

Group Structural Proximity Shapes Exploration-Exploitation in Tool-Use Reinforcement Learning

Lexiao Zou^{1,2} Yitong Yao² Gongwei Chen¹ Wenmin Deng²

Chao Wang² Shuangyong Song^{2,†} Miao Zhang^{1,†}

¹Harbin Institute of Technology, Shenzhen

²Xingchen AGI Lab, China Telecom Artificial Intelligence Technology (Beijing) Co., Ltd

[†]Corresponding authors.

{lokshawchau@gmail.com, zhangmiao@hit.edu.cn, songshy@chinatelecom.cn}

Abstract

Group-based reinforcement learning typically relies on coarse outcome-level signals, such as group accuracy or variance, for data selection or replay. Yet in multi-turn tool-use rollout groups, these signals miss structural differences between successful and failed trajectories: a failure may diverge into an unrelated tool chain or differ from a successful trajectory only at a local decision. We therefore introduce Group Structural Proximity (GSP), which measures failed-to-successful rollout alignment within a group using tool dependency graphs induced by tool calls and argument flow. A controlled diagnostic study with outcome-statistics-matched subsets further shows that GSP captures distinct learning dynamics: high-GSP data drives localized policy refinement, while low-GSP data induces broader policy movement and higher-entropy exploration. This positions GSP as an effective metric on the exploration–exploitation tradeoff. Building on this insight, we introduce Proximity-Guided Policy Optimization (PGPO), which considers replay buffers with high- and low- GSP under an Exploration-to-Refinement Schedule (ERS) for group-based training. On τ^2 -Bench and ACEBench Agent, PGPO improves sample efficiency and downstream tool-use performance, demonstrating the benefit of structure-aware group comparison in multi-turn tool-use reinforcement learning.

1 Introduction

Large language models are increasingly being trained as interactive agents rather than standalone text generators, learning to invoke external tools across many turns of user-agent-environment interaction (Yao et al., 2023, 2025; Barres et al., 2025; Tu et al., 2026). In this multi-turn tool-use setting, reinforcement learning is rollout-expensive: each episode requires many tool calls, argument bindings, and environment transitions before a single terminal reward is observed. The currently

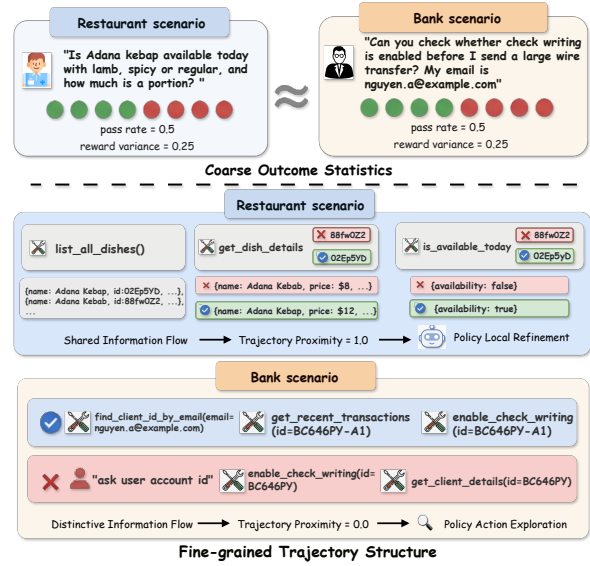


Figure 1: Coarse outcome statistics versus fine-grained rollout structure. GSP quantifies the structural contrast by measuring dependency-graph overlap between successful and failed rollouts in the same group.

dominant paradigm for this regime is group-based RL (Shao et al., 2024; Feng et al., 2025a; He et al., 2026), which samples multiple rollouts per scenario and computes group-wise advantages—making the cost of an additional rollout an explicit budget item. Understanding what kind of data drives what kind of policy update is therefore valuable for allocating a fixed rollout budget effectively.

A growing body of work scores training queries using statistics of rollout outcomes. The most direct signal is reward variance within a rollout group, used to retain non-degenerate gradient updates (Wang et al., 2026; Zhan et al., 2025); pass-rate-based scores instead treat outcome dispersion that peaks at intermediate pass rates as a learnability signal (Baroian and Berger, 2026; Cui et al., 2026); and recent extensions refine this with success-rate-based learnability (Li et al., 2025) and

Bayesian outcome-uncertainty (Zhou et al., 2026a). For multi-turn tool use, however, such reward-distribution summaries are insufficient: they ignore the structural relationship between successful and failed rollouts within the same group—which tools are called, in what order, and how their outputs feed into subsequent decisions. Recent work (Zou et al., 2026b,a) on credit assignment in long-horizon agents further shows not all failures are equally amenable to RL learning—some expose localized repair amenable to precise credit assignment, while others propagate an early error through the entire execution, calling for broader structural exploration. This structural relationship, invisible to outcome statistics, determines what kind of learning signal the success–failure contrast actually provides. Figure 1 illustrates both regimes with concrete rollout examples.

We introduce *Group Structural Proximity* (GSP) to quantify this structural relationship, measuring how closely the tool dependency graphs of failed rollouts align with those of successful ones within the same rollout group. To validate that GSP captures a training-relevant signal, we conduct a controlled study outcome-statistics-matched subsets drawn from different GSP ranges and measure KL divergence and policy entropy over training. High-GSP scenarios induce sharp, localized policy updates with faster KL growth and lower entropy, consistent with exploitation; low-GSP scenarios drive diffuse policy movement with slower KL growth and sustained higher entropy, consistent with structural exploration. We turn this diagnostic signal into *Proximity-Guided Policy Optimization* (PGPO), which preserves both sides of the tradeoff rather than committing to a single replay regime: low-proximity replay keeps early training exploratory, while high-proximity replay later concentrates updates on local repair through an Exploration-to-Refinement Schedule (ERS).

Experiments on τ^2 -Bench and ACEBench Agent demonstrate that PGPO consistently improves downstream tool-use performance over curriculum and replay baselines. On τ^2 -Bench, PGPO raises average pass@1 from 35.4 to 46.0 over the base model. On ACEBench Agent, it improves multi-step end-to-end accuracy by 20.0 points and process accuracy by 19.1 points, indicating the learned policy follows more valid intermediate tool-use behavior. Ablation results confirm that both the high- and low-proximity buffers are necessary, and that GSP-based replay outperforms reward-variance re-

play.

Our contributions are:

- We propose Group Structural Proximity (GSP) to capture structural proximity within rollout groups, revealing training signals beyond outcome statistics that separate local exploitation from broader exploration.
- We introduce Proximity-Guided Policy Optimization (PGPO), a GSP-guided policy optimization method that uses exploration-to-refinement schedule to arrange data flow by GSP score.

2 Related Work

Multi-turn agent tool-use policy learning.

Early tool-use agents showed that language models can interleave reasoning with actions and call external tools (Yao et al., 2023; Schick et al., 2023; Li et al., 2024; Wang et al., 2024, 2025c; Liu et al., 2025a; Lyu et al., 2025; Jiang et al., 2026; Wang et al., 2025b; Zhao et al., 2025). ToolLLM scaled API supervision (Qin et al., 2024), while ToolSand-box, τ -bench, τ^2 -Bench, and ACEBench moved evaluation toward stateful user-agent-tool interaction (Lu et al., 2025b; Yao et al., 2025; Barres et al., 2025; Chen et al., 2025a). Data and environment synthesis work further expands this setting through simulated multi-turn interaction, programmatic environment generation, and graph-based tool-use synthesis (Prabhakar et al., 2025; Song et al., 2026; Yin et al., 2025; Tu et al., 2026). These works make multi-turn tool-use RL feasible, but they mainly define tasks, tools, or executable environments. Policy learning methods such as GRPO provide critic-free group-based optimization (Shao et al., 2024); DAPO and Dr.GRPO analyze or stabilize this family of objectives (Yu et al., 2025; Liu et al., 2025b); and GiGPO/HGPO adapt group construction to long-horizon agents (Feng et al., 2025a; He et al., 2026). Our work is complementary: rather than changing only the optimization objective, it asks whether the successful and failed rollouts in a group share a proximate tool dependency backbone.

Data-centric policy training. A separate line of work improves sample efficiency by selecting, scheduling, or replaying informative data. Classical prioritized replay samples transitions by estimated utility (Schaul et al., 2015), and prioritized level replay reuses levels with high learning po-

tential (Jiang et al., 2021). In LLM and agent RL, reward-variance filtering and adaptive rollout methods select non-degenerate groups (Wang et al., 2026), while long-horizon interactive RL and GUI-agent training use replay to reduce expensive environment interaction (Chen et al., 2025d; Lu et al., 2025a; Chen et al., 2025c; Zhou et al., 2026b; Chen et al., 2025b). Recent reasoning replay methods further manage experiences using correctness, entropy, or stale-policy correction (Zhan et al., 2025; Zhang et al., 2025a). Curriculum and online data-loop methods similarly adapt sampling or problem construction using prompt length, reward, confidence, difficulty, distribution-level utility, or model failure patterns (Song et al., 2025; Feng et al., 2025b; Cai and Sugiyama, 2026; Zhang et al., 2025c; Chen et al., 2025e; Wang et al., 2025a; Qi et al., 2024; Zhang et al., 2025b; Shi et al., 2026). These approaches treat exploration and exploitation as a function of scalar outcomes, uncertainty, difficulty, past success, or weakness coverage. GSP instead measures the structure of the success-failure contrast within a rollout group: high-proximity replay supports local repair, while low-proximity replay preserves broader structural exploration.

3 Group Structural Proximity

We study reinforcement learning for multi-turn tool-use agents under sparse terminal rewards. Group-based RL methods compare multiple rollouts from the same scenario, but their training signal is usually derived from scalar outcome statistics. Our method adds a structural view of the same rollout group.

3.1 Preliminaries

Multi-turn tool use as a POMDP. We treat each training data point as a *scenario*—the multi-turn analogue of a query, in which a user intent unfolds over one or more dialogue turns against an executable environment and tool set. We associate each scenario x with a POMDP whose terminal reward $R(\tau; x) \in \{0, 1\}$ judges whether a rollout $\tau = (o_0, a_0, \dots, o_T)$ sampled from $\pi_\theta(a_t | h_t)$ satisfies the user intent and constraints. The full state, action, and observation spaces are detailed in Appendix A. The terminal reward decides whether a trajectory succeeds, but does not localize which tool dependency caused the outcome—motivating a representation that preserves tool ordering and inter-turn information flow.

Group-based reinforcement learning. For each scenario x , group-based RL samples a rollout group $\mathcal{R}(x) = \{\tau_i\}_{i=1}^G$ with terminal rewards $r_i = R(\tau_i; x)$. GRPO is a representative group-based objective. For a token or action at step t in rollout τ_i , the importance ratio is $\rho_{i,t}(\theta) = \pi_\theta(a_{i,t} | h_{i,t}) / \pi_{\theta_{\text{old}}}(a_{i,t} | h_{i,t})$, and the group-normalized advantage uses the centered reward r_i relative to the group mean. The clipped objective applies a min over the clipped and unclipped surrogate together with a KL regularizer to a reference model (Shao et al., 2024); the full equation is given in Appendix B. When all rollouts succeed or all fail, the group has zero reward variance and contributes no gradient signal under GRPO. More broadly, prior work uses outcome-level statistics to identify informative scenarios: RAGEN-2 filters prompts by reward variance as a proxy for group-level signal strength (Wang et al., 2026), while ExGRPO manages rollout experiences using correctness and entropy to balance exploration and exploitation (Zhan et al., 2025). These scalar criteria are useful but incomplete for long-horizon tool use. They summarize the outcome distribution, but not the rollout structure that generated it—leaving a gap between scalar signal and the dependency relations that determine credit assignment.

3.2 Group Structural Proximity

From tool dependency graphs to success-failure comparison. Tool dependency graphs represent the information flow between tool calls in multi-turn agent tasks. Prior work constructs such graphs offline from environment specifications or demonstration corpora—nodes represent tools, edges encode output-to-input flow—to support environment synthesis and training data generation (Tu et al., 2026; Yin et al., 2025). Our motivation is different. Rather than describing what a task requires, we use tool dependency graphs to compare what a policy actually did, measuring how closely the execution dependency graph of failed rollouts aligns with that of successful ones within the same rollout group. This success-failure contrast is invisible to outcome statistics but directly relevant to what the policy can learn from the group.

Node representation. For each tool call at step t , we construct a node

$$v_t = (\text{tool}_t, \text{schema}_t, \text{parent}_t),$$

where tool_t is the normalized tool name, schema_t is the set of argument keys with extractable scalar values, and parent_t records the earliest semantic sources of those values (the user request or a previous tool observation). Two nodes match only when their tool, argument schema, and parent signature agree.

Trajectory proximity and Group Structural Proximity. Let $V^+ = V(\tau^+) = (v_1^+, \dots, v_m^+)$ and $V^- = V(\tau^-) = (v_1^-, \dots, v_n^-)$ be the node sequences of a successful and a failed rollout. We define the number of matched nodes $K(\tau^+, \tau^-)$ as the length of the longest common subsequence of V^+ and V^- under the node-match criterion:

$$K(\tau^+, \tau^-) = \max\{k \mid \text{Match}(v_{i_\ell}^+, v_{j_\ell}^-) = 1 \forall \ell\},$$

where $i_1 < \dots < i_k$ and $j_1 < \dots < j_k$. Trajectory proximity is

$$\text{TP}(\tau^+, \tau^-) = \frac{K(\tau^+, \tau^-)}{|V^+| + |V^-| - K(\tau^+, \tau^-)}.$$

The Jaccard-style normalization makes trajectory proximity high when success and failure reuse most of the same tool dependency graph, and low when either rollout contains unmatched detours. Denoting $\mathcal{S}(x) = \{\tau_i : r_i = 1\}$ and $\mathcal{F}(x) = \{\tau_i : r_i = 0\}$, the group-level GSP is

$$\text{GSP}(x) = \frac{1}{|\mathcal{S}(x)| + |\mathcal{F}(x)|} \sum_{\tau^+ \in \mathcal{S}(x)} \sum_{\tau^- \in \mathcal{F}(x)} \text{TP}(\tau^+, \tau^-). \quad (1)$$

4 Diagnostic Study: GSP Shapes Training Dynamics

4.1 Study Design

To investigate how group-level GSP affects training dynamics, we use Qwen3-4B-Instruct-2507 (Yang et al., 2025a) as the base model and sample eight rollouts per scenario on the ToolScale training set (Su et al., 2025), with Qwen3.5-35B-A3B (Yang et al., 2025a) as the user simulator. To isolate structural proximity from outcome difficulty, we group scenarios by their pass rate over the eight rollouts; within each pass-rate bucket we take the GSP top and bottom quartiles, and pool 500 scenarios across buckets to form the *high-GSP* and *low-GSP* subsets, with a same-sized *random* subset drawn uniformly from the same buckets. By construction the three subsets share the same pass-rate distribution but lie at the two extremes (and the middle) of the GSP

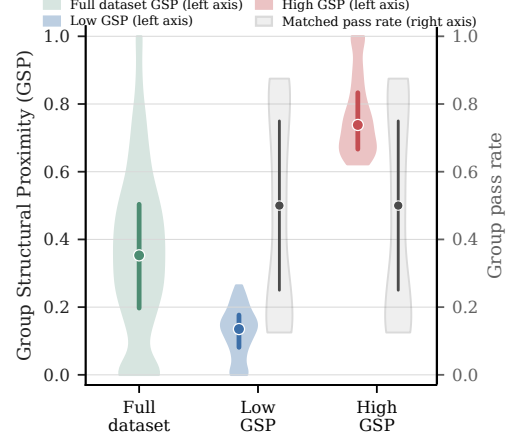


Figure 2: Distribution of GSP and pass rates over the full set and the high-/low-GSP subsets.

axis (Figure 2), so any difference in training dynamics is attributable to structural proximity rather than outcome difficulty. Detailed subset composition statistics are provided in Appendix E. We then run independent GRPO training on each subset and track policy entropy and KL divergence from the reference model.

4.2 Optimization Dynamics

Figure 3 shows the results. The entropy and KL curves reveal complementary dynamics. On the high-GSP subset, KL divergence grows faster and policy entropy drops more sharply: the policy concentrates its updates around the locatable deviation between the shared dependency backbone and the failure, converging quickly toward a refined local solution. On the low-GSP subset, KL growth is slower and entropy remains elevated throughout training: without a shared dependency backbone, the gradient signal is diffuse and the policy explores a broader range of tool-use behaviors. Random selection falls between these two regimes in both metrics. The avg@1 curve reveals the downstream consequence of this tradeoff: high-GSP training reaches a stronger intermediate checkpoint at step 30 (38.6 vs. 37.9), reflecting faster early exploitation, but low-GSP training overtakes it by the end of training (40.0 vs. 38.4), reflecting the long-term benefit of sustained exploration.

4.3 Implications for Replay

These dynamics confirm that GSP partitions scenarios along an exploitation-exploration axis that basic outcome statistics alone cannot capture. High-GSP scenarios support local refinement, while low-GSP

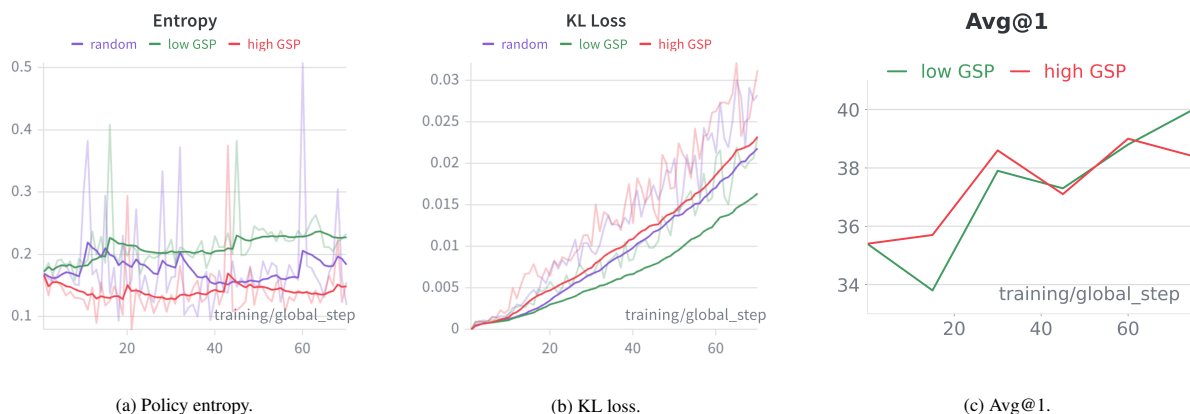


Figure 3: Training dynamics of GRPO on outcome-statistics-matched high-GSP, low-GSP, and random subsets, evaluated on τ^2 -Bench.

scenarios preserve structural exploration. This suggests that GSP should not be used as a one-sided filter: neither high- nor low-proximity scenarios alone provide the full training signal. The diagnostic study therefore motivates a replay mechanism that retains both regimes and schedules their mixture over training.

5 Proximity-Guided Policy Optimization

Motivation. The diagnostic study in Section 4 shows that high- and low-GSP scenarios induce qualitatively different training dynamics, and that neither regime alone achieves the best final performance. Prioritizing high-GSP scenarios accelerates early exploitation but limits long-run exploration, while prioritizing low-GSP scenarios sustains exploration but slows convergence. We therefore introduce *Proximity-Guided Policy Optimization* (PGPO), a group-based policy optimization method that uses GSP to decide which scenarios should be replayed and how replay should be mixed over training.

PGPO has two core designs. First, it maintains two proximity-specific replay buffers that separately retain high-GSP and low-GSP scenarios. Second, it uses a scheduled buffer sampler that changes the replay mixture over training, allocating more budget to low-proximity structural exploration early and shifting toward high-proximity local refinement later. The rest of the RL update remains standard: replayed scenarios are re-rolled by the current policy, and the resulting rollout groups are optimized with the same GRPO objective.

5.1 Dual Proximity Buffers

PGPO maintains two replay buffers partitioned by GSP score. The *high-proximity buffer* stores scenarios where failed rollouts share most of the successful dependency backbone: the policy has already found the correct tool chain, and replay should refine the localized deviation that separates success from failure. The *low-proximity buffer* stores scenarios where failed rollouts diverge structurally from successful ones: the policy has not yet converged on a shared procedure, and replay should encourage broader behavioral search. Figure 4 illustrates the full pipeline.

Each replayed item is a scenario identifier, not a stored trajectory. This choice is important for policy optimization: PGPO does not train on stale rollouts. Instead, the current policy re-rolls every selected scenario to obtain fresh rollout groups, and the standard GRPO update is applied to the combined batch of fresh and replayed scenarios.

After rollout, PGPO recomputes GSP for each scenario. Scenarios where all rollouts succeed or all fail are excluded. For the remaining scenarios, each computed GSP value is compared against both buffers: it replaces the current minimum of the high-proximity buffer if it exceeds that minimum, and replaces the current maximum of the low-proximity buffer if it falls below that maximum, maintaining each buffer as a heap ordered by GSP. This keeps the buffers focused on the most structurally informative scenarios under the current policy. Buffer warm-start and edge cases are detailed in Appendix C.

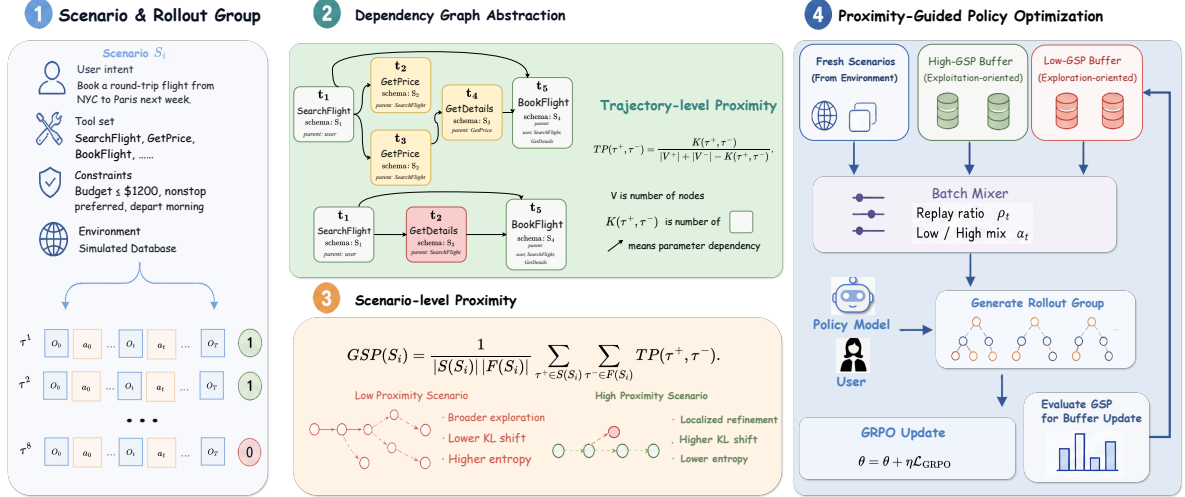


Figure 4: Pipeline of PGPO. Dual Proximity Buffers retain high- and low-proximity scenarios separately, while an Exploration-to-Refinement Schedule mixes them with fresh samples during training. The two buffers are updated by comparing the recomputed GSP of each scenario against the current buffer contents.

5.2 Exploration-to-Refinement Schedule

The second design controls how the two buffers are mixed during training. We call this rule the *Exploration-to-Refinement Schedule* (ERS), since it shifts replay from low-proximity structural exploration toward high-proximity local refinement over training. Annealed schedules are common in curriculum and replay-based reinforcement learning, where the training distribution is adjusted as the policy improves (Yang et al., 2025b; Wang et al., 2025a; Chen et al., 2025e; Zhang et al., 2025c). ERS differs in what it schedules: rather than annealing by task difficulty, reward, or prompt-level uncertainty, it schedules the structural relation between successful and failed rollouts within a group.

The direction of the schedule follows directly from the diagnostic study. Low-GSP scenarios keep entropy high and encourage broader structural search, which is most valuable before the policy has settled on reliable tool-use procedures. High-GSP scenarios drive sharper KL movement and localized repair, which becomes more useful once the policy has discovered a plausible dependency backbone. A static mixture would treat these two regimes as equally useful throughout training, while ERS makes the replay distribution match the changing role of exploration and refinement. The ablation later confirms this design choice: keeping both buffers but removing ERS reduces average pass@1 from 46.0 to 44.5.

At each training step, ERS draws $M_t = \lfloor \rho_t B \rfloor$ scenarios from the replay buffers, where B is the

number of fresh scenarios and $\rho_t \in [0, 1]$ is the replay ratio. The M_t slots are split by a mixing ratio α_t : $M_t^{\text{low}} = \lfloor \alpha_t M_t \rfloor$ from the low-proximity buffer and $M_t^{\text{high}} = M_t - M_t^{\text{low}}$ from the high-proximity buffer. Within each buffer, scenarios are sampled by their proximity extremeness: high-buffer scenarios with probability proportional to GSP, low-buffer scenarios with probability proportional to $(1 - \text{GSP})$. We schedule α_t as a decaying exponential

$$\alpha_t = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \exp(-t/\tau),$$

starting high to favor structural exploration and decaying to shift the replay budget toward local refinement (Schaul et al., 2015).

6 Experiments

6.1 Experimental Setup

Training data and protocol. We train on the ToolScale split from ToolOrchestra (Su et al., 2025), which contains executable multi-turn tool-use scenarios across ten virtual domains. Each scenario is evaluated by a sparse binary reward that jointly checks final database state, action, and dialogue-content matching. Before training, we sample eight rollouts per scenario and remove all-correct or all-incorrect groups, yielding 3,840 training scenarios. Training uses a per-step batch of 64 scenarios, KL coefficient $\beta = 0.001$, sequence length 32768, 30 interaction turns, and rollout temperature 1.0. The user simulator is Qwen3.5-35B-A3B without thinking mode.

Table 1: Comparison of PGPO with baselines on τ^2 -Bench. pass@4 counts a task as solved if any of its four trials succeeds; pass^4 requires all four to succeed. Trained methods report mean $\pm$ std. over 3 seeds.

Model	Airline			Retail			Telecom			Avg.		
	p@1	p@4	p ⁴	p@1	p@4	p ⁴	p@1	p@4	p ⁴	p@1	p@4	p ⁴
<i>Qwen3-4B-Instruct-2507</i>												
Base	34.0	54.0	12.0	47.6	66.7	14.0	24.6	43.9	7.9	35.4	54.8	11.3
+ DAPO	42.0 $\pm$ 2.0	60.0 $\pm$ 2.0	14.7 $\pm$ 1.2	51.8 $\pm$ 0.9	71.7 $\pm$ 1.0	15.8 $\pm$ 0.9	33.6 $\pm$ 1.8	47.9 $\pm$ 1.3	8.4 $\pm$ 1.3	42.5 $\pm$ 1.6	59.9 $\pm$ 1.4	13.0 $\pm$ 1.1
+ ExGRPO	41.3 $\pm$ 1.2	58.7 $\pm$ 1.2	14.0 $\pm$ 2.0	52.1 $\pm$ 0.5	72.8 $\pm$ 1.8	17.2 $\pm$ 1.0	32.5 $\pm$ 0.9	48.8 $\pm$ 1.3	7.9 $\pm$ 1.5	42.0 $\pm$ 0.9	60.1 $\pm$ 1.4	13.0 $\pm$ 1.5
+ Ours	46.7 $\pm$ 1.2	64.0 $\pm$ 2.0	16.7 $\pm$ 1.2	53.5 $\pm$ 1.8	75.7 $\pm$ 0.5	20.5 $\pm$ 1.3	36.8 $\pm$ 1.8	52.3 $\pm$ 1.0	9.9 $\pm$ 1.3	45.7 $\pm$ 1.6	64.0 $\pm$ 1.2	15.7 $\pm$ 1.3
<i>Qwen3.5-2B</i>												
Base	64.0	78.0	44.0	54.8	86.8	21.9	84.2	98.5	57.9	67.7	87.8	41.3
+ DAPO	66.0 $\pm$ 1.2	80.0 $\pm$ 2.0	48.0 $\pm$ 2.0	57.6 $\pm$ 1.3	87.1 $\pm$ 0.5	23.1 $\pm$ 1.8	86.5 $\pm$ 1.4	98.5 $\pm$ 0.8	58.4 $\pm$ 1.0	70.0 $\pm$ 1.3	88.5 $\pm$ 1.1	43.2 $\pm$ 1.6
+ ExGRPO	66.0 $\pm$ 2.0	82.0 $\pm$ 1.2	46.0 $\pm$ 1.2	58.1 $\pm$ 1.3	88.5 $\pm$ 0.8	24.5 $\pm$ 1.8	86.2 $\pm$ 1.8	99.1 $\pm$ 0.8	58.1 $\pm$ 1.3	70.1 $\pm$ 1.7	89.9 $\pm$ 0.9	42.9 $\pm$ 1.4
+ Ours	70.0 $\pm$ 2.0	84.0 $\pm$ 2.0	50.0 $\pm$ 1.2	61.4 $\pm$ 1.8	90.1 $\pm$ 0.5	26.9 $\pm$ 1.0	89.7 $\pm$ 0.5	99.1 $\pm$ 0.5	60.5 $\pm$ 1.7	73.7 $\pm$ 1.4	91.1 $\pm$ 1.0	45.8 $\pm$ 1.3

Evaluation benchmarks. We evaluate on τ^2 -Bench and the Agent subset of ACEBench, both of which require stateful user-agent-tool interaction. On τ^2 -Bench, we report domain-level and average pass rates under multiple sampling criteria. On ACEBench Agent, we report both end-to-end success and process accuracy.

6.2 Main Results

Table 1 reports τ^2 -Bench results.

PGPO outperforms RL baselines across all metrics and domains. PGPO raises average pass@1 from 35.4 to 45.7 over the base model, exceeding DAPO (42.5) and ExGRPO (42.0) under the same rollout budget. The gain is consistent across domains: Airline +4.7, Retail +1.7, Telecom +3.2 over DAPO. Telecom, which requires the most complex multi-turn state tracking, shows a large margin, suggesting that GSP-guided replay is particularly effective when the dependency graph is deep. The pattern also generalizes beyond the main backbone: on Qwen3.5-2B, the same ordering holds on all nine domain $\times$ metric combinations, with PGPO improving average pass@1 by +6.0 over the untrained backbone (73.7 vs. 67.7), indicating that the observed gains are not specific to one backbone’s scale or architecture.

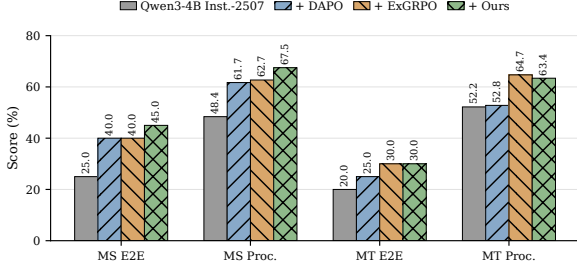
Replay improves consistency, not just peak performance. The three sampling metrics expose different aspects of policy quality. pass@1 reflects the expected quality of a single deployment, directly measuring what a user experiences in practice. pass@4 measures coverage: whether the model has at least one viable solution strategy in its distribution, which is informative about the breadth of learned behaviors. pass^4 is the strictest criterion, requiring all four trials to succeed, and is therefore sensitive to the tail of the failure distribution rather than the mean—a model with high

pass@4 but low pass^4 has learned to solve the task occasionally but has not eliminated the failure modes that cause it to fail most of the time. PGPO improves all three average metrics simultaneously: pass@1 from 35.4 to 45.7, pass@4 from 54.8 to 64.0, and pass^4 from 11.3 to 15.7. The joint improvement is meaningful: pass@4 rising alongside pass^4 indicates that PGPO both broadens the set of solvable scenarios and reduces within-scenario variance. The strongest pass^4 improvement appears in Retail, where PGPO reaches 20.5 compared with 17.2 for ExGRPO and 14.0 for the base model, consistent with the GSP hypothesis: high-proximity replay targets the localized failure modes that cause repeated failures on the same scenario, directly attacking the variance that pass^4 penalizes.

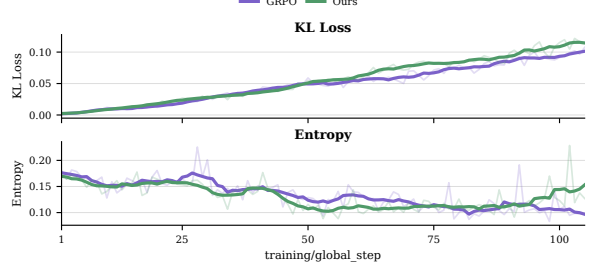
Process-level gains on ACEBench confirm structural improvement. Figure 5 summarizes the ACEBench Agent results and training dynamics. PGPO improves the base model by 20.0 points on multi-step end-to-end accuracy and 19.1 points on multi-step process accuracy. The process score measures the quality of intermediate tool-use steps independently of the final outcome, so a large process-score gain indicates that the learned policy is following more valid tool-call sequences, not merely reaching correct terminal states by chance. The near-equal magnitude of the end-to-end and process gains (20.0 vs. 19.1) suggests that the improvement is structurally grounded: the model has learned better intermediate tool-use behavior, and the final-outcome improvement follows from that.

6.3 Ablation Study

Table 2 isolates each design component. All variants use the same rollout budget and are evaluated on τ^2 -Bench average pass@1.



(a) ACEBench Agent.



(b) KL and entropy dynamics.

Figure 5: ACEBench Agent results and training dynamics under PGPO. The method combines Dual Proximity Buffers with an Exploration-to-Refinement Schedule (ERS) to preserve early structural exploration and later local refinement. MS/MT denote multi-step/multi-turn; E2E and Proc. denote end-to-end and process scores.

Table 2: Ablation on Qwen3-4B-Instruct-2507. *ERS*: whether the schedule on α_t is enabled. *Buffer*: which GSP-based buffer or alternative criterion is used.

Variant	ERS	Buffer	Avg.
Full method	✓	Dual GSP	46.0
w/o ERS	×	Dual GSP	44.5
w/o Low-GSP Buffer	×	High GSP	42.7
w/o High-GSP Buffer	×	Low GSP	42.3
Reward-variance Replay	×	Reward variance	43.4

Both buffers are necessary and neither dominates. Removing either buffer causes a larger drop than removing the schedule alone. Dropping the low-GSP buffer reduces average pass@1 to 42.7, and dropping the high-GSP buffer reduces it to 42.3, both well below the 44.5 obtained when the schedule is removed but both buffers are retained. This ordering reveals that the performance gain is not attributable to one preferred replay source: the high-proximity buffer alone cannot sustain exploration, and the low-proximity buffer alone cannot drive local refinement. The full method requires both to alternate between structural search and localized improvement across the training trajectory.

ERS provides additional gain beyond static mixing. Retaining both buffers but removing ERS drops average pass@1 from 46.0 to 44.5. This 1.5-point gap shows that the composition of replay, not just its presence, matters. A static mixture treats early and late training identically, whereas ERS allocates more low-GSP replay early—when the policy has not yet converged on shared tool-use procedures—and shifts toward high-GSP replay later, when local refinement becomes the binding constraint. ERS therefore acts as a form of curriculum over the replay distribution.

Structural proximity outperforms outcome dispersion as a replay criterion. Replacing GSP with reward variance as the replay criterion yields 43.4 average pass@1, 2.6 points below the full method. Reward variance is a natural baseline because it identifies groups with non-degenerate outcomes, which is a necessary condition for informative RL updates. However, it does not distinguish whether the contrast between success and failure reflects a localized deviation or a deep structural divergence. Two scenarios with identical reward variance can require fundamentally different learning responses, and routing them to the same replay pool conflates these cases. GSP resolves this ambiguity by measuring the dependency-graph overlap between successful and failed rollouts within the same group.

Larger KL movement and sustained entropy characterize the optimization regime. Figure 5(b) compares KL divergence and policy entropy between PGPO and GRPO. PGPO induces slightly larger KL movement throughout training. Because replayed scenarios are re-rolled by the current policy rather than replayed from stored trajectories, the larger KL reflects genuine policy updates on structurally informative scenarios rather than gradient noise from stale samples. At the same time, PGPO maintains higher late-stage entropy than the control. Premature entropy collapse in multi-turn tool use is particularly harmful because it commits the agent to narrow tool-call patterns before it has encountered the full range of user responses and tool observations. The combination of stronger KL movement and higher late-stage entropy is consistent with the ablation results: ERS keeps the policy moving toward better solutions while preserving the behavioral diversity needed

for long-horizon interaction.

7 Conclusion

We introduced Group Structural Proximity (GSP), a provenance-aware metric that captures how closely failed rollouts align with successful ones in the tool dependency graph of the same group. A controlled diagnostic study shows that GSP partitions scenarios along an exploitation–exploration axis orthogonal to outcome statistics. Building on this, PGPO maintains GSP-routed dual replay buffers and uses an Exploration-to-Refinement Schedule to mix them, consistently outperforming GRPO-family baselines on τ^2 -Bench and ACEBench Agent. The results show that structural group comparison provides a complementary training signal for multi-turn tool-use RL.

Limitations

Our study focuses on multi-turn tool-use agents whose interactions are mediated by structured tool calls with traceable argument provenance, and the conclusions we draw should be read within this scope. Domains that operate on richer or less typed action surfaces—for example, code generation and editing, web navigation, or open-ended GUI manipulation—do not naturally expose a tool-dependency graph with stable schemas and explicit information flow, so GSP and PGPO are not directly applicable to such settings without a graph front-end that first constructs dependency graphs from those unstructured traces. Since the core machinery of GSP operates on an already-built dependency graph regardless of how it was extracted, providing such a front-end gives a concrete path to extending our approach to these domains; building and validating it is left for future work. The exploration-to-refinement schedule is currently parameterized rather than learned, and the optimal annealing profile likely depends on dataset difficulty and base-model capability; a fully adaptive controller, together with verification across additional model families and scales, is left for future work. Finally, our evaluation uses two simulated tool-use benchmarks under verifiable rewards, so transferring the observed exploitation-exploration tradeoff to noisier real-world deployments remains an empirical question we plan to address.

Acknowledgements

This work was conducted during Lexiao Zou’s internship at Xingchen AGI Lab, China Telecom Artificial Intelligence Technology (Beijing) Co., Ltd. Miao Zhang was partially sponsored by the National Natural Science Foundation of China under Grants 62306084, 62676118 and U23B2051, the Shenzhen College Stability Support Plan under Grant GXWD20231128102243003, the Shenzhen Science and Technology Program under Grant KJZD20230923115113026, and the Guangdong Basic and Applied Basic Research Foundation under Grant 2026A1515011654.

References

- Andrei Baroian and Rutger Berger. 2026. [Prompt replay: Speeding up GRPO with on-policy reuse of high-signal prompts](#). *Preprint*, arXiv:2603.21177.
- Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. [\$\tau^2\$ -bench: Evaluating conversational agents in a dual-control environment](#). *Preprint*, arXiv:2506.07982.
- Xin-Qiang Cai and Masashi Sugiyama. 2026. [Vicurl: Stabilizing verifier-independent RL reasoning via confidence-guided variance reduction](#). *Preprint*, arXiv:2602.12579.
- Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Yuefeng Huang, Xiangcheng Liu, Wang Xinzhi, and Wu Liu. 2025a. Acebench: A comprehensive evaluation of LLM tool usage. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 12970–12998. Association for Computational Linguistics.
- Gongwei Chen, Lirong Jie, Lexiao Zou, Weili Guan, Miao Zhang, and Liqiang Nie. 2025b. [Enhancing gui agent with uncertainty-aware self-trained evaluator](#). In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 142026–142058. Curran Associates, Inc.
- Gongwei Chen, Xurui Zhou, Rui Shao, Yibo Lyu, Kaiwen Zhou, Shuai Wang, Wentao Li, Yinchuan Li, Zhongang Qi, and Liqiang Nie. 2025c. [Less is more: Empowering gui agent with context-aware simplification](#). In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5901–5911.
- Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. 2025d. [Reinforcement learning for long-horizon interactive llm agents](#). *Preprint*, arXiv:2502.01600.
- Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier,

- Yoshua Bengio, and Ehsan Kamaloo. 2025e. [Self-evolving curriculum for LLM reasoning](#). *Preprint*, arXiv:2505.14970.
- Peng Cui, Boyao Yang, and Jun Zhu. 2026. [Learning-zone energy: Online data selection for efficient RL post-training](#). *Preprint*, arXiv:2605.17003.
- Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025a. [Group-in-group policy optimization for LLM agent training](#). *CoRR*, arXiv:2505.10978.
- Zihao Feng, Xiaoxue Wang, Ziwei Bai, Donghang Su, Bowen Wu, Qun Yu, and Baoxun Wang. 2025b. [Improving generalization in intent detection: GRPO with reward-based curriculum sampling](#). *Preprint*, arXiv:2504.13592.
- Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. 2026. [Hierarchy-of-groups policy optimization for long-horizon agentic tasks](#). *Preprint*, arXiv:2602.22817.
- Junpeng Jiang, Gangyi Hong, Miao Zhang, Hengtong Hu, Kun Zhan, Rui Shao, and Liqiang Nie. 2026. [Dive: Efficient multi-view driving scenes generation based on video diffusion transformer](#). *IEEE Transactions on Multimedia*, pages 1–14.
- Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. 2021. [Prioritized level replay](#). In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 4940–4950. PMLR.
- Shipeng Li, Zhiqin Yang, Shikun Li, Xiaobo Xia, Hengyu Liu, Xinghua Zhang, Gaode Chen, Dong Fang, Ying Tai, and Zhe Peng. 2025. [LearnAlign: Data selection for LLM reinforcement learning with improved gradient alignment](#). *Preprint*, arXiv:2506.11480.
- Xiang Li, Yiqun Yao, Xin Jiang, Xuezhi Fang, Chao Wang, Xinzhang Liu, Zihan Wang, Yu Zhao, Xin Wang, Yuyao Huang, Shuangyong Song, Yongxiang Li, Zheng Zhang, Bo Zhao, Aixin Sun, Yequan Wang, Zhongjiang He, Zhongyuan Wang, Xuelong Li, and Tiejun Huang. 2024. 52b to 1t: Lessons learned via tele-flm series. *CoRR*, abs/2407.02783.
- Xinzhang Liu, Chao Wang, Zhihao Yang, Zhuo Jiang, Xudong Zhao, Haoran Wang, Lei Li, Dongdong He, Luobin Liu, Kaizhe Yuan, Han Gao, Zihan Wang, Yitong Yao, Sishi Xiong, Wenmin Deng, Haowei He, Kaidong Yu, Yu Zhao, Ruiyu Fang, and 35 others. 2025a. Training report of telechat3-moe. *CoRR*, abs/2512.24157.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025b. [Understanding rl-zero-like training: A critical perspective](#). *Preprint*, arXiv:2503.20783.
- Fanbin Lu, Zhisheng Zhong, Shu Liu, Chi-Wing Fu, and Jiaya Jia. 2025a. [Arpo: End-to-end policy optimization for gui agents with experience replay](#). *Preprint*, arXiv:2505.16282.
- Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. 2025b. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *NAACL (Findings)*, Findings of ACL, pages 1160–1183. Association for Computational Linguistics.
- Yibo Lyu, Rui Shao, Gongwei Chen, Yijie Zhu, Weili Guan, and Liqiang Nie. 2025. [Puma: Layer-pruned language model for efficient unified multimodal retrieval with modality-adaptive learning](#). In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, page 7653–7662, New York, NY, USA. Association for Computing Machinery.
- Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Manoj Awalganekar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, Shelby Heinecke, Weiran Yao, Huan Wang, Silvio Savarese, and Caiming Xiong. 2025. Apigentmt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*.
- Zehan Qi, Xiao Liu, Iat Long Long, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. 2024. [Webrl: Training LLM web agents via self-evolving online curriculum reinforcement learning](#). *Preprint*, arXiv:2411.02337.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations (ICLR)*. OpenReview.net.
- Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. [Prioritized experience replay](#). *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. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). *Preprint*, arXiv:2402.03300.
- Ji Shi, Peiming Guo, Meishan Zhang, Miao Zhang, Xuebo Liu, Min Zhang, and Weili Guan. 2026. [Cvedrl: An efficient code verifier via difficulty-aware reinforcement learning](#). *Preprint*, arXiv:2601.22803.

- Mingyang Song, Mao Zheng, Zheng Li, Wenjie Yang, and Xuan Luo. 2025. Fastcurl: Curriculum reinforcement learning with stage-wise context scaling for efficient training rl-like reasoning models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 8856–8866. Association for Computational Linguistics.
- Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. 2026. [Envscaler: Scaling tool-interactive environments for LLM agent via programmatic synthesis](#). *Preprint*, arXiv:2601.05808.
- Hongjin Su, Shizhe Diao, Ximing Lu, Mingjie Liu, Jiacheng Xu, Xin Dong, Yonggan Fu, Peter Belcak, Hanrong Ye, Hongxu Yin, Yi Dong, Evelina Bakhurina, Tao Yu, Yejin Choi, Jan Kautz, and Pavlo Molchanov. 2025. [Toolorchestra: Elevating intelligence via efficient model and tool orchestration](#). *Preprint*, arXiv:2511.21689.
- Dunwei Tu, Hongyan Hao, Hansi Yang, Yihao Chen, Yi-Kai Zhang, Zhikang Xia, Yu Yang, Yueqing Sun, Xingchen Liu, Furao Shen, Qi Gu, Hui Su, and Xunliang Cai. 2026. [Scaleenv: Scaling environment synthesis from scratch for generalist interactive tool-use agent training](#). *Preprint*, arXiv:2602.06820.
- Zhenting Wang, Guofeng Cui, Kun Wan, and Wentian Zhao. 2025a. [DUMP: automated distribution-level curriculum learning for rl-based LLM post-training](#). *Preprint*, arXiv:2504.09710.
- Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. 2025b. [Agent-Dropout: Dynamic agent elimination for token-efficient and high-performance LLM-based multi-agent collaboration](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24013–24035, Vienna, Austria. Association for Computational Linguistics.
- Zihan Wang, Chi Gui, Xing Jin, Qineng Wang, Licheng Liu, Kangrui Wang, Shiqi Chen, Linjie Li, Zhengyuan Yang, Pingyue Zhang, Yiping Lu, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. 2026. [RAGEN-2: reasoning collapse in agentic RL](#). *Preprint*, arXiv:2604.06268.
- Zihan Wang, XinZhang Liu, Shixuan Liu, Yitong Yao, Yunyao Huang, Mengxiang Li, Zhongjiang He, Yongxian Li, Luwen Pu, Huinan Xu, Chao Wang, and Shuangyong Song. 2024. [TeleChat: An open-source bilingual large language model](#). In *Proceedings of the 10th SIGHAN Workshop on Chinese Language Processing (SIGHAN-10)*, pages 10–20, Bangkok, Thailand. Association for Computational Linguistics.
- Zihan Wang, Xinzhang Liu, Yitong Yao, Chao Wang, Yu Zhao, Zhihao Yang, Wenmin Deng, Kaipeng Jia, Jiabin Peng, Yuyao Huang, Sishi Xiong, Zhuo Jiang, Kaidong Yu, Xiaohui Hu, Fubei Yao, Ruiyu Fang, Zhuoru Jiang, Ruiting Song, Qiyi Xie, and 19 others. 2025c. Technical report of telechat2, telechat2.5 and T1. *CoRR*, abs/2507.18013.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Chenghao Yang, Lin Gui, Chenxiao Yang, Victor Veitch, Lizhu Zhang, and Zhuokai Zhao. 2025b. [Let it calm: Exploratory annealed decoding for verifiable reinforcement learning](#). *Preprint*, arXiv:2510.05251.
- Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R. Narasimhan. 2025. τ -bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations (ICLR)*.
- 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 *The Eleventh International Conference on Learning Representations (ICLR)*. OpenReview.net.
- Fan Yin, Zifeng Wang, I-Hung Hsu, Jun Yan, Ke Jiang, Yanfei Chen, Jindong Gu, Long Le, Kai-Wei Chang, Chen-Yu Lee, Hamid Palangi, and Tomas Pfister. 2025. Magnet: Multi-turn tool-use data synthesis and distillation via graph translation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32600–32616. Association for Computational Linguistics.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gao-hong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pages 113222–113244.
- Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F. Wong, and Yu Cheng. 2025. [Exgrpo: Learning to reason from experience](#). *Preprint*, arXiv:2510.02245.
- Hongzhi Zhang, Jia Fu, Jingyuan Zhang, Kai Fu, Qi Wang, Fuzheng Zhang, and Guorui Zhou. 2025a. [Rlep: Reinforcement learning with experience replay for llm reasoning](#). *Preprint*, arXiv:2507.07451.
- Kangning Zhang, Wenxiang Jiao, Kounianhua Du, Yuan Lu, Weiwen Liu, Weinan Zhang, Lei Zhang, and Yong Yu. 2025b. [Looptool: Closing the data-training loop for robust LLM tool calls](#). *Preprint*, arXiv:2511.09148.

- Shijie Zhang, Guohao Sun, Kevin Zhang, Xiang Guo, and Rujun Guo. 2025c. [CLPO: curriculum learning meets policy optimization for LLM reasoning](#). *Preprint*, arXiv:2509.25004.
- Jiaqi Zhao, Chao Zeng, Ming Wang, Linxuan Han, Yuzhang Shang, Miao Zhang, and Liqiang Nie. 2025. [Lrquant+: A unified and learnable framework to post-training quantization for transformer-based large foundation models](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(12):11817–11833.
- Xinyu Zhou, Boyu Zhu, Haotian Zhang, Huiming Wang, and Zhijiang Guo. 2026a. [Efficient RLVR training via weighted mutual information data selection](#). *Preprint*, arXiv:2603.01907.
- Xurui Zhou, Gongwei Chen, Yuquan Xie, Zaijing Li, Kaiwen Zhou, Shuai Wang, Shuo Yang, Zhuotao Tian, and Rui Shao. 2026b. Hiconagent: History context-aware policy optimization for gui agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13028–13038.
- Deyu Zou, Yongqiang Chen, Fan Feng, Mufei Li, Pan Li, Yu Gong, and James Cheng. 2026a. [On information self-locking in reinforcement learning for active reasoning of llm agents](#). *Preprint*, arXiv:2603.12109.
- Deyu Zou, Yongqiang Chen, Jianxiang Wang, Garry Yang, Mufei Li, Qing Da, James Cheng, Pan Li, and Yu Gong. 2026b. Reducing belief deviation in reinforcement learning for active reasoning of LLM agents. In *The Fourteenth International Conference on Learning Representations (ICLR)*.

A POMDP Formalization

We associate each scenario x with a POMDP $\mathcal{M}_x = \langle \mathcal{S}, \mathcal{A}, \mathcal{O}, P, R, \gamma \rangle$. A scenario consists of an initial executable environment state, a user intent, an available tool set with schemas, and domain constraints. At turn t , the latent state can be written as $s_t = (e_t, h_t, u, c)$, where e_t is the environment, h_t is the interaction history, u is the user intent, and c denotes hidden policy or domain constraints. The agent observes h_t but not the full latent state. Its action space is $\mathcal{A} = \mathcal{A}_{\text{resp}} \cup \mathcal{A}_{\text{tool}}$, where $\mathcal{A}_{\text{resp}}$ contains natural-language responses and $\mathcal{A}_{\text{tool}}$ contains executable tool calls with structured arguments. The observation space contains user responses and tool results. A tool action updates the environment state and appends a tool observation to the history; a response action typically leaves the environment unchanged and appends user feedback or terminates the interaction.

B GRPO Objective

For a token or action at step t in rollout τ_i , the importance ratio is $\rho_{i,t}(\theta) = \pi_\theta(a_{i,t} | h_{i,t}) / \pi_{\theta_{\text{old}}}(a_{i,t} | h_{i,t})$, and the group-normalized advantage uses the centered reward r_i relative to the group mean. The clipped objective is

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{x, \{\tau_i\}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{T_i} \sum_{t=1}^{T_i} \ell_{i,t} \right], \quad (2)$$

where

$$\begin{aligned} \ell_{i,t} = & \min \left(\rho_{i,t}(\theta) \hat{A}_i, \right. \\ & \left. \text{clip}(\rho_{i,t}(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_i \right) \\ & - \beta \text{KL}(\pi_\theta(\cdot | h_{i,t}) \| \pi_{\text{ref}}(\cdot | h_{i,t})). \end{aligned} \quad (3)$$

C Buffer Update Rule

At the start of training, both buffers are filled with randomly sampled scenarios. Their GSP scores are initialized as placeholders: 0 in the high-proximity buffer and 1 in the low-proximity buffer, so observed high- and low-GSP scenarios can replace them under the buffer update rule. When a recomputed GSP value would qualify for both buffers (i.e., it exceeds the current minimum of the high-proximity buffer and falls below the current maximum of the low-proximity buffer), it is admitted to whichever buffer it is more extreme relative to (the buffer where the displaced item is closer to the median GSP), avoiding double-counting.

D Implementation Notes

Trace parsing. The current implementation parses a rollout trace as an alternating sequence of user turns, assistant tool calls, and tool responses. User utterances contribute candidate source values for later arguments. Tool responses are canonically normalized and cached as potential provenance anchors for later calls.

Canonicalization. Argument dictionaries and tool observations are normalized before hashing into parent signatures. Canonicalization removes JSON key-order sensitivity and reduces brittle mismatches caused by formatting differences.

One-source policy. When an argument value can be linked to multiple earlier occurrences, the implementation keeps only the earliest source. This choice avoids multi-parent explosion and makes the backbone focus on primary information flow rather than every repeated mention of the same value.

Dynamic-programming backbone extraction.

The longest common dependency backbone is computed with standard dynamic programming for longest common subsequence. For a pair of rollouts of lengths n^+ and n^- , the exact runtime is $O(n^+ n^-)$ after node extraction.

E Diagnostic Subset Composition and Length Confound

The high- and low-GSP subsets used in the diagnostic study (Section 4) are matched on the pass-rate distribution by construction. Other factors are not balanced, and we report them plainly here. Table 3 lists the domain composition of the two subsets: the domain mix differs substantially between them. We argue this imbalance is the expected signature of GSP working as intended rather than a confound: different domains carry different tool sets and interaction settings that inherently shape RL dynamics differently, and GSP is precisely the metric that empirically captures this domain-related effect.

Tool-chain length, a more natural candidate confound, is minor: it explains only $R^2 = 20.7\%/24.9\%$ of GSP’s variance on the two subsets, and 79.1% of the variance falls within, rather than between, length tertiles. In short, GSP is not a proxy for length, and the training-dynamics differences in Section 4 are not explained away by trajectory length.

Table 3: Domain composition (%) of the high-GSP and low-GSP diagnostic subsets.

Domain	High-GSP %	Low-GSP %
bank	15.5	4.8
basketball	8.3	8.3
ecommerce	13.1	2.4
medicine	10.7	22.6
movie	13.1	31.0
railway	19.0	3.6
restaurant	8.3	2.4
school	6.0	9.5
travel	2.4	13.1
weather	3.6	2.4

F Node-Match Sensitivity

To verify that the reported GSP separation is a structural property of the data rather than an artifact of one specific matching rule, we recomputed GSP under five node-match criteria of increasing strictness (`tool_only` through `strict`) on 500 real scenarios. All relaxed and tightened variants retain strong rank agreement with our design ($\rho = 0.83$ – 0.94), and none correlates significantly with pass rate (Table 4).

Table 4: Node-match sensitivity. Spearman ρ is computed over 500 scenarios; the pass-rate column tests outcome-orthogonality of each variant.

Match mode	Mean GSP	ρ vs. full	ρ vs. pass rate
<code>tool_only</code>	0.492	0.828	0.032 ($p = 0.60$)
<code>schema_only</code>	0.462	0.922	0.072 ($p = 0.23$)
<code>provenance_free</code>	0.447	0.938	0.093 ($p = 0.12$)
ours (full)	0.395	—	0.080 ($p = 0.18$)
<code>strict</code>	0.318	0.885	0.073 ($p = 0.22$)

On brittleness specifically: a failed rollout calls `search_packages` with the exact same tool name and argument schema as a successful rollout, but with argument values reflecting a completely different product category (e.g., “small-group/coastal” vs. the user’s requested “honeymoon/romantic”). `tool_only/schema_only` score this pair as highly similar (0.625) because they see only matching tool names and argument keys, while `full` correctly drops it to 0.345 because provenance detects that the arguments no longer trace back to the user’s stated request. Such semantically meaningful divergences—not superficial formatting noise, which canonicalization (Appendix D) already removes—are exactly what the stricter criteria are designed to catch.

G LCS vs. Order-Invariant Similarity

Order-sensitivity is a deliberate design choice, necessary whenever call order carries causal information, which is the common case in τ^2 -Bench. To validate it, we implemented graph edit distance (GED, bipartite approximation on the tool-dependency graph) and a WL-kernel-style similarity (dependency-graph-aware relabeling, $h = 2$), and evaluated both on 450 real τ^2 -Bench success-failure pairs across Airline/Retail/Telecom. Both agree strongly and significantly with our LCS-based proximity (Spearman $\rho = 0.748$, $p = 8.2 \times 10^{-26}$ for GED; $\rho = 0.627$, $p = 2.3 \times 10^{-16}$ for WL; consistently per domain, $\rho(\text{GED}) = 0.94/0.72/0.77$), and PGPO’s dual-buffer routing agrees on 68.3% of scenarios. Their correlation with pass rate is near-zero ($|\rho| < 0.08$, $p > 0.39$), indicating that the outcome-orthogonality our diagnostic study relies on is shared across all three structural metrics.

Where they diverge is informative: GED/WL assign uniformly higher proximity specifically on “retry-storm” failures (e.g., 10+ near-identical tool calls differing only in a shifting argument subset), since their position-agnostic substitution/deletion cost cheaply absorbs repeated near-duplicate calls into low-cost edits—masking a real divergence behind superficial similarity. On the reorderability property, swapping two causally-independent calls (no dependency edge between them) leaves both GED and WL unchanged (1.0), confirming they behave as intended for genuinely independent calls. Table 5 shows a controlled stress test.

Table 5: Controlled stress test: LCS vs. order-invariant (Bag-Jaccard) similarity.

Case	LCS proximity	Bag-Jaccard
A (locally repairable): book vs. cancel tail	0.600	0.600
B (structurally derailed): same tools, reversed	0.143	1.000

H Computational Overhead

PGPO’s only overhead relative to vanilla GRPO is the GSP computation itself. The pairwise LCS cost is exactly $O(n_+n_-)$, and comparing every success-failure pair in a group costs $O(|S||F|\bar{n}_+\bar{n}_-)$, i.e. $O(G^2)$ in the worst case. Measured on real trajectories (mean node count 5.51, $p_{95} = 14$ over 500 rollout scenarios), a single-pair LCS costs 10.7–511 μs for $n = 5$ –40 nodes, confirming $O(n^2)$ scaling in trajectory length (Table 6). For a typical

group of size $G = 8$, this yields $\sim 1\text{--}2$ ms of GSP computation per scenario. With 32 scenarios per training step (our default batch at evaluation time), the total GSP cost is $\sim 32\text{--}64$ ms per step—a pure-CPU cost that fully overlaps with the GPU/network I/O of the next rollout batch—against a ~ 14 -minute (840 s) end-to-end GRPO/DAPO training step, i.e. under 0.01% overhead.

Table 6: CPU micro-benchmark of single-pair LCS computation (1 CPU core), confirming $O(n^2)$ scaling.

Trajectory length n	5	10	20	40
Single-pair LCS	10.7 μ s	36.4 μ s	136.3 μ s	511.0 μ s

I Aggregation over Heterogeneous Groups

When a group contains multiple successful and multiple failed trajectories with very different graph lengths, one may ask whether averaging all pairwise proximities obscures heterogeneous groups. We tested this directly on 500 multi-success/multi-fail groups ($|S|, |F| \geq 2$). Two pieces of evidence indicate that mean remains the more reliable aggregator.

First, mean tracks median—a robust central-tendency statistic—almost perfectly even at the highest length heterogeneity (Table 7): $\rho = 0.934$ at the top tercile of the coefficient of variation of node count, only marginally below the low tercile. If mean were obscuring heterogeneous groups, it would drift away from median precisely where heterogeneity is highest; instead it is min whose agreement with mean drops most ($0.836 \rightarrow 0.625$).

Table 7: Rank agreement (Spearman ρ) between mean and alternative aggregators, by length-heterogeneity tercile (coefficient of variation of node count).

Heterogeneity tercile	$\rho(\text{mean, median})$	$\rho(\text{mean, min})$	$\rho(\text{mean, max})$
Low	0.964	0.836	0.695
Mid	0.951	0.787	0.725
High	0.934	0.625	0.750

Second, max/min are order statistics of the $|S||F|$ pairwise-proximity sample, so their expectation is mechanically pulled toward $1/0$ as $|S||F|$ grows— independent of any change in structural similarity— while mean stays essentially flat ($0.396 \rightarrow 0.370$) over the same range (Table 8). Adopting max/min would therefore make GSP confound structural proximity with group size, the very outcome-statistic contamination GSP is designed to avoid;

mean is the aggregator whose expectation stays invariant to $|S|, |F|$, which we view as a prerequisite for GSP to remain comparable across groups.

Table 8: Expected aggregate value as the pair count $|S| \times |F|$ grows.

$ S \times F $	$E[\text{max}]$	$E[\text{min}]$	$E[\text{mean}]$
[4, 8)	0.588	0.218	0.396
[12, 16)	0.685	0.163	0.381
[16, 24)	0.688	0.159	0.370

J Failure Case Analysis

We present one concrete failure case in which GSP is high while outcomes are mostly failures. In a bank scenario with pass rate 0.125 and full-GSP 0.657, 7 of 8 rollouts retrieve identical, correct evidence via an identical tool sequence, but fail because the simulated user talks the agent out of reporting it—not because retrieval diverges. The divergence here is genuinely local, just one layer downstream of where node-matching looks (response generation rather than tool calls). This is consistent with, not a counterexample to, GSP’s “locally repairable divergence” construct, and simply marks a boundary of its current scope.

K Comparison with Group-Construction Methods

GSP/PGPO is a data-centric contribution—a replay/selection criterion within a group-based RL loop—whereas GiGPO and HGPO modify group construction or advantage estimation itself, requiring exact “anchor” states to recur across rollouts. That mechanism suits closed, near-deterministic environments such as ALFWorld/WebShop. Our τ^2 -Bench/ACEBench setting has continuous, LLM-simulated-user states that essentially never repeat exactly, so their anchor-state mechanism is not directly applicable without a nontrivial state-abstraction step of its own. We therefore compared against DAPO (pass-rate-related filtering) and Ex-GRPO (reward-variance-related buffer replay) to isolate the data-selection axis under a fixed GRPO-family objective.

L Evaluation-Protocol Noise

To bound execution noise of the LLM-simulator-based evaluation protocol, we reran the frozen, untrained base model 4 times on Airline, isolating

sampling/judge variance from training randomness: $\text{pass@1} = 66.0, 64.0, 64.0, 62.0$ (mean 64.0, std 1.4). This noise is far smaller than PGPO’s 6.0-point average pass@1 gain over the same base (Table 1), so the reported improvements are not a protocol-noise artifact.

M Artifact Licenses and Intended Use

All artifacts used in this work—ToolScale, τ^2 -Bench, ACEBench, the Qwen3 model family, and the verl framework—are existing public releases used as-is for non-commercial academic research, which is consistent with their respective licenses and with the intended use stated on each release page (training data for tool-use agents, benchmarks for evaluation, base models for fine-tuning and simulation, and a training framework used as a library); we did not modify, repurpose, or redistribute any of them.

N Hyperparameters

Table 9 lists all hyperparameters used in our GRPO and PGPO training runs. All values are kept constant across the diagnostic study, the main results, and the ablation study, except where the variant under study explicitly modifies a single component (e.g., disabling ERS or replacing GSP with reward variance). We did not perform a systematic hyperparameter search; the values below were chosen as reasonable defaults based on prior GRPO-family work and our preliminary smoke tests, and we explicitly note this so that the reported gains are not confounded with extensive tuning. The full training and evaluation code is available at <https://github.com/Lokshaw-Chau/GSP>.

O Potential Risks

PGPO improves the sample efficiency of multi-turn tool-use RL and therefore inherits the broader risks of strengthening autonomous tool-using agents: by accelerating convergence toward locally refined behavior under the current reward verifier, it can also entrench unsafe tool effects, dataset biases, and over-reliance on automation if the surrounding pipeline is not adequately gated. We recommend that adopters pair PGPO with sandboxed execution, action-level safety verifiers, audits of the high-GSP buffer composition, and explicit privacy controls over the persistent tool-call traces required by GSP, and retain human oversight for consequential actions.

Table 9: Hyperparameter settings used for all GRPO and PGPO runs. Values are shared across the diagnostic study, main results, and ablations unless an ablation explicitly modifies the listed component.

Hyperparameter	Value
<i>Optimization</i>	
Learning rate	1×10^{-6}
Optimizer	AdamW
Train batch size (scenarios/step)	64
Total training data	3,840
KL coefficient β	0.001
<i>Rollout</i>	
Group size G	8
Rollout temperature	1.0
Max sequence length	32,768
Max interaction turns	30
User simulator	Qwen3.5-35B-A3B
<i>Replay buffers</i>	
High-GSP buffer capacity	512
Low-GSP buffer capacity	512
Replay ratio ρ_t	0.5
<i>Exploration-to-Refinement Schedule</i>	
$\alpha_{\min}$	0.20
$\alpha_{\max}$	0.60
Time constant τ	42
<i>Infrastructure</i>	
Hardware	8× NVIDIA H800 80 GB
Training framework	verl
Wall-clock per run	≈ 192 GPU-hours

Privacy and data retention of stored traces.

Unlike standard GRPO pipelines, which consume rollouts transiently within a training step, PGPO persists tool-call traces in the replay buffers over the course of training: GSP’s node construction stores tool names, argument values, and their provenance, and the dual-buffer update keeps these parsed traces retrievable long after the rollout that produced them. In our setting the traces come from simulated scenarios and contain no real user data, but in deployments where rollouts interact with real users or production tools, stored traces can carry personal or otherwise sensitive content (e.g., identifiers, account details, locations, or the content of user utterances used as provenance anchors). We therefore recommend that adopters treat the replay buffers as personal-data stores: minimize the retention window by expiring buffer entries rather than keeping them for the full training run, strip or pseudonymize argument values that do not affect graph structure before trace storage, restrict buffer access to the training pipeline, and provide deletion guarantees so that removing a user’s data also removes their traces from the buffers. When traces are derived from real interactions, retention

and secondary use should additionally be reviewed under the applicable data-protection regulations.

P Use of AI Assistants

The core technical contributions—the formulation of GSP, the design of PGPO and its replay mechanism, the diagnostic protocol, the experiments, and all analyses—were conceived, implemented, and verified by the authors. AI assistants were used only in supporting roles: a coding assistant provided autocomplete and refactoring suggestions for boilerplate and utility code (every fragment that entered the experimental pipeline was read, edited, and tested by the authors), and a writing assistant was used as a paragraph-level polishing pass over author-written drafts to tighten phrasing and grammar (no claims, results, or citations were generated by the assistant, and every revision was reviewed and accepted or rejected by the authors). Any residual errors are the authors' own.