CRPS: Curriculum Replay via Progressive Suffixes from Successful Trajectories for Long-Horizon LLM Agents

Zijing Zhang[1]*, **Xiajie Yang**[2]†

[1]Peking University, [2]Baidu Inc.

Abstract

Long-horizon LLM agents trained with sparse terminal rewards tend to experience slow and unstable learning, and the issue is amplified by group-normalized on-policy objectives commonly used for LLM training (e.g., GRPO). When rollout groups collapse to nearly all failures early in training, within-group normalization yields collapsed advantages and weak learning signals. To address this, we propose **Curriculum Replay via Progressive Suffixes from Successful Trajectories (CRPS)**, a lightweight RL-training strategy that turns serendipitous terminal successes into a within-trajectory curriculum. CRPS maintains a buffer of successful trajectories and restarts rollouts from suffix states, with an online controller adapting k to match agent competence and keep replay outcomes away from all-failure/all-success extremes. Across ALFWorld and WebShop with different foundation models, CRPS consistently outperforms full-episode GRPO and naive experience replay. Group-level diagnostics further show that CRPS reduces the fraction of extreme groups and increases within-group outcome diversity, aligning with faster and more stable training.

1 Introduction

Large language models (LLMs) have recently been extended into interactive agents capable of multi-step reasoning, tool use, and real world interaction (Gur et al., 2023; Shi et al., 2024; Xi et al., 2025). While such agents can solve non-trivial tasks in embodied-like text environments and web navigation tasks (Shridhar et al., 2021; He et al., 2024; Wei et al., 2025), improving them with end-to-end reinforcement learning remains challenging, especially in long-horizon settings with sparse and delayed rewards (Wang et al., 2025; Feng et al., 2025b).

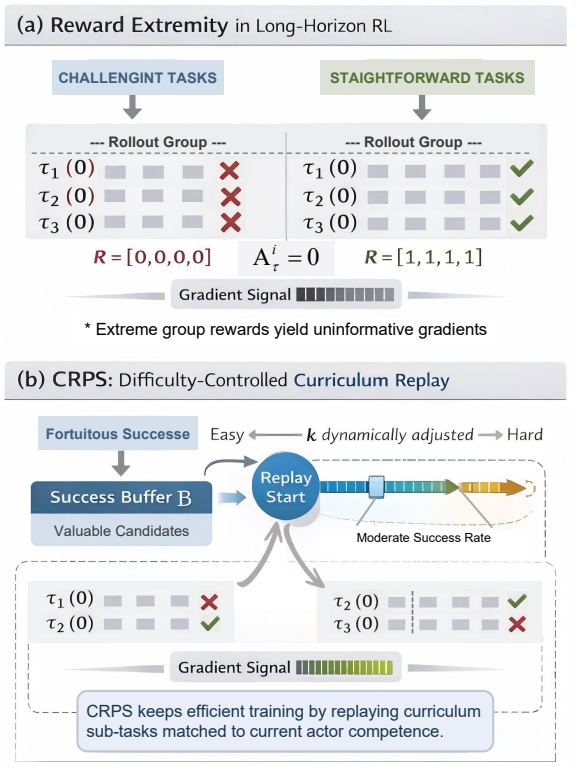

Figure 1: Sparse long-horizon agent RL often yields collapsed within-group reward distributions (e.g., all failures), making GRPO-style normalization ineffective. CRPS replays from successful suffix states to keep group success rates away from extremes.

In many realistic settings, an episode can span tens of environment steps, while success is only observed at termination: the agent must plan, explore, and execute a long sequence of actions before receiving a single terminal success signal (Shridhar et al., 2021; Yao et al., 2022). As a result, most early rollouts fail, yielding sparse learning signals and slow progress under outcome-only reinforcement learning (Ecoffet et al., 2021; Pathak et al., 2017).

This challenge is further amplified for group-based policy optimization methods commonly used for LLM fine-tuning (e.g., GRPO-style train-

* Correspondence: zijingzhang@stu.pku.edu.cn
† Corresponding author.

ing (Shao et al., 2024)). These methods typically sample multiple rollouts per instruction and normalize rewards within each group to form relative advantages. When the reward distribution within a group becomes extreme (e.g., almost all failures early in training, or almost all successes for "too easy" replay), advantage estimates collapse and gradients become weak, wasting environment interactions.

These observations suggest that, beyond the choice of optimizer, the *training distribution* (which tasks and which start states we sample) matters critically in sparse long-horizon agent RL (Bengio et al., 2009; Narvekar et al., 2020; Portelas et al., 2020). In particular, within-group normalization is most effective when groups contain a mix of successes and failures; however, long-horizon sampling naturally produces many all-failure groups early on, while overly easy replay can later produce all-success groups. Both extremes waste interactions because the resulting advantages provide little usable learning signal.

At the same time, even in the difficult regime, agents occasionally succeed due to exploration and stochasticity. Such rare successes are expensive to obtain but contain privileged information: they identify states from which the agent can sometimes finish the task under the current policy class (Oh et al., 2018; Andrychowicz et al., 2017). A natural question, then, is how to systematically *amplify the learning value* of these occasional successes—turning a small number of terminal rewards into a sequence of learnable training problems of controllable difficulty, without requiring dense reward shaping or external task generators (Bengio et al., 2009; Narvekar et al., 2020; Portelas et al., 2020).

At the same time, even in the difficult regime, agents occasionally succeed due to exploration and stochasticity. Such rare successes are expensive to obtain but are informative for GRPO-style learning: they expose intermediate states where success is neither impossible nor trivial under the current policy class (Oh et al., 2018; Andrychowicz et al., 2017). A natural question, then, is how to *amplify the learning value* of these occasional successes—turning a small number of terminal rewards into a sequence of learnable training problems of controllable difficulty, without dense shaping or task generators, and avoiding degenerate groups (Bengio et al., 2009; Narvekar et al., 2020; Portelas et al., 2020).

In this paper, we propose Curriculum Replay via Progressive Suffixes from Successful Trajectories (CRPS), a simple add-on that couples success-only replay with progressive suffix restarts and adaptive difficulty control for GRPO-style training (Mnih et al., 2015; Schaul et al., 2015; Oh et al., 2018). The key idea is to repeatedly train on *appropriately difficult* subproblems—neither trivially easy nor hopelessly hard—because these "partially solvable" cases tend to yield the strongest within-group learning signal. Concretely, given a stored successful trajectory of length T, CRPS chooses a replay start index $t_0 = T - k$ (i.e., a suffix length k), resets the environment to the corresponding state, and re-executes from t_0 to termination with the current policy. Here, k serves as a simple difficulty knob: smaller k produces easier near-goal subproblems, while larger k yields harder ones starting farther from the goal. We further adapt k online so that replayed subproblems track the agent's capability and keep group success rates away from extremes as training progresses.

In summary, our contributions are as follows:

- We introduce CRPS, a minimal replay-based add-on that turns serendipitous terminal successes into a within-trajectory curriculum with a single difficulty knob, without requiring dense rewards or external task generation.
- We show that combining suffix-state replay with adaptive difficulty control constructs less-extreme group reward distributions for GRPO-style optimization, mitigating reward extremity and improving data utilization under sparse terminal rewards.
- We evaluate CRPS on ALFWorld and WebShop, demonstrating consistent improvements over full-episode baselines and naive experience replay across model scales.

2 Related Work

Reinforcement learning for LLM agents. LLM agents combine language generation with environment interaction, where a small change in the generated text can induce a different tool call or state transition. Early agent frameworks such as ReAct (Yao et al., 2023) highlight the tight coupling between reasoning traces and executable actions, and subsequent work explores how to endow base models with stronger tool and agent behaviors via self-training or fine-tuning (Schick et al., 2023; Shinn et al., 2023; Zeng et al., 2023; Chen et al.,

2023). To optimize these agents under sparse, verifiable objectives (e.g., task success), recent pipelines increasingly adopt RL-style policy optimization. Compared to standard PPO (Schulman et al., 2017), group-based variants such as GRPO (Shao et al., 2024) and REINFORCE-style baselines such as RLOO (Kool et al., 2019) are attractive for LLM training because they weaken dependence on a learned critic. However, long-horizon interaction benchmarks (e.g., ALFWorld and WebShop (Shridhar et al., 2021; Yao et al., 2022)) make learning particularly challenging: rewards arrive at the end, while optimization must improve decisions far from the goal.

Experience enhanced LLM agents. Experience replay is a canonical mechanism for reusing past interactions in RL, popularized in deep RL systems such as DQN (Mnih et al., 2015) and extended with prioritized and distributed variants (Schaul et al., 2015; Horgan et al., 2018). In sparse-reward settings, related ideas include self-imitation (Oh et al., 2018) and hindsight relabeling (Andrychowicz et al., 2017; Fang et al., 2019). For LLM agents, the emphasis is often not purely on off-policy sample reuse, but on *experience augmentation*: recording, organizing, and reusing high-value trajectories or memories to improve long-horizon behavior. Representative directions include experience replay for GUI agents (Lu et al., 2025; Xu et al., 2025), record-and-replay style agent training (Feng et al., 2025a), contextual replay for self-improvement (Liu et al., 2025), and analyses of memory management for LLM agents (Xiong et al., 2025). A common limitation is coarse control over *where* to restart or what subproblem to practice within a trajectory. CRPS complements these efforts as a GRPO-oriented instantiation with *within-trajectory* control over the start state: it restores an intermediate state by replaying a prefix of a successful trajectory and then collects a fresh on-policy suffix, inducing a graded curriculum along the same task instance.

Curriculum learning in RL. Curriculum learning trains on easier problems first and gradually increases difficulty (Bengio et al., 2009). Prior work studies curriculum design and automation in RL, including survey frameworks and automatic curricula (Narvekar et al., 2020; Portelas et al., 2020), self-paced and distributionally robust curricula (Satheesh et al., 2025a), and other adaptive strategies (Chaudhary and Behera, 2025; Hübotter et al., 2025). For LLM agents, curricula are often induced via additional machinery (e.g., process-level supervision or reward shaping, constraint-aware training, or environment/task design) (Tan et al., 2025; Tzannetos et al., 2025; Satheesh et al., 2025b; Xia et al., 2025). CRPS instead provides a simple curriculum mechanism without dense intermediate rewards or external task generators: for a fixed task instance, it reshapes the start-state distribution along a successful trajectory from near-goal to farther-from-goal.

3 Method

3.1 Problem Formulation

We model an instruction-conditioned interactive agent as a partially observable Markov decision process (POMDP)

$$\mathcal{M}_x = \big(\mathcal{S}, \mathcal{A}, P, \Omega, O, r, \gamma\big), \tag{1}$$

where x is a task instruction, $s_t \in \mathcal{S}$ is the latent environment state, $a_t \in \mathcal{A}$ is a textual action executed by the environment, $s_{t+1} \sim P(\cdot \mid s_t, a_t)$, and the agent observes $o_t \in \Omega$ with $o_t \sim O(\cdot \mid s_t)$. We focus on sparse long-horizon benchmarks where the reward is terminal:

$$r_t = \begin{cases} 1 & \text{if episode succeeds at } t = T, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

We denote the terminal return by $R(\tau) = r_T \in \{0, 1\}$. The agent is an autoregressive language policy π_θ that conditions on the instruction and interaction history $h_t = (x, o_1, a_1, \ldots, o_t)$ and samples actions $a_t \sim \pi_\theta(\cdot \mid h_t)$. The objective is to maximize success probability:

$$\max_\theta \ \mathbb{E}_{\tau \sim \pi_\theta(\cdot|x)}\big[R(\tau)\big]. \tag{3}$$

3.2 Reward Extremity

Group-based on-policy methods for LLM fine-tuning (e.g., GRPO) sample G rollouts per instruction and compute *relative* advantages by normalizing rewards within the group:

$$\begin{aligned} A^{(g)} &= \frac{R^{(g)} - \mu_R}{\sigma_R + \epsilon}, \\ \mu_R &= \frac{1}{G}\sum_{g=1}^{G} R^{(g)}, \\ \sigma_R^2 &= \frac{1}{G}\sum_{g=1}^{G} (R^{(g)} - \mu_R)^2. \end{aligned} \tag{4}$$

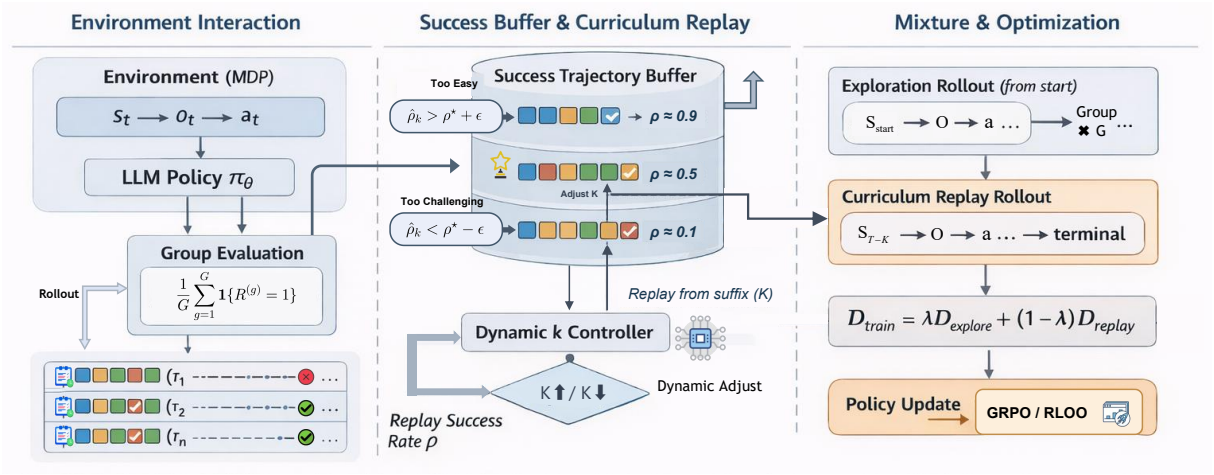

Figure 2: CRPS overview: store successful trajectories, replay from suffix states with adaptive k, and mix replay with fresh exploration for on-policy updates.

When all rewards in the group are identical (all 0 or all 1), $\sigma_R \approx 0$ and the resulting learning signal collapses. This "reward extremity" is common early in long-horizon training (all failures) and can also happen under overly easy replay (all successes).

3.3 CRPS Overview

CRPS (Curriculum Replay via Progressive Suffixes from Successful Trajectories) builds a within-trajectory curriculum from occasional terminal successes. The core mechanism is to replay from suffix states of a successful trajectory, where the suffix length k directly controls replay difficulty and horizon. By dynamically adjusting k, CRPS selects replay problems that match the agent's current competence, keeping replay outcomes away from all-failure/all-success extremes and preventing group-normalized advantages from collapsing.

3.4 Success Buffer

We maintain a buffer $\mathcal{B}$ of *task instances* (instructions) for which we have observed at least one successful rollout and that remain non-trivial for the current policy. Each buffer entry stores (i) the instruction and a successful interaction trace, and (ii) metadata that allows reconstructing an intermediate state by replaying a prefix: an environment initialization identifier z (e.g., a seed) and the action sequence. Under deterministic environment dynamics conditioned on $(z, a_{0:t-1})$, we can restore s_t by resetting with z and re-applying the first t actions. We store each successful trace in the buffer as

$$b = \Big(x,\ z,\ T,\ \{(o_t, a_t)\}_{t=0}^{T-1}\Big), \quad \text{with } R(b) = 1. \tag{5}$$

In addition to the trajectory itself, we optionally maintain per-entry replay statistics (e.g., moving-average replay success) to support difficulty-aware sampling and eviction.

Trajectory Insertion. Among tasks for which we observe at least one successful trajectory, successes can correspond to (i) *already-easy* tasks with high group success rates (which can yield near-deterministic replay and all-success groups), (ii) *brittle* one-off successes that are hard to reproduce, or (iii) *moderately difficult* tasks where success is achievable but not guaranteed. Since CRPS aims to replay these moderately difficult, mixed-outcome cases for group-normalized updates, we gate buffer insertion using the group success rate

$$\mathrm{acc}(x) = \frac{1}{G}\sum_{g=1}^{G} \mathbf{1}\{R^{(g)} = 1\}. \tag{6}$$

To prioritize non-trivial tasks and filter out overly easy (mastered) successes, we insert b into $\mathcal{B}$ only if its rollout succeeds and

$$\mathrm{acc}(x) \le \alpha_{\max} \quad \text{and} \quad \mathbf{1}\{R(\hat{\tau}) = 1\} = 1, \tag{7}$$

where $\alpha_{\max}$ is an "easiness" threshold that excludes near-deterministic successes. Importantly, tasks that remain challenging are not discarded: CRPS can make them learnable by starting from a shorter suffix (smaller k) and then gradually expanding the replay horizon as the policy improves.

3.5 Suffix Replay with Dynamic k

Given a sampled buffer entry $b \in \mathcal{B}$ with horizon T, CRPS selects a suffix length k and defines the replay start index

$$t_0 = \max(0, T - k). \tag{8}$$

CRPS restores s_{t_0} by resetting the environment using the stored initialization metadata z and replaying the first t_0 actions from the stored successful trajectory, then rolls out the current policy π_θ from t_0 until termination, producing an on-policy trajectory segment $\hat{\tau}$ with terminal reward $R(\hat{\tau})$. Smaller k starts closer to success (easier, higher replay success); larger k moves farther from the goal (harder, longer horizon).

Initializing k. We initialize replay difficulty proportionally to the successful trajectory length and the observed task difficulty. Concretely, for a task instance with group success rate $\mathrm{acc}(x)$ (Eq. 6) and a successful trajectory length T, we set

$$k_0 = \mathrm{clip}\Big(\big\lfloor (\beta_{\min} + (\beta_{\max} - \beta_{\min})\,\mathrm{acc}(x))\,T \big\rceil, \; k_{\min},\; k_{\max}\Big), \tag{9}$$

so harder tasks (smaller $\mathrm{acc}(x)$) start from shorter suffixes (easier replay), and easier tasks start from longer suffixes.

Adapting k to avoid extreme groups. To prevent group-based advantages from collapsing (Eq. 4), we want replay groups to contain a mixture of successes and failures rather than collapsing to all-0 or all-1 outcomes. CRPS achieves this by adapting k to maintain an intermediate replay success rate.

Let ρ_k denote the replay success probability when starting from suffix length k. Since the optimization signal is computed *within groups*, we monitor replay outcomes at the group level. For a replay group (same instruction, G rollouts) we define

$$\mathrm{acc}_{\text{replay}}(x) = \frac{1}{G}\sum_{g=1}^{G} \mathbf{1}\{R^{(g)} = 1\}. \tag{10}$$

We track an exponential moving average $\hat{\rho}_k$ over recent replay groups:

$$\hat{\rho}_k \leftarrow (1-\lambda)\hat{\rho}_k \;+\; \lambda \cdot \mathrm{acc}_{\text{replay}}(x), \tag{11}$$

with smoothing factor $\lambda \in (0, 1]$. We then adjust k with step size Δ to keep $\hat{\rho}_k$ inside a target band $\rho^\star = [\rho_l, \rho_u] \subset (0, 1)$:

$$k \leftarrow \begin{cases} \min(k + \Delta, k_{\max}) & \text{if } \hat{\rho}_k > \rho_u, \\ \max(k - \Delta, k_{\min}) & \text{if } \hat{\rho}_k < \rho_l, \\ k & \text{otherwise.} \end{cases} \tag{12}$$

If replay is too successful ($\hat{\rho}_k > \rho_u$), we increase k to start farther from the goal, making replay harder and preventing all-success groups; if replay is too unsuccessful ($\hat{\rho}_k < \rho_l$), we decrease k to move closer to success, preventing all-failure groups. This keeps the reward distribution away from extremes and maintains useful gradients under within-group normalization.

Buffer maintenance (mastery removal). As training progresses, the controller may expand replay toward the full trajectory (i.e., $k \to T$). When replay remains highly successful even at the maximal horizon (e.g., the replay group success rate consistently exceeds a threshold), the corresponding task/trajectory is considered mastered and can be removed from $\mathcal{B}$ to keep replay focused on currently learnable examples.

3.6 Exploration–Replay Mixture

Pure replay can overfit to narrow success modes and stop discovering new solutions. We therefore mix: (i) **fresh rollouts** from the environment initial state distribution to discover new successes, and (ii) **CRPS replay rollouts** from suffix states to stabilize learning and propagate success backward along the trajectory.

Replay also chooses *which instruction* to train on (an instruction stored in $\mathcal{B}$) in addition to *where to start* (the suffix state). We keep a non-zero exploration mass to continually discover new successes and prevent the buffer from collapsing to a narrow subset of tasks. Concretely, CRPS induces a mixture over start states:

$$d^{\text{start}} = \begin{cases} (1 - p_{\text{replay}})\, d_0 & \text{(fresh exploration)}, \\ p_{\text{replay}}\, d_{\mathcal{B}}^{(k)} & \text{(suffix replay)}, \end{cases} \tag{13}$$

where d_0 is the environment's initial-state distribution and $d_{\mathcal{B}}^{(k)}$ samples an entry $b \sim \mathcal{B}$, restores s_{t_0} by resetting with z and replaying the first t_0 actions from the stored trace, and uses t_0 from Eq. 8.

4 Experimental Setup

4.1 Agent Interface and Training

Action interface. We follow a ReAct-style loop (Yao et al., 2023): at each environment step, the model is prompted with the task instruction and a truncated interaction history, and then outputs a structured `Action` string executed by the environment. The model may also emit intermediate reasoning (e.g., `Thought`) tokens, but only the `Action` is parsed and executed.

extbfRL algorithm. Unless otherwise stated, we focus on terminal-only sparse-reward policy-gradient training (PPO/RLOO/GRPO family), using a group on-policy optimizer (GRPO-style; Shao et al., 2024) with group size G rollouts per instruction and terminal outcome reward $R \in \{0, 1\}$. We compare full-episode training (from initial states) against replay variants and CRPS.

CRPS configuration. We maintain a buffer $\mathcal{B}$ of historical successful trajectories. During data collection, we sample replay groups with probability p_{replay} (and otherwise sample fresh groups from the initial state distribution). For a replay group, we sample $\tau \sim \mathcal{B}$, restore a suffix start state s_{t_0} by replaying the prefix actions up to t_0 (Eq. 8), and then roll out the current policy from s_{t_0} until termination. We adapt the suffix length k online using the controller in Eq. 11–12 to keep the replay success rate close to a target band (we use $\rho^\star \in [0.2, 0.8]$ throughout unless otherwise stated). Key hyperparameters such as p_{replay} are reported in the Appendix for reproducibility.

4.2 Benchmarks

ALFWorld (Shridhar et al., 2021) is a text-based household environment aligned with ALFRED-style tasks (e.g., pick-and-place, clean, heat/cool). Episodes require multi-step exploration and state tracking, and reward is typically sparse and terminal.

WebShop (Yao et al., 2022) is an interactive e-commerce website environment where the agent must search, browse product pages, and finally click "buy" to select an item satisfying a natural-language instruction. The environment provides a normalized task score in $[0, 1]$ based on how well the selected item matches the instruction constraints (e.g., attributes), and we treat score $= 1$ as success.

4.3 Baselines

We include: (i) **Imitation learning** (if demonstrations are available), including supervised fine-tuning (SFT) and rejection-filtered fine-tuning (RFT). We include imitation baselines to separate the gains from offline behavior cloning vs. online RL. (ii) **Full-episode on-policy RL** from initial states, including PPO (Schulman et al., 2017), RLOO (Kool et al., 2019), and GRPO-style training (Shao et al., 2024). (iii) **Success replay baseline** that reuses historical successful trajectories by restarting from the beginning of a previously successful episode. This baseline is motivated by the most recent and strongest replay-buffer based training paradigms for agentic RL (Lu et al., 2025; Xu et al., 2025); we implement a streamlined but competitive variant tailored to sparse, terminal-reward text environments.

4.4 Metrics

We report (1) **final success rate** at a fixed interaction budget, and (2) **task score** when the benchmark provides a graded score (e.g., WebShop score in $[0, 1]$, measuring the degree of constraint satisfaction).

5 Results

We present main results on ALFWorld and WebShop, then analyze whether CRPS mitigates reward extremity and whether the replay start automatically expands over training.

5.1 Main Results

Table 1 reports the final performance under a fixed interaction budget. Across both model scales, GRPO+CRPS achieves the strongest overall results on ALFWorld (All) and WebShop (Score/Succ.), outperforming full-episode RL baselines as well as vanilla experience replay. On Qwen2.5-1.5B, CRPS improves ALFWorld-All from 0.73 (GRPO) to 0.83 (+10%), and raises WebShop Score/Succ. from 0.67/0.53 to 0.73/0.61. On Qwen2.5-7B, CRPS improves ALFWorld-All from 0.78 (GRPO) to 0.86 (+8%), and WebShop Score/Succ. from 0.77/0.68 to 0.82/0.74.

Compared with `+ExpReplay`, CRPS yields consistent additional gains, indicating that *curriculum replay from successful suffixes* provides more fine-grained and targeted control over the curriculum, and is therefore more effective than simple replay alone. For example, on 1.5B, CRPS improves over

Base Model	Method	ALFWorld							WebShop	
		Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Closed-source	GPT-5	0.88	0.68	0.43	0.46	0.34	0.41	0.56	0.61	0.46
	Gemini-2.5Pro	0.85	0.53	0.67	0.68	0.29	0.59	0.59	0.43	0.36
Qwen2.5-1.5B	SFT	0.79	0.55	0.64	0.70	0.23	0.50	0.52	0.28	0.20
	RFT	0.82	0.55	0.75	0.74	0.42	0.52	0.63	0.43	0.30
	PPO	0.78	0.60	0.77	0.74	0.56	0.55	0.66	0.65	0.53
	RLOO	0.88	0.53	0.71	0.63	0.66	0.57	0.70	0.58	0.46
	GRPO	0.85	0.54	0.85	0.78	0.60	0.54	0.73	0.67	0.53
	+ExpReplay	0.88	0.63	0.87	0.84	0.67	0.65	0.78	0.67	0.55
	+CRPS (ours)	**0.91**	**0.72**	**0.88**	**0.90**	**0.73**	**0.76**	**0.83**	**0.73**	**0.61**
Qwen2.5-7B	SFT	0.64	0.71	0.60	0.68	0.32	0.57	0.58	0.56	0.39
	RFT	0.82	0.66	0.70	0.73	0.64	0.59	0.68	0.63	0.45
	PPO	**0.91**	0.64	**0.92**	**0.89**	**0.80**	0.68	0.80	0.77	0.64
	RLOO	0.88	0.78	0.87	0.81	0.72	0.49	0.76	0.69	0.59
	GRPO	0.90	0.66	0.89	0.75	0.73	0.65	0.78	0.77	0.68
	+ExpReplay	**0.91**	0.73	0.88	0.79	0.77	0.68	0.82	0.80	0.70
	+CRPS (ours)	**0.91**	**0.80**	0.86	0.83	**0.80**	**0.71**	**0.86**	**0.82**	**0.74**

Table 1: Performance on ALFWorld and WebShop. We report ALFWorld success rates for each subtask category and overall (All). For WebShop, we report task Score and Success Rate (Succ.).

+ExpReplay from 0.78 to 0.83 on ALFWorld-All and from 0.67/0.55 to 0.73/0.61 on WebShop; on 7B, CRPS further improves ALFWorld-All from 0.82 to 0.86 and WebShop Score/Succ. from 0.80/0.70 to 0.82/0.74.

We also observe that CRPS brings especially large gains on longer-horizon categories (e.g., Cool/Pick2), consistent with the motivation that replaying near-goal suffixes early and gradually expanding the replay horizon helps propagate learning signal to decisions far from the goal.

Learning curves. Figure 3 shows the learning dynamics on WebShop. Compared with GRPO, CRPS achieves a clear left-shift in the learning curve and exhibits smoother, more stable improvement. Notably, GRPO undergoes an inefficient early phase where exploration is dominated by long-horizon failures, producing many all-zero rollout groups and thus weak learning signal. By replaying successful suffixes, CRPS increases the prevalence of mixed-outcome groups early, which shortens the cold-start period and improves sample efficiency (see Figure 4 for reward-extremity diagnostics).

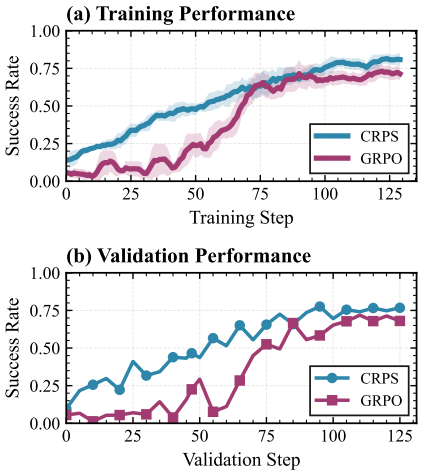

Figure 3: WebShop training and validation success rate over optimization steps for GRPO vs. CRPS. CRPS exhibits faster and more stable improvement, and reaches a higher final success rate.

5.2 Diagnosing Reward Extremity

To test whether CRPS prevents group advantages from collapsing (Eq. 4), we measure reward-distribution statistics over training: (i) the fraction of *all-zero groups* (all rollouts fail), and (ii) group reward entropy.

With binary terminal rewards $R \in \{0, 1\}$, let k

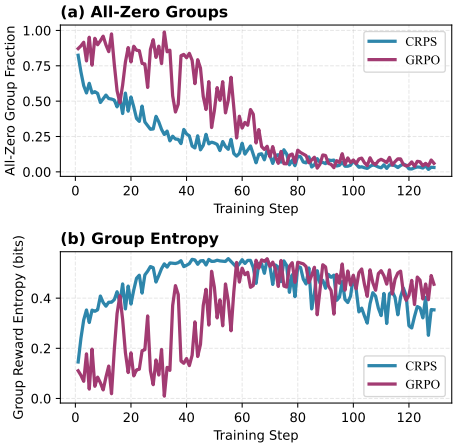

Figure 4: WebShop reward-extremity diagnostics during training. CRPS reduces extreme groups (all-0/all-1) and yields a more mixed reward distribution within groups.

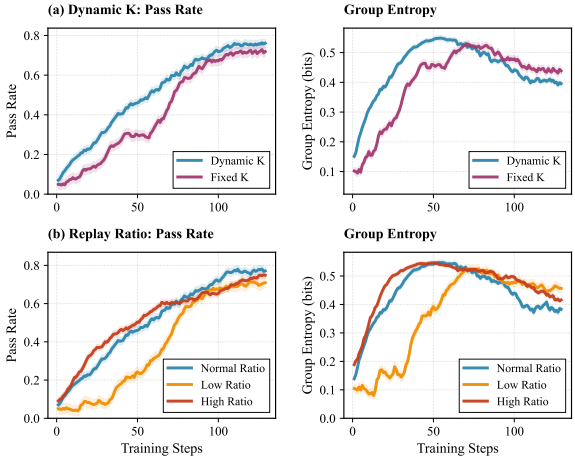

Figure 5: Ablations on ALFWorld with Qwen2.5-1.5B. Top: dynamic k vs. fixed k. Bottom: replay ratio sensitivity. Left: pass rate. Right: group reward entropy (bits).

be the number of successes among the G rollouts for a prompt. The empirical distribution is $\hat{p}(1) = k/G$ and $\hat{p}(0) = 1 - k/G$, and the *group reward entropy* (in bits) is

$$H_{\text{group}} = -\hat{p}(1)\log_2 \hat{p}(1) - \hat{p}(0)\log_2 \hat{p}(0), \quad (14)$$

We report the *all-zero group fraction* (the fraction of prompts with $k = 0$), and summarize entropy at each optimization step by averaging over groups in the batch, $H_{\text{batch}} = \frac{1}{B}\sum_{i=1}^{B} H_{\text{group}}^{(i)}$. Higher entropy indicates more mixed outcomes within groups (less reward extremity), yielding more discriminative relative comparisons and a stronger effective learning signal for group-based RL.

Results and analysis. Figure 4(a) shows that GRPO suffers from a long early stage where a large fraction of rollout groups are *all-zero* (all sampled trajectories fail), implying near-zero within-group reward variance and thus collapsed group advantages in Eq. 4. In contrast, CRPS quickly drives down the all-zero-group fraction, consistent with the idea that starting from successful suffixes makes it substantially more likely to obtain at least some successful rollouts within a group. This effect is also reflected in Figure 4(b): CRPS attains higher group reward entropy earlier in training, indicating more mixed outcomes within groups and therefore more discriminative relative comparisons. Concretely, during the early phase where GRPO frequently produces nearly homogeneous (mostly failing) groups, CRPS maintains a substantially lower all-zero fraction and a higher entropy signal, suggesting that it alleviates the "no-signal" regime caused by reward extremity. Together, these trends explain the faster and more stable learning curves in Figure 3: by reducing reward extremity (all-0/all-1 groups) and increasing within-group outcome diversity, CRPS improves the quality and frequency of effective updates, especially during the cold-start phase of long-horizon tasks.

5.3 Ablations

We report ablations on ALFWorld using Qwen2.5-1.5B as the base model, following the same GRPO-style on-policy optimizer and CRPS pipeline described in Section 4. We track both pass rate and group reward entropy (Section 5.2) to jointly reflect end performance and whether training avoids extreme groups.

We ablate two core design choices in CRPS: (i) the *within-trajectory curriculum* implemented by dynamic suffix-length control, and (ii) the *replay ratio* p_{replay} that trades off exploiting known successes vs. exploring fresh trajectories. Figure 5(a) removes the controller and uses a fixed suffix length for replay. Dynamic k improves early-stage learning and maintains higher reward entropy, consistent with keeping replay outcomes in a learnable band and thus providing stronger within-group learning signal. Figure 5(b) evaluates replay ratios $p_{\text{replay}} \in \{0.05, 0.2, 0.8\}$. Too small a replay ratio (0.05) underutilizes the success buffer and degrades toward full-episode on-policy training, while too large a ratio (0.8) accelerates early learning but reduces exploration and can slow later-stage convergence; the default setting (0.2) best balances these effects.

6 Conclusion

This paper we introduced **CRPS**, a within-trajectory curriculum replay strategy for long-horizon LLM agents trained with sparse terminal rewards. CRPS maintains a buffer of successful trajectories and restarts rollouts from suffix states, while dynamically adapting the suffix length k to keep replay outcomes in a learnable band. This mitigates reward extremity in group-based policy optimization and improves data efficiency without requiring dense reward models or task generation. We outlined an evaluation protocol on ALFWorld and WebShop that emphasizes both success-rate improvements and mechanism-oriented diagnostics. Future work can consider combining CRPS with more fine-grained credit assignment.

Limitations

Like many recent LLM-agent RL studies, our experiments primarily assume environments with (approximately) deterministic state transitions given an action sequence, which makes prefix replay reliable when restoring intermediate states. While CRPS does not conceptually require determinism, stochastic transitions or non-replayable side effects may introduce state mismatch when replaying prefixes, potentially reducing the effectiveness of within-trajectory curricula. An important future direction is to extend CRPS to more stochastic and partially observable settings by incorporating robust state restoration (e.g., snapshotting when available), uncertainty-aware replay selection, and replay strategies that tolerate imperfect reconstruction.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. *Advances in Neural Information Processing Systems*, 30.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48.

Gaurav Chaudhary and Laxmidhar Behera. 2025. Teach: Temporal variance-driven curriculum for reinforcement learning. *arXiv preprint arXiv:2512.22824*.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. 2021. First return, then explore. In *Nature*, volume 590, pages 580–586. Nature Publishing Group.

Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhen Zhang. 2019. Curriculum-guided hindsight experience replay. *Advances in Neural Information Processing Systems*, 32.

Erhu Feng, Wenbo Zhou, Zibin Liu, Le Chen, Yunpeng Dong, Cheng Zhang, Yisheng Zhao, Dong Du, Zhichao Hua, Yubin Xia, and 1 others. 2025a. Get experience from practice: Llm agents with record & replay. *arXiv preprint arXiv:2505.17716*.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025b. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. 2023. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. 2024. Webvoyager: Building an end-to-end web agent with large multimodal models. *arXiv preprint arXiv:2401.13919*.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. 2018. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*.

Jonas Hübotter, Leander Diaz-Bone, Ido Hakimi, Andreas Krause, and Moritz Hardt. 2025. Learning on the job: Test-time curricula for targeted reinforcement learning. *arXiv preprint arXiv:2510.04786*.

Wouter Kool, Herke van Hoof, and Max Welling. 2019. Buy 4 reinforce samples, get a baseline for free! In *ICLR 2019 Workshop*.

Yitao Liu, Chenglei Si, Karthik Narasimhan, and Shunyu Yao. 2025. Contextual experience replay for self-improvement of language agents. *arXiv preprint arXiv:2506.06698*. Accepted to ACL 2025.

Fanbin Lu, Zhisheng Zhong, Shu Liu, Chi-Wing Fu, and Jiaya Jia. 2025. Arpo: End-to-end policy optimization for gui agents with experience replay. *arXiv preprint arXiv:2505.16282*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, and 1 others. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. 2020. Curriculum learning for reinforcement learning domains: A framework and survey. In *Journal of Machine Learning Research*, volume 21, pages 7382–7431.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. 2018. Self-imitation learning. In *International Conference on Machine Learning*, pages 3878–3887. PMLR.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pages 2778–2787. PMLR.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. 2020. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*.

Anirudh Satheesh, Keenan Powell, and Vaneet Aggarwal. 2025a. Distributionally robust self paced curriculum reinforcement learning. *arXiv preprint arXiv:2511.05694*.

Anirudh Satheesh, Keenan Powell, and Hua Wei. 2025b. cmalc-d: Contextual multi-agent llm-guided curriculum learning with diversity-based context blending. *arXiv preprint arXiv:2508.20818*. Accepted to CIKM 2025.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Wentao Shi, Mengqi Yuan, Junkang Wu, Qifan Wang, and Fuli Feng. 2024. Direct multi-turn preference optimization for language agents. *arXiv preprint arXiv:2406.14868*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

Weiting Tan, Xinghua Qu, Ming Tu, Meng Ge, Andy T Liu, Philipp Koehn, and Lu Lu. 2025. Process-supervised reinforcement learning for interactive multimodal tool-use agents. *arXiv preprint arXiv:2509.14480*.

Georgios Tzannetos, Parameswaran Kamalaruban, and Adish Singla. 2025. Curriculum design for trajectory-constrained agent: Compressing chain-of-thought tokens in llms. *Advances in Neural Information Processing Systems*, 38. NeurIPS 2025.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, and 1 others. 2025. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, and 1 others. 2025. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *arXiv preprint arXiv:2505.16421*.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, and 1 others. 2025. Agentgym: Evaluating and training large language model-based agents across diverse environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27914–27961.

Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. 2025. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. *arXiv preprint arXiv:2511.16043*.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. 2025. How memory management impacts llm agents: An empirical study of experience-following behavior. *arXiv preprint arXiv:2505.16067*.

Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Wenyi Zhao, and Yuxiao Dong. 2025. Mobilerl: Online agentic reinforcement learning for mobile gui agents. *arXiv preprint arXiv:2509.18119*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. WebShop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. AgentTuning: Enabling generalized agent abilities for LLMs. *arXiv preprint arXiv:2310.12823*.

A Benchmarks and Interaction Interfaces

We evaluate CRPS on two long-horizon, sparse-reward agent benchmarks that require multi-step interaction.

ALFWorld. ALFWorld (Shridhar et al., 2021) is a text-based embodied environment aligned with ALFRED-style household tasks. Each episode provides a natural-language goal and requires the agent to complete it through multi-turn interaction. The benchmark contains 3,827 task instances spanning six common activity categories: Pick & Place (Pick), Examine in Light (Look), Clean & Place (Clean), Heat & Place (Heat), Cool & Place (Cool), and Pick Two & Place (Pick2). At each step, the agent receives a textual observation and an explicit list of admissible actions; the episode terminates after a fixed horizon, and reward is sparse and success-based.

WebShop. WebShop (Yao et al., 2022) is an interactive e-commerce environment where the agent must search, browse product pages, and eventually select an item that satisfies a natural-language shopping instruction. The environment is presented as a simulated HTML-based website with a large action space (e.g., searching, clicking links/buttons, navigating pages), and includes over 1.1 million products and 12k instructions. Following standard practice in WebShop, the environment returns a normalized score in $[0, 1]$; we treat score $= 1$ as success.

B Environment Interface and Prompts

We use a ReAct-style loop: at each environment step, the policy is prompted with the task instruction and a truncated interaction history, and produces an output of the form

$$\langle \texttt{think} \rangle \, \tau \, \langle / \texttt{think} \rangle \, \langle \texttt{action} \rangle \, a \, \langle / \texttt{action} \rangle,$$

where only the <action> span is parsed and executed by the environment. Each step's prompt includes: (i) the task instruction, (ii) the current step count, (iii) a short window of recent action–observation history (history length 2 for ALFWorld and WebShop), (iv) the current observation, and (v) the current action space provided by the environment (admissible actions in ALFWorld; available actions in WebShop).

C Prompt Templates

This section lists the exact prompts used to format the agent input for ALFWorld and WebShop.

Prompts for ALFWorld Environment

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

Now it's your turn to take an action.
You should first reason step-by-step
    about the current situation. This
    reasoning process MUST be enclosed
    within <think> </think> tags.
Once you've finished your reasoning, you
    should choose an admissible action
    for current step and present it
    within <action> </action> tags.
```

Prompts for WebShop Environment

```
You are an expert autonomous agent
    operating in the WebShop e-commerce
    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:
[
{available_actions}
].

Now it's your turn to take one action for
     the current step.
You should first reason step-by-step
    about the current situation, then
    think carefully which admissible
    action best advances the shopping
    goal. This reasoning process MUST be
    enclosed within <think> </think> tags
    .
Once you've finished your reasoning, you
    should choose an admissible action
    for current step and present it
    within <action> </action> tags.
```

D Environment Reset and State Snapshots

CRPS requires resetting the environment to an intermediate state s_{t_0} from a successful trajectory. Depending on the benchmark implementation, this can be supported via: (i) state serialization (the environment exposes a function to serialize and restore the full simulator state), or (ii) deterministic replay (store the initial seed/snapshot and deterministically re-apply the action sequence up to t_0 to reconstruct s_{t_0}). In either case, the replay rollout from s_{t_0} is executed with the *current* policy to remain on-policy.

E Dynamic k Hyperparameters

The dynamic controller in Eq. 11–12 introduces $\rho^\star = [\rho_l, \rho_u]$ (target band for replay success rate), λ (EMA smoothing), and Δ (step size). In all experiments, we use the following values unless otherwise stated:

- target band $\rho^\star = [0.2, 0.8]$;
- EMA smoothing $\lambda = 0.9$;
- step size $\Delta = 2$.

E.1 Dynamic Controller Parameter Analysis

To understand the impact of the step size Δ and the EMA smoothing parameter λ on the dynamic k controller, we evaluate the average convergence steps of curriculum replay samples under different parameter settings. Convergence steps measure how many iterations are required, on average, for samples with extreme reward distributions to converge into the target success rate interval. All experiments use Qwen2.5-1.5B-Instruct with settings consistent with the main paper.

Step size Δ. Table 2 reports convergence steps for varying Δ. Larger step sizes accelerate convergence but risk instability: on ALFWorld, $\Delta \geq 8$ leads to unstable convergence; on WebShop, which requires fewer average steps to converge, this instability threshold is lower ($\Delta \geq 6$). Small values ($\Delta \in \{1, 2\}$) provide stable, acceptable performance across both tasks.

Smoothing λ. Table 3 reports convergence steps for varying λ. Smaller λ causes a lagged estimation of the replay success probability: as the policy improves, new high-success rollouts receive lower weight, leading to higher convergence steps. At $\lambda = 1.0$ (no smoothing), the success rate estimate shows some oscillation but remains acceptable.

Δ	ALFWorld	WebShop
1	4.13	2.96
2	2.96	2.01
4	1.88	1.33
6	1.58	—
8	—	—

Table 2: Average convergence steps under varying step size Δ ($\lambda = 0.9$). "—" indicates unstable convergence.

λ	ALFWorld	WebShop
0.1	17.68	10.74
0.5	6.67	5.53
0.9	2.96	2.01
1.0	2.87	2.03

Table 3: Average convergence steps under varying smoothing λ ($\Delta = 2$).

Overall, CRPS's negative-feedback dynamic curriculum mechanism is insensitive to these control parameters. Reasonable parameter combinations have minimal impact on final performance, and convergence step variation constitutes only a small fraction of the total RL training process.

F Additional Discussion for Ablations

We provide additional interpretation for the ablations in Section 5.3.

Removing dynamic k. On **ALFWorld**, we remove dynamic suffix-length control and use a fixed $k = 10$ for replay. Dynamic k improves early-stage learning dynamics by keeping replay rollouts in a learnable band (avoiding both all-fail and all-success groups), which increases group reward entropy and provides a stronger within-group learning signal for group-based on-policy optimization.

Replay ratio sensitivity. We evaluate replay ratios $p_{\text{replay}} \in \{0.05, 0.2, 0.8\}$. When $p_{\text{replay}} = 0.05$, CRPS underutilizes the success buffer and becomes close to full-episode on-policy training; when $p_{\text{replay}} = 0.8$, replay stabilizes and accelerates early training but reduces exploration and can slow the discovery of new successes.

G LLM Assistance

We used large language models (LLMs) in a limited, assistive manner during manuscript preparation. Specifically, LLMs were used for English grammar/style polishing and spelling checks, and

to help surface potentially relevant literature for subsequent manual verification and citation. In addition, some icon assets used in the method schematic were generated with AI-based tools and then manually curated/edited by the authors. All technical contributions, experimental design, implementation, and result interpretation were performed and verified by the authors.