GRAR: GRAPH-RUBRIC-GUIDED AGENTIC REWARD FOR REINFORCEMENT LEARNING OF LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning for large language model (LLM) agents is naturally supervised by the environment's terminal signal: whether the task was completed. Under group-relative policy optimization (GRPO), terminal supervision is uninformative in two distinct regimes. In an all-failed group, every sampled trajectory receives the same reward, so the group produces no policy-gradient signal. In a mixed group containing both successful and failed trajectories, the terminal reward distinguishes their outcomes but does not identify which decisions contributed to success or failure. We introduce GRAR (Graph-Rubric-guided Agentic Reward), a post-hoc credit-assignment framework that preserves the terminal reward while providing process-level feedback in both regimes. For groups containing at least one success, *graph settlement* assigns step credit from verified outcomes represented in a group-local semantic state graph. For all-failed groups, *rubric settlement* scores a small set of graph-selected steps with a single guarded LLM-judge call. Trajectory advantages remain determined solely by terminal rewards, while process credit enters as a bounded step-aligned bonus; under the deployed configuration, this preserves the ordering between successful and failed trajectories. A single environment-agnostic adapter spans τ^2-bench, WebShop, and ALFWorld. Across the evaluated benchmark settings, the best GRAR variant improves on the terminal-only baseline on both metrics.

1 INTRODUCTION

LLM agents operate in multi-turn interactive environments, where the most defensible training signal is the environment's own completion verdict. Terminal-only reinforcement learning with verifiable rewards has this property and is a standard paradigm for agentic post-training, instantiated as GRPO-style group-relative optimization (Shao et al., 2024): trajectories are sampled in groups for each prompt, scored by task completion, and updated through within-group advantage normalization.

The resulting limitation concerns *coverage*, rather than correctness. When every sampled trajectory fails, all rewards coincide, the group-relative advantages are identically zero, and the group contributes no gradient. In our runs this silent regime accounts for roughly half of the training groups (Section 2.1), although the otherwise unused rollouts often contain evidence about the decisions that led to failure. When some trajectories instead succeed, the terminal signal establishes *that* the goal was reached but not *which* steps were decisive: it cannot prefer a concise success over one that reaches the same outcome through redundant steps, nor identify the branch at which a failed sibling diverged (Figure 1).

Dense process rewards—including step-level reward models (Lightman et al., 2024; Wang et al., 2024), LLM judges (Zheng et al., 2023), and hand-designed shaping—address this gap, but introduce two risks in agentic settings. First, *reward hacking* can arise when a learned or judged process score is manipulated, allowing scoring errors to enter the policy gradient directly. Second, *terminal dilution* can occur when process bonuses added to the normalized scalar reward cause a failed trajectory to outrank a successful one, weakening the ordering property of the terminal signal. Among remedies for GRPO's all-failed degeneracy, including dynamic resampling (Yu et al., 2025) and debiased normalization (Liu et al., 2025), GRAR retains every group while extracting step-level information from failing groups. Among step-level signals, GiGPO (Feng et al., 2025) groups steps by exact anchor states, whereas GRAR groups them by normalized action signatures and applies two settlement procedures with a guarded judge (Appendix F).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

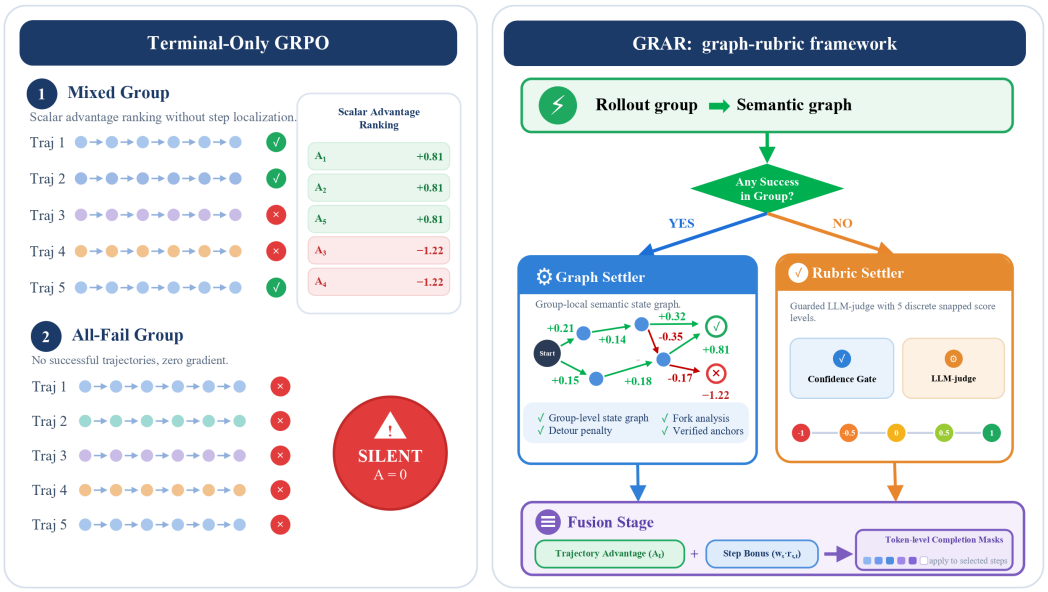

Figure 1: Overview of GRAR. In the left panel, terminal-only GRPO assigns scalar advantages from terminal outcomes alone. These advantages distinguish trajectories in a mixed group but do not localize decisive steps, and they vanish in an all-failed group. In the right panel, GRAR assigns each group to graph settlement or guarded rubric settlement and combines the resulting step bonus with the original trajectory advantage.

GRAR separates terminal supervision from process credit. The terminal reward remains the sole source of trajectory advantage, whereas process credit is assigned only after the group rollouts are complete. The trainer replays the recorded trajectories into a group-local semantic state graph and applies a single ownership rule: the group's verified outcomes determine which settlement procedure is applied. Groups containing at least one success use *graph settlement*, which derives step credit from those outcomes. All-failed groups use *rubric settlement*, in which a small set of graph-selected decision steps is scored in a single batched judge call with deterministic overrides and a confidence gate. The judge is therefore invoked only for all-failed groups, where no successful trajectory provides positive outcome evidence for graph settlement. Process credit enters the update as a small step-aligned bonus whose default weight satisfies the ranking-preservation bound, so a failed trajectory can receive step-level feedback without outranking a successful one. The graph is rebuilt for each group and discarded after the update, and its node potentials are calibrated within the group using the environment's verified progress signal when available. A minimal environment-agnostic rollout adapter maps actions and observations from all three environments to a common record, allowing the same trainer to process them.

Our contributions are as follows:

- **A unified credit-assignment framework**: GRAR addresses the two limitations of terminal supervision by recovering process-level signal in all-failed groups and localizing decisive steps in mixed groups. The resulting process credit is introduced only as a bounded, step-aligned bonus, preserving the ordering between successful and failed trajectories under the deployed configuration.
- **Graph settlement for mixed groups**: a group-local semantic state graph, rebuilt from the group's own rollouts, calibrates node potentials from the verified outcomes in the current group, so that step credit concentrates on decisions that genuinely advanced the task while penalizing redundant detours.
- **A guarded rubric settlement procedure for all-failed groups**: where terminal outcomes provide no positive graph evidence, a small set of graph-selected decision steps is scored in a single batched judge call under deterministic overrides and a confidence gate, recovering signal from the silent regime without letting the judge overturn a terminal verdict.

- **A reproducible evaluation protocol** spanning τ^2-bench, WebShop, and ALFWorld under a shared trainer and adapter: the best GRAR variant improves on the terminal-only baseline on the aggregate results for both metrics and more than doubles exact success where the baseline is weakest.

2 BACKGROUND AND MOTIVATION

2.1 PRELIMINARIES: THE TERMINAL-ONLY BASELINE AND ITS SILENT REGIME

An agent policy π_θ interacts with an environment over T_i turns per trajectory τ_i; at turn t it emits an action $a_{i,t}$, such as a tool call, click, admissible command, or user-facing message, and receives an observation. The episode terminates with the environment's terminal reward $r_i^{\text{env}} \in \{0, 1\}$, indicating whether the task was completed. For each prompt q, GRPO samples a group of G trajectories $\mathcal{G}(q) = \{\tau_1, \ldots, \tau_G\}$ and computes the trajectory advantage

$$A_i \;=\; \frac{r_i^{\text{env}} - \text{mean}_{j\in\mathcal{G}}\, r_j^{\text{env}}}{\text{std}_{j\in\mathcal{G}}\, r_j^{\text{env}} + \epsilon}. \tag{2.1}$$

When all trajectories fail ($\forall i : r_i^{\text{env}} = 0$), Eq. 2.1 yields $A_i = 0$ for every i, so the group contributes no gradient. The all-success case ($\forall i : r_i^{\text{env}} = 1$) is likewise silent, although it occurs rarely in our runs. In our training runs (Section 4), the silent regime is common rather than exceptional: the fraction of groups with zero reward standard deviation averages between 0.44 and 0.51 across the five trained WebShop runs in Table 1. The regime becomes less frequent as training raises the success rate, but it does not disappear (Figure 6 in Appendix E). In mixed groups, the terminal signal likewise cannot distinguish a direct successful course from one that reaches the same outcome through redundant steps.

2.2 OVERVIEW AND THE ROLLOUT INTERFACE

The group-local design was motivated by a failure observed in an earlier GRAR prototype that computed graph rewards *per trajectory*. In one representative group, four trajectories succeeded and two failed, with the outcomes differing at a single decision point: the choice of payment card. Nevertheless, the logged branch-comparison counter was zero for every rollout, and the information associated with this decision was not used. GRAR therefore rebuilds the graph *group-locally* from all G trajectories and preserves task-determining arguments such as the payment card in its signatures; Appendix D provides the corresponding analysis.

GRAR is implemented as a settlement pass inside the GRPO trainer and is invoked after all G terminal rewards for a group have been received (Figure 2). Environments are integrated through a minimal adapter interface. At each step, the adapter records a *signature*: a normalized action identity that combines the action type, the tool or command, and the environment's outcome class, including terminal signatures such as `terminal:success` and `terminal:loop_failure`. It also records a compact *state payload*, consisting of the observation with entity identifiers normalized away, and *hindsight*, consisting of the terminal reward and episode flags. Two normalization rules apply uniformly: consecutive repeats of a base signature are marked `:repeat`, and a run exceeding a domain-configured bound terminates as `terminal:loop_failure`. When the environment exposes a deterministic progress scalar $s_t \in [0, 1]$, the adapter records both s_t and its per-step change Δs_t as verified progress signals; per-domain details appear in Section 4 and Appendix G.

The settlement stage is a deterministic function of the recorded group. For each group of G trajectories, it rebuilds a temporary semantic state graph from the recorded paths and selects a settlement procedure according to the following ownership rule:

$$\text{settlement}(\mathcal{G}) \;=\; \begin{cases} \text{graph settlement}, & \exists\, i :\; r_i^{\text{env}} = 1, \\ \text{rubric settlement}, & \forall\, i :\; r_i^{\text{env}} = 0, \end{cases} \tag{2.2}$$

the two single-branch variants of Section 4 each disable one of these branches. Both settlement branches write per-step arrays $(r_{i,t}, m_{i,t})$ of rewards and activity masks and leave r_i^{env} untouched; Algorithm 1 in Appendix B summarizes the pass.

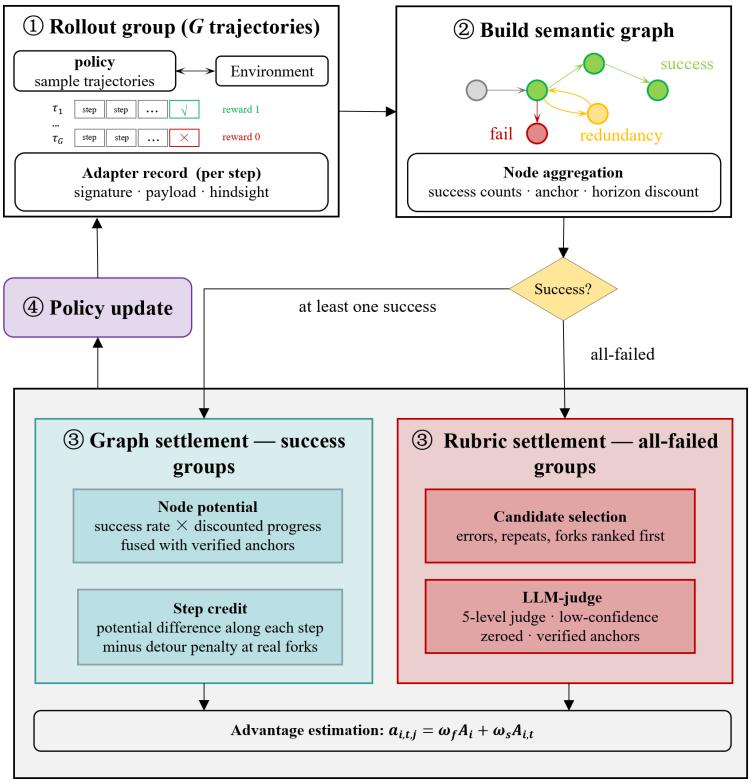

Figure 2: Overview of the GRAR workflow. After the G rollouts for a prompt group are completed, their recorded actions, observations, and terminal outcomes are replayed to construct a group-local semantic state graph. A success gate then selects graph settlement for groups containing at least one successful trajectory and guarded rubric settlement for all-failed groups. The resulting step-aligned process bonus is combined with the terminal-reward advantage, which remains unchanged, to form the token-level advantages used for the policy update.

3 METHOD

3.1 GROUP-LOCAL SEMANTIC STATE GRAPH

Nodes are states induced by normalized signatures, and directed edges represent transitions. The graph is rebuilt from scratch for each group in a temporary cache, preventing historical states from influencing credit assignment; all settlement statistics are computed exclusively from the current group's trajectories. Construction records per-node outcome statistics, $\texttt{success_count}(v)$ and $\texttt{terminal_count}(v)$, incremented for each recorded terminal outcome passing through v; per-node anchor values, defined as the mean deterministic progress of the payloads mapped to v; and per-edge remaining-success lengths among successful trajectories.

3.2 GRAPH SETTLEMENT

Horizon-adaptive discount. Let H denote the number of steps in the longest successful trajectory, equivalently one less than its number of unique nodes, and let $\gamma = H/(H+1)$. The discount is thus determined by the group's successful trajectories rather than by a separately tuned constant.

Node potential. For a node v with $q(v) = \texttt{success_count}(v)/\texttt{terminal_count}(v)$ and mean remaining steps $\bar{d}(v)$, the graph component of the potential is

$$\Phi_{\text{graph}}(v) \;=\; q(v) \cdot \frac{\gamma^{\bar{d}(v)} - \gamma^{H}}{1 - \gamma^{H}}, \tag{3.1}$$

i.e., the empirical success rate multiplied by a normalized discounted progress term, which equals one at a verified success and decays with the remaining horizon. The anchor component averages the environment's own verified progress over the anchor-bearing visits to v:

$$\Phi_{\text{anchor}}(v) \;=\; \frac{1}{|T_v|} \sum_{(i,t)\in T_v} \big[\, s_{i,t} \,\big]_0^1, \tag{3.2}$$

where T_v collects steps whose payload maps to v and whose anchor is marked *available* by the adapter, $s_{i,t} \geq 0$ is the environment's progress score at step t of trajectory i, and $[\cdot]_0^1$ denotes clamping to $[0, 1]$. If no anchor-bearing visit reaches v, we set $\Phi_{\text{anchor}}(v) = 0$, so the potential reduces to its graph component; anchor availability is environment-defined (Section 4, Appendix G). The two components are then fused as independent evidence about the value of the same state:

$$\Phi(v) \;=\; 1 - \big(1 - \Phi_{\text{graph}}(v)\big)\big(1 - \Phi_{\text{anchor}}(v)\big). \tag{3.3}$$

Unvisited nodes default to $\Phi = 0$; a node visited only by failed trajectories receives no outcome-statistic credit ($q(v) = 0$), and any positive Φ it carries comes only from verified anchor progress.

Detour penalty at genuine decision points. A detour is identifiable only when at least two *successful* continuations leave the same source state; otherwise, a shorter alternative may simply be unobserved. Let $B(v)$ denote the successful outgoing edges of v. For an edge (v, u), let $\bar{\ell}(v, u)$ denote the mean number of unique nodes remaining after that edge among successful trajectories that take it. The detour penalty is

$$\text{detour}(v, u) \;=\; \begin{cases} \dfrac{\max\big(0,\; \bar{\ell}(v,u) - \min_{(v,u')\in B(v)} \bar{\ell}(v,u')\big)}{H+1}, & |B(v)| \geq 2, \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

The penalty captures only *relative* verbosity among verified successful alternatives; interchangeable action orders are unaffected, and no additional scale parameter is introduced.

Per-step reward. For step t with source v_{t-1} and target v_t,

$$r_{i,t} \;=\; \begin{cases} \Delta\Phi(v_{t-1} \to v_t) - \text{detour}(v_{t-1}, v_t), & \text{non-terminal step,} \\ 0, & \text{terminal step,} \end{cases} \tag{3.5}$$

with two deterministic clamps: steps whose signature indicates a tool error (`:error`) or a repeat (`:repeat`) are clamped to $r_{i,t} \leq 0$. The first clamp treats a failed tool call as verified negative evidence even if a subsequent recovery occurs; the second assigns no positive credit to repeated states. Steps whose signature the adapter marks terminal (`terminal:*`) receive zero process reward by construction: the terminal outcome is represented solely by r_i^{env} and is neither amplified nor discounted by the graph. If the environment's winning action is an ordinary step rather than a terminal-marked step, this rule does not apply, and the final edge is treated in the same manner as other edges; Appendix C.1 provides an example. Since $|\Delta\Phi| \leq 1$ and the detour penalty is below one, every graph step satisfies $|r_{i,t}| < 2$; the ranking guarantee in Section 3.4 is stated with respect to this bound.

3.3 Rubric settlement

When every trajectory fails, the episode-level reward is uniformly zero and the group graph contains no verified success evidence, so positive credit derived solely from outcome statistics would be unsupported. Failure, however, is binary only at the episode level: the rubric decomposes the objective into separate conditions, and individual steps can satisfy some of them while the episode fails others. A search that correctly narrows the catalog, for example, is still informative even when the final purchase misses an attribute. GRAR exploits precisely this partial satisfaction: a frozen LLM judge scores each selected step against the rubric criteria, so all-failed groups retain a discriminative step-level signal that the terminal reward alone cannot provide.

Graph-based candidate selection. Candidate selection is likewise performed on the group-local graph. Each non-terminal step receives a priority weight: error steps rank first, followed by repeats,

branch-point steps (that is, steps whose source nodes have at least two distinct continuations), and all remaining steps. Any strictly decreasing set of weights induces this ordering, and the top L steps from each trajectory are selected as candidates. For each candidate, the graph provides structural context, including node-visit and branch counts, together with the truncated action, observation summary, four preceding signatures, and deterministic progress values s_t and Δs_t.

Deterministic overrides and the judge. Repeat and error candidates are assigned scores deterministically without querying the judge: errors receive the lowest score and repeats receive the intermediate negative score, both with maximal confidence (numeric values in Appendix G). The remaining candidates in the group are evaluated in a *single batched call* to a frozen instruction-tuned judge conditioned on the task objective and the domain-specific rubric. The judge returns per-candidate JSON containing `score`, `confidence`, `reason`, and validity flags, with scores restricted to the five-level set $\mathcal{S} = \{1, 0.5, 0, -0.5, -1\}$ (per-level semantics in Appendix G). We then (i) snap each score to the nearest element of $\mathcal{S}$ and (ii) set it to zero when the reported confidence falls below the gate τ_{conf}. The gate uses the judge's *self-reported* confidence (Appendix I). Snapping limits the effect of an individual judgment to half a point and makes judge behavior auditable because the trace records both raw and snapped values.

Anchor lower bound. Along each trajectory, the adapter derives the per-step anchor delta $\Delta s_t = s_t - s_{t-1}$. If a candidate has a verified positive delta ($\Delta s_t > 0$) and is neither an error nor a repeat, its reward is set to $\max(r_{\text{judge}}, \Delta s_t)$. Thus, verified progress provides a lower bound on step quality independently of the judge.

Credit bounds. Every trajectory in an all-failed group keeps its terminal reward 0; the rubric contributes only step-aligned terms with $|r_{i,t}| \leq 1$ on at most L steps per trajectory, and the per-group trace of candidates, judge output, masks, and rewards is written to disk before training consumes it.

3.4 Step-aligned advantage fusion

The trajectory-level GRPO advantage is computed *only* from terminal rewards, as in Eq. 2.1; settlement never modifies the scalar reward. Process rewards enter at the token level: if step t of trajectory i generated the token segment of length $L_{i,t}$, every token in that segment receives

$$a_{i,t,j} \;=\; w_f A_i \;+\; w_s\, r_{i,t}, \qquad j = 1, \ldots, L_{i,t}, \tag{3.6}$$

with $w_f = 1$ and a small step weight w_s in all experiments (numeric values in Appendix G; sensitivity in Section 5.4); tokens outside any active step receive $w_f A_i$ only, and the step-to-token alignment advances a cursor over the completion mask turn by turn, preventing rewards from shifting across turns. The guarantee has two parts. Trajectory advantages are computed from the terminal reward alone, by construction; and since process credit enters only as the bounded bonus $w_s r_{i,t}$ with $|r_{i,t}| < 2$ (Section 3.2), cross-trajectory token ranking is preserved whenever $4\, w_s(\sigma + \epsilon) < 1$, where σ is the group's terminal-reward standard deviation; the deployed configuration $w_s{=}0.2$ satisfies this bound at the worst case $\sigma{=}\frac{1}{2}$ (proof in Appendix A). In all-failed groups the trajectory advantage is zero and the update is entirely process-derived, recovering signal from the silent regime described in Section 2.1. Section 4 instantiates the framework as three training variants.

4 Experimental Setup

Environments. (i) τ^2-*bench* (Barres et al., 2025): tool-calling customer-service dialogues in the airline, retail, and telecom task domains with a simulated user. The terminal reward is the official binary database check. (ii) *WebShop* (Yao et al., 2022): goal-directed shopping with search and click actions over a 52-task validation split and a 100-product catalog. The terminal reward is exact task success (binary), and the environment's per-step task score is used as a deterministic anchor. (iii) *ALFWorld* (Shridhar et al., 2021): text-based household tasks issued as natural-language commands, with episodes capped at 50 environment steps. The terminal reward is binary task success, with optional subgoal progress used as an anchor. All three environments are integrated through the same three-field adapter. Per-domain signature formats, loop guards, and adapter defaults are summarized in Appendix G (Table 7).

Models and training. We train Qwen3.5-9B (Qwen Team, 2026a) with ms-swift and generate rollouts using vLLM (Kwon et al., 2023) in server mode. We use G=8 in all domains and keep the optimizer settings fixed across experiments (full hyperparameters are provided in Appendix G). GRAR uses w_s=0.2, w_f=1.0, L=6 candidates per trajectory, a confidence gate of τ_{conf}=0.4, and a frozen judge (Qwen3.6-35B-A3B (Qwen Team, 2026b)) served separately from both the policy and the rollout engine. Process rewards are introduced only through the settlement stage; no online shaping or dynamic sampling is used.

Variants. We compare three training variants against the terminal-only baseline:

- **GRAR w/o rubric**: graph settlement only for groups with successes; all-failed groups receive no process credit.
- **GRAR w/o graph**: rubric settlement only; groups with successes receive terminal-only training.
- **GRAR (combined)**: the full method, with the ownership split of Eq. 2.2, rubric settlement owning all-failed groups and graph settlement the rest.

GiGPO baseline. We evaluate GiGPO (Feng et al., 2025) within the same trainer using the same data, rollouts, optimizer, and evaluation protocol as the other methods. Step groups pool turns across the G trajectories that start from an identical observation. Within each anchor group, step advantages are computed as return-to-go differences with γ=0.95, mean-normalized, and set to zero for singleton groups, following the authors' released implementation. Token advantages are then given by $A_i + A^{\text{step}}_{i,t}$.

Evaluation. We evaluate all environments using their official benchmark success checks and draw five samples per task. **pass@1** denotes the per-sample success rate, whereas **pass@5** denotes the fraction of tasks solved by at least one of the five samples; all table entries are percentages. Each training configuration uses a single seed.

5 RESULTS AND ANALYSIS

5.1 WEBSHOP: TRAINING DYNAMICS, PRECISION, AND COVERAGE

Table 1 reports pass@1 and pass@5 for the three GRAR variants and the terminal-only baseline under identical training and evaluation conditions. The WebShop results are consistent with the complementary roles of the two settlement branches. GRAR w/o rubric, which shapes only groups containing successes, increases the best pass@1 from 15.4 to 32.7; GRAR w/o graph, which settles only all-failed groups, increases the best pass@5 by 15.4 percentage points to 71.2; and the combined variant integrates both branches and reaches the best pass@1. Figure 3 shows the training dynamics, with the variants separating from the baseline after the warmup period. GiGPO improves pass@1 over terminal-only GRPO, but its pass@5 remains slightly below the baseline; every GRAR variant exceeds it on both WebShop metrics by a clear margin. Averaged over the five benchmark columns of Table 1, the combined variant obtains the highest values on both metrics (57.7 pass@1, 79.5 pass@5).

The GiGPO results also illustrate a limitation of exact-observation step grouping: step credit is assigned only when trajectories share an identical observation, leaving singleton or rarely recurring states without signal. The WebShop results reflect this: GiGPO raises pass@1 from 15.4 to 23.1, yet does not improve pass@5 over terminal-only GRPO (53.8 versus 55.8). In contrast, GRAR uses normalized action signatures and explicitly addresses all-failed groups, providing broader coverage of recurring decisions and additional process signal when terminal outcomes are uniformly negative.

5.2 τ^2-BENCH: CONSISTENCY UNDER STRICT EVALUATION SETTINGS

The same pattern is observed across τ^2-bench. On airline, terminal-only GRPO improves on neither metric relative to the untrained base, whereas the combined variant raises pass@1 from 69.0 to 77.5 and pass@5 from 80.0 to 83.0. On retail, the lowest-performing τ^2-bench domain, the combined variant achieves the highest pass@1 at 57.6, while GRAR w/o rubric achieves the highest pass@5 at 73.5. Averaged over the three domains, the combined variant's margins over the baseline are 4.9 percentage points on pass@1 and 1.8 on pass@5. The combined variant also attains the best

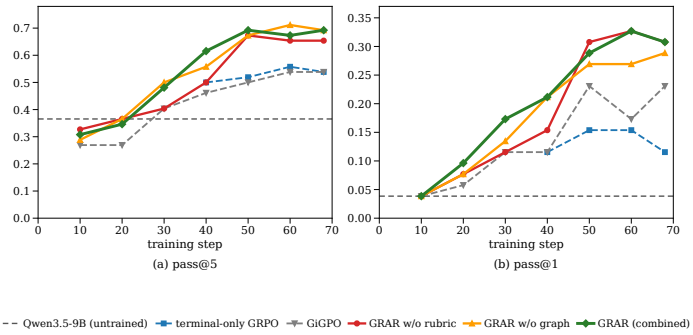

Figure 3: WebShop training curves comparing the terminal-only baseline, GiGPO, and GRAR variants, evaluated every 10 steps: (a) pass@5 and (b) pass@1. The dashed line denotes the untrained Qwen3.5-9B model.

Table 1: Performance of GRAR and baseline methods across the evaluated environments. Arrows indicate percentage-point changes relative to the untrained Qwen3.5-9B model; bold marks the best result in each column.

Method (pass@1)	τ^2-bench			WebShop	ALFWorld	Avg.
	airline	retail	telecom			
Qwen3.5-9B (untrained)	69.0	44.0	65.3	3.8	30.3	42.5
terminal-only GRPO	69.2↑0.2	51.1↑7.1	71.2↑5.9	15.4↑11.6	38.1↑7.8	49.0↑6.5
GiGPO	72.2↑3.2	50.4↑6.4	65.0↓0.3	23.1↑19.3	42.2↑11.9	50.6↑8.1
GRAR w/o rubric	68.2↓0.8	55.7↑11.7	**72.7**↑7.4	**32.7**↑28.9	41.5↑11.2	54.2↑11.7
GRAR w/o graph	73.4↑4.4	51.9↑7.9	69.2↑3.9	28.8↑25.0	**49.6**↑19.3	54.6↑12.1
GRAR (combined)	**77.5**↑8.5	**57.6**↑13.6	71.0↑5.7	**32.7**↑28.9	49.5↑19.2	**57.7**↑15.2

Method (pass@5)	τ^2-bench			WebShop	ALFWorld	Avg.
	airline	retail	telecom			
Qwen3.5-9B (untrained)	80.0	64.0	84.0	36.5	68.8	66.7
terminal-only GRPO	80.0↑0.0	72.0↑8.0	86.0↑2.0	55.8↑19.3	72.0↑3.2	73.2↑6.5
GiGPO	81.0↑1.0	71.0↑7.0	84.0↑0.0	53.8↑17.3	75.3↑6.5	73.0↑6.3
GRAR w/o rubric	79.0↓1.0	**73.5**↑9.5	**87.5**↑3.5	67.3↑30.8	74.2↑5.4	76.3↑9.6
GRAR w/o graph	**83.0**↑3.0	71.0↑7.0	85.5↑1.5	**71.2**↑34.7	**80.4**↑11.6	78.2↑11.6
GRAR (combined)	**83.0**↑3.0	73.0↑9.0	**87.5**↑3.5	69.2↑32.7	**85.0**↑16.2	**79.5**↑12.8

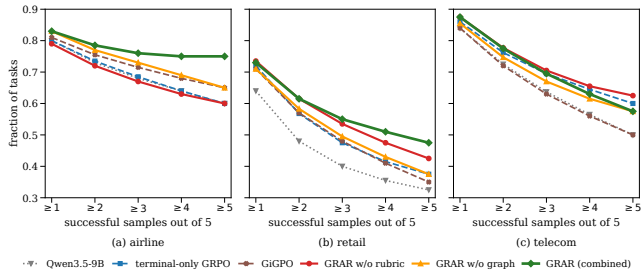

Figure 4: τ^2-bench consistency profiles: the fraction of tasks solved by at least k of the five evaluation samples, from pass@5 coverage (k=1) to all-five success (k=5); (a) airline, (b) retail, and (c) telecom. Curve labels match the methods in Table 1.

three-domain averages, 68.7 pass@1 (against 65.5 for GRAR w/o rubric and 64.8 for GRAR w/o graph) and the highest pass@5 at 81.2. The consistency profiles in Figure 4 show a related trend: on airline and retail, the combined variant's margin over the baseline increases monotonically with the consistency requirement k. At k=1, which is equivalent to pass@5, its average margin over the three domains is 7.5 percentage points; while the k=5 endpoint measures the fraction of tasks solved by all five samples.

Telecom, the highest-baseline domain, is the only τ^2-bench setting in which the baseline retains a pass@1 advantage over the combined variant; GRAR w/o rubric attains the domain's best values,

Table 2: Sensitivity of GRAR w/o rubric on WebShop to the step weight w_s (Eq. 3.6) and to the deterministic anchor fusion (Eq. 3.3). Full 68-step runs under identical prompt pool, optimizer, and evaluation; each cell is the per-metric best over checkpoints 10–68 ($\times 100$, single seed). The default configuration (w_s=0.2, anchors on) is the main-table GRAR w/o rubric row.

Configuration	pass@1	pass@5
$w_s = 0.05$	19.2	55.8
$w_s = 0.1$	23.1	65.4
$w_s = 0.2$ (default, anchors on)	**32.7**	**67.3**
$w_s = 0.4$	30.8	**67.3**
$w_s = 0.6$	25.0	53.8
$w_s = 0.2$, anchors off	23.1	59.6

72.7 pass@1 and 87.5 pass@5. GiGPO improves airline slightly, remains roughly at the baseline's level on retail, and is comparable to the untrained base on telecom; no GiGPO cell reaches the best GRAR value in its column.

5.3 ALFWORLD: LONG-HORIZON TRANSFER

The ALFWorld results are computed on the 93 valid-seen tasks from the four retained task families. Every trained arm improves on the untrained base's 30.3 pass@1 and 68.8 pass@5, with a different ordering from WebShop: GRAR w/o graph attains the best pass@1 at 49.6, while the combined variant obtains the best coverage at 85.0 pass@5. The per-task-type breakdown (Table 9 in Appendix H) further characterizes these gains: GRAR w/o graph leads three of the four types at pass@1, and at pass@5 it leads on clean while tying for the best on pick and look; the combined variant attains the best two-object placement score at 96.2 and the best overall coverage; and pick-and-place reaches 100 pass@5 for every trained arm. On tasks whose successful rollouts span up to fifty admissible commands, the largest gains are associated with the guarded rubric channel. GRAR w/o rubric, whose graph settlement is grounded in verified subgoal stages rather than a judge, still improves pass@1 by 11.2 percentage points over the untrained base; the rubric branch accounts for the additional coverage achieved by the best-performing variants.

5.4 SENSITIVITY TO THE STEP WEIGHT AND THE ANCHORS

The fused advantage of Eq. 3.6 has one primary hyperparameter, the step weight w_s, and one binary choice, whether the deterministic anchor enters the node potential; Table 2 ablates both on GRAR w/o rubric under the identical WebShop protocol.

Three observations follow. (i) The response to w_s is approximately inverted-U shaped: an under-weighted signal at 0.05 loses 13.5 pass@1 percentage points, an over-weighted signal at 0.6 allows process credit to compete with the terminal advantage and degrades both metrics, and the default range 0.2–0.4 forms a plateau. (ii) Disabling the anchors, which removes the Φ_{anchor} term of Eq. 3.2 and reduces Eq. 3.3 to the graph component alone, costs 9.6 pass@1 and 7.7 pass@5 percentage points. Verified environment progress is therefore the largest isolated channel in this ablation, although outcome statistics alone remain above the baseline; the channels are not additive (the anchor component is fused, not added). (iii) We report 0.2–0.4 as a plateau rather than as a strict optimum.

Across the completed runs, the settlement traces themselves support an audit of the mechanism along six trace-only criteria that bound the judge's influence (Appendix I).

6 CONCLUSION

We presented GRAR, a post-hoc credit-assignment framework that keeps the terminal reward as the sole trajectory-level signal while recovering the groups terminal-only GRPO discards: graph settlement converts verified outcomes into step credit, and guarded rubric settlement scores all-failed groups under deterministic overrides, a snapped scale, and a confidence gate. Future work includes extending graph credit beyond a single group's rollouts, validating judge accuracy rather than only bounding judge influence, and testing generality across model scales and further environments.

AI USE STATEMENT

In this work, we used generative AI tools for three tasks with required disclosure: implementing methods (the training and evaluation code), providing feedback on research methodology and experiments, and interpreting results. We have not used generative AI tools for the remaining required-disclosure tasks, such as generating synthetic datasets, developing the theoretical framework or its mathematical claims and proofs, proposing or refining hypotheses, translating or reformatting data, and qualitative data analysis. We additionally used them for tasks with recommended disclosure: creating and modifying scientific figures, writing and editing code, editing the manuscript for readability and structure, and searching for information; we did not use them to search for or summarize the literature. All AI-assisted code and every number reported in this paper were verified by the authors against the raw training records, and every AI-suggested edit to the text was reviewed by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The complete experimental protocol is specified in Section 4: environments, models, optimizer settings, and evaluation metrics, with all hyperparameters and per-domain adapter details in Appendix G. Appendix B provides pseudocode for the settlement pass, Appendix C excerpts a full settlement trace and judge transcript together with the verbatim judge prompt and a per-domain rubric file, and Appendix I lists the trace-audit criteria applied to every run; these materials are intended to support independent reimplementation of the complete pipeline.

REFERENCES

V. Barres, D. Shin, H. Xu, Z. Chen, K. Gopalakrishnan, et al. τ^2-bench: Evaluating conversational agents in a dual-control environment. arXiv:2506.07982, 2025.

H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and G. Cobbe. Let's verify step by step. In *ICLR*, 2024.

A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, 1999.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv:2402.03300, 2024.

M. Shridhar, X. Yuan, M.-A. Côté, Y. Bisk, and A. Trivedi. ALFWorld: Aligning text and embodied environments for interactive learning. In *ICLR*, 2021.

P. Wang, L. Li, Z. Shao, R. Xu, D. Dai, Y. Li, et al. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *ACL*, 2024.

S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. WebShop: Towards scalable real-world web interaction with grounded language agents. In *NeurIPS*, 2022.

L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Li, et al. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *NeurIPS Datasets and Benchmarks*, 2023.

L. Feng, Z. Xue, T. Liu, and B. An. Group-in-group policy optimization for LLM agent training. In *NeurIPS*, 2025.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. arXiv:1707.06347, 2017.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

A. Ahmadian, C. Cremer, M. Gallé, M. Fadaee, J. Kreutzer, O. Pietquin, A. Üstün, and S. Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback with comparison to PPO. In *ACL*, 2024.

D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv:2501.12948, 2025.

Q. Yu, Z. Zhang, R. Zhu, Y. Yuan, X. Zuo, Y. Yue, T. Fan, G. Liu, L. Liu, X. Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. arXiv:2503.14476, 2025.

Z. Liu, C. Chen, W. Li, P. Qi, T. Pang, C. Du, W. Lee, and M. Lin. Understanding R1-Zero-like training: A critical perspective. arXiv:2503.20783, 2025.

G. Sheng, C. Zhang, Z. Ye, X. Wu, W. Zhang, R. Zhang, Y. Peng, H. Lin, C. Wu, et al. HybridFlow: A flexible and efficient RLHF framework. In *EuroSys*, 2025.

W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with PagedAttention. In *SOSP*, 2023.

S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. ReAct: Synergizing reasoning and acting in language models. In *ICLR*, 2023.

L. Luo, Y. Liu, R. Liu, S. Phatale, M. Guo, H. Lara, Y. Li, and A. Agarwal. Improve mathematical reasoning in language models by automated process supervision. arXiv:2406.06592, 2024.

A. Setlur, S. Garg, X. Geng, V. Smith, V. Kumar, and A. Levine. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *ICLR*, 2025.

A. Kazemnejad, A. Agarwal, T. Kothari, S. Kakade, A. Sordoni, et al. VinePPO: Unlocking RL potential for LLM reasoning through refined credit assignment. arXiv:2410.01679, 2024.

Y. Zhou, et al. ArCHer: Training language model agents via hierarchical multi-turn RL. In *ICML*, 2024.

Y. Zhou, et al. SWEET-RL: Training multi-turn LLM agents on collaborative reasoning tasks. arXiv:2503.15478, 2025.

S. Choudhury. Process reward models for LLM agents: Practical framework and directions. arXiv:2502.10325, 2025.

Z. Qi, X. Liu, I. L. Iong, H. Lai, et al. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *ICLR*, 2025.

J. Y. Koh, S. McAleer, D. Fried, and R. Salakhutdinov. Tree search for language model agents. arXiv:2407.01476, 2024.

P. Putta, E. Mills, N. Garg, et al. Agent Q: Advanced reasoning and learning for autonomous AI agents. arXiv:2408.07199, 2024.

S. Kim, J. Suk, S. Longpre, B. Y. Lin, J. Shin, et al. Prometheus: Inducing fine-grained evaluation capability in language models. In *ICLR*, 2024.

W. Yuan, R. Y. Pang, K. Cho, X. Li, S. Sukhbaatar, J. Xu, and J. E. Weston. Self-rewarding language models. In *ICML*, 2024.

Qwen Team. Qwen3.5-9B model card. https://huggingface.co/Qwen/Qwen3.5-9B, 2026.

Qwen Team. Qwen3.6-35B-A3B model card. https://huggingface.co/Qwen/Qwen3.6-35B-A3B, 2026.

M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. In *NeurIPS*, 2017.

S. Devlin and D. Kudenko. Dynamic potential-based reward shaping. In *AAMAS*, 2012.

A. Panickssery, S. R. Bowman, and S. Feng. LLM evaluators recognize and favor their own generations. arXiv:2404.13076, 2024.

Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. EUREKA: Human-level reward design via coding large language models. In *ICLR*, 2024.

L. Yuan, W. Li, H. Chen, G. Cui, N. Ding, et al. Free process rewards without process labels. arXiv:2412.01981, 2024.

G. Cui, L. Yuan, Z. Wang, H. Wang, Y. Zhang, J. Chen, W. Li, et al. Process reinforcement through implicit rewards. arXiv:2502.01456, 2025.

A Token-level ranking preservation under step-aligned fusion

Lemma 1. *Consider a group of $G \geq 2$ trajectories with binary terminal rewards, containing at least one success and at least one failure; let A_i be the group-relative advantage of Eq. 2.1 with the population standard deviation σ of the group's terminal rewards in the denominator; let step rewards satisfy $|r_{i,t}| \leq 2$; and let fused token advantages be $a_{i,t,j} = A_i + w_s\, r_{i,t}$ as in Eq. 3.6 with $w_f = 1$. If $4\, w_s\, (\sigma + \epsilon) < 1$, then every token of a successful trajectory receives a strictly larger fused advantage than every token of a failed trajectory. Since $\sigma \leq \frac{1}{2}$ for binary rewards, the condition holds for every $w_s < \frac{1}{2}$ when ϵ is negligible.*

Proof. Let $p = \frac{1}{G}\sum_i r_i^{\text{env}} \in (0, 1)$ be the group's success fraction, so $\sigma = \sqrt{p(1-p)}$. A successful trajectory has $A_i = (1-p)/(\sigma + \epsilon)$ and a failed trajectory $A_i = -p/(\sigma + \epsilon)$. The smallest fused advantage on a success and the largest on a failure therefore satisfy

$$\min_{\text{succ}} a - \max_{\text{fail}} a \;\geq\; \frac{1}{\sigma + \epsilon} - 4w_s \;>\; 0,$$

where the last inequality is the hypothesis (with $|r_{i,t}| \leq 2$ bounding the step term by $2w_s$ on each side). □

Scope. The lemma is a statement about *token-level cross-trajectory ranking within a group* rather than Ng-style policy invariance: the group-local, per-iteration-rebuilt potential already forfeits the fixed-potential premise of potential-based shaping (Devlin and Kudenko, 2012), independent of the detour penalty. It is stated for binary terminal rewards and for w_s within the bound above; under the sample-standard-deviation convention $\sigma\sqrt{G/(G-1)}$ the bound tightens only mildly ($w_s < 0.47$ at G=8), and the deployed configuration uses $w_s = 0.2$, satisfying the hypothesis with margin; the w_s=0.6 point of the step-weight ablation exceeds the worst-case bound, and no ranking claim is made for it.

B Group settlement pseudocode

Algorithm 1 gives the settlement pass that Section 3 describes; it executes once per prompt group, after all G terminals have arrived.

C Worked examples of group settlement

Both examples are real groups drawn from the completed WebShop combined run (Sections 4–5). Objectives, action signatures, node statistics, candidate lists, judge scores, and confidence values are taken from the recorded per-group settlement traces and rubric files. Derived quantities are recomputed from these statistics using Eqs. 3.1 and 3.4–3.5, except where the text explicitly quotes recorded per-step rewards (Table 3). For clarity, the worked example displays only the graph component Φ_{graph} of Eq. 3.1: the trace records outcome statistics but not per-node anchor means, whereas the actual settlement fuses the graph and anchor components according to Eq. 3.3.

Algorithm 1 GRAR group settlement (one pass per prompt group of G trajectories)

Require: recorded paths $\{(\sigma_{i,t}, \text{payload}_{i,t})_t\}_{i=1}^{G}$, terminal rewards $\{r_i^{\text{env}}\}$, anchors $\{s_{i,t}\}$

build group-local graph from all G paths; record node/edge outcome statistics and anchor means
$H \leftarrow$ (max unique-node length over successful paths) $- 1$; $\gamma \leftarrow H/(H{+}1)$
if some $r_i^{\text{env}} = 1$ **and** graph settlement enabled **then**
  for each trajectory i, step t **do**
    $\Phi(v_t) \leftarrow 1 - (1 - q(v_t)\frac{\gamma^{\bar{d}(v_t)} - \gamma^H}{1-\gamma^H})(1 - \Phi_{\text{anchor}}(v_t))$ {Eqs. 3.1, 3.3}
    $r_{i,t} \leftarrow \Delta\Phi - \text{detour}$; clamp ≤ 0 if error/repeat; 0 if terminal {Eqs. 3.4, 3.5}
  end for
else if all $r_i^{\text{env}} = 0$ **and** rubric settlement enabled **then**
  select $\leq L$ candidates per trajectory by priority (error $>$ repeat $>$ branch $>$ remaining)
  score repeats/errors by deterministic rule; batch-score the rest with one judge call
  snap scores to $\mathcal{S}$; zero if confidence $< \tau_{\text{conf}}$; raise to $\max(r, \Delta s_t)$ when anchor-verified
end if
return step rewards and masks; terminal rewards and GRPO advantages unchanged

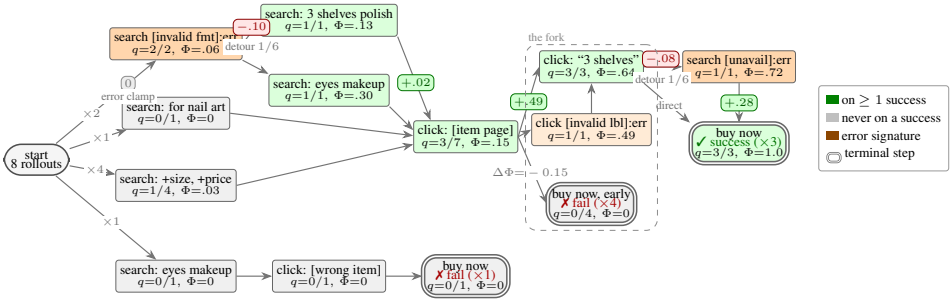

Figure 5: Group-local graph for the worked mixed group. The stadium node denotes the common task start; each path ends at a terminal `buy now`, marked by outcome, with ✓ denoting success and ✗ failure. Green nodes lie on at least one successful trajectory, gray nodes on none; orange marks an `:error` signature, and double outlines mark terminal steps. Node labels show $q{=}s/t$, outcome counts, and the potential from Eq. 3.1. Signed pills show the per-step rewards of the longest successful trajectory (Table 3).

C.1 A MIXED GROUP: THREE OF EIGHT ROLLOUTS SUCCEED

The task asks for "eyes makeup for nail polish, nail art with size: 3 shelves, and price lower than 50.00 dollars", a three-shelf organizer in which *3 shelves* is a product variant that must be selected before purchase. Of the eight rollouts, three succeed. Seven pass through the product-page click (7 visits); the three successful trajectories then click the "3 shelves" variant, whereas four failures buy immediately from the product page and fail, exactly the branch that terminal-only credit cannot localize. The eighth rollout never reaches the product page and instead purchases an incorrect product after an underspecified search. Figure 5 reconstructs the group-local graph from the trace, showing all eight rollouts from the common task start to each rollout's terminal `buy now`, marked by its outcome; it also records $H{=}5$ and $\gamma = 5/6 \approx 0.83$.

Potentials. Recomputing Eq. 3.1 from the trace counts, the successful `buy now` node has $\Phi = 1.0$ (adjacent to success), the variant click has $\Phi = 0.64$, the product-page hub has $\Phi = 0.15$ (its success fraction is $3/7$), and the premature `buy now` node has $\Phi = 0$, because no successful trajectory passed through it. The outcome statistics alone therefore assign every product-page→premature-purchase edge $\Delta\Phi = -0.15$: the graph localizes the missing variant click without any judge call.

Step credit on the longest success. Table 3 reports the step rewards recorded in the settlement trace for the successful trajectory that recovers from an invalid-format search at step 1. The opening error step receives no credit despite the later recovery, since the clamp of Section 3.2 treats a failed tool call as verified negative evidence. The well-formed search receives -0.10, its potential gain being outweighed by the detour charge at the entry fork; the variant click receives the largest single credit ($+0.49$); and the recovery error-search still receives -0.08, again because the detour charge

Table 3: Step rewards for the longest successful trajectory of Figure 5, as recorded in the settlement trace (rounded to two decimals; the detour penalty is $1/(H{+}1) = 1/6 \approx .17$ on both detoured edges, and components are themselves rounded, so recomputing from the displayed parts can differ in the last digit). The final `buy now` is a plain click in WebShop, so the terminal rule of Section 3.2 does not apply. w_s=0.2 scales these onto tokens as $w_s\, r_{i,t}$.

Step (signature, abridged)	Credit	$r_{i,t}$
`search[...] :error`	error clamp	0
`search: 3 shelves eyes makeup nail polish`	$\Delta\Phi = .07$, less detour $1/6$	$-.10$
`click: [item page]`	$\Delta\Phi = .15 - .13$	$+.02$
`click: "3 shelves"`	$\Delta\Phi = .64 - .15$	$+.49$
`search[...] :error` (recovery route)	$\Delta\Phi = .08$, less detour $1/6$	$-.08$
`click: buy now` (final action)	$\Delta\Phi = 1 - .72$ (not gated)	$+.28$

exceeds the potential gain. Finally, `buy now` receives $+0.28$: in WebShop, the winning action is an ordinary click rather than a terminal-marked step, so the terminal rule of Section 3.2 does not apply.

C.2 An all-failed group: the judge under its constraints

The task asks for "machine wash, wash cold women's fashion hoodies & sweatshirts for dry clean, tumble dry with color: cotton candy, and size: small, and price lower than 60.00 dollars." All eight rollouts fail, so every node in the group graph has success fraction 0 and there is no verified positive evidence to distribute (Section 3.3). The selector nominates 33 candidates (≈4 per trajectory), dominated by error steps: 21 are settled by the deterministic error override at score -1.0 with confidence 1.0, requiring no judge call, and the remaining 12 are scored in one batched call at temperature 0. The per-domain rubric file is passed to the judge verbatim; the entry for this task (`webshop_74`) reads:

Domain rubric file: entry `webshop_74`, quoted verbatim

Objective: "Find me machine wash, wash cold women's fashion hoodies & sweatshirts for dry clean, tumble dry with color: cotton candy, and size: small, and price lower than 60.00 dollars."

Criteria:

1. "Purchase exactly one item that satisfies every stated attribute in the objective."
2. "If the objective states a price ceiling, the purchased item must respect it."
3. "Each search must use informative keywords; each click must narrow down or verify a candidate product."
4. "Repeated clicks on the same page element or identical searches are redundant and scored negatively."
5. "Purchasing an item that misses any stated attribute is a failure regardless of partial match."

The judge prompt is quoted verbatim:

Judge prompt: quoted verbatim; the four payload fields are elided

[System] You are a strict grouped process reward judge. Return JSON only.

[User] You are judging process quality for one agent task. All eight rollout trajectories failed, so do not infer success from the terminal result. Rank only whether each selected action is a necessary, valid, non-redundant step toward the task objective.

Task={...} Task rubric={...} Eight-rollout summary={...} Candidates={...}

Evaluation criteria: prerequisite satisfaction and grounding in earlier observations; useful information gain or required environment state change; consistency with task rubric and tool schema; detours, redundant lookups, unsupported assumptions, and unnecessary transfers. Do not award positive credit merely because a tool call is syntactically valid.

Return JSON only: { "scores": [{ "id": "...", "score": 0, "confidence": 0, "reason": "..." }] }. Allowed score values are exactly 1, 0.5, 0, -0.5, -1. Keep each reason under 80 characters.

The template's opening line names τ^2-bench, where the judge module originated; the task objective, the per-domain rubric, and the group summary are passed in the corresponding fields, and all runs

reported in this paper place eight rollouts per group. The call is issued at temperature 0 with reasoning mode disabled; on return, each score is snapped to the nearest element of $\mathcal{S}$ and each confidence clamped to $[0, 1]$ before the gate of Section 3.3. Snapping has no effect when the judge already returns values of $\mathcal{S}$, as it did for all twelve candidates here.

One failing trajectory comes within a single verification click of success. The judge's three verdicts on this trajectory (Table 4) are consistent with the task rubric:

Table 4: The batched judge's verdicts on one all-failed trajectory of the worked group, as recorded in the settlement trace; scores already lie on $\mathcal{S}$.

Action	Score / conf.	Judge's reason (verbatim)
`search[machine wash ... under 60]`	$+0.5$ / 0.7	"Search query is comprehensive and includes all constraints. Valid initial step."
`click[B09P5CRVQ6]`	0.0 / 0.8	"Clicks a product without verifying attributes. Premature action before confirming product details."
`click[Buy Now]`	-1.0 / 1.0	"Purchases item without verifying attributes. Fails rubric requirement to purchase exactly one item satisfying all attributes."
`search[keywords]` (another traj.)	-1.0 / 1.0	"Search term 'keywords' is nonsensical and yields zero results. Completely redundant and invalid."

Across the twelve judged candidates, four earn $+0.5$ (well-formed searches), six earn 0.0 (five clicks or page views not grounded in earlier observations, one search too verbose and carrying an irrelevant brand), and two earn -1.0; confidences range 0.7–1.0, all above $\tau_{\text{conf}} = 0.4$, so the confidence gate never triggers in this group. When the judge call or its JSON parse fails, the trace records reward 0 and confidence 0 for the affected candidates, a fail-safe that injects no credit in either direction. Because every $A_i = 0$ here, the fused advantages of Eq. 3.6 reduce to $w_s\, r_{i,t}$: the entire update on this group is process-derived, the silent regime of Section 2.1. This example also clarifies the scope of the structural audits in Appendix I: the judge's -1.0 on the purchase is plausible and well supported by the rubric, but is not independently verified.

D Full analysis of the motivating failure

This appendix expands the motivating failure that Section 2.2 summarizes.

We analyzed 2,503 airline trajectories (1,410 successes, 1,093 failures) from an earlier GRAR prototype that computed graph rewards *per trajectory* with per-rollout graph objects. In one representative group of six rollouts for the same booking-upgrade task, four trajectories used the correct payment card and succeeded, while two used an incorrect card and failed. The relevant branch is the payment-card choice at booking, a decision that process credit should identify. Yet in all six trajectories the logged branch-comparison counter was zero: success replays credited every step up to a fixed per-trajectory budget, and failures received only self-loop penalties. The group therefore contained four positive and two negative observations about the same decision, but the prototype did not exploit this contrast. Two structural fixes follow and are adopted here: (i) the graph is *group-local*, rebuilt from all G trajectories for the same prompt so that successes and failures populate a common graph; and (ii) action signatures preserve task-determining arguments such as the payment card while normalizing away irrelevant entity identifiers.

E The silent-group regime over training

Figure 6 plots, for the five trained WebShop runs of Table 1, the per-iteration fraction of groups whose within-group terminal-reward standard deviation is zero (30 groups per iteration). The regime is most prevalent when the policy is weakest: the fraction starts high, with the untrained policy failing most groups; it declines as training proceeds, yet never approaches zero, remaining at or above one half in 38–44 of the 68 iterations of every run (Section 2.1 quantifies this).

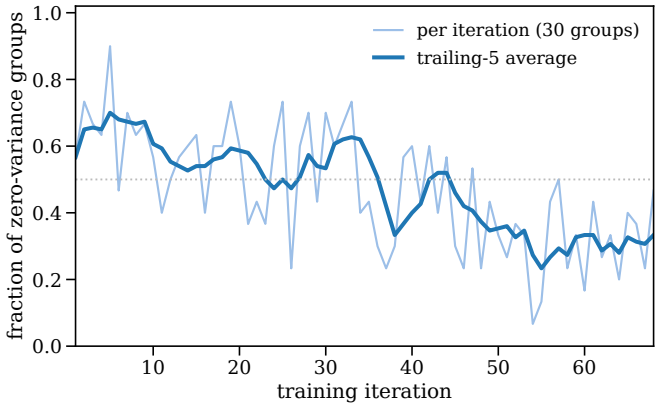

Figure 6: Per-iteration fraction of zero-variance groups, mostly all-failed groups, during WebShop training. The curve is averaged over the five trained runs in Table 1; each run contains 30 groups per iteration. The dark line is a trailing five-iteration average. The fraction starts high, declines as training proceeds, and remains nonzero throughout training.

F Extended related work

This appendix expands the related-work discussion sketched in Section 1 with the full per-family discussion.

Work that attacks sparse or terminal-only supervision falls into three families: change which data the optimizer trains on, inject step-level process credit, or shape the reward function directly. GRAR draws on the latter two while keeping the terminal verdict as the trajectory-level source of supervision.

Agentic RL and group-relative optimization. Agents that interleave reasoning and acting (Yao et al., 2023), whether browsing a web shop (Yao et al., 2022), calling tools in dialogue (Barres et al., 2025), or manipulating a household (Shridhar et al., 2021), are trained with RL from verifiable rewards (Ouyang et al., 2022; Guo et al., 2025). This includes PPO (Schulman et al., 2017), baseline-free methods such as GRPO (Shao et al., 2024) and RLOO (Ahmadian et al., 2024), and multi-turn training frameworks such as verl (Sheng et al., 2025). GRPO's all-failed degeneracy, in which the advantage and hence the gradient contribution are zero whenever all G samples fail, is well known. Existing remedies intervene elsewhere: dynamic resampling (Yu et al., 2025) changes *which groups are trained on*, whereas debiased normalization (Liu et al., 2025) removes the standard-deviation and length terms but leaves all-failed groups silent. GRAR instead retains every group and extracts process credit from failing groups. The closest step-level comparison is GiGPO (Feng et al., 2025), which groups steps by recurring *exact* anchor states and adds a return-to-go advantage within each group to the episode-level GRPO advantage. In contrast, GRAR groups steps by recurring *normalized* action signatures within the group. This grouping is coarser but more frequently recurring, and GRAR additionally records per-step settlement traces to support auditing. We include GiGPO as a step-level-credit baseline evaluated under an identical trainer and evaluation protocol.

Process reward models and step-level credit. Process reward models score reasoning steps using human labels (Lightman et al., 2024), Monte Carlo completion statistics (Wang et al., 2024), MCTS-based supervision (Luo et al., 2024), or learned progress verifiers (Setlur et al., 2025); VinePPO (Kazemnejad et al., 2024) estimates step credit from Monte Carlo returns at branch points. In agent training, step-level signals have been introduced through hierarchical value functions (Zhou et al., 2024), critic-guided multi-turn RL (Zhou et al., 2025), agent-specific process reward models (Choudhury, 2025), curriculum-based web RL (Qi et al., 2025), and action-tree search (Koh et al., 2024; Putta et al., 2024), typically at the cost of additional rollouts, a trained scorer, or both. Implicit process rewards (Yuan et al., 2024b; Cui et al., 2025) and LLM-authored reward code (Ma et al., 2024) remove the need for step labels but not the associated infrastructure. Hindsight replay (Andrychowicz et al., 2017) also extracts signal from failures by relabeling goals; GRAR instead scores steps under the original task objective. Graph settlement derives step scores from the group's *own* terminal

Table 5: Training configuration and settlement hyperparameters. The optimization settings and the main settlement hyperparameters are shared across τ^2-bench, WebShop, and ALFWorld; domain-specific adapter details are given below.

Setting	Value
Trainer and optimization	
policy; rollout engine	Qwen3.5-9B (Qwen Team, 2026a); vLLM (Kwon et al., 2023) server mode (sleep level 2)
optimizer	ms-swift
group size G; global batch	8; 48 ($= 6$ prompts $\times$ 8 rollouts)
learning rate; KL coefficient β	2×10^{-7}; 0.05
sampling temperature	1.0
max prompt / completion length	32,768 / 8,192 tokens
GRAR *settlement*	
step / trajectory weights	w_s=0.2; w_f=1.0
candidates per trajectory L	6
confidence gate τ_{conf}	0.4
deterministic overrides	error -1.0; repeat -0.5
judge	frozen Qwen3.6-35B-A3B (Qwen Team, 2026b), temperature 0

outcomes in a single replay, whereas rubric settlement invokes a judge only when terminal outcomes provide no positive evidence.

LLM-as-judge for RL. Judges provide rubric-based scores for open-ended outputs (Zheng et al., 2023), can be trained for fine-grained evaluation (Kim et al., 2024), and can generate training rewards outright (Yuan et al., 2024a); their known biases of position, verbosity, score drift, and self-preference (Panickssery et al., 2024) are usually mitigated with pairwise comparisons and ensembles. We instead bound the judge's *influence*: a five-level snapped score set, a confidence gate, deterministic overrides on errors and repeats, and restriction to all-failed groups, so it can never overturn a terminal verdict; we do not claim to measure its *accuracy*.

Potential-based shaping. Potential-based reward shaping (Ng et al., 1999) guarantees policy invariance when the shaped reward is a potential difference of a fixed state function. Our potential is group-local and rebuilt every iteration, placing it in the dynamic-shaping setting of Devlin and Kudenko (2012); the detour penalty is a second deliberate deviation, introduced to prefer concise successful paths even though strict policy invariance no longer applies.

G TRAINING CONFIGURATION AND JUDGE SCORING SEMANTICS

This appendix collects the numeric training configuration, the judge's five-level score semantics, and the per-domain adapter details that Sections 3.3 and 4 summarize.

Hyperparameters. Table 5 collects the training configuration and the settlement hyperparameters; all values are shared across the three domains.

Judge score semantics. The five levels carry explicit semantics, stated as scoring rules in the judge module; Table 6 quotes them verbatim. Two levels are additionally enforced *structurally*: repeat and error candidates never reach the judge, because the deterministic overrides pre-score them at -0.5 and -1.0 with maximal confidence (Section 3.3); in the group-level batched call, the remaining candidates are scored against the evaluation criteria and the per-task rubric exhibited in Appendix C.

Per-domain adapter details. τ^2-bench action signatures take the form `tool:<name>:<outcome>` or `message:<type>`, e.g. `tool:get_user_details:success` and `tool:update_reservation_flights:error`. WebShop signatures include `tool:search:<query>:<outcome>` and `tool:click:<label>:<outcome>`, and a loop guard terminates a run of ≥ 3 consecutive repeats as `terminal:loop_failure`. ALFWorld signatures take the form `action:<verb>:<obj>:<outcome>`. The adapter interface (signature, state payload, hindsight) is identical across domains; Table 7 summarizes the three. Two settlement details are specific to the GRAR w/o rubric run on ALFWorld, which

Table 6: The judge's five-level score set $\mathcal{S} = \{1, 0.5, 0, -0.5, -1\}$: the defining rule of each level, quoted verbatim from the judge module. A blanket rule additionally withholds positive credit for merely syntactically valid calls.

Score	Defining rule (verbatim)
$+1$	"Give 1 only when the step is necessary and makes critical progress or changes the environment state."
$+0.5$	"Give 0.5 only for useful valid progress that is not critical."
0	"A normal information lookup is score 0 unless it is a necessary key step."
-0.5	"If action is repeated [...] score MUST be ≤ -0.5"; enforced as the deterministic repeat override.
-1	"If the tool result is an error, invalid, empty, unavailable, or not found, score MUST be -1"; enforced as the deterministic error override.

Table 7: The three evaluation domains. Each environment provides a binary terminal reward, and the adapter exposes the same signature, payload, and hindsight interface across domains. "Anchor" denotes the environment-specific deterministic progress signal consumed by settlement when available.

Domain	Task type	Action space	Terminal reward	Anchor	G
τ^2-bench	tool-calling dialogue	tool calls, messages	database check	—	8
WebShop	goal-directed shopping	search, click	exact success	task score	8
ALFWorld	embodied household	text commands	task success	subgoals (opt.)	8

exposes no progress scalar: the anchor is a verified subgoal ladder derived from the action signatures, with the trajectory's stage score s_t rising $0 \to 0.4 \to 0.7 \to 1.0$ as it accumulates successful `take`, `heat`/`cool`/`clean`/`use`, and `put` actions (`:error` steps excluded), and $q(v)$ is Laplace-smoothed to $(\texttt{success_count} + 0.5)/(\texttt{terminal_count} + 1)$, damping single-visit potentials.

H ALFWorld per-task-type results

Table 9 expands the ALFWorld column of Table 1 into the four task types at both metrics; it is the breakdown underlying the per-type pattern described in Section 5.3.

I Settlement diagnostics

The settlement traces provide a structural audit of the mechanism: deterministic overrides resolve 34.6% of candidate settlements, the batched judge resolves 56.6% (mean confidence 0.91; snapping and the confidence gate never altered a reward), and the fail-safe resolves the remaining 8.8% of failed calls; the anchor lower bound of Section 3.3 covers 4.3% of candidates. Every run is audited from its settlement traces alone along the six structural criteria of Table 8, which together bound the judge's influence.

Table 8: Six audit criteria that can be evaluated from settlement traces alone.

Criterion	Check
Terminal invariance	terminal rewards are bit-identical to the baseline pipeline's
Ownership discipline	no rubric credit in groups with a success; no graph credit in all-failed groups
Clamp effectiveness	error and repeat steps never carry positive credit
Judge restraint	gate and snap statistics reported; influence bounded by L steps at $\lvert r_{i,t}\rvert \leq 1$
Detour specificity	penalties only at sources with ≥ 2 successful out-edges
No scale drift	the $\lvert r_{i,t}\rvert$ distribution stays stationary over training

Table 9: ALFWorld results by task type under the same runs, protocol, and metric definitions as Table 1. The "overall" column repeats the ALFWorld aggregate from Table 1; bold marks the best value in each column. Task types are pick & place, two-object placement, clean & place, and look at in light.

pass@1					
Variant	pick	two-obj.	clean	look	overall
Qwen3.5-9B (untrained)	55.6	25.4	7.9	36.7	30.3
terminal-only GRPO	71.9	25.4	16.4	40.0	38.1
GiGPO (Feng et al., 2025)	77.0	32.3	17.9	41.7	42.2
GRAR w/o rubric	77.0	27.7	19.3	43.3	41.5
GRAR w/o graph	**83.0**	42.4	21.4	**55.0**	**49.6**
GRAR (combined)	**83.0**	**46.2**	**22.9**	43.3	49.5
pass@5					
Variant	pick	two-obj.	clean	look	overall
Qwen3.5-9B (untrained)	96.3	69.2	32.1	**91.7**	68.8
terminal-only GRPO	**100.0**	69.2	46.4	75.0	72.0
GiGPO (Feng et al., 2025)	**100.0**	76.9	46.4	83.3	75.3
GRAR w/o rubric	**100.0**	76.9	42.9	83.3	74.2
GRAR w/o graph	**100.0**	84.0	53.6	**91.7**	80.4
GRAR (combined)	**100.0**	**96.2**	**60.7**	75.0	**85.0**