trajectories ▷ Tree-Based Policy Update
14: end for
```

where $b_t^{(i)} \in \{1, B\}$ specifies the branching factor—either linear continuation ($b = 1$) or multi-branch exploration ($b = B$).

Branching Criterion. The function \mathcal{B} identifies states where exploration is valuable by detecting signals of epistemic uncertainty in the reasoning trace, which is explicitly manifested as a dedicated <explore> tag. When uncertainty is detected, we set $b_t^{(i)} = B$; otherwise $b_t^{(i)} = 1$. Function \mathcal{B} can be formalized as:

$$\mathcal{B}(z_t^{(i)}) = \begin{cases} B, & \text{if <explore> in } z_t^{(i)} \\ 1, & \text{otherwise.} \end{cases} \quad (5)$$

Based on $b_t^{(i)}$, we sample continuations:

$$\{(z_{t+1}^{(i,j)}, a_{t+1}^{(i,j)})\}_{j=1}^{b_t^{(i)}} \sim \pi_\theta(\cdot | h_{t+1}^{(i)}), \quad (6)$$

where $h_{t+1}^{(i)} = (h_t^{(i)}, z_t^{(i)}, a_t^{(i)}, o_{t+1}^{(i)})$ is the updated history. This creates $b_t^{(i)}$ child trajectories from the current node.

Stage 3: Budget Enforcement. Expansion is constrained by the global budget N . Let N_{current} denote the current number of active leaf nodes. The actual branching factor is adjusted dynamically:

$$b_{t,\text{eff}}^{(i)} = \min(b_t^{(i)}, N - N_{\text{current}} + 1). \quad (7)$$

This ensures we never exceed budget N . Rollout terminates when a trajectory reaches a terminal state (success/failure) or maximum horizon K .

This process yields a collection of M trajectory trees containing $|G|$ completed trajectories, where $M \leq |G| \leq N$. We denote this collection as a group $G = \{\tau^{(1)}, \tau^{(2)}, \dots, \tau^{(|G|)}\}$.

Stage 4: Tree-Based Policy Update. The trajectory trees generated by SPARK are used to update the policy without modifying the underlying optimization algorithm. Completed leaf trajectories from the same task are grouped and assigned binary terminal rewards, which are propagated to all steps. Shared prefixes induce comparable alternatives, enabling relative credit assignment via group-normalized advantages. Policy updates are performed using standard clipped optimization with KL term against a reference policy, remaining compatible with existing GRPO-style pipelines while incorporating decision-focused exploration signals.

2.3 Theoretical Perspective on SPARK

We present a concise theoretical argument explaining why SPARK achieves substantially improved performance under *consistently lower token budgets* in long-horizon decision-making tasks.

Key Insight. Let K denote the episode horizon and $\mathcal{C} \subseteq \{0, \dots, K-1\}$ the set of *pivotal decision steps* (SPARK points) whose action choices critically affect task success, with $|\mathcal{C}| = m \ll K$. Standard GRPO-style linear sampling allocates computation uniformly across all steps of a trajectory. In contrast, SPARK redistributes intermediate computation toward pivotal steps without increasing the

final number of completed trajectories.

Decision Coverage at Pivotal Steps. At a pivotal step $t \in \mathcal{C}$, let $q_t \in (0, 1)$ denote the probability mass assigned by the current policy to desirable actions when sampled once. Under linear sampling, each trajectory attempts this decision only once, yielding a coverage probability of q_t .

In SPARK, dynamic branching generates B conditional continuations at pivotal steps, increasing the probability of selecting at least one desirable action to

$$q_t^{\text{branch}} = 1 - (1 - q_t)^B, \quad (8)$$

which strictly dominates q_t for any $B \geq 2$.

Implications for Long-Horizon Tasks. Task success typically depends on a sequence of correct decisions at pivotal steps. By replacing each q_t with q_t^{branch} , SPARK amplifies the probability of discovering high-quality trajectories through a multiplicative effect across \mathcal{C} . Meanwhile, SPARK avoids repeatedly rolling out suffixes induced by suboptimal pivotal decisions, leading to fewer redundant computations at non-pivotal steps and a modest reduction in overall token usage.

3 Experiments

We analyze our framework from four aspects: (1) **Performance**: superior success rates versus strong baselines; (2) **Efficiency**: sample and token efficiency; (3) **Generalization**: robustness on unseen scenarios; (4) **Ablation Study**: contribution of dynamic branching and exploration quality analysis.

3.1 Experimental Settings

Benchmarks. Following prior work (Feng et al., 2025; Zhang et al., 2025e), we evaluate on three challenging long-horizon agentic domains: (1) **Embodied Decision Making**: ALFWorld (Shridhar et al., 2020) for household task execution; (2) **Scientific Reasoning**: ScienceWorld (Wang et al., 2022) with complex task horizons up to 30+ steps; (3) **Web Navigation**: WebShop (Yao et al., 2022a) with 1.1M products to test open-ended exploration.

Baselines. We compare SPARK against: (1) **Closed-source LLMs**: GPT-4o (OpenAI, 2024a), GPT-5-mini (OpenAI, 2025), GPT-5 (OpenAI, 2025), Gemini-2.5-Pro (Comanici et al., 2025); (2) **Prompting Methods**: ReAct (Yao et al., 2022b); and (3) **RL Methods**: GRPO (Shao et al., 2024),

ETO (Song et al., 2024), GiGPO (Feng et al., 2025), and RLVMR (Zhang et al., 2025e).

Evaluation Metrics. Following Shridhar et al. (2020); Yao et al. (2022a), we report *Success Rate (SR)* for task completion and *Average Score* (WebShop only) for partial success (e.g., selecting a product with correct attributes but wrong color).

Implementation Details. We use Qwen2.5-1.5B/7B-instruct (Qwen Team, 2024) as base models. A cold-start SFT with 300 trajectories enables initial `<explore>` tag generation in RL. We set $N = 8$, $M = 4$, $B = 2$ for rollout budget, initial roots, and branching factor, respectively. RL training runs for 120 steps with batch size 16. Full details are provided in Appendix B.4.

3.2 Main Results

Table 1 presents results across long-horizon agentic planning tasks, revealing three key findings:

✧ *SPARK establishes leading performance across model scales and task domains.* SPARK consistently outperforms prompting (ReAct), SFT, and RL methods (GiGPO, RLVMR) on both 1.5B and 7B backbones. Notably, SPARK-1.5B attains 49.2% on the hardest ScienceWorld L2, exceeding GPT-5 (33.6%) and Gemini-2.5-Pro (30.5%). This reveals that strategic exploration enables smaller models to rival significantly larger proprietary systems.

✧ *Dynamic branching at critical states substantially outperforms uniform exploration.* Compared to GRPO’s uniform sampling, SPARK yields substantial gains on 7B: +23.3% on ALFWorld Look and +39.4% on Pick2. The amplified improvement on Pick2 (multi-step coordination) versus Look (simple navigation) validates our hypothesis: selective branching allocates budget to critical steps where additional search provides maximum value.

✧ *SPARK exhibits robust generalization to unseen environments.* On L2 out-of-domain tasks, SPARK-1.5B significantly outperforms GRPO on ALFWorld (80.5% vs. 29.7%). On the highly complex ScienceWorld L2, SPARK (49.2%) establishes a $10.5\times$ lead over GiGPO (4.7%) and $1.9\times$ over RLVMR (26.5%). This highlights the effectiveness of autonomous strategic exploration in novel scenarios where conventional methods struggle.

3.3 Exploration Efficiency Analysis

Sample Efficiency. To assess whether SPARK’s strategic exploration enhances sample efficiency, we evaluate performance under varying data scales

Method	ALFWorld						ScienceWorld			WebShop	
	Look	Clean	Pick2	L0	L1	L2	L0	L1	L2	score	succ.
<i>Closed-Source Models</i>											
GPT-4o	50.0	34.4	15.8	52.3	48.4	38.3	32.0	32.8	32.8	12.7	6.2
GPT-5-mini	58.3	28.1	52.6	57.0	54.7	46.1	39.8	31.3	31.4	14.2	7.0
GPT-5	75.0	43.8	57.9	70.0	60.2	63.3	53.9	33.6	33.6	34.9	29.7
Gemini-2.5-Pro	75.0	40.6	84.2	56.3	55.0	55.5	32.8	27.3	30.5	42.9	32.0
<i>Advanced Method Comparison</i>											
<i>Qwen2.5-1.5B</i>											
ReAct	18.3	10.8	0.0	11.3	13.7	10.2	1.2	0.8	0.8	40.1	11.3
SFT	42.0	35.9	29.7	43.0	38.7	17.6	20.3	18.0	12.5	65.8	39.1
ETO	46.3	66.2	55.6	64.1	66.4	25.8	39.1	22.7	15.6	-	-
GiGPO	66.5	90.7	73.8	86.7	83.2	48.0	25.8	15.2	4.7	83.1	65.0
RLVMR	78.8	91.2	77.6	89.1	87.9	56.3	46.9	34.4	26.5	86.7	71.1
GRPO	55.6	88.1	72.1	76.6	71.1	29.7	21.1	13.7	10.9	75.8	56.8
SPARK (Ours)	91.7	100.0	89.5	96.9	93.8	80.5	69.5	68.0	49.2	88.8	75.8
Δ vs. GRPO	+36.1	+11.9	+17.4	+20.3	+22.7	+50.8	+48.4	+54.3	+38.3	+13.0	+19.0
<i>Qwen2.5-7B</i>											
ReAct	33.2	18.7	12.8	23.1	28.5	27.0	7.8	11.3	6.3	46.2	19.5
SFT	63.0	61.1	33.2	63.3	57.0	37.5	36.7	32.0	23.4	62.2	38.3
ETO	70.5	82.3	51.2	70.3	74.2	51.6	62.5	40.6	28.1	-	-
GiGPO	85.9	93.3	83.6	89.5	90.2	67.2	53.4	35.2	25.8	84.4	72.8
RLVMR	88.2	90.1	86.7	91.4	91.8	83.6	67.2	43.0	32.2	87.3	74.2
GRPO	76.7	86.0	56.4	79.3	77.3	52.3	49.1	30.1	26.6	79.3	66.1
SPARK (Ours)	100.0	92.5	95.8	96.1	92.2	88.3	75.0	74.1	57.8	89.8	82.8
Δ vs. GRPO	+23.3	+6.5	+39.4	+16.8	+14.9	+36.0	+25.9	+44.0	+31.2	+10.5	+16.7

Table 1: **Performance comparison on long-horizon agentic tasks.** We evaluate success rates (%) on ALFWorld and ScienceWorld, and task completion score/success rate on WebShop. **L0/L1/L2** denote seen task categories and instance variants, seen task categories but unseen instance variants, unseen task categories and instance variants, respectively. Look, Clean, and Pick2 represent distinct task types in ALFWorld. Best results per model are in **bold**.

on ALFWorld and ScienceWorld L0 (1.5B). As shown in Figure 4, SPARK demonstrates dramatic efficiency advantages. For example, on ALFWorld, with only 20% training data, SPARK achieves 84.4% success rate, surpassing GRPO’s 76.6% at full data (100%). At 40% data, SPARK reaches 89.1%, matching RLVMR and exceeding GiGPO (86.7%) at full data. In contrast, baselines collapse at low data regimes: GRPO and GiGPO achieve only 22.7% at 20% data, revealing their inability to efficiently learn from limited samples. This 5 \times data reduction over GRPO confirms that prioritizing exploration at critical decision points enables higher sample efficiency (details in Appendix C.1).

Token Efficiency. Table 2 compares token consumption of SPARK against chain-like methods. By sharing common prefixes across trajectories, our method substantially reduces token generation: 6.9%, 47.0%, and 11.2% on ALFWorld, ScienceWorld, and WebShop, respectively. These results

Method	ALFWorld \downarrow	ScienceWorld \downarrow	WebShop \downarrow
GRPO	100.0	100.0	100.0
SPARK	93.1	53.0	88.8
Reduction	6.9	47.0	11.2

Table 2: **Token efficiency comparison on 1.5B backbone.** We report the relative token consumption of SPARK compared to chain-like methods, with the latter normalized to 100 for clear comparison.

validate our core thesis: by concentrating the exploration budget at critical decision points rather than uniformly across all steps, dynamic branching not only improves sample quality but also eliminates redundant token generation on routine actions, thereby maximizing both performance and computational efficiency with limited resources.

3.4 Cross-Domain Generalization

To further evaluate generalization to unseen scenarios, we analyze performance degradation from in-domain (ID, L0) to out-of-domain (OOD, L2)

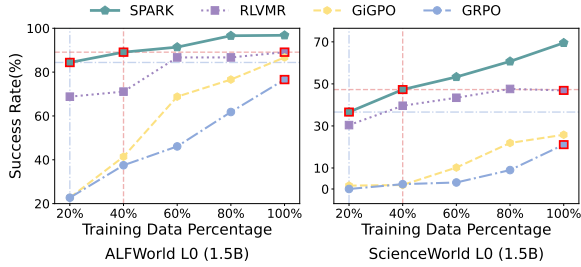


Figure 4: **Sample efficiency comparison.** SPARK surpasses GRPO@100% using only 20% data, while baselines often collapse in low-data regimes.

Method	ALFWorld			ScienceWorld		
	ID \uparrow	OOD \uparrow	Δ \downarrow	ID \uparrow	OOD \uparrow	Δ \downarrow
SFT	43.0	17.6	59.1%	20.3	12.5	38.4%
GRPO	76.6	29.7	61.2%	21.1	10.9	48.3%
Ours	96.9	80.5	16.9%	69.5	49.2	29.2%

Table 3: **Cross-domain generalization comparison.** Δ denotes the performance drop ratio from seen (in-domain, ID) to unseen scenarios (out-of-domain, OOD).

splits. As shown in Table 3, GRPO exhibits severe overfitting with performance drops of 61.2% on ALFWorld and 48.3% on ScienceWorld. In contrast, SPARK maintains robust performance, limiting degradation to 16.9% and 29.2% respectively. Moreover, SPARK achieves the highest absolute OOD success rates (80.5% and 49.2%), substantially outperforming GRPO (29.7% and 10.9%). These results indicate that SPARK’s dynamic branching enables effective learning of transferable exploration strategies rather than memorization of task-specific trajectories, yielding policies that generalize well to novel environments.

3.5 Ablation Study and Analysis

Impact of Dynamic Branching. To verify dynamic branching, we compare SPARK against a “Fixed Probability” baseline that randomly triggers branching at every step with fixed probability, independent of decision criticality. As shown in Table 4, stochastic branching degrades performance across all datasets, most severely on ScienceWorld (69.5% \rightarrow 45.3%) where long horizons amplify the costs of misallocated exploration. This validates our thesis: effective exploration requires *selective* branching at high-uncertainty states rather than uniform expansion, as fixed methods waste budget on trivial steps while under-exploring critical states.

Impact of Initial Roots. We investigate the trade-off between initial diversity and subsequent exploration depth by varying initial roots (M) under

Method	ALFWorld \uparrow	ScienceWorld \uparrow	WebShop \uparrow
Dynamic	96.9	69.5	75.8
Fixed	90.6	45.3	70.3
Δ	-6.3	-24.2	-5.5

Table 4: **Ablation on RL branching strategy.** We report success rates (%) on 1.5B model. Fixed branching randomly triggers expansion at each step with constant probability, independent of state uncertainty.

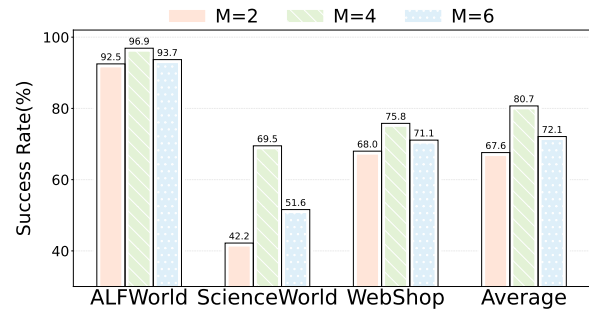


Figure 5: **Ablation on initial root count (M).** We report success rates (%) with fixed total budget $N = 8$.

fixed budget $N = 8$. As shown in Figure 5, performance peaks at $M = 4$ (80.7%), but drops to 67.6% at $M = 2$ and 72.1% at $M = 6$. Too few roots ($M = 2$) constrain initial diversity, risking early convergence to local optima despite remaining budget. Conversely, excessive roots ($M \geq 6$) exhaust budget at initialization, leaving insufficient resources for later branching. This reveals that the sweet spot $M = 4$ balances broad initial coverage with deep adaptive search, splitting budget between diverse starts and strategic exploration.

B	ALFWorld L0	WebShop Succ.	WebShop Score
2	96.9	75.8	88.8
3	94.5	73.4	87.6
4	93.8	72.7	85.5

Table 5: **Sensitivity analysis on branching factor B** (1.5B backbone). Performance drops as B increases due to premature budget exhaustion.

Impact of Branching Factor. We conduct an additional sensitivity analysis on the branching factor B , which governs the number of parallel continuations generated at each triggered `<explore>` state. As shown in Table 5, increasing B from 2 to 4 leads to a consistent performance decline across tasks. This occurs because a larger B rapidly exhausts the global leaf budget N at a single critical state, leaving insufficient resources for exploration at subsequent pivotal steps later in the trajectory.

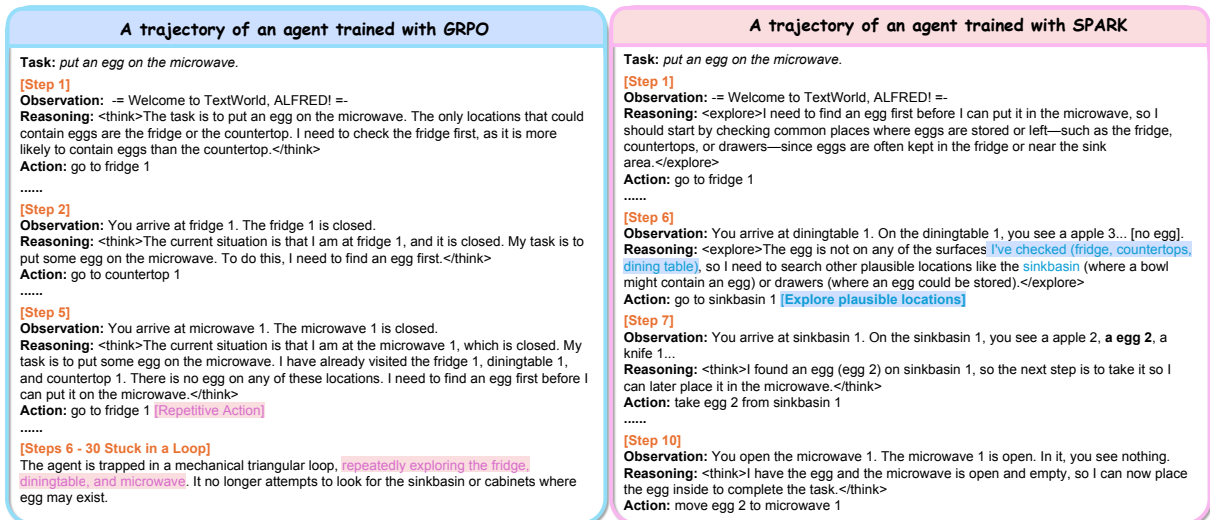


Figure 6: Qualitative comparison on the embodied planning task (ALFWorld): “put an egg on the microwave”.

Method	1.5B			7B		
	L0 ↓	L1 ↓	L2 ↓	L0 ↓	L1 ↓	L2 ↓
SFT	10.7	20.2	21.4	13.9	24.5	14.4
GRPO	18.4	17.6	27.1	21.5	20.3	31.2
Ours	10.1	7.0	15.4	10.0	8.6	14.1

Table 6: **Repetitive action ratio on ALFWorld** (lower is better). SPARK produces fewer repetitive actions on in-domain (L0) and out-of-domain (L1, L2) splits.

This phenomenon mirrors our findings on initial root count M (Section 3.5): just as an excessive M prematurely depletes the budget at initialization, an overly large B over-allocates resources locally, impairing overall trajectory quality. Thus, $B = 2$ strikes the optimal balance between local exploration depth and global budget preservation.

Exploration Quality: Repetitive Action Ratio. Beyond success rates, we analyze exploration quality via repetitive action rates. Table 6 shows that SPARK produces significantly fewer repetitive actions across model scales. On 1.5B’s L2 (OOD), SPARK achieves 15.4% repetitive ratio versus GRPO’s 27.1% (43% reduction). By concentrating search at uncertain states, SPARK explores more purposefully, avoiding redundant trial-and-error on routine actions. The gap widens on harder splits (L1, L2), confirming that strategic exploration particularly benefits novel scenarios where indiscriminate search is wasteful.

Triggering Frequency Evolution. We track the evolution of the exploration ratio (proportion of

Training Steps	1	30	60	90	120	150
ALFWorld	0.34	0.37	0.41	0.48	0.47	0.29
WebShop	0.45	0.30	0.22	0.14	0.20	0.17

Table 7: **Evolution of <explore> triggering ratio during training.** The mechanism self-regulates, decreasing as the policy matures.

steps triggering <explore> per trajectory) throughout training. As shown in Table 7, the frequency self-regulates rather than saturating or vanishing. On ALFWorld, the ratio rises initially (Steps 1–90) as the policy explores diverse strategies, then drops at Step 150 (0.29), indicating *policy consolidation*: as the policy improves, the agent autonomously reduces reliance on branching, reserving it for genuinely uncertain states. On WebShop, the model rapidly identifies that excessive branching is inefficient for linear search-click tasks, converging quickly to a lean exploration rate (~ 0.17). This self-regulating behavior demonstrates that the <explore> mechanism dynamically calibrates to task complexity and policy competency.

Qualitative Analysis. Figure 6 illustrates exploration behaviors on an ALFWorld egg-retrieval task. The GRPO-trained agent becomes trapped in a mechanical loop (Steps 6-30), repeatedly checking the same three locations without exploring alternatives like sinkbasin or cabinets. In contrast, SPARK exhibits strategic exploration: at Step 6, upon encountering uncertainty about egg location, the agent emits an <explore> signal and reasons about plausible locations (sinkbasin, drawers). This triggers a

systematic search across multiple candidates, successfully locating the egg at Step 7 and completing the task by Step 10. This reveals that autonomous identification of critical decision points enables purposeful exploration that avoids redundant loops.

4 Related Work

Agentic Reinforcement Learning. Applying RL to train LLM-based agents has shown promise in reasoning tasks (Shao et al., 2024), yet long-horizon scenarios remain challenging due to trajectory scarcity with constrained resources. To obtain better training signals, recent work explores process supervision via Process Reward Models (Zhang et al., 2025e,d) for dense intermediate feedback, but requires expensive annotations that scale poorly to open-ended environments (Khalifa et al., 2025; Shridhar et al., 2020; Zhou et al., 2024). Alternative methods attempt to generate high-quality trajectories by expanding exploration budget (Xing et al., 2025; Ji et al., 2025), but they suffer from indiscriminate resource allocation that treats all steps uniformly. Our method enables agents to autonomously allocate exploration budget, reducing reliance on external supervision while improving sample quality.

Strategic Exploration. Effective exploration is critical for discovering optimal policies in vast state-action spaces. Inference-time search methods (Zhang et al., 2025c; Wu et al., 2024), such as ToT (Yao et al., 2023) and RAP (Hao et al., 2023), have demonstrated the value of exploring diverse reasoning paths. Recent efforts aim to internalize this capability into training through tree-structured RL (Hou et al., 2025; Wu et al., 2025). However, these methods typically apply uniform branching at every step, wasting considerable resources on trivial steps while often under-exploring critical strategic junctures (Ji et al., 2025). In contrast, SPARK leverages intrinsic decision-making signals to selectively branch only at high-uncertainty states. This design prioritizes sampling quality over exhaustive coverage, achieving superior trajectory quality under budget constraints.

5 Conclusion

We present SPARK, a novel RL framework for long-horizon agentic learning. By enabling strategic exploration through dynamic branching, SPARK selectively expands exploration at critical decision points and achieves precise resource allocation.

Empirical results confirm that SPARK achieves superior success rates with better exploration efficiency, exhibiting robust generalization. Our work highlights strategic exploration’s value in advancing capable and efficient agentic systems.

Limitations

Our study is comprehensive, but has certain limitations that we plan to address in future research. In this study, we leverage the agent’s intrinsic decision-making signals to identify critical decision points for dynamic branching. While this design minimizes reliance on external supervision and enables autonomous strategic exploration, it may not fully exploit exploration opportunities in extremely low-capability base models where the agent’s self-awareness is limited. We believe these are minor issues and we will explore learning-based calibration mechanisms, such as combining internal signals with external feedback to enhance state awareness, thereby further improving the robustness and applicability of strategic branching across diverse model capabilities.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. U21B2010).

References

- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 34 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.
- Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.
- Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. 2026. Agentic reinforced policy optimization. In

- The Fourteenth International Conference on Learning Representations.*
- Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for LLM agent training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems.*
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948.*
- Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore. Association for Computational Linguistics.
- Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. 2025. 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)*, pages 12355–12369, Vienna, Austria. Association for Computational Linguistics.
- Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. 2025. Tree search for llm agent reinforcement learning. *arXiv preprint arXiv:2509.21240.*
- Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process reward models that think. *arXiv preprint arXiv:2504.16828.*
- Samuel Kotz and Norman L. Johnson, editors. 1992. *Breakthroughs in Statistics: Methodology and Distribution.* Springer New York, New York, NY.
- Yizhi Li, Qingshui Gu, Zhoufutu Wen, Ziniu Li, Tianshun Xing, Shuyue Guo, Tianyu Zheng, Xin Zhou, Xingwei Qu, Wangchunshu Zhou, and 1 others. 2025. Treepo: Bridging the gap of policy optimization and efficacy and inference efficiency with heuristic tree-based modeling. *arXiv preprint arXiv:2508.17445.*
- Wenlong Liang, Rui Zhou, Yang Ma, Bing Zhang, Songlin Li, Yijia Liao, and Ping Kuang. 2025. Large model empowered embodied ai: A survey on decision-making and embodied learning. *arXiv preprint arXiv:2508.10399.*
- MoonshotAI. 2025. [Kimi-researcher: End-to-end rl training for emerging agentic capabilities.](#) Accessed: 2025-06.
- OpenAI. 2024a. [Hello gpt-4o.](#) Accessed: 2024-05.
- OpenAI. 2024b. [Learning to reason with LLMs.](#) Accessed: 2024-09.
- OpenAI. 2025. [Introducing gpt-5.](#) Accessed: 2025-08.
- Qwen Team. 2024. [Qwen2.5: A party of foundation models.](#) Accessed: 2024-09.
- Sahar Salimpour, Lei Fu, Farhad Keramat, Leonardo Militano, Giovanni Toffetti, Harry Edelman, and Jorge Peña Queralta. 2025. Towards embodied agentic ai: Review and classification of llm-and vlm-driven robot autonomy and interaction. *arXiv preprint arXiv:2508.05294.*
- Max-Philipp B. Schrader. 2018. gym-sokoban. [https://github.com/mpSchrader/gym-sokoban.](https://github.com/mpSchrader/gym-sokoban)
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseek-math: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300.*
- Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfvorld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768.*
- Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441.*
- Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization for llm agents. *arXiv preprint arXiv:2403.02502.*
- Ruosong Wang, Simon S Du, Lin Yang, and Sham Kakade. 2020. Is long horizon rl more difficult than short horizon rl? In *The Thirty-fourth Annual Conference on Neural Information Processing Systems*, volume 33, pages 9075–9085.
- Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Fang Wu, Weihao Xuan, Heli Qi, Aaron Tu, Ximing Lu, Li Erran Li, and Yejin Choi. 2026a. Deepsearch: Overcome the bottleneck of reinforcement learning with verifiable rewards via tree-based search. In *The Fourteenth International Conference on Learning Representations.*
- Jinyang Wu, Mingkuan Feng, Shuai Zhang, Feihu Che, Zengqi Wen, and Jianhua Tao. 2024. Beyond examples: High-level automated reasoning paradigm in in-context learning via mcts. *arXiv preprint arXiv:2411.18478.*

- Jinyang Wu, Chonghua Liao, Mingkuan Feng, Shuai Zhang, Zhengqi Wen, Haoran Luo, Ling Yang, Huazhe Xu, and Jianhua Tao. 2025. Templaterl: Structured template-guided reinforcement learning for llm reasoning. *arXiv preprint arXiv:2505.15692*.
- Jinyang Wu, Guocheng Zhai, Ruihan Jin, Jiahao Yuan, Yuhao Shen, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. 2026b. Atlas: Orchestrating heterogeneous models and tools for multi-domain complex reasoning. *arXiv preprint arXiv:2601.03872*.
- Shangyu Xing, Siyuan Wang, Chenyuan Yang, Xinyu Dai, and Xiang Ren. 2025. Lookahead tree-based rollouts for enhanced trajectory-level exploration in reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2510.24302*.
- Zhicheng Yang, Zhijiang Guo, Yinya Huang, Xiaodan Liang, Yiwei Wang, and Jing Tang. 2025. Treerpo: Tree relative policy optimization. *arXiv preprint arXiv:2506.05183*.
- Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.
- Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*.
- Yuexiang Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Shengbang Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, and Sergey Levine. 2024. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 37, pages 110935–110971. Curran Associates, Inc.
- Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, and 1 others. 2025a. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*.
- Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, and 1 others. 2025b. A survey of reinforcement learning for large reasoning models. *arXiv preprint arXiv:2509.08827*.
- Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, and 1 others. 2025c. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*.
- Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. 2025d. Process vs. outcome reward: Which is better for agentic RAG reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. 2025e. Rlvmr: Reinforcement learning with verifiable meta-reasoning rewards for robust long-horizon agents. *arXiv preprint arXiv:2507.22844*.
- Tianyu Zheng, Tianshun Xing, Qingshui Gu, Taoran Liang, Xingwei Qu, Xin Zhou, Yizhi Li, Zhoufutu Wen, Chenghua Lin, Wenhao Huang, and 1 others. 2025. First return, entropy-eliciting explore. *arXiv preprint arXiv:2507.07017*.
- Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*.
- Sining Zhoubian, Dan Zhang, and Jie Tang. 2025. Rest-rl: Achieving accurate code reasoning of llms with optimized self-training and decoding. *arXiv preprint arXiv:2508.19576*.

Appendix of SPARK

This supplementary material provides in-depth insights into our SPARK method, covering additional theoretical analysis, additional experimental details, results and analysis. The appendix is organized as follows:

Contents

A Theoretical Analysis	12
A.1 Problem Setup and Notations . . .	12
A.2 Exploration Efficiency under Budget Constraints	13
A.3 Sample Complexity Advantage . .	13
A.4 Summary	13
B Additional Experimental Details	14
B.1 Datasets	14
B.2 Baselines	14
B.3 Evaluation Details	15
B.4 Implementation Details	16
C Supplementary Results	17
C.1 Sample Efficiency	17
C.2 Inference Scalability	17
C.3 Multimodal Extension	17
C.4 Sensitivity Analysis on Total Budget	18
C.5 Test-Time Branching	19
C.6 Statistical Analysis	19
D Case Study	19
E Additional Discussion	19
E.1 Discussion on Tree-Based Baselines	19
E.2 Discussion on Related works . . .	20
E.3 Discussion on Reducing Dependence on Human Priors	20
E.4 Failure Mode Analysis	20

A Theoretical Analysis

In this section, we provide *heuristic theoretical justification* for why SPARK’s dynamic branching achieves superior results. Our analysis offers conceptual insights into the exploration efficiency under constrained computational budget, serving to motivate our method design and interpret empirical findings. While we present mathematical formulations, we emphasize that these are *informal analyses* rather than rigorous proofs, intended to build intuition for the mechanisms underlying SPARK’s performance.

A.1 Problem Setup and Notations

Consider a long-horizon agentic task with maximum horizon K (as defined in §2.1) and action space \mathcal{A} . Let $V_\pi(s)$ denote the value function under policy π at state s , and define the **epistemic uncertainty** at state s as:

$$\begin{aligned} U(s) &= \text{Var}_\pi[Q_\pi(s, a)] \\ &= \mathbb{E}_a[(Q_\pi(s, a) - V_\pi(s))^2], \end{aligned}$$

where the variance is computed over the action distribution $\pi(\cdot|s)$. Intuitively, $U(s)$ measures the agent’s uncertainty regarding the optimal action at state s .

Remark on Implicit Estimation: In our critic-free GRPO framework, while Q_π is not explicitly modeled by a value network, the agent performs an *implicit estimation* of this uncertainty through its reasoning trace z_t . High $U(s)$ is explicitly manifested as a dedicated <explore> tag in the reasoning trace when the agent recognizes semantic ambiguity or potential missteps.

We partition states along a trajectory into two disjoint sets:

- *Critical states* $\mathcal{S}_{\text{crit}} = \{s : U(s) > \tau\}$, where the agent identifies that exploration is valuable.
- *Routine states* $\mathcal{S}_{\text{rout}} = \{s : U(s) \leq \tau\}$, where the optimal action is relatively clear and the agent proceeds linearly.

Let $K_c = |\mathcal{S}_{\text{crit}} \cap \text{trajectory}|$ denote the number of critical states encountered in a trajectory. For long-horizon agentic tasks, we typically observe $K_c \ll K$ (i.e., most steps are routine navigation).

A.2 Exploration Efficiency under Budget Constraints

Analysis 1. (Exploration efficiency of tree-structured branching) *Consider two exploration strategies with the same rollout budget N (total leaf nodes): (i) **Independent Sampling**: generate N independent chains, each of length up to K ; (ii) **SPARK**: construct a tree with M roots and selective branching at critical states (SPARK points). Suppose critical states appear at expected count K_c , and let Δ_{crit} and Δ_{rout} denote the value gain from exploring critical vs. routine states. Our analysis suggests:*

$$\mathbb{E}_{\tau \sim \pi_{\text{SPARK}}} [R(\tau)] \gtrsim \mathbb{E}_{\tau \sim \pi_{\text{indep}}} [R(\tau)] + \Omega\left(\frac{K}{K_c}\right) \cdot (\Delta_{\text{crit}} - \Delta_{\text{rout}}),$$

where the advantage scales with the sparsity ratio K/K_c .

Heuristic Argument. Independent sampling distributes computational effort uniformly across $N \times K$ state-action pairs, resulting in an insufficient exploration depth of $O(1)$ per critical state. SPARK instead shares early routine steps (prefixes) across branches, concentrating the budget $B \approx \frac{N}{M \cdot K_c}$ specifically at critical decision points.

Let $f(b)$ denote the value improvement from b exploratory branches. Under the assumption of diminishing returns (concavity of f), the total gain can be approximated as:

$$\begin{aligned} \text{Gain}_{\text{indep}} &\approx N \cdot [K_c \cdot \Delta_{\text{crit}} \cdot f(1) + (K - K_c) \cdot \Delta_{\text{rout}} \cdot f(1)] \\ \text{Gain}_{\text{SPARK}} &\approx M \cdot K_c \cdot \Delta_{\text{crit}} \cdot f\left(\frac{N}{M \cdot K_c}\right) + N \cdot (K - K_c) \cdot \Delta_{\text{rout}} \cdot f(1), \end{aligned}$$

When f is concave and $\frac{N}{M \cdot K_c} \gg 1$, the tree structure enables deeper exploration where it matters most, yielding an advantage proportional to the sparsity of critical junctures. \square

A.3 Sample Complexity Advantage

Analysis 2. (Reduced sample complexity) *Let ϵ denote the target approximation error and $\rho = K_c/K$ the fraction of critical states along a trajectory. We argue that SPARK can achieve comparable performance using only an $O(M \cdot \rho)$ fraction of samples relative to uniform exploration strategies.*

Heuristic Argument. In long-horizon agentic tasks, learning efficiency is primarily governed by the ability to resolve action preferences at a small number of *critical decision points*, rather than by uniformly improving behavior at all steps. Let $T(\epsilon)$ denote the number of comparable action samples required to identify a near-optimal action at a given critical state within error ϵ .

Under independent sampling, each trajectory contributes at most one action sample per critical state. Consequently, after collecting N trajectories, the effective number of samples available for decision making at each critical state scales as $O(N)$.

In contrast, SPARK explicitly branches multiple alternatives under a shared prefix at each critical state. With a fixed leaf budget N distributed across M roots and K_c critical states, the effective number of comparable action samples per critical state scales as $O\left(\frac{N}{M \cdot K_c}\right)$. This concentrated sampling significantly accelerates the resolution of action uncertainty at critical decision points.

As a result, to achieve the same decision accuracy $T(\epsilon)$ at each critical state, SPARK requires only an $O(M \cdot \rho)$ fraction of the total number of trajectories needed by uniform sampling methods, where $\rho = K_c/K$. Although shared prefixes introduce correlations between samples, the paired nature of branch comparisons increases the **information density** per generated token by avoiding redundant re-sampling of routine interaction histories. \square

Empirical Validation. This heuristic scaling behavior is consistent with our empirical findings in Section 3.3: SPARK matches RLVMR’s peak performance using only 40% of the training data, indicating that comparable performance can be achieved with a constant fraction of samples. This observation falls within the same order of magnitude as the estimated critical-state ratio $\rho \approx 0.4$, up to constant factors.

A.4 Summary

Our heuristic theoretical analysis establishes three key insights:

1. **Exploration Efficiency** (Analysis 1): Tree-structured branching concentrates budget at critical states, suggesting an $\Omega(K/K_c)$ advantage over independent sampling when $K_c \ll K$.

Category	Dataset	#Test Samples
Embodied Reasoning (Shridhar et al., 2020)	ALFWorld-L0 (Seen)	140
	ALFWorld-L1 (Unseen Instance)	134
	ALFWorld-L2 (Unseen Category)	38
Scientific Reasoning (Wang et al., 2022)	ScienceWorld-L0 (Seen)	1661
	ScienceWorld-L1 (Unseen Instance)	1684
	ScienceWorld-L2 (Unseen Task)	549
Web Navigation (Yao et al., 2022a)	WebShop	500

Table 8: Detailed information on the agentic benchmarks and test set split sizes.

2. **Sample Complexity** (Analysis 2): SPARK may require approximately an $M \cdot \rho$ fraction of samples ($\rho = K_c/K$) to match uniform methods’ performance under favorable conditions.

These analyses provide conceptual justification and intuition for SPARK’s empirical success: superior performance (Sec. 3.2), and sample efficiency (Sec. 3.3). While not rigorous proofs, they offer valuable theoretical perspective on the mechanisms underlying our method’s effectiveness.

B Additional Experimental Details

B.1 Datasets

The evaluation benchmarks used in this paper are detailed in Table 8. Following prior work (Feng et al., 2025; Zhang et al., 2025e), we evaluate our method on three long-horizon agentic domains. To rigorously test the generalization of our SPARK paradigm, we categorize tasks into three levels: **L0** (seen categories/instances), **L1** (seen categories but unseen instances), and **L2** (unseen categories and instances). Details are listed as follows:

- **ALFWorld (Shridhar et al., 2020)**: This benchmark aligns TextWorld with the ALFRED environment, requiring agents to execute household tasks (e.g., *Look*, *Clean*, *Pick2*) in a text-based interactive world. It poses challenges in high-level goal decomposition and low-level action execution. We specifically focus on the *Unseen* split (L2) to evaluate the agent’s ability to generalize its learned exploration strategies to completely novel domestic environments.
- **ScienceWorld (Wang et al., 2022)**: ScienceWorld is a complex text-based environment simulating 30 distinct scientific tasks across 10 virtual environments. It requires the agent to apply the scientific method (e.g., conducting experiments, measuring substances) with deep horizons often exceeding 30+ steps. Due to its enormous state space and strict logic requirements, it serves as a robust testbed for our strategic exploration via dynamic branching mechanism.
- **WebShop (Yao et al., 2022a)**: This dataset provides a realistic e-commerce simulation containing over 1.1 million products. Agents must navigate via search and button clicks to purchase items that match specific user requirements. Evaluation is conducted on 500 test queries, where we report both the *task completion score* and *success rate*. WebShop tests the model’s ability to perform open-ended exploration and attribute-matching in large-scale, semi-structured observation spaces.

Task Illustration. We illustrate interactions between agent and Webshop environment in Figure 7.

B.2 Baselines

We evaluate SPARK against three representative classes of baselines, which are briefly described as follows:

(1) Closed-source LLMs:

- **GPT-4o (OpenAI, 2024a)**: A natively multi-modal large language model that can process and generate text, images, and audio within a single architecture, enabling real-time interaction, improved cross-modal reasoning, and low-latency conversational performance.
- **GPT-5-mini (OpenAI, 2025)**: A compact and resource-efficient variant of the GPT-5 family, designed to deliver reliable language understanding and reasoning while minimizing computational cost.

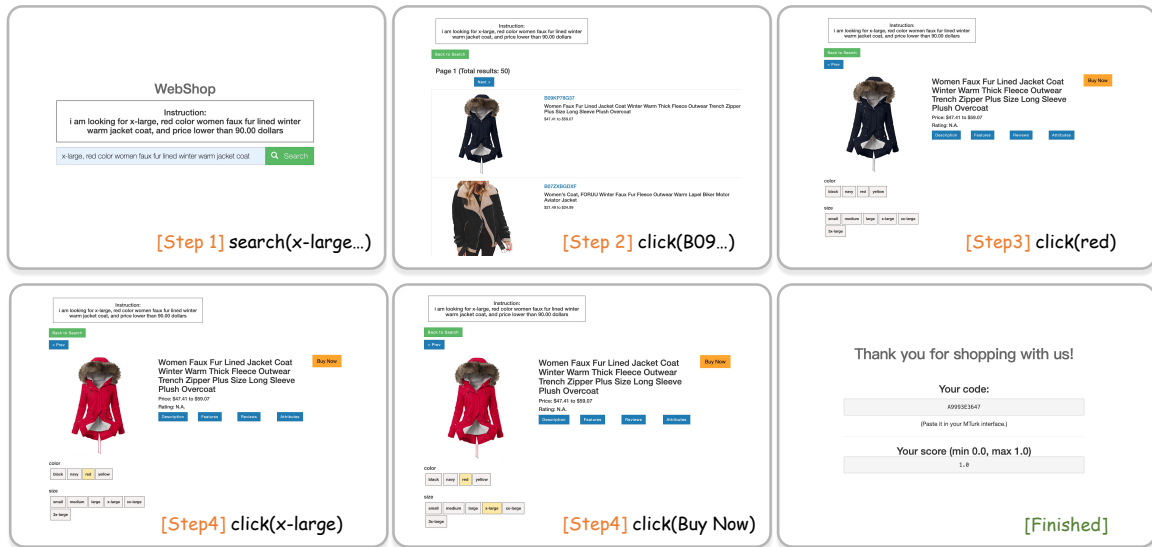


Figure 7: **Illustration of Webshop Environment.** We illustrate the agent’s execution path: first searching for items, then clicking on a suitable product, selecting its color and size, and finally completing the purchase.

- **GPT-5 (OpenAI, 2025):** A next-generation flagship model focused on deep reasoning, long-context understanding, and complex problem solving across domains such as science, coding, and analysis, with enhanced alignment and safety mechanisms.
- **Gemini-2.5-Pro (Comanici et al., 2025):** An advanced multimodal model developed by Google DeepMind, optimized for strong logical reasoning, mathematical problem solving, and code generation, with robust long-context processing capabilities.

(2) Prompting Methods:

- **ReAct (Yao et al., 2022b):** A synergistic framework that interleaves reasoning traces with task-specific actions, enabling LLMs to interact with external environments in real time. ReAct has been widely adopted in agentic tasks and serves as a foundational paradigm shared by many methods.

(3) RL Training Methods:

- **ETO (Song et al., 2024):** An exploration-centric paradigm that enhances model policies through a trial-and-error mechanism. By constructing contrastive success-failure pairs from collected trajectories, it enables LLMs to iteratively refine their decision-making logic and learn from past procedural errors.
- **GRPO (Shao et al., 2024):** A critic-free reinforcement learning framework that estimates advantages through relative rewards

within sampled groups. By eliminating the need for a separate value model, it significantly optimizes computational efficiency while maintaining robust performance in reasoning-intensive tasks.

- **GiGPO (Feng et al., 2025):** A hierarchical policy optimization method that introduces a group-in-group structure for fine-grained credit assignment. By clustering anchor states across trajectories, it enables precise optimization of long-horizon tasks without the dependency on dense external reward models.
- **RLVMR (Zhang et al., 2025e):** A meta-reasoning approach that integrates verifiable cognitive steps into the reinforcement learning loop. By rewarding explicit planning and reflection behaviors via rule-based verification, it enhances the transparency and generalizability of agents in complex problem-solving.

B.3 Evaluation Details

In our evaluation, we consistently adopt Success Rate (SR.) as the primary performance metric. The success rate measures whether an agent successfully completes a given task. For the WebShop environment, in addition to success rate, we also report the task completion score (Score), which reflects the degree of task completion. Specifically, even when a task is not fully completed, the agent can still receive partial credit based on the completion of intermediate sub-goals.

System Prompt For SPARK

[Context & State]

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

Your current observation is: {current_observation}
Your admissible actions of the current situation are: [{admissible_actions}].

[Reasoning Strategies]

You have two reasoning modes:

<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.

</explore>

<think>

Continuously track the current progress and history of reasoning and execution throughout the task. Typically used when task outcomes are as expected and no other mode of reasoning is required.

</think>

You need to choose one of the two reasoning modes mentioned above as your reasoning mode, based on its definition and applicable situation.

[Output Format]

Now it's your turn to first reason and then take an action.

1. You should first reason about the current situation in one concise, brief sentence. This reasoning process MUST be enclosed within ONLY ONE of the above reasoning tags, like: <tag> your reasoning process </tag>.
2. Once you've finished your reasoning, you should choose an admissible action for current step and present it within <action> </action> tags.

Figure 8: System prompt for RL in SPARK.

For all evaluation experiments, we set the test batch size to 128, and fix the random seed to 0 to ensure reproducibility and fair method comparison.

B.4 Implementation Details

Details for SFT. Our SFT dataset consists of 300 successful trajectories designed to initialize the model’s capability to generate intrinsic exploration signals. The dataset is a hybrid constructed from two distinct sources: 90% via a *retro-annotation strategy* and 10% via *real interactions*.

Step Interval	1–5	6–10	11–15	16–20	21–25	26–30
Ratio	22.97%	10.43%	14.20%	7.69%	3.57%	0.00%

Table 9: **Distribution of <explore> tags across step intervals in ALFWorld SFT data.** Tags are concentrated in early steps, reflecting temporal exploration necessity.

(1) Retro-annotation (90%). The “golden paths” are sourced from the datasets used by the RLVMR baseline (Zhang et al., 2025e), which contain only observation-action (obs, action) pairs without intermediate reasoning steps. We employ a teacher model (Kimi-K2) to perform retro-annotation: it reviews these static sequences and synthesizes missing reasoning traces, selectively inserting

<explore> tags at steps where epistemic uncertainty logically arises.

(2) Real interactions (10%). These are entirely new successful trajectories generated organically from scratch by deploying the teacher model to interact directly with the environments. When the teacher model successfully completes a task, the full trajectory—including spontaneous reasoning steps with <explore> and <think> tags—is recorded and added to the SFT dataset.

Tag Placement Robustness. We find the method robust to variations in tag quality and placement. Replacing <explore> with a semantically equivalent alternative (<exploring>) yields no meaningful performance degradation. Furthermore, as shown in Table 9, the <explore> tags are concentrated in early trajectory steps to address initial ambiguity, and decrease as the task progresses, confirming that exploration signals are distributed based on temporal necessity rather than random placement.

Effect of SFT Phase. To further validate that the cold-start SFT phase functions as *format alignment* rather than distilling strategic priors from a specific

SFT Data Source	ALFWorld		ScienceWorld	
	L0	L2	L0	L2
Kimi-K2 (Default)	96.9	80.5	69.5	49.2
Gemini-2.5-Flash	95.8	79.5	67.6	48.1
Qwen2.5-32B	95.6	78.9	67.7	47.8

Table 10: **Effect of SFT teacher model on final performance** (1.5B backbone). Performance differences are marginal across teacher models, confirming that SFT serves as format alignment.

frontier model, we compare the final performance of SPARK when the SFT trajectories are synthesized by different teacher models. As shown in Table 10, substituting Kimi-K2 with Gemini-2.5-Flash or Qwen2.5-32B yields only marginal differences in downstream performance. This demonstrates that SPARK does not heavily rely on the advanced capabilities of any particular teacher model; the true strategy optimization occurs during the RL phase through environment interaction.

Details for RL. Response lengths are set to 512 tokens. Each episode is capped at 30 environment steps. We employ a learning rate of 1×10^{-6} for the actor. A rule-based reward system is adopted, assigning a reward of 10 for task success and 0 for failure. To encourage adherence to the environment’s action space, a penalty of -0.1 is applied for any invalid actions generated by the agent. For all group-based RL methods, the total budget (i.e. group size N) is set to 8, with a batch size of 16. We use a temperature of 0.4 in rollout phase. The KL-divergence loss coefficient is fixed at 0.01. To prevent context window overflow, we restrict the conversation history to a maximum length of 5. Specifically for SPARK, the number of initial roots is set to 4 and branching factor is set to 2 by default. For ALFWorld, ScienceWorld, and WebShop, the maximum prompt lengths are set to 2,048, 4,096, and 6,000 tokens, respectively. The system prompt is shown in Figure 8.

Computing Details. All experiments are conducted on 4 NVIDIA A100-80GB GPUs.

C Supplementary Results

C.1 Sample Efficiency

We provide detailed sample efficiency results in Table 11. The results demonstrate that SPARK exhibits superior sample-efficient scaling properties: with only 20% of the training data, our method al-

Method	20%	40%	60%	80%	100%
GRPO	22.7	37.5	46.1	61.8	76.6
GiGPO	22.7	41.4	68.8	76.6	86.7
RLVMR	68.8	71.1	86.7	86.7	89.1
Ours	84.4	89.1	91.4	96.6	96.9

Table 11: **Sample efficiency comparison on ALF-World L0.** We report success rates (%) on Qwen2.5-1.5B under varying training data percentages. SPARK consistently outperforms representative RL baselines.

ready achieves a success rate of 84.4%, which significantly surpasses the performance of the GRPO baseline trained on the full (100%) dataset (76.6%). Furthermore, SPARK matches the strong RLVMR baseline’s peak performance using only 40% total samples, highlighting the efficacy of our strategic exploration mechanism in extracting higher information density from limited environment interactions. Similar results are also observed in Table 12.

C.2 Inference Scalability

To further investigate the performance potential of our model under extended computational budgets, we evaluate the inference scalability of SPARK compared to other baselines. For each task across the three environments, we generate $n = 32$ candidate trajectories and report the success rate using the Pass@ k metric with $k = 16$. Following standard definition (Chen et al., 2021), the Pass@ k is:

$$\text{Pass}@k := \mathbb{E}_{\text{Problems}} \left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right]$$

where n is the total number of sampled trajectories, c is the number of successful trajectories among n samples, and k is the evaluation threshold. As shown in Table 13, our method consistently outperforms all competitive baselines across all benchmarks. Notably, in the complex ScienceWorld environment, SPARK achieves a success rate of 94.9%, surpassing the strongest baseline by 30.1%. Results show that by prioritizing sampling quality over blind coverage, SPARK significantly enhances model ability to solve long-horizon tasks when more inference-time budget is allocated.

C.3 Multimodal Extension

To validate the generalizability of our framework beyond text-only tasks, we extend our evaluation to the multimodal setting using Qwen2.5-VL-3B-Instruct (Bai et al., 2025). We employ two distinct visual benchmarks: Sokoban (Schrader, 2018)

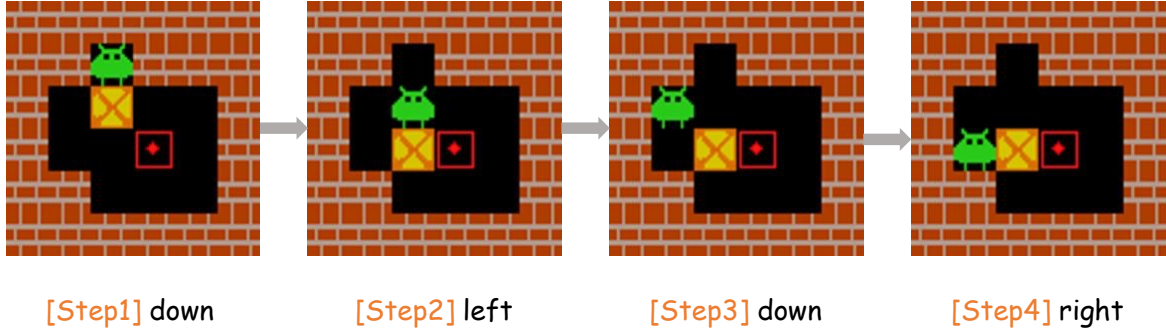


Figure 9: **Illustration of Sokoban Environment.** We illustrate the agent’s execution path: down, left, down, right and finally reach the goal.

Method	20%	40%	60%	80%	100%
GRPO	0.0	2.3	3.1	9.0	21.1
GiGPO	1.6	1.9	10.2	21.9	25.8
RLVMR	30.4	39.6	43.4	47.6	46.9
Ours	36.6	47.3	53.3	60.7	69.5

Table 12: **Sample efficiency comparison on ScienceWorld L0.** We report success rates (%) on Qwen2.5-1.5B under varying training data percentages. SPARK consistently outperforms RL baselines.

Method	ALFWorld \uparrow	ScienceWorld \uparrow	WebShop \uparrow
ReAct	19.8	4.5	18.6
SFT	93.7	64.8	59.7
GRPO	90.9	45.5	78.8
GiGPO	97.0	53.1	74.0
RLVMR	99.9	64.8	75.4
Ours	100.0	94.9	82.0

Table 13: **Inference Scalability Analysis.** We report the Pass@16 success rates (%) on ALFWorld, ScienceWorld, and WebShop using Qwen2.5-1.5B. Results show that SPARK significantly outperforms competitive baselines across all environments.

(6×6 size), a classic grid-based puzzle that necessitates spatial reasoning and long-term planning to push boxes onto designated targets, and EZPoints in Gym Cards (Zhai et al., 2024), a visual reasoning task where the agent must perceive playing cards and formulate arithmetic steps to reach a target value of 12. As shown in Table 14, SPARK achieves substantial improvements across all baseline approaches. Compared to GRPO, our method demonstrates 11.3 points gain in average success rate (88.3% vs. 77.0%), with particularly notable improvements on Sokoban (17.3 points) and consistent gains on EZPoints (5.2 points). More remarkably, our approach surpasses GPT-5 by 16.4 points (88.3% vs. 71.9%) and dramatically out-

Method	Sokoban[6×6] \uparrow	EZPoints \uparrow	Average \uparrow
GPT-5	45.3	98.4	71.9
ReAct	11.7	3.1	7.4
GRPO	67.1	86.9	77.0
Ours	84.4	92.1	88.3
Δ vs. GRPO	+17.3	+5.2	+11.3

Table 14: **Multimodal Extension Analysis.** We present success rates (%) on two multimodal visual benchmarks, Sokoban[6×6] and EZPoints, using Qwen2.5-VL-3B-Instruct as the underlying backbone model.

Budget (N)	4	8	16
L0	89.1	96.9	97.7
L1	81.3	93.8	91.4

Table 15: **Sensitivity Analysis on Total Budget(N).** We investigate how the model performance varies with different budget constraints ($N \in \{4, 8, 16\}$) under ALFWorld L0 and L1, based on 1.5B backbone.

performs the ReAct prompting strategy by 80.9 points (88.3% vs. 7.4%). These results highlight the broad applicability of our framework, demonstrating that our strategic exploration approach is modality-agnostic. As shown in Figure 9, we provide a trace visualization on Sokoban.

C.4 Sensitivity Analysis on Total Budget

As shown in Table 15, we conduct a sensitivity analysis on the total budget N . Increasing N from 4 to 8 yields substantial gains (+7.8% on L0, +12.5% on L1), demonstrating that SPARK effectively leverages additional computational resources for enhanced exploration. However, further scaling to $N = 16$ shows diminishing returns, with performance plateauing. This pattern suggests that $N = 8$ strikes an optimal balance between exploration breadth and efficiency, beyond which

marginal benefits diminish. These results confirm that our method achieves strong cost-effectiveness without requiring excessive budget allocation.

Method	ALFWorld	ScienceWorld	WebShop
baseline-inference	96.9	69.5	75.8
spark-inference	98.4	71.9	78.8

Table 16: **Test-time branching results.** Applying <explore>-conditioned branching at inference yields consistent improvements, indicating potential for Test-Time Scaling.

C.5 Test-Time Branching

The <explore> signal, originally designed to govern training-time rollout topology, also serves as a principled indicator of decision uncertainty at inference time. To assess its test-time utility, we implement a *spark-inference* strategy that applies conditional branching when the model emits <explore> during standard autoregressive generation, selecting the best continuation via majority voting.

As shown in Table 16, spark-inference yields consistent gains across all benchmarks, demonstrating that the learned <explore> signal retains meaningful utility beyond the training phase and aligns with the emerging paradigm of Test-Time Scaling (Zhang et al., 2025c).

C.6 Statistical Analysis

To statistically evaluate the significance of performance differences between SPARK and baseline methods, we use the nonparametric Wilcoxon signed-rank test (Kotz and Johnson, 1992). This statistical test is specifically designed to compare paired observations when data may not follow a normal distribution, making it particularly suitable for performance analysis across methods. The Wilcoxon signed-rank test evaluates whether there is a significant difference between paired observations through the following procedure:

- Calculate differences:** For each benchmark pair, compute the difference $D_i = X_i - Y_i$ where X_i represents SPARK performance and Y_i represents baseline performance.
- Rank differences:** Take absolute values $|D_i|$ and rank them from smallest to largest as R_i , with average ranks assigned for ties.
- Assign signs to ranks:** For each difference D_i , assign its sign to the corresponding rank: $R'_i = \text{sign}(D_i) \cdot R_i$.

- Calculate rank sums:** Compute positive and negative rank sums: $W^+ = \sum_{D_i > 0} R'_i$ and $W^- = \sum_{D_i < 0} R'_i$.

- Determine test statistic:** The test statistic is $W = \min(W^+, W^-)$.

- Calculate p-value:** Derive the p-value from the distribution of test statistic W .

We conducted the Wilcoxon test by pairing SPARK results against GRPO across all benchmarks and model scales (1.5B and 7B). The null hypothesis is H_0 : no significant difference between methods, while the alternative hypothesis is H_1 : significant difference exists. **The analysis yields a p-value of 9.7e-4** (calculated based on the consistent dominance across multiple task domains shown in Table 1). At a significance level of $\alpha = 0.05$, we **reject the null hypothesis** ($p < \alpha$), providing compelling statistical evidence that SPARK possesses a significant advantage over standard GRPO. This confirms that autonomous strategic exploration facilitates a robust and reproducible improvement in agentic reasoning.

D Case Study

Figures 10–13 provide illustrative examples from the ALFWorld and WebShop benchmarks.

E Additional Discussion

E.1 Discussion on Tree-Based Baselines

Tree-structured exploration methods have gained significant attention in recent reinforcement learning research (Li et al., 2025; Wu et al., 2025; Yang et al., 2025). However, the majority of existing tree-based approaches, such as TreeRL (Hou et al., 2025), have been primarily developed and evaluated within the math domain, where problem structures are relatively self-contained and solution verification is straightforward.

The long-horizon agentic tasks considered in our work present fundamentally different challenges (Zhang et al., 2025a; Singh et al., 2025). In embodied planning (Shridhar et al., 2020) and web navigation (Yao et al., 2022a) scenarios, trajectories often span 20-30+ sequential interactions with dynamic environments. Applying conventional tree-structured exploration to such settings would incur prohibitive computational overhead, as the branching factor compounds across the extended horizon. For instance, even a modest branching factor of

2 at each step would yield $2^{30} > 10^9$ potential trajectories for a 30-step episode, rendering exhaustive tree search practically infeasible. Furthermore, while some prior methods incorporate forms of adaptive branching, they typically rely on external signals (e.g., process reward models) or predefined heuristics that do not transfer well to open-ended agentic environments where dense supervision is unavailable. In contrast, SPARK leverages intrinsic decision-making signals to selectively trigger branching only at critical states, achieving a principled balance between exploration depth and computational efficiency.

Given these considerations, we focus our empirical comparisons on chain-like RL methods (e.g., GRPO) that represent the current practical paradigm for long-horizon agent training. Adapting tree methods to long-horizon agentic domains remains an important direction for future work.

E.2 Discussion on Related works

Several concurrent works also explore uncertainty-aware or tree-structured rollout allocation. ARPO (Dong et al., 2026) targets distribution shifts after tool calls, while FR3E (Zheng et al., 2025) identifies high-entropy tokens to provide fine-grained feedback signals. DeepSearch (Wu et al., 2026a) employs MCTS-based state-space coverage to overcome RL training plateaus. While these methods employ uncertainty-aware exploration, they fundamentally differ from SPARK in both motivation and mechanism. First, in terms of motivation, ARPO and FR3E adopt a *reactive, local* perspective—correcting errors after they occur—whereas SPARK pursues *proactive, strategic* resource allocation across the entire trajectory horizon. Second, in terms of mechanism, ARPO and FR3E rely on *external statistical signals* (e.g., token-level entropy), treating the agent as a passive subject of statistical fluctuations; DeepSearch applies algorithmic heuristics (UCT scores) from an external controller. In contrast, SPARK introduces *intrinsic semantic reasoning* via the <explore> tag, enabling the agent to autonomously identify and act on its own epistemic uncertainty within its cognitive flow—transitioning from a passive subject of external measurement to an active, self-aware participant in exploration strategy. This intrinsic mechanism generalizes robustly to unseen domains, as evidenced by SPARK’s strong out-of-domain performance (Table 3).

E.3 Discussion on Reducing Dependence on Human Priors

A key design principle of SPARK is to minimize reliance on human-engineered heuristics for determining when and where to explore. Traditional exploration strategies often depend on handcrafted rules, domain-specific knowledge, or external reward models that require substantial human effort to design and may not generalize across tasks. In contrast, SPARK enables the agent to autonomously identify critical decision points through its own reasoning process. The <explore> signal emerges from the agent’s internal deliberation when it recognizes epistemic uncertainty or semantic ambiguity in the current state. This intrinsic mechanism offers two advantages: (1) it eliminates the need for task-specific annotations, and (2) it allows the exploration strategy to adapt naturally to novel scenarios, as the agent learns to recognize uncertainty patterns rather than relying on fixed rules.

Our experimental results support this claim. SPARK demonstrates robust generalization on out-of-domain tasks (L2 splits), where human-defined heuristics would likely fail due to the absence of prior knowledge about unseen task categories. The agent’s ability to autonomously trigger exploration at appropriate junctures without explicit supervision, validates that strategic exploration can emerge from learned intrinsic signals rather than external human guidance.

E.4 Failure Mode Analysis

While SPARK demonstrates strong performance across diverse benchmarks, we identify several scenarios where it may underperform. First, when the base model has limited capability to recognize uncertainty, the <explore> signal may not be reliably triggered at genuinely critical states, leading to either missed exploration opportunities or spurious branching at routine steps. Second, in tasks where critical decisions are densely distributed throughout the trajectory, the advantage of selective branching diminishes, and SPARK may not significantly outperform uniform exploration strategies. Future work could address these limitations by incorporating learned uncertainty estimators to calibrate branching decisions, or developing curriculum strategies that progressively refine exploration signals.

A trajectory of an agent trained with SPARK

Task: *put an egg on the microwave.*

[Step 1]

Observation: -= Welcome to TextWorld, ALFRED! -=

Reasoning: <explore>I need to find an egg first before I can put it in the microwave, so I should start by checking common places where eggs are stored or left—such as the fridge, countertops, or drawers—since eggs are often kept in the fridge or near the sink area.</explore>

Action: go to fridge 1

[Step 2]

Observation: You arrive at fridge 1. The fridge 1 is closed.

Reasoning: <think>I am at the fridge, which is closed, and I need to find an egg to place in the microwave, so I should open the fridge to check its contents.</think>

Action: open fridge 1

[Step 3]

Observation: You open the fridge 1. The fridge 1 is open. In it, you see a bowl 2, a bowl 1, a lettuce 1, a mug 2, a mug 1, and a pot 1.

Reasoning: <explore>The task is to put an egg on the microwave, but so far I've only opened the fridge and see a bowl and other items—none of which are eggs—so I need to search for an egg elsewhere; the microwave is not in the fridge, so I should check other plausible locations like countertops or the dining table where an egg might be left.</explore>

Action: go to countertop 1

[Step 4]

Observation: You arrive at countertop 1. On the countertop 1, you see nothing.

Reasoning: <explore>Countertop 1 is empty, so I need to check other common places where an egg might be—such as countertops 2 or 3, dining table, or even the sinkbasin where a dropped egg could be—before heading to the microwave.</explore>

Action: go to countertop 2

[Step 5]

Observation: You arrive at countertop 2... you see a apple 1, a bowl 3... [no egg].

Reasoning: <explore>The task is to put an egg on the microwave, but I haven't seen an egg yet; so far I've only checked the fridge... and countertops... so I need to keep searching for an egg by checking other common places like the dining table or sink basin where an egg might be left.</explore>

Action: go to diningtable 1

[Step 6]

Observation: You arrive at diningtable 1. On the diningtable 1, you see a apple 3... [no egg].

Reasoning: <explore>The egg is not on any of the surfaces I've checked (fridge, countertops, dining table), so I need to search other plausible locations like the sinkbasin (where a bowl might contain an egg) or drawers (where an egg could be stored).</explore>

Action: go to sinkbasin 1

[Step 7]

Observation: You arrive at sinkbasin 1. On the sinkbasin 1, you see a apple 2, a egg 2, a knife 1...

Reasoning: <think>I found an egg (egg 2) on sinkbasin 1, so the next step is to take it so I can later place it in the microwave.</think>

Action: take egg 2 from sinkbasin 1

[Step 8]

Observation: You pick up the egg 2 from the sinkbasin 1.

Reasoning: <think>I have successfully picked up egg 2 from the sinkbasin 1, and now I need to go to the microwave 1 to place the egg inside it.</think>

Action: go to microwave 1

[Step 9]

Observation: You arrive at microwave 1. The microwave 1 is closed.

Reasoning: <think>I have the egg and I am now at the microwave, which is closed; to place the egg inside I must first open the microwave.</think>

Action: open microwave 1

[Step 10]

Observation: You open the microwave 1. The microwave 1 is open. In it, you see nothing.

Reasoning: <think>I have the egg and the microwave is open and empty, so I can now place the egg inside to complete the task.</think>

Action: move egg 2 to microwave 1

Figure 10: A full trajectory of SPARK in ALFWorld (Example 1).

A trajectory of an agent trained with SPARK

Task: *put two watch in dresser.*

[Step 1]

Observation: -= Welcome to TextWorld, ALFRED! -=

Reasoning: <explore>I need to find two watches first before I can put them in the dresser, so I should start by exploring likely locations where watches might be—such as sidetables, drawers, or tables—since they are small objects unlikely to be on the sofa or armchair.</explore>

Action: go to sidetable 1

[Step 2]

Observation: You arrive at sidetable 1. On the sidetable 1, you see a remotecontrol 1.

Reasoning: <explore>I need to find two watches, but I don't see any watches on sidetable 1; they could be on any of the many tables, drawers, or dresser surfaces, so I should systematically check the diningtable, other sidetables, and drawers next since they are common places for small items.</explore>

Action: go to diningtable 1

[Step 3]

Observation: You arrive at diningtable 1. On the diningtable 1, you see a laptop 3, a laptop 2, a remotecontrol 3, a statue 3, a statue 2, a watch 2, and a watch 1.

Reasoning: <think>I need to find two watches, and I can see watch 1 and watch 2 on diningtable 1; I should pick them up first before heading to the dresser.</think>

Action: take watch 1 from diningtable 1

[Step 4]

Observation: You pick up the watch 1 from the diningtable 1.

Reasoning: <think>I have picked up watch 1 from diningtable 1, and now I need to go to dresser 1 to place it inside.</think>

Action: go to dresser 1

[Step 5]

Observation: You arrive at dresser 1. On the dresser 1, you see a television 1.

Reasoning: <think>I have watch 1 in inventory and am now at dresser 1, so I can complete the task by placing watch 1 on the dresser.</think>

Action: move watch 1 to dresser 1

[Step 6]

Observation: You move the watch 1 to the dresser 1.

Reasoning: <think>I have successfully moved one watch (watch 1) to the dresser, so now I need to go back to the dining table to pick up the second watch (watch 2) so I can place both watches in the dresser.</think>

Action: go to diningtable 1

[Step 7]

Observation: You arrive at diningtable 1. On the diningtable 1, you see a laptop 3, a laptop 2, a remotecontrol 3, a statue 3, a statue 2, and a watch 2.

Reasoning: <think>I have already placed one watch (watch 1) on dresser 1, and I am now back at diningtable 1 where the second watch (watch 2) is still present; to complete the task I must pick up this remaining watch and then go back to dresser 1 to place it alongside the first one.</think>

Action: take watch 2 from diningtable 1

[Step 8]

Observation: You pick up the watch 2 from the diningtable 1.

Reasoning: <think>I have just picked up watch 2 from diningtable 1, and I already placed watch 1 on dresser 1; to complete the task I simply need to move watch 2 to the dresser as well.</think>

Action: go to dresser 1

[Step 9]

Observation: You arrive at dresser 1. On the dresser 1, you see a television 1, and a watch 1.

Reasoning: <think>I have already placed watch 1 on dresser 1, and I am now holding watch 2, so I just need to place it on the dresser to complete the task.</think>

Action: move watch 2 to dresser 1

Figure 11: A full trajectory of SPARK in ALFWorld (Example 2).

A trajectory of an agent trained with SPARK

Task: Find me loose fit women's tops, tees & blouses with short sleeve, long sleeve with color: b2-wine, and size: medium, and price lower than 50.00 dollars

[Step 1]

Observation: Search

Reasoning: <explore>The user wants women's tops/tees/blouses in b2-wine color, short or long sleeve, medium size, under \$50. The search bar is the fastest way to surface relevant products; I need to enter a query that includes these key descriptors.</explore>

Action: search[loose fit women's tops tees blouses b2-wine medium size under 50]

[Step 2]

Observation: Back to Search [SEP] Page 1 (Total results: 50) [SEP] Next > [SEP] B097RK2B2Q [SEP] FABIURT Summer Tops for Women, Women Fashion Graphic Plus Tank Top Sleeveless Casual Tunic Loose Cross Tee Shirt Blouses [SEP] \$10.99 to \$11.99 [SEP] B09QGK5XHZ [SEP] ...

Reasoning: <think>The search returned 50 results. The first item (B097RK2B2Q) is a loose-fit women's top/tee/blouse priced \$10.99–\$11.99, which satisfies color “b2-wine” and size “medium” (likely medium/mediumsized women's tops are available in medium), so I need to inspect its details to confirm the remaining constraints before purchasing.</think>

Action: click[b097rk2b2q]

[Step 3]

Observation: Back to Search [SEP] < Prev [SEP] size [SEP] small [SEP] medium [SEP] large [SEP] x-large [SEP] xx-large [SEP] color [SEP] a1-blue [SEP] ... [SEP] b2-wine [SEP] ... [SEP] FABIURT Summer Tops for Women... [SEP] Price: \$10.99 to \$11.99

Reasoning: <think>I'm now on the product page for the FABIURT loose top. The color filter shows “b2-wine” as an available option, so I need to select that color before proceeding.</think>

Action: click[b2-wine]

[Step 4]

Observation: Back to Search [SEP] < Prev [SEP] size [SEP] small [SEP] medium [SEP] large [SEP] x-large [SEP] xx-large [SEP] color [SEP] a1-blue [SEP] ... [SEP] b2-wine [SEP] ... [SEP] FABIURT Summer Tops for Women... [SEP] Price: \$10.99 to \$11.99

Reasoning: <think>Color is now set to b2-wine, price is still under \$50, and size is medium. All constraints are satisfied, so the next logical step is to choose the required size.</think>

Action: click[medium]

[Step 5]

Observation: Back to Search [SEP] < Prev [SEP] size [SEP] small [SEP] medium [SEP] large [SEP] x-large [SEP] xx-large [SEP] color [SEP] a1-blue [SEP] ... [SEP] b2-wine [SEP] ... [SEP] FABIURT Summer Tops for Women... [SEP] Price: \$10.99 to \$11.99

Reasoning: <think>Color is now set to b2-wine and size to medium; the price is still within \$10.99–\$11.99, so the only remaining step is to purchase the item.</think>

Action: click[buy now]

Figure 12: A full trajectory of SPARK in WebShop (Example 3).

A trajectory of an agent trained with SPARK

Task: Find me machine wash, moisture wicking men's t-shirts & tanks with polyester spandex, long sleeve with color: grey | red, and size: small, and price lower than 40.00 dollars

[Step 1]

Observation: Search

Reasoning: <explore>The user wants men's long-sleeve, machine-washable, moisture-wicking, polyester-spandex, grey/red, small, <\$40 t-shirts/tanks. I need to search broadly first to see what options exist, then narrow down based on the constraints.</explore>

Action: search[men's long sleeve moisture wicking polyester spandex t-shirt tank grey red small under 40]

[Step 2]

Observation: Back to Search [SEP] Page 1 (Total results: 50) [SEP] Next > [SEP] B09KLQLLT2 [SEP] Long Sleeve Superhero T Shirt Tank Top Mens Compression Shirt Men Workout Fitness Gym Shirt [SEP] \$19.99 [SEP] B09QQP3356 [SEP] HAUKLIE Men's Sports Waffle Ribbed Polo Shirts...

Reasoning: <think>I opened the result list and saw 50 total hits. The first item (B09KLQLLT2) is a long-sleeve compression shirt priced at \$19.99, which satisfies the long-sleeve and price constraints. I need to inspect its details to verify moisture-wicking, polyester-spandex, grey/red color, and small size availability.</think>

Action: click[b09klqllt2]

[Step 3]

Observation: Back to Search [SEP] < Prev [SEP] color [SEP] black | blue [SEP] blue [SEP] ... [SEP] grey | red [SEP] size [SEP] small [SEP] ... [SEP] Long Sleeve Superhero T Shirt ... [SEP] Price: \$19.99

Reasoning: <think>I'm now on the product page. The color filter shows "grey | red" as an available option, which matches the required color constraint. I need to select this color variant before proceeding.</think>

Action: click[grey | red]

[Step 4]

Observation: Back to Search [SEP] < Prev [SEP] color [SEP] black | blue [SEP] ... [SEP] grey | red [SEP] size [SEP] small [SEP] ... [SEP] Long Sleeve Superhero T Shirt ... [SEP] Price: \$19.99

Reasoning: <think>Color is now set to grey | red, so the shirt now meets one of the required color constraints. I still need to check the size and material, so I will proceed to select size "small".</think>

Action: click[small]

[Step 5]

Observation: Back to Search [SEP] < Prev [SEP] color [SEP] black | blue [SEP] ... [SEP] grey | red [SEP] size [SEP] small [SEP] ... [SEP] Long Sleeve Superhero T Shirt ... [SEP] Price: \$19.99

Reasoning: Color and size are now set to grey | red and small, respectively, and the price (\$19.99) is under \$40, so the only remaining step is to purchase the item.</think>

Action: click[buy now]

Figure 13: A full trajectory of SPARK in WebShop (Example 4).