ADAPTIVE ADVANTAGE REDISTRIBUTION FOR SOFTWARE-ENGINEERING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning for software-engineering agents uses terminal test outcomes to supervise trajectories of commands and edits. Episode-level advantages provide uniform supervision within each trajectory, limiting step-level credit assignment. We propose adaptive advantage redistribution, which groups steps across trajectories by their pre-action state. Local advantages are then estimated by comparing the terminal returns associated with different actions taken at the same state. When only a few rollouts visit a state, its mean return can be unreliable. Adaptive shrinkage combines this state-group mean with the task-level mean to estimate the comparison baseline. Direct addition of the resulting local advantages can alter the original sum of token advantages by different amounts across trajectories, changing their relative contributions to the policy update. We address this through constrained redistribution, which converts local advantages into step-specific corrections while preserving each trajectory's original advantage sum. The policy objective then clips the episode-level and correction terms separately, retaining a restoring gradient when their signs oppose and the probability ratio exceeds the clipping bounds. With Qwen3-Coder-30B-A3B-Instruct, the method achieves resolve rates of 57.0% on SWE-bench Verified and 26.0% on SWE-bench Pro Public, compared with 50.8% and 21.9%, respectively, for episode-only training. Ablation experiments demonstrate the effectiveness of step-level credit assignment, adaptive shrinkage, constrained redistribution, and separate clipping.

1 INTRODUCTION

Software-engineering agents resolve repository issues through code inspection, editing, and test execution (Jimenez et al., 2024; Yang et al., 2024). Verifier-based reinforcement learning supervises these interactions using tests on the submitted patch (Agentica & AI, 2025). The outcome indicates success but does not assign credit to the preceding commands and edits.

Group-relative optimization computes an episode-level advantage from sampled terminal outcomes and assigns it to every generated token (Shao et al., 2024). State-relative comparisons provide finer information through returns from trajectories visiting a shared decision state (Feng et al., 2025; Wang et al., 2026). Each command or edit remains a complete action sample, including its arguments and replacement text. Comparing their associated returns yields local advantages without requiring identical patches to recur.

Each rollout uses a baseline estimated from the other rollouts in its task group. Adaptive shrinkage combines state-group and task-level means according to visitation support and observed return differences, balancing state-specific information with the task context when few trajectories reach a state.

Directly adding local advantages changes each trajectory's original sum of token advantages by an amount that can differ across rollouts, altering their relative contributions to the batch update. We instead redistribute credit within each trajectory while preserving that sum. A weighted projection subtracts the token-weighted trajectory mean from each local advantage, producing zero-mean corrections to the episode-level coefficient. Matching the projection weights to the policy loss ensures conservation for turns of unequal length. The projection also removes trajectory-wide offsets and contracts the weighted norm of the local coefficients while retaining within-trajectory variation.

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.

The episode-level advantage and its correction can have opposite signs. Clipping their sum can place both components on a plateau, suppressing the corrective update. We clip them separately using the same probability ratio and bounds. Separate and joint clipping agree inside the clipping interval and whenever the components agree in sign. Otherwise, separate clipping adds an out-of-interval penalty that retains a restoring term, distinct from the conservation of advantage coefficients.

These operations form adaptive advantage redistribution (Figure 1): adaptive shrinkage estimates local advantages, constrained redistribution allocates them across steps, and separate clipping incorporates the resulting components into policy optimization.

We evaluate the method on SWE-bench Verified and SWE-bench Pro Public. With Qwen3-Coder-30B-A3B-Instruct, it reaches 57.0% and 26.0% resolve rate, respectively, compared with 50.8% and 21.9% for episode-only training. Within-model improvements on Verified also hold for Qwen3-8B and Qwen2.5-Coder-32B-Instruct. Matched-budget ablations test adaptive shrinkage, redistribution, and separate clipping. Objective analysis characterizes the resulting update rule.

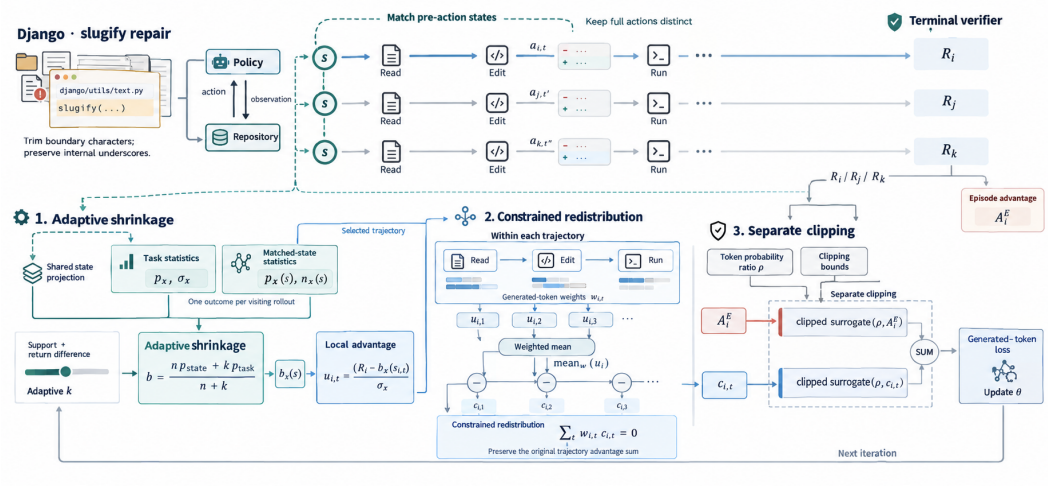

Figure 1: Overview of adaptive advantage redistribution for a Django `slugify` repair. Steps are matched by their pre-action states while complete actions remain distinct. State-group and task-level baseline statistics exclude the target rollout. Adaptive shrinkage combines these statistics to estimate local advantages. Token-weighted centering converts these advantages into corrections that preserve each trajectory's original advantage sum. The episode advantage and local correction then enter separate clipped surrogates before their contributions are combined for the policy update.

2 PRELIMINARIES

Repository interaction. An episode begins with an issue description and a repository snapshot. At step t the agent observes o_t, the output of its previous command, and emits an action a_t: reasoning followed by a tool call, typically a shell command that reads, searches, edits, or runs tests. After at most T steps, the submitted patch is evaluated by a held-out test suite, yielding a binary terminal reward $R \in \{0, 1\}$. This is the sole reward, following repository-level evaluation (Jimenez et al., 2024) and verifier-based training (Agentica & AI, 2025).

Group-relative advantages. For task x, sample $N \geq 2$ rollouts $\{\tau_1, \ldots, \tau_N\}$ from the current policy with rewards $\{R_1, \ldots, R_N\}$. The episode-level advantage is

$$A_i^E = \frac{R_i - \bar{R}}{\sigma_x}, \qquad \bar{R} = \frac{1}{N}\sum_j R_j, \qquad \sigma_x^2 = \frac{1}{N}\sum_j (R_j - \bar{R})^2. \tag{1}$$

This value is assigned to every model-generated token of τ_i (Shao et al., 2024).

Loss weights. Let $L_{i,t}$ count the unmasked generated tokens in turn t of rollout i, including reasoning and tool arguments. Define $L_i = \sum_t L_{i,t}$ and $w_{i,t} = L_{i,t}/L_i$. Observations and padding have zero weight. Uniform-outcome task groups have zero reward variance and are omitted from updates. Rollouts with $L_i = 0$ contribute no policy-loss tokens.

3 METHOD

We estimate local advantages from pre-action state groups using adaptive baselines, convert them into corrections that preserve each trajectory's original token-advantage sum, and clip the episode and correction terms separately during policy optimization.

3.1 STATE-RELATIVE LOCAL ADVANTAGES

For task x, we record the pre-action state

$$s_{i,t} = \phi(h_{i,t}) = (\text{current file}, \text{has edited}, \text{has executed}, \text{last command failed}). \tag{2}$$

The key is updated after each command's result. Groups are task-local; the policy remains conditioned on the full interaction history. Each action retains its reasoning, tool name, and complete arguments, including replacement text for edits; a multi-file shell command remains one action turn.

Let $\mathcal{I}_x(s) = \{j : \exists t,\ s_{j,t} = s\}$ denote the distinct rollouts visiting state s. For target rollout i, define $\mathcal{I}_x^{-i}(s) = \mathcal{I}_x(s) \setminus \{i\}$ and $M = N - 1$. The comparison statistics exclude all visits and the outcome of rollout i:

$$n_x^{-i}(s) = |\mathcal{I}_x^{-i}(s)|, \qquad p_x^{-i}(s) = \frac{\sum_{j \in \mathcal{I}_x^{-i}(s)} R_j}{n_x^{-i}(s)}, \qquad p_x^{-i} = \frac{\sum_{j \neq i} R_j}{M}. \tag{3}$$

The state mean is defined only for nonzero support, counting each other rollout once regardless of repeat visits. Action turns remain separate samples; the baseline and adaptive weight use these leave-one-rollout-out statistics.

We define the local advantage as the return residual relative to this baseline:

$$u_{i,t} = \frac{R_i - b_x^{-i}(s_{i,t})}{\sigma_x}. \tag{4}$$

The episode advantage and normalization retain the full-group definitions in Equation 1. This residual supplies a local credit signal for redistribution, not an unbiased estimate of the history-conditioned advantage. Different target rollouts can have different baselines at the same state. Appendix C illustrates the calculation.

3.2 ADAPTIVE BASELINE SHRINKAGE

For a target rollout and state, abbreviate the excluded statistics as $n = n_x^{-i}(s)$, $p_s = p_x^{-i}(s)$, and $p = p_x^{-i}$. We combine the state-group and task-level means using an adaptive weight:

$$b_x^{-i}(s) = (1 - \alpha)p + \alpha p_s = \frac{np_s + kp}{n + k}, \tag{5}$$

where $\alpha = \alpha_x^{-i}(s)$ and $k = k_x^{-i}(s) = n(1 - \alpha)/\alpha$ for $\alpha > 0$. For $\alpha = 0$, we use $b_x^{-i}(s) = p$ and write $k = \infty$.

Reference variation. For $2 \leq n < M$ and $0 < p < 1$, define

$$v = \frac{p(1-p)}{n} \frac{M - n}{M - 1}, \qquad D = (p_s - p)^2. \tag{6}$$

The variance v of a uniformly sampled subset mean, conditional on the other M outcomes, calibrates the squared difference D. Since state visits depend on the policy, it is a reference scale rather than the actual sampling variance (Appendix B).

Adaptive weight. The adaptive weight retains the part of the squared mean difference that exceeds the reference variance:

$$d^2 = [D - v]_+, \qquad \alpha = \frac{d^2}{d^2 + v}. \tag{7}$$

Here $[z]_+ = \max(z, 0)$. When $D \leq v$, the baseline equals the excluded task mean; otherwise, $\alpha = 1 - v/D$. At a fixed nonzero mean difference, greater support reduces v and increases the state mean's weight. We use $\alpha = 0$ when fewer than two other rollouts visit the state, all M others visit it, or their task outcomes are uniform. These cases require no division by v or an undefined state mean.

For eight rollouts with four successes, suppose a state has three successful visitors and one failed visitor. Excluding a successful target leaves two successes among three state visitors, giving $k = 72$ and $b_x^{-i}(s) = 46/105$. Excluding the failed target instead gives $k = 24/19$ and $b_x^{-i}(s) = 55/63$. Appendix C continues the example through redistribution.

3.3 Trajectory-Constrained Advantage Redistribution

To incorporate local advantages without changing the original trajectory-level advantage sum, we compute corrections $c_{i,t}$ by solving

$$\min_{\{c_{i,t}\}} \frac{1}{2} \sum_t w_{i,t}(c_{i,t} - u_{i,t})^2 \quad \text{subject to} \quad \sum_t w_{i,t} c_{i,t} = 0. \tag{8}$$

On turns with positive loss weight, the unique solution is

$$\bar{u}_i = \sum_t w_{i,t} u_{i,t}, \qquad c_{i,t} = u_{i,t} - \bar{u}_i, \qquad A_{i,t} = A_i^E + c_{i,t}. \tag{9}$$

Thus $\sum_t L_{i,t} A_{i,t} = L_i A_i^E$: using the policy loss's token weights preserves the original advantage sum even when turns have different lengths.

For terminal-only returns, the correction has the equivalent expression

$$c_{i,t} = \frac{\bar{b}_i - b_x^{-i}(s_{i,t})}{\sigma_x}, \qquad \bar{b}_i = \sum_t w_{i,t} b_x^{-i}(s_{i,t}). \tag{10}$$

The outcome enters through A_i^E, while relative state baselines determine the correction: turns below the weighted baseline mean receive positive corrections, and those above receive negative corrections. Constant baselines recover the episode-level advantage.

Effect on the policy gradient. At the rollout policy, with fixed advantages and inactive clipping, redistribution removes the gradient component due to the trajectory-wide mean local advantage while retaining within-trajectory variation. The correction is invariant to constant offsets and has no greater weighted norm than the original local coefficients; this does not imply conservation of the policy-gradient norm. Appendix B gives the decomposition and norm bound.

3.4 Separate Clipping of Episode and Step Components

The episode advantage A_i^E and zero-mean correction $c_{i,t}$ can have opposite signs. We retain them separately in the policy objective. For token ℓ in turn t, define $\rho_{i,t,\ell}(\theta) = \pi_\theta(y_{i,t,\ell} \mid h_{i,t,\ell}) / \pi_{\text{old}}(y_{i,t,\ell} \mid h_{i,t,\ell})$, and write

$$\bar{\rho} = \text{clip}(\rho, 1 - \epsilon_-, 1 + \epsilon_+), \qquad \ell_{\text{clip}}(\rho, a) = \min(\rho a, \bar{\rho} a). \tag{11}$$

This is the usual clipped surrogate (Schulman et al., 2017). Applying it to the sum gives $\ell_{\text{clip}}(\rho, A_i^E + c_{i,t})$, whose clipping behavior is determined by the sign of the combined coefficient. We instead maximize

$$J_{\text{sep}}(\theta) = \frac{1}{\sum_i L_i} \sum_{i,t} \sum_{\ell=1}^{L_{i,t}} \left[\ell_{\text{clip}}(\rho_{i,t,\ell}, A_i^E) + \ell_{\text{clip}}(\rho_{i,t,\ell}, c_{i,t}) \right]. \tag{12}$$

The terms share the same ratio, clipping bounds, and token-loss normalization; there is no additional mixing coefficient. Advantages are held fixed during policy optimization.

Disagreement-dependent penalty. Let $\lambda_{i,t} = \mathbf{1}[A_i^E c_{i,t} < 0] \min(|A_i^E|, |c_{i,t}|)$. For any nonnegative probability ratio, separate clipping satisfies

$$\ell_{\text{clip}}(\rho, A_i^E) + \ell_{\text{clip}}(\rho, c_{i,t}) = \ell_{\text{clip}}(\rho, A_i^E + c_{i,t}) - \lambda_{i,t}|\rho - \bar{\rho}|. \tag{13}$$

The identity also holds for asymmetric bounds. Separate and joint clipping coincide inside the interval or when the components agree in sign. Otherwise, separate clipping subtracts a piecewise-linear penalty for ratio excursions, weighted by the smaller component magnitude. Appendix B provides the proof and Figure 4 visualizes the objectives.

Corrective updates beyond the interval. For $A_i^E = 1$, $c_{i,t} = -0.3$, and bounds $[0.8, 1.28]$, both objectives have derivative 0.7 with respect to ρ at $\rho = 1$. At $\rho = 1.4$, joint clipping is on a plateau, while separate clipping retains derivative -0.3: the episode term stops increasing probability and the correction supplies a restoring term. The reverse-sign case is given in Appendix B.

Separate clipping preserves the unclipped gradient at the rollout policy but changes the surrogate outside the interval. A negative correction expresses relative credit, not action incorrectness; the restoring term does not guarantee bounds on actual policy ratios, KL, or gradient variance.

Procedure and complexity. One pass over the sampled turns accumulates distinct-rollout state statistics. A second pass subtracts each target rollout's contribution from the task and visited-state totals, computes its baselines, and centers its local advantages. Each excluded statistic takes constant time from the accumulated totals. Time and auxiliary storage are linear in the number of sampled turns; no additional environment interaction is required. The policy loss evaluates two scalar clipping terms per generated token, without additional model forward passes. Algorithm 1 gives the complete procedure.

4 Experiments

4.1 Experimental Setup

Data and initialization. We use R2E-Gym-Subset: 4,578 executable tasks from 10 repositories after excluding repositories represented in the SWE-bench test set (Jain et al., 2025). Executable tests supply the binary terminal reward. The starting checkpoints are Qwen3-8B, Qwen3-Coder-30B-A3B-Instruct (Yang et al., 2025a), and Qwen2.5-Coder-32B-Instruct (Hui et al., 2024). Training starts directly from these checkpoints.

Agent and optimization. We use mini-swe-agent (Yang et al., 2025c) with a 32k-token context and at most 100 steps. Our training configuration uses 32 GPUs with veRL (Sheng et al., 2024), 16 tasks per update, and $N = 8$ rollouts per task for 200 updates. Uniform-outcome groups are omitted from policy updates. The full estimator uses Equation 7; the variant without adaptation uses a constant $k = 4$. Both clipping terms use $\epsilon_- = 0.2$ and $\epsilon_+ = 0.28$. The joint-clipping control uses these same bounds.

Each batch contains 128 rollouts before uniform-outcome filtering. Our comparison baselines use the other rollouts' task-local outcomes, without intermediate rewards or additional continuations. The loss includes generated reasoning and tool-call tokens and masks environment observations.

Benchmarks and reporting. We evaluate on SWE-bench Verified (500 issues) (Jimenez et al., 2024) and SWE-bench Pro Public (731 issues) (Deng et al., 2025), using the same checkpoint across benchmarks without benchmark-specific fine-tuning. Single-attempt resolve rate is $100m/M$, where m is the number of resolved issues and M includes all issues, including timeouts. Scores use one-decimal round-half-up rounding; differences are computed before rounding. Table 1 reports the reference evaluation (decoding seed 11). Tables 2 and 3 report mean $\pm$ sample standard deviation across three complete evaluations (seeds 11, 29, 47), with fixed checkpoints and decoding settings. Each repeat makes one attempt per issue, without combining predictions across repeats. Standard deviations use denominator $3 - 1$ and reflect decoding, not training-run, variability. Shared configurations reuse the same three evaluations.

Table 1: Results on SWE-bench Verified and SWE-bench Pro Public with Qwen3-Coder-30B-A3B-Instruct. Each baseline retains its complete training recipe, adapted to the common backbone, task pool, and sampling budget. Training reward and credit estimation summarize the configurations; tests denote terminal execution feedback. Values are single-attempt resolve rates (%) from the reference evaluation (seed 11).

Method	Training reward	Credit estimation	Verified	Pro Public
GRPO	Tests	Group-relative	50.8	21.9
RLOO	Tests	Leave-one-out	50.4	21.5
SWE-RL	Patch similarity	Group-relative	49.8	18.7
DeepSWE	Tests	Leave-one-out	52.0	23.4
SkyRL-Agent (SA-SWE)	Tests	Leave-one-out	52.4	24.1
RTMC	Tests + step shaping	Rollout-tree MC	51.8	23.8
Ours	Tests	Adaptive state-relative	**57.0**	**26.0**

4.2 Comparison of Complete Training Methods

We compare SWE RL methods on Qwen3-Coder-30B-A3B-Instruct (Table 1). The main baselines are SWE-RL (Wei et al., 2025), DeepSWE (Agentica & AI, 2025), SkyRL-Agent's SA-SWE recipe (Cao et al., 2025), and RTMC (Wang et al., 2026). GRPO (Shao et al., 2024) and RLOO (Ahmadian et al., 2024) provide general-purpose RL references. Each SWE baseline retains its complete training recipe, including its reward function, interaction workflow, training tools, trajectory filtering, and optimization rules. We adapt these recipes to the common backbone, task pool, and sampling budget, and evaluate the resulting checkpoints with the single-attempt protocol in Section 4.1. This compares complete training methods, not isolated optimization components.

Baseline configurations. SWE-RL retains its patch-generation workflow and reference-patch similarity reward. DeepSWE retains its trajectory-level RL recipe, including leave-one-out advantages and compact filtering. SA-SWE retains training-time AST search, recovery hints, and its on-policy optimization procedure. RTMC retains rollout-tree credit estimation, prior smoothing, and step rewards. Section 4.4 isolates our estimator's components.

On Verified, our method reaches 57.0%, exceeding the transferred DeepSWE and SA-SWE recipes by 5.0 and 4.6 points, respectively. Both SWE-specific recipes outperform the GRPO and RLOO references, showing the value of training choices beyond the generic optimizer.

SWE-RL reaches 49.8%, below the execution-reward recipes in this comparison. It optimizes similarity to a reference patch, whereas evaluation tests whether the submitted patch resolves the issue. This may contribute to the ordering, although the different repair workflows prevent attributing the gap to reward design alone.

The gain extends to SWE-bench Pro Public. Our method resolves 190 of 731 issues (26.0%), compared with 21.9% for GRPO and 24.1% for SA-SWE, the strongest baseline in this column. The respective margins are 4.1 and 1.9 points, compared with 6.2 and 4.6 points on Verified. The method's advantage therefore persists beyond the benchmark used for the detailed ablations. Absolute resolve rates remain lower on Pro Public for every method, leaving substantial headroom even with step-level credit assignment.

DeepSWE and RTMC reach 23.4% and 23.8% on Pro Public, respectively, compared with 18.7% for SWE-RL. The smaller margins on Pro Public indicate that the benefit varies across benchmarks rather than providing a uniform gain.

4.3 Generalization Across Backbones

Table 2 compares the initial checkpoints and RL methods across three backbones. GRPO and RLOO provide general-purpose references, while DeepSWE represents a SWE-specific training recipe. The configurations follow Section 4.2, with data and rollout budgets matched within each backbone.

RLOO performs similarly to GRPO across the three backbones, while DeepSWE achieves higher resolve rates than both. Our method further improves on DeepSWE by 1.7 points on Qwen3-8B,

Table 2: Resolve rates (%) across backbones on SWE-bench Verified. Columns compare the initial checkpoint, GRPO, RLOO, DeepSWE, and our method; entries are mean ± sample standard deviation across three evaluations.

Backbone	Initial	GRPO	RLOO	DeepSWE	Ours
Qwen3-8B	18.3 ± 0.5	24.1 ± 0.7	23.5 ± 0.7	25.3 ± 0.6	$\mathbf{27.1} \pm 0.7$
Qwen3-Coder-30B-A3B	46.9 ± 0.8	50.9 ± 0.7	50.2 ± 0.7	52.1 ± 0.7	$\mathbf{56.9} \pm 0.7$
Qwen2.5-Coder-32B	41.1 ± 0.5	45.9 ± 0.8	45.7 ± 0.8	47.5 ± 0.7	$\mathbf{49.1} \pm 0.8$

4.9 points on Qwen3-Coder-30B-A3B, and 1.5 points on Qwen2.5-Coder-32B. Improvements over GRPO range from 3.0 to 6.1 points. The gains hold across the tested backbones, with the largest margin on Qwen3-Coder.

The initial-checkpoint column separates improvements from RL training from those obtained by changing the training method. For example, Qwen3-8B gains 5.7 points from GRPO and a further 3.0 points from our method, whereas Qwen2.5-Coder-32B gains 4.7 and 3.2 points, respectively. The additional benefit appears at both lower and higher starting resolve rates.

4.4 Ablation Study

Table 3 separates baseline estimation, redistribution, and policy clipping under the shared training setup in Section 4.1. Without adaptation, the baseline uses fixed $k = 4$ and retains both redistribution and separate clipping. The joint-clipping control retains the same A_i^E and $c_{i,t}$ but applies one clipped term to their sum. Direct addition removes centering and uses $\ell_{\text{clip}}(\rho, A_i^E) + \ell_{\text{clip}}(\rho, u_{i,t})$, retaining separate clipping to isolate the allocation constraint. Episode-only training sets the local component to zero, recovering $\ell_{\text{clip}}(\rho, A_i^E)$.

Without adaptation, mean resolve rate falls from 56.9% to 53.1%, still 2.3 points above episode-only training. State-relative comparisons remain useful with a fixed prior, while adaptive weighting adds a further 3.8 points with redistribution and separate clipping held fixed.

Direct addition reaches 52.3%, 4.6 points below the full method. For given local advantages, centering preserves every within-trajectory difference, $c_{i,t} - c_{i,t'} = u_{i,t} - u_{i,t'}$, while removing the common offset. This comparison therefore tests the aggregate allocation of credit, rather than merely whether step coefficients differ. Removing local advantages altogether recovers episode-only training at 50.9%.

Joint clipping reaches 53.3%, 3.6 points below separate clipping, despite using the same conserved coefficients. This isolates the policy objective from local estimation and allocation; Appendix B characterizes where the objectives differ. Together, the ablations support adaptive baseline estimation, constrained allocation, and separate clipping. Their effects are conditional comparisons, not additive shares of the total improvement.

4.5 State Support and Credit Activation

We examine state support and credit activation over 200 updates, pooling 722,423 positive-weight turns from 23,376 rollouts in non-uniform task groups. Each sampled turn is counted once, independently of optimizer reuse. Support $n_x^{-i}(s)$ counts distinct other rollouts visiting the state. Baseline activation is measured by $\alpha_x^{-i}(s) > 0$, equivalent under our rule to $b_x^{-i}(s) \neq p_x^{-i}$. Correction activity uses $|c_{i,t}| > 10^{-8}$; its magnitude is summarized by

$$C_{\text{RMS}} = \sqrt{\frac{\sum_{(i,t)\in\mathcal{T}} L_{i,t} c_{i,t}^2}{\sum_{(i,t)\in\mathcal{T}} L_{i,t}}}, \tag{14}$$

where $\mathcal{T}$ contains all included turns. Rates pool their numerators and denominators across updates rather than averaging update-level percentages.

Table 3: Component ablations on SWE-bench Verified with Qwen3-Coder-30B-A3B-Instruct. Joint clipping changes only the surrogate applied to the two advantage components; direct addition removes centering but retains separate clipping. Resolve rates are mean ± sample standard deviation across three evaluations; differences use unrounded means and are given in percentage points.

Configuration	Resolve rate (%)	Difference (pp)
Full method	56.9 ± 0.7	0.0
w/o adaptation (fixed $k = 4$)	53.1 ± 0.9	-3.8
Joint clipping (sum before clipping)	53.3 ± 0.7	-3.6
Direct addition (no centering)	52.3 ± 0.8	-4.6
Episode-only (A_i^E)	50.9 ± 0.7	-6.1

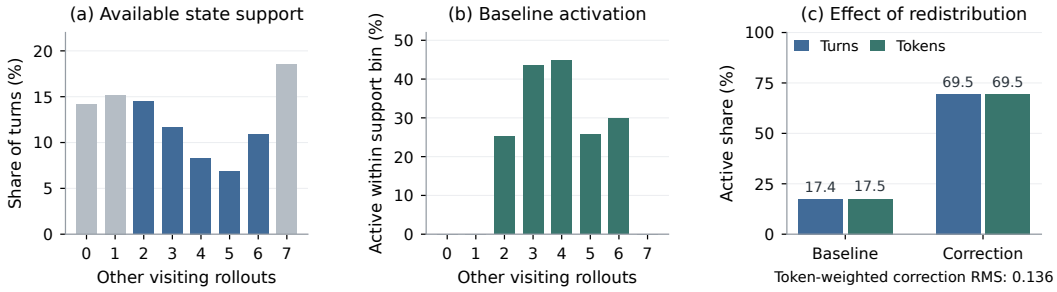

Figure 2: State support and credit activation. (a) Turn-weighted distribution of the number of other visiting rollouts. (b) Fraction of turns with a nonzero adaptive weight within each support bin. (c) Baseline and correction activation under turn and generated-token weighting. Baseline activation compares against the target-excluded task mean; correction activity is measured after centering.

Figure 2(a) shows that 29.3% of turns have fewer than two other visitors, while 18.5% share their state with all seven other rollouts. Both cases use the task-level baseline. The remaining 52.1% have intermediate support, but only 17.4% of all turns receive a nonzero adaptive weight. Activation is not monotonic in support (Figure 2(b)): shrinkage depends on both support and return differences, rather than retaining every available comparison.

After centering, 69.5% of turns have nonzero corrections. Turns with an unchanged task-level baseline but a nonzero correction account for 52.1% of all included turns. Subtracting the trajectory-weighted mean thus propagates local baseline adjustments across the trajectory. Token-weighted activation is 17.5% for baselines and 69.5% for corrections, close to the turn-level rates, while $C_{\mathrm{RMS}} = 0.136$. These diagnostics describe the frequency and magnitude of credit redistribution, not causal effects of individual actions.

4.6 Clipping Under Matched Initial Conditions

We sample 4,096 turns from Section 4.5 and optimize independent Bernoulli logits under both objectives, matching coefficients, initial ratios, clipping bounds, and step size. We track fixed opposing-sign cohorts on initial joint-clipping plateaus: 236 upper-bound and 137 lower-bound cases over 60 scalar optimization steps (Appendix E).

Joint clipping leaves the initial excursions unchanged at 0.1572 and 0.0860 for the upper and lower cohorts, respectively. Separate clipping reduces them to 0.0744 and 0.0345, decreases of 52.7% and 59.9%. Separate clipping retains a restoring derivative from the opposing component on the joint plateau. This isolates the scalar mechanism, not its effect under coupled token updates in agent training.

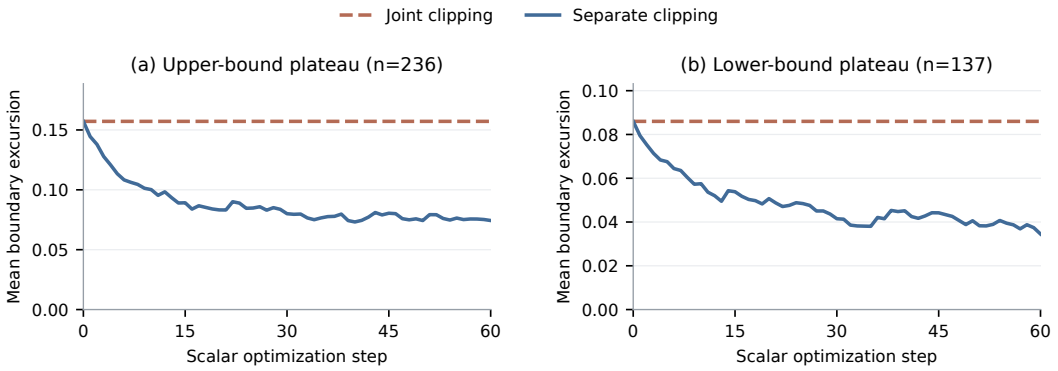

Figure 3: Clipping under matched initial conditions. Token-weighted mean distance beyond the violated boundary for fixed upper- and lower-plateau cohorts. Curves show every scalar step without smoothing, including zero excursions after re-entry.

5 Related Work

Reinforcement learning for software-engineering agents. SWE-Gym and R2E-Gym provide executable environments for training agents and verifiers (Pan et al., 2025; Jain et al., 2025), while SWE-smith synthesizes test-breaking changes to scale task construction (Yang et al., 2025b). SWE-RL learns from software evolution using reference-patch similarity rewards (Wei et al., 2025), whereas DeepSWE trains interactive agents with binary execution feedback (Agentica & AI, 2025). SkyRL-Agent combines efficient rollout collection with a SWE training recipe using AST-based search (Cao et al., 2025). These systems address supervision, interaction tools, and training efficiency; our focus is how terminal outcomes supervise actions within a trajectory.

Outcome-grounded step-level credit. GRPO and RLOO use sampled outcomes to construct group-relative and leave-one-out baselines, respectively (Shao et al., 2024; Ahmadian et al., 2024). For finer-grained credit, PRIME learns implicit process rewards from outcome labels (Cui et al., 2025), and VinePPO uses Monte Carlo continuation estimates (Kazemnejad et al., 2025). GiGPO forms step-level groups from repeated decision states (Feng et al., 2025), while RTMC aggregates returns through rollout trees and state–action signatures (Wang et al., 2026). RUDDER studies return decomposition and return-equivalent reward redistribution (Arjona-Medina et al., 2019). Our method combines target-excluded state comparisons with adaptive shrinkage, then redistributes advantage coefficients under a token-weighted conservation constraint and clips the two components separately.

6 Conclusion

Adaptive advantage redistribution connects state-relative estimation with trajectory-level credit allocation and policy optimization. It estimates local advantages from terminal returns, uses adaptive shrinkage to balance state-group and task-level means, and converts these advantages into zero-mean corrections. The resulting action coefficients preserve each trajectory's original advantage sum. Separate clipping retains the episode-level and correction terms in the policy objective, adding a restoring penalty when their signs oppose and the probability ratio leaves the clipping interval. With Qwen3-Coder-30B-A3B-Instruct, the method reaches 57.0% on SWE-bench Verified and 26.0% on Pro Public, improving on episode-only training by 6.2 and 4.1 points, respectively. Component ablations support each stage of the method, while the objective analysis clarifies the distinction between coefficient conservation and the subsequent clipped update. Extending this formulation to other tool-use tasks and reward structures is a natural direction for future work.

AI USE STATEMENT

Generative AI tools were used solely as auxiliary tools in this work, including assisting with experimental code implementation and debugging, routine code and consistency checks, and drafting, editing, and polishing the manuscript. AI was also used to assist with mathematical notation and presentation.

REFERENCES

Agentica and Together AI. Deepswe: Training a fully open-sourced, state-of-the-art coding agent by scaling rl, 2025. `https://www.together.ai/blog/deepswe`.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return decomposition for delayed rewards. *arXiv preprint arXiv:1806.07857*, 2019.

Shiyi Cao, Dacheng Li, Fangzhou Zhao, Shuo Yuan, Sumanth R. Hegde, Connor Chen, Charlie Ruan, Tyler Griggs, Shu Liu, Eric Tang, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-agent: Efficient rl training for multi-turn llm agent. *arXiv preprint arXiv:2511.16108*, 2025.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, et al. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025.

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

Binyuan Hui, Jian Yang, Zeyu Cui, et al. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights swe agents. *arXiv preprint arXiv:2504.07164*, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations (ICLR)*, 2024.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in rl training of llms. In *International Conference on Machine Learning (ICML)*, 2025.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. In *International Conference on Machine Learning (ICML)*, 2025.

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

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.

Tao Wang, Suhang Zheng, and Xiaoxiao Xu. Rtmc: Step-level credit assignment via rollout trees. *arXiv preprint arXiv:2604.11037*, 2026.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024.

John Yang, Kilian Lieret, Carlos E Jimenez, et al. Swe-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2025b.

John Yang, Kilian Lieret, et al. mini-swe-agent, 2025c. `https://github.com/SWE-agent/mini-swe-agent`.

A The Algorithm in Full

B Properties of the Estimator

Finite-population reference variance. Fix a target rollout and the $M = N - 1$ remaining binary rewards, with mean p. For a uniformly sampled subset of size n, the population variance with denominator $M-1$ is $S^2 = Mp(1-p)/(M-1)$. The subset-mean variance is $(1-n/M)S^2/n = p(1-p)(M-n)/(n(M-1))$, giving Equation 6. This calculation conditions only on the non-target outcomes; it calibrates, rather than models, policy-induced visits.

Equivalent shrinkage strength. For $D > v$, substituting $\alpha = 1 - v/D$ into $k = n(1-\alpha)/\alpha$ gives

$$k = \frac{nv}{D - v}. \tag{15}$$

The quantities are specific to the excluded target and queried state, as in Section 3.2. If the remaining task outcomes are uniform, the state and task means coincide wherever the state mean exists, and we use $\alpha = 0$.

Projection solution. For turns with $w_t > 0$, the Lagrangian of Equation 8 is

$$\mathcal{L}(c, \lambda) = \tfrac{1}{2} \sum_t w_t (c_t - u_t)^2 + \lambda \sum_t w_t c_t. \tag{16}$$

Stationarity gives $c_t = u_t - \lambda$. Substituting into the constraint yields $\lambda = \sum_t w_t u_t$. Strict convexity on positive-weight turns gives uniqueness. Zero-token turns are outside the loss; the implementation assigns them the same algebraic expression for consistent logging.

The projection satisfies

$$\sum_t w_{i,t} c_{i,t}^2 = \sum_t w_{i,t} u_{i,t}^2 - \bar{u}_i^2 \le \sum_t w_{i,t} u_{i,t}^2. \tag{17}$$

The correction is orthogonal, in the weighted inner product, to the constant vector. Expanding $u_t = c_t + \bar{u}$ yields $\sum_t w_t u_t^2 = \sum_t w_t c_t^2 + \bar{u}^2$, which proves Equation 17. For any other feasible correction $c'_t = c_t + z_t$, where $\sum_t w_t z_t = 0$, its objective exceeds the minimum by $\frac{1}{2} \sum_t w_t z_t^2$.

Algorithm 1 Adaptive advantage redistribution with separate clipping.

Require: task groups $\{(\tau_i, R_i)\}$, token counts $L_{i,t}$, clipping widths ϵ_-, ϵ_+
Ensure: fixed coefficients $\{A_i^E, c_{i,t}, A_{i,t}\}$ and a policy update

for each task x with non-uniform rewards **do**
  compute N, $\bar{R}$, σ_x, and A_i^E from the full group
  for each rollout i **do**
    initialize state s and empty visited-state set H_i
    for each action turn t **do**
      record $(i, t, s, a_{i,t}, L_{i,t})$
      if $s \notin H_i$ **then**
        add R_i to the statistics for s; insert s into H_i
      end if
      update s from the command and its observation
    end for
  end for
  for each rollout i with $L_i > 0$ **do**
    $M \leftarrow N - 1$; $p \leftarrow (\sum_j R_j - R_i)/M$
    for each distinct state s visited by rollout i **do**
      derive excluded state sum and count n from unchanged totals, subtracting i once
      $\alpha \leftarrow 0$; $b_x^{-i}(s) \leftarrow p$
      if $2 \le n < M$ and $0 < p < 1$ **then**
        $p_s \leftarrow$ excluded state return sum divided by n
        compute v, D, d^2, α by Equations 6–7
        $b_x^{-i}(s) \leftarrow p + \alpha(p_s - p)$
      end if
    end for
    $u_{i,t} \leftarrow (R_i - b_x^{-i}(s_{i,t}))/\sigma_x$
    $\bar{u}_i \leftarrow L_i^{-1} \sum_t L_{i,t} u_{i,t}$
    $c_{i,t} \leftarrow u_{i,t} - \bar{u}_i$; $A_{i,t} \leftarrow A_i^E + c_{i,t}$
    attach $(A_i^E, c_{i,t})$ to the generated tokens of action $a_{i,t}$
  end for
end for
omit uniform groups and empty token sequences; mask observations and padding
hold all advantage coefficients fixed
for each policy optimization step **do**
  compute token ratios $\rho_{i,t,\ell}$ against the rollout policy
  sum $\ell_{\text{clip}}(\rho_{i,t,\ell}, A_i^E)$ and $\ell_{\text{clip}}(\rho_{i,t,\ell}, c_{i,t})$
  maximize their generated-token mean by Equation 12
end for

Gradient decomposition. Holding rewards, weights, and advantages fixed, define $g_t = L_t^{-1} \sum_\ell \nabla_\theta \log \pi_\theta(y_{t,\ell} \mid h_{t,\ell})$ on positive-weight turns at $\theta = \theta_{\text{old}}$, and let $\bar{g} = \sum_t w_t g_t$. With clipping inactive, substituting $u_t = \bar{u} + c_t$ and using $\sum_t w_t c_t = 0$ gives

$$\sum_t w_t u_t g_t = \bar{u}\bar{g} + \sum_t w_t c_t (g_t - \bar{g}). \tag{18}$$

Direct addition includes the common-offset term $\bar{u}\bar{g}$; redistribution removes this term while retaining the within-trajectory component. Each trajectory's contribution to the globally token-normalized loss additionally has the fixed factor $L_i / \sum_j L_j$. Summing these weighted contributions gives the local-advantage contribution to the token-normalized gradient at the rollout policy; the episode term contributes separately.

Separate-clipping identity. Write $q = \text{clip}(\rho, 1-\epsilon_-, 1+\epsilon_+)$ and $m_+ = (\rho - q)_+$, $m_- = (q - \rho)_+$. For any real coefficient a,

$$\ell_{\text{clip}}(\rho, a) = \rho a - a_+ m_+ - (-a)_+ m_-. \tag{19}$$

For real a, b, both $a_+ + b_+ - (a+b)_+$ and $(-a)_+ + (-b)_+ - (-(a+b))_+$ equal $\mathbf{1}[ab < 0] \min(|a|, |b|)$. Substituting these identities gives

$$\ell_{\text{clip}}(\rho, a+b) - \ell_{\text{clip}}(\rho, a) - \ell_{\text{clip}}(\rho, b) = \mathbf{1}[ab < 0] \min(|a|, |b|)(m_+ + m_-). \tag{20}$$

Since $m_+ + m_- = |\rho - q|$, this proves Equation 13, including zero coefficients and the clipping boundaries. For fixed coefficients, the additional penalty vanishes throughout the interval. Its derivative outside the interval points back toward the nearest boundary in gradient ascent. The complete surrogate also includes the joint-clipped term; interactions among token gradients and optimizer updates can still move ratios outside the interval.

For the lower-bound counterpart, $A_i^E = -1$, $c_{i,t} = 0.3$, and $\rho = 0.7$ give derivative $+0.3$ for separate clipping, while joint clipping is on a plateau. These derivatives are taken with respect to the scalar ratio, away from clipping boundaries.

Illustrating the clipping objective. Figure 4 isolates the effect of the objective using $A^E = 1$, $c = -0.3$, and the training bounds $[0.8, 1.28]$. Inside the interval, joint and separate clipping give the same objective and gradient. Above the upper bound, joint clipping reaches a plateau, whereas separate clipping retains the negative correction term. At $\rho = 1.4$, their derivatives with respect to the ratio are 0 and -0.3, respectively. Separate clipping therefore provides a restoring term in this disagreement case without changing the unclipped update.

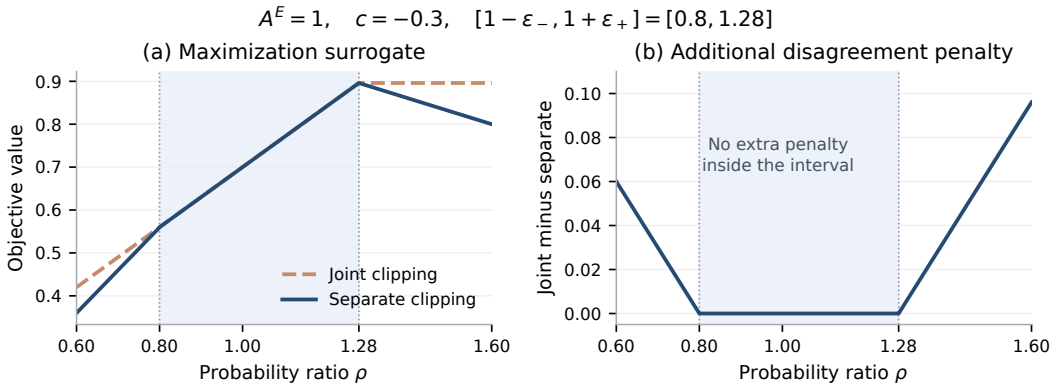

Figure 4: Exact objective functions for $A^E = 1$, $c = -0.3$, and clipping bounds $[0.8, 1.28]$. (a) Separate and joint clipping coincide inside the shaded interval and diverge outside it. (b) Their difference is the nonnegative penalty $0.3|\rho - \bar{\rho}|$, evaluated directly from Equation 13.

The second panel isolates the additional penalty. It is zero inside the interval and increases linearly with an out-of-interval excursion. Its coefficient is $\min(|A^E|, |c|)$ when the components disagree, and zero when they have the same sign. Separate clipping therefore targets out-of-interval updates whose components disagree. The curves evaluate the objective functions directly; the frequency of these events in training is a separate empirical quantity.

Scope of the analysis. The conservation identity concerns advantage coefficients, and the norm bound concerns the local correction rather than the full policy gradient. Importance ratios, clipping, and correlations between token gradients affect the optimized update. Separate clipping introduces the explicit extra term in Equation 13; it is not equivalent to joint clipping outside the interval when components disagree. A negative local correction expresses relative allocation, not an independent negative reward. Each comparison baseline and adaptive weight excludes the target rollout's outcome and visits. This removes direct self-inclusion in those statistics, but the full-group scale, trajectory-dependent centering, and clipping remain batch-dependent operations; we make no unbiasedness claim for the resulting policy update. State-relative returns describe observed continuations, not causal effects of individual edits. The finite-population variance calibrates the shrinkage rule but does not imply optimal estimation for policy-dependent state visits.

C Worked Action Comparison

Consider eight rollouts with rewards $(1, 1, 1, 1, 0, 0, 0, 0)$ and full-group standard deviation $\sigma_x = 1/2$. A state is visited by three successful rollouts and one failed rollout. For a successful visitor i, excluding it gives

$$p_x^{-i} = \tfrac{3}{7}, \quad p_x^{-i}(s) = \tfrac{2}{3}, \quad n = 3, \quad M = 7, \quad v = \tfrac{8}{147}, \quad D = \tfrac{25}{441}. \tag{21}$$

Hence $\alpha = 1/25$, $k = 72$, $b_x^{-i}(s) = 46/105$, and $u_{i,t} = 118/105$. For the failed visitor j, exclusion gives $p_x^{-j} = 4/7$, $p_x^{-j}(s) = 1$, $\alpha = 19/27$, $k = 24/19$, and $b_x^{-j}(s) = 55/63$, yielding $u_{j,t'} = -110/63$. The baselines depend on the excluded rollout, so paired local advantages obey

$$u_{i,t} - u_{j,t'} = \frac{R_i - R_j - b_x^{-i}(s) + b_x^{-j}(s)}{\sigma_x}, \qquad s_{i,t} = s_{j,t'} = s. \tag{22}$$

Concrete action at the shared state	R_i	$u_{i,t}$
Insert a null guard preserving non-null behavior	1	$+1.124$
Add a separate null-handling branch	1	$+1.124$
Update the caller to handle the empty-input case	1	$+1.124$
Replace the return expression by an unconditional zero	0	-1.746

Successful visitors receive equal coefficients here because excluding any one of them leaves identical counts and outcome sums, not because patches are grouped.

For a successful two-turn trajectory, let the first state be visited by all eight rollouts and the second be the state above. The target-excluded baselines are $3/7$ and $46/105$. With 10 and 30 generated tokens, the local signals are $(8/7, 118/105)$ and their weighted mean is $79/70$. Redistribution gives

$$(c_{i,1}, c_{i,2}) = (1/70, -1/210), \qquad (A_{i,1}, A_{i,2}) = (71/70, 209/210). \tag{23}$$

Their weighted average remains $1 = A_i^E$. Direct addition instead has mean $149/70$. For the failed visitor, the two baselines are $4/7$ and $55/63$, with $\bar{u}_j = -67/42$, giving

$$(c_{j,1}, c_{j,2}) = (19/42, -19/126), \qquad (A_{j,1}, A_{j,2}) = (-23/42, -145/126). \tag{24}$$

Its redistributed mean remains -1, whereas direct addition gives $-109/42$.

Visualizing the allocation constraint. Figure 5 compares these successful and failed continuations. Both use the same state keys and token weights, but different target-excluded baselines. Centering preserves within-trajectory differences between action coefficients while restoring the episode-level mean.

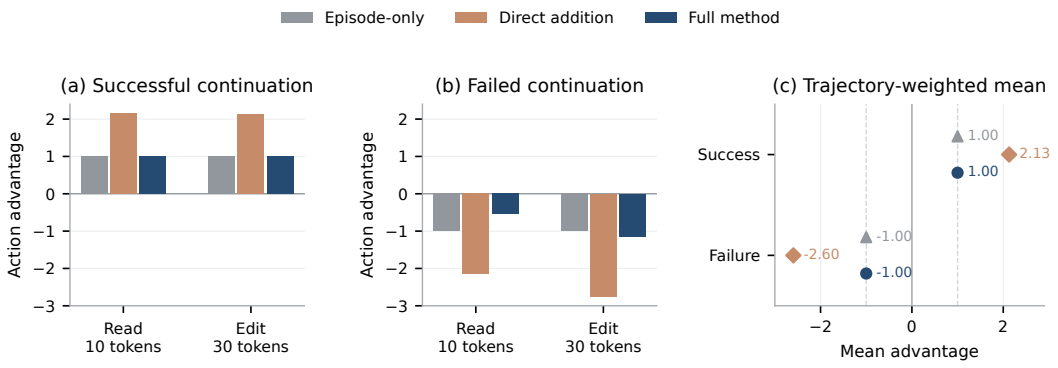

Figure 5: Advantage allocation in the leave-one-rollout-out worked example. Panels (a) and (b) show successful and failed continuations with 10 and 30 generated tokens. Panel (c) shows trajectory-weighted means: direct addition gives 2.13 and -2.60, whereas redistribution preserves 1 and -1. Values follow the exact arithmetic above.

The within-trajectory coefficient differences are $2/105$ for the successful visitor and $38/63$ for the failed visitor, unchanged by centering. A negative correction reduces a step's coefficient relative to its trajectory average; it need not make the final advantage negative or identify an incorrect action.

D Implementation Conventions

The estimator uses pre-action state keys and generated-token counts. State reconstruction belongs to the agent interface: the current file is the most recent unambiguous file target; commands with no unique target retain the previous file. The initial target is a sentinel. Edit and execution flags record whether those operations have occurred, and the failure flag records the previous command's reported execution status. These fields use only the history available before the scored action.

The implementation first accumulates task and distinct-rollout state totals. For each target rollout, its outcome is subtracted once from the task total and once from every state it visited, regardless of repeat visits. The remaining counts and sums determine the task mean, state mean, reference variance, and adaptive weight. The target's full-group episode advantage and population standard deviation remain unchanged.

With $M = N - 1$, the adaptive formula is evaluated only for $2 \le n < M$ and $0 < p < 1$, which ensures $v > 0$. Otherwise the baseline is the target-excluded task mean. This covers unseen states, single other visitors, states visited by all other rollouts, and uniform remaining task outcomes without undefined ratios. The mixture form avoids multiplying by an infinite k.

The fixed-k control uses the same excluded statistics and fallback cases, replacing only the adaptive weight with $n/(n+k)$. At $k = 0$ it uses the state mean on supported states; at $k = \infty$ all steps use the excluded task mean, so centering recovers the original episode-level advantage. Zero-token turns have zero projection and loss weight but still enter state membership. Baselines can be cached per target–state pair to handle repeated visits in linear time.

The policy objective uses $\epsilon_- = 0.2$ and $\epsilon_+ = 0.28$, with shared bounds for both terms. The default objective sums the two clipped components and averages over included generated tokens; it does not average the components themselves. Excluded tokens are skipped before ratio and coefficient arithmetic. The negative of this surrogate is the loss minimized in a gradient-based implementation. Baselines, the adaptive weights, and advantage components are detached during policy updates.

E Matched Clipping Optimization Protocol

We uniformly sample 4,096 turns without replacement from the coefficient set used in Section 4.5, using seed 20260926. Each sample retains A_i^E, $c_{i,t}$, and its generated-token count $L_{i,t}$. Corrections with magnitude at most 10^{-8} are treated as zero. We draw initial ratios from $\mathrm{Uniform}(0.6, 1.6)$, shared between objectives. Each turn uses an independent Bernoulli policy with reference probability $q_0 = 0.25$ for the selected action:

$$p(z) = \mathrm{sigmoid}(z), \qquad \rho(z) = p(z)/q_0, \qquad z_0 = \log \frac{q_0\rho_0}{1 - q_0\rho_0}. \tag{25}$$

For $F_{\mathrm{joint}}(\rho) = \ell_{\mathrm{clip}}(\rho, A^E{+}c)$ and $F_{\mathrm{sep}}(\rho) = \ell_{\mathrm{clip}}(\rho, A^E){+}\ell_{\mathrm{clip}}(\rho, c)$, we perform 60 independent gradient-ascent steps per sample with $\eta = 0.2$:

$$z_{m+1} = z_m + \eta F'(\rho(z_m))\rho(z_m)(1 - p(z_m)). \tag{26}$$

Both objectives use bounds $[0.8, 1.28]$, fixed coefficients, and no update noise. The derivative of a positive-coefficient term is zero above the upper bound, and that of a negative-coefficient term is zero below the lower bound; otherwise it equals the coefficient. At a boundary we use the unclipped-side derivative.

The upper cohort satisfies $A^E c < 0$, $A^E{+}c > 0$, and $\rho_0 > 1.28$; the lower cohort satisfies $A^E c < 0$, $A^E + c < 0$, and $\rho_0 < 0.8$. These fixed cohorts contain 236 and 137 samples, respectively. For each cohort $\mathcal{C}$, define

$$d_m = \frac{\sum_{j\in\mathcal{C}} L_j e_{j,m}}{\sum_{j\in\mathcal{C}} L_j}, \qquad e_{j,m} = \begin{cases} [\rho_{j,m} - 1.28]_+, & \text{upper cohort,} \\ [0.8 - \rho_{j,m}]_+, & \text{lower cohort.} \end{cases} \tag{27}$$

Token counts weight only the diagnostic, not gradient steps. The fixed denominator includes samples after re-entry; no checkpoint selection or smoothing is applied. The logits are independent, without shared model parameters.