CROSS-TRAJECTORY SEMANTIC CREDIT SHARING FOR MATHEMATICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Outcome-supervised methods such as GRPO assign one advantage throughout each response, even when solutions with different final outcomes contain the same local mathematical derivation. Prefix-based branching pools terminal feedback through shared generation histories, yet related local derivations can also arise in independently generated solutions. We introduce Semantic Credit Sharing (SCS), which forms local sharing groups after rollout collection. To match related content expressed in different forms, a frozen encoder represents selected segments using their original text, preceding context, and auxiliary normalized mathematical expressions. SCS clusters these segments within each question, averages the advantages of the distinct responses represented in each cluster, and applies the resulting coefficients to the matched token spans. Other token advantages remain unchanged. Sharing reuses the terminal rewards of the sampled responses. Across three independent training runs on Qwen3-1.7B-Base, SCS improves mean accuracy across five benchmarks by 1.88 percentage points over Dr.GRPO. It also outperforms a within-response assignment-shuffling control by 1.10 points, demonstrating the value of preserving credit-to-segment correspondence. On Qwen3-4B-Base, a single paired run yields a 1.49-percentage-point improvement over Dr.GRPO. These results show that local mathematical content can support feedback sharing beyond common generation histories.

1 INTRODUCTION

Reinforcement learning with verifiable rewards trains language models using the final correctness of their responses. In outcome-supervised GRPO, one group-relative advantage is applied throughout each response (Shao et al., 2024). Yet responses with different outcomes can contain the same local mathematical derivation. One solution may establish a useful relation before making a later mistake, while another reaches the same relation and finishes correctly. Their corresponding segments inherit different update coefficients from their respective terminal outcomes.

Several continuations from an intermediate prefix can provide local feedback. VinePPO uses auxiliary prefix-conditioned rollouts for Monte Carlo value estimation (Kazemnejad et al., 2025). SPO-tree estimates values from descendant outcomes and reuses the sampled tree for policy optimization (Guo et al., 2025). PPPO accumulates the terminal scores of an original response and new continuations into feedback for its retained prefix (Sun et al., 2026). Together, these approaches use multiple possible completions to provide feedback for earlier parts of a solution.

While repeated continuations provide more observations for a chosen prefix, branching also determines where the sampling budget is spent. With sixteen completed responses, a four-by-four scheme samples four prefixes independently and assigns four continuations to each, trading independent prefix draws for repeated outcome observations. More importantly, shared ancestry is only one basis for local correspondence: different prefixes can still lead to related mathematical derivations.

GraphPO connects semantic states during graph-based rollout (Zhan et al., 2026). We take a different route: we identify sharing opportunities after sampling complete responses independently. We introduce **Semantic Credit Sharing (SCS)**, which builds on post-rollout advantage sharing (Li et al., 2026) and uses local mathematical content and context to define sharing groups. Instead of using common ancestry to decide which outcomes inform a prefix, SCS uses local correspondence

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.

to decide which completed responses inform a segment's update. Figure 1 contrasts the generation structure with the credit-sharing structure.

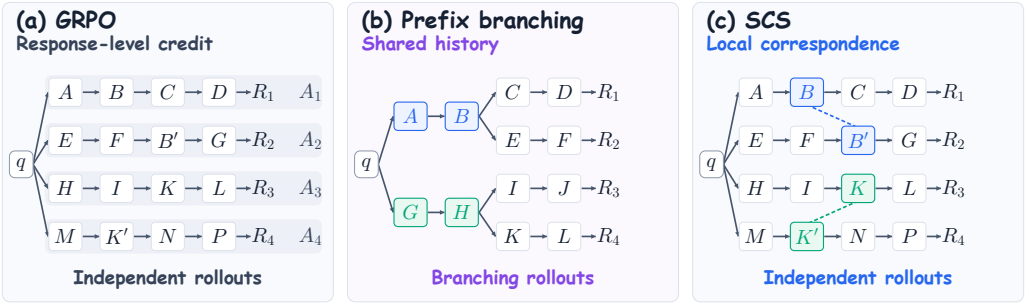

Figure 1: Generation structure and credit-sharing structure. (a) GRPO applies one response-level advantage A_i along each independent rollout. (b) Prefix branching pools terminal feedback through shared histories. (c) SCS keeps independent rollouts and links local segments after generation. Arrows denote generation, dashed links credit sharing, and colors sharing groups; R_i is the terminal reward. Primes mark matched variants. The tree depicts two prefixes with two continuations each.

Matching independently generated solutions requires a correspondence rule beyond shared prefixes. Exact text matching is simple, but misses related derivations expressed with different wording, notation, or algebraic forms. We therefore compare local segments in an embedding space using a frozen Qwen3 encoder (Zhang et al., 2025). The preceding segment supplies context, while auxiliary normalized mathematical fields give supported algebraic variants a common representation. Similarity-based clustering forms sharing groups within each question, retaining the original response text and token positions.

For each group, SCS averages the advantages of its distinct source responses and applies the shared coefficient at their original token spans. Unmatched tokens retain their response-level advantages. The sharing scores are computed from already sampled responses, without generating additional continuations. Figure 2 illustrates the calculation on a concrete mathematical example.

We evaluate SCS on Qwen3-1.7B-Base across five benchmarks and three independent training runs. It improves the benchmark average by 1.88 percentage points over Dr.GRPO and by 1.10 points over a within-response assignment control that perturbs where shared credit is applied. Both aggregate comparisons favor SCS in every paired run. A further comparison on Qwen3-4B-Base uses one paired run and improves the five-benchmark mean over Dr.GRPO by 1.49 points. Training-trace analysis examines the affected coefficients, computational cost, and recorded correspondence groups.

2 Semantic Credit Sharing

SCS uses local correspondence groups to pool terminal outcomes and assign shared advantages to original token positions.

Given question x, sample G responses $y_1, \ldots, y_G$ independently from behavior policy π_{old}, with terminal correctness rewards $R_i \in \{0, 1\}$. Our direct baseline, Dr.GRPO (Liu et al., 2025), uses

$$p_x = \frac{1}{G}\sum_{j=1}^{G} R_j, \qquad A_i^{\text{base}} = R_i - p_x, \tag{1}$$

without reward-standard-deviation normalization. Each valid generated token initially receives its response's advantage.

2.1 SHARING TERMINAL EVIDENCE

A correspondence group identifies the response outcomes used to construct a shared local update. Terminal correctness is observed once per response, so each source response contributes one vote regardless of how many matching segments it contains. The group selects which outcomes to pool; the question-level success rate supplies the common reference against which that subset is compared.

Consider a group C of corresponding segments from responses to the same question. Let I_C contain the indices of the distinct responses containing these segments. We share only when $|I_C| \geq 2$. The pooled outcome and its centered advantage are

$$q_C = \frac{1}{|I_C|} \sum_{i \in I_C} R_i, \qquad A_C = q_C - p_x = \frac{1}{|I_C|} \sum_{i \in I_C} A_i^{\text{base}}. \tag{2}$$

The cluster mean supplies correspondence-specific outcome evidence, while the question mean provides a common reference. A group therefore receives positive shared advantage when its represented responses succeed more often than the question group as a whole.

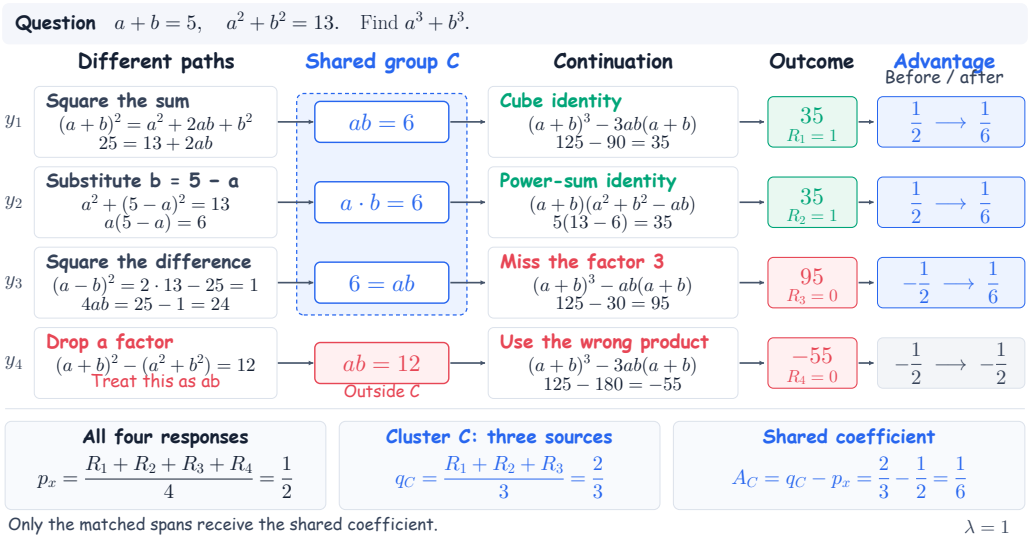

Figure 2: A worked example of semantic credit sharing. Three independent responses reach $ab = 6$, then diverge through correct or erroneous continuations. Their rewards $(1, 1, 0)$ give $q_C = 2/3$; all four responses give $p_x = 1/2$, so $A_C = 1/6$. The incorrect product in the fourth response is outside this group. The enclosure marks the shared segments; the right column shows their coefficient changes and the unchanged coefficient outside the group. This constructed example stipulates membership; Figure 4 provides actual recorded matches.

In Figure 2, the shared coefficient reflects the group's outcomes rather than any one completion. It reduces the original positive coefficients on the two successful responses and raises the coefficient on the failed response.

2.2 FINDING LOCAL CORRESPONDENCES

We use line breaks and sentence-ending punctuation outside protected mathematical regions as practical cues for local segments. Keeping recognized mathematics intact avoids splitting an expression across matching units. We then select up to $K = 4$ candidates across each response's segment sequence. Candidates must contain a complete mathematical region and have a verified mapping to original tokens. Appendix C specifies the position-selection rule, length limits, and boundary handling.

A deterministic normalizer produces canonical forms for supported expressions while preserving variable names; unsupported expressions retain their raw form. As a rule illustration, $2u + 2v = 10$ and $u + v = 5$ share the coefficient representation of $u + v - 5 = 0$. A frozen Qwen3-Embedding-0.6B encoder (Zhang et al., 2025) receives these ordered auxiliary fields alongside the candidate's

original text and preceding segment. The preceding segment provides the context in which the local expression is used.

Thresholded pairwise similarity is not transitive: two segments can each be similar to a third without being similar to one another. We therefore use complete-link clustering, requiring every pair within a group to meet the similarity threshold. Embeddings are normalized to unit length, so their dot product is cosine similarity. Within each question, we merge the most similar pair of clusters, using the minimum cross-cluster cosine similarity between their members. Merging stops below $\tau = 0.90$.

2.3 Token assignment and policy update

A cluster supplies one coefficient, but training acts on token positions. We assign that coefficient only to the original spans of its members. Let $\mathcal{C}_{i,t}$ contain the distinct eligible clusters covering token (i, t). If several clusters cover a token, their coefficients are averaged, with each cluster counted once:

$$\widetilde{A}_{i,t} = \begin{cases} (1-\lambda)A_i^{\text{base}} + \dfrac{\lambda}{|\mathcal{C}_{i,t}|} \displaystyle\sum_{C \in \mathcal{C}_{i,t}} A_C, & |\mathcal{C}_{i,t}| > 0, \\ A_i^{\text{base}}, & |\mathcal{C}_{i,t}| = 0. \end{cases} \tag{3}$$

Here, λ controls the sharing strength; we use $\lambda = 1$. Unmatched tokens retain their response-level advantages. Define $\Delta A_{i,t} = \widetilde{A}_{i,t} - A_i^{\text{base}}$. For one covering cluster, $\Delta A_{i,t} = \lambda(q_C - R_i)$. Sharing increases the advantages on matched spans of failed responses and decreases those on matched spans of successful responses whenever the covering group contains different outcomes. Groups containing different outcomes produce the adjustments examined in Section 4.3; these can reduce a magnitude, cancel a coefficient, or reverse its sign.

With these coefficients fixed, the policy-gradient direction used in our single-update configuration is

$$g_{\text{SCS}} = \frac{1}{NH} \sum_{i,t} M_{i,t} \widetilde{A}_{i,t} \nabla_\theta \log \pi_\theta(y_{i,t} \mid x_i, y_{i,<t}), \tag{4}$$

where N is the number of responses in the full batch, H is the generation limit, and M masks valid response tokens. The denominator stays fixed. SCS changes the coefficients while each log-probability gradient is evaluated in its original response context. One AdamW step uses this direction, with gradient clipping and optimizer settings given in Appendix C. That appendix also gives the clipped-surrogate implementation and its reduction to Equation 4 in the single-update branch.

Algorithm 1 isolates the advantage-construction operation. It receives an already sampled batch and returns coefficients at the original token positions; sampling precedes this call, and one policy update follows it. All encoding, grouping, and coefficient construction are detached from policy gradients.

Algorithm 1 Constructing token advantages with Semantic Credit Sharing

Input: Question-grouped rollout batch $\mathcal{R} = \{(x, y^x_{1:G}, R^x_{1:G})\}_{x\in\mathcal{B}}$;
frozen encoder f_ϕ, valid-token masks, candidate budget K, threshold τ, strength λ

Output: Token advantages $\{\widetilde{A}^x\}_{x\in\mathcal{B}}$ at the original response positions

for each question group $(x, y_{1:G}, R_{1:G})$ in $\mathcal{R}$ **do**
  $p_x \leftarrow G^{-1}\sum_{i=1}^{G} R_i$; $A_i^{\text{base}} \leftarrow R_i - p_x$ for $i = 1, \ldots, G$
  $\widetilde{A}^x_{i,t} \leftarrow A_i^{\text{base}}$ for every valid response token (i, t)
  $\mathcal{S} \leftarrow \text{ExtractCandidates}(y_{1:G}, K)$
  $z_s \leftarrow \text{Normalize}(f_\phi(\text{EncoderInput}(s)))$ for each $s \in \mathcal{S}$
  $\mathcal{G} \leftarrow \text{CompleteLinkClustering}(\mathcal{S}, z, \tau)$; $\mathcal{E} \leftarrow \varnothing$
  for each cluster $C \in \mathcal{G}$ **do**
    $I_C \leftarrow \{\text{SourceResponse}(s) : s \in C\}$
    if $|I_C| \geq 2$ **then**
      $A_C \leftarrow |I_C|^{-1}\sum_{i\in I_C} R_i - p_x$; $\mathcal{E} \leftarrow \mathcal{E} \cup \{C\}$
    end if
  end for
  for each valid response token (i, t) **do**
    $\mathcal{C}_{i,t} \leftarrow \{C \in \mathcal{E} : C \text{ covers original token } (i, t)\}$
    if $\mathcal{C}_{i,t} \neq \varnothing$ **then**
      $\widetilde{A}^x_{i,t} \leftarrow (1-\lambda)A_i^{\text{base}} + \lambda|\mathcal{C}_{i,t}|^{-1}\sum_{C\in\mathcal{C}_{i,t}} A_C$
    end if
  end for
end for
return $\text{stopgrad}(\{\widetilde{A}^x\}_{x\in\mathcal{B}})$

EXTRACTCANDIDATES applies the selection and eligibility rules in Section 2.2, retaining source-response identities and original token mappings. ENCODERINPUT combines the preceding segment, candidate text, and ordered mathematical fields; NORMALIZE gives unit-length embeddings. The superscript x in Algorithm 1 identifies each question's output block.

Design choices. $K = 4$ bounds encoding work while covering different response positions. $\lambda = 1$ replaces the coefficient on the small matched subset and leaves other tokens unchanged. We use $\tau = 0.90$ for the main comparison.

3 EXPERIMENTAL SETUP

Our experiments evaluate whether local sharing improves the underlying training rule and whether its benefit depends on where the shared advantages are applied.

Training and model selection. The main comparison starts from Qwen3-1.7B-Base; an additional comparison uses Qwen3-4B-Base. Both use a frozen MATH-derived split of 6,439 training and 1,024 development questions (Hendrycks et al., 2021). Each run uses 400 updates with 64 questions and 16 responses per question (409,600 responses), a 4,096-token response limit, temperature 1, and learning rate 2×10^{-6}. Each update uses one global minibatch and one optimization epoch. We evaluate every 40 updates and select the most accurate development checkpoint for all test sets, breaking ties toward earlier steps. Sampled response and update counts are matched.

Evaluation. We report one greedy response per question on MATH500 (500), GSM8K (1,319), OCW (272), the English text-only open-ended mathematics subset of OlympiadBench (674), and AMC83 (83), with fixed prompts, scoring, and a 4,096-token limit (Hendrycks et al., 2021; Cobbe et al., 2021; Lewkowycz et al., 2022; He et al., 2024). AMC83 pools 43 AMC2022 and 40 AMC2023 questions by correct-answer counts. We report accuracy on each benchmark and their unweighted arithmetic mean.

Baselines and scope. Dr.GRPO uses the same centered advantages, fixed-NH normalization, and zero KL penalty as SCS, isolating the effect of sharing. GRPO additionally normalizes by reward standard deviation and uses an actor KL coefficient of 0.001. DAPO w/o DS disables dynamic group filtering but retains token-mean loss, reward normalization, and overlength treatment. GRPO and DAPO thus compare broader training recipes; Appendix C gives their implementation details.

We include SCS-shuffle as an assignment control (Section 4.2). At 1.7B, all five methods use three independent runs with matched randomization settings, and we report means and sample standard deviations. At 4B, Dr.GRPO and SCS use one paired run with the same sampling and update budgets; SCS retains $K = 4$, $\tau = 0.90$, and $\lambda = 1$. Appendix A gives per-run results and selected checkpoints.

4 RESULTS

4.1 LOCAL CREDIT SHARING IMPROVES MATHEMATICAL REASONING

Table 1: Test accuracy (%). The 1.7B block reports means and sample standard deviations across three runs; the 4B block reports one run per method. The final column is the equal-weight mean of five task accuracies. All aggregates and differences are computed before rounding. Bold indicates the highest value within each model block.

Method	n	MATH	GSM8K	OCW	Olympiad	AMC83	Mean
Qwen3-1.7B-Base							
GRPO	3	68.53 ± 1.17	83.78 ± 1.37	30.15 ± 2.57	32.34 ± 0.39	30.92 ± 4.23	49.14 ± 0.63
Dr.GRPO	3	70.87 ± 0.81	83.80 ± 0.42	29.66 ± 1.53	32.54 ± 1.04	33.33 ± 2.51	50.04 ± 0.55
SCS-shuffle	3	70.67 ± 0.50	84.03 ± 0.12	29.66 ± 0.21	32.84 ± 0.45	36.95 ± 1.39	50.83 ± 0.47
DAPO w/o DS	3	70.73 ± 0.58	**84.96** ± 0.27	29.78 ± 1.10	32.39 ± 1.89	38.96 ± 1.39	51.37 ± 0.28
SCS	3	**71.33** ± 1.01	84.23 ± 0.35	**30.27** ± 1.74	**34.03** ± 0.70	**39.76** ± 2.09	**51.92** ± 0.53
Qwen3-4B-Base							
Dr.GRPO	1	80.40	93.10	**36.40**	46.14	49.40	61.09
SCS	1	**81.60**	**93.56**	36.03	**47.48**	**54.22**	**62.58**

On Qwen3-1.7B-Base, SCS achieves the highest five-benchmark mean among the evaluated methods and the highest mean on four of the five tasks (Table 1). It improves the aggregate over Dr.GRPO by 1.88 percentage points and over DAPO w/o DS by 0.56 points. Relative to Dr.GRPO, the three-run mean improves on every benchmark, with the largest gains on AMC83 and OlympiadBench. The aggregate improvement is positive in all three paired runs (Figure 3a). Against DAPO w/o DS, two paired runs favor SCS and the third slightly favors DAPO; all per-run scores are in Appendix A.

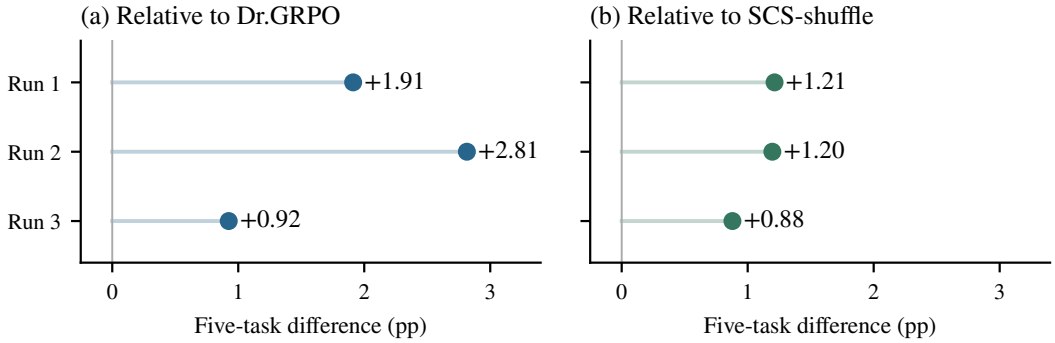

Figure 3: SCS improves over both the no-sharing baseline and randomized assignment. Paired differences in five-task accuracy on Qwen3-1.7B-Base relative to (a) Dr.GRPO and (b) SCS-shuffle. Each point denotes one paired run difference.

Extension to 4B. On Qwen3-4B-Base, SCS reaches 62.58% compared with 61.09% for Dr.GRPO, a gain of 1.49 percentage points in the single paired run (Table 1). Accuracy improves on four of the five benchmarks. This comparison extends the sharing rule to a larger base model using the same $\tau = 0.90$ configuration.

4.2 CREDIT-TO-SEGMENT ALIGNMENT MATTERS

SCS-shuffle tests whether shared credit is more useful when applied to its corresponding local segments than under a partial assignment perturbation. It uses SCS's candidate extraction, encoder, and initial clustering, then exchanges the cluster assignments of at most one eligible segment pair within each response. Because both segments come from the same response, every cluster retains its source-response set I_C, and hence its pooled outcome q_C and advantage A_C, on that fixed batch. The control holds the source evidence fixed while perturbing where credit is applied; text, token positions, and generation contexts remain unchanged. This is a partial perturbation: unexchanged assignments retain their semantic correspondence. Appendix E gives the complete rule.

At 1.7B, SCS improves the five-benchmark mean over SCS-shuffle by 1.10 percentage points, with a positive difference in every paired run (Figure 3b). The two methods change a similar fraction of token advantages—1.95% for SCS and 2.03% for SCS-shuffle on average—and incur similar training cost (Table 2). At this comparable intervention scale, applying the pooled advantages to their original matched segments yields better performance than the within-response shuffle. The partially shuffled control also retains a gain over the no-sharing baseline.

Table 2: Sharing extent and cost. Mean and sample SD across three 1.7B runs. Shared measures coverage by eligible cross-response clusters; Changed measures differences from the baseline token advantage; Sign-flipped counts strict sign reversals, excluding transitions to or from zero. Percentages use all valid generated response tokens in each full run.

Method	Shared (%)	Changed (%)	Sign-flipped (%)	GPU-hours
SCS-shuffle	11.15 ± 2.49	2.03 ± 0.45	0.96 ± 0.19	67.40 ± 5.99
SCS	11.04 ± 2.59	1.95 ± 0.51	0.95 ± 0.25	67.11 ± 5.93

4.3 SHARING BEHAVIOR AND SOURCE SELECTION

A matched segment does not necessarily receive a different advantage. When all responses represented in a cluster have the same terminal reward, its shared advantage equals each member's original advantage. Changes therefore arise from groups containing different outcomes, as characterized by $\Delta A_{i,t}$ in Section 2. Across the three 1.7B SCS runs, eligible groups cover 8.16–13.20% of generated tokens, while 1.38–2.36% receive a different advantage. This separates the extent of correspondence from the extent of the resulting intervention.

Figure 4 shows how these changes arise in recorded groups. In the first case, a recurring valid relation receives the same positive coefficient despite different final outcomes. In the second, a valid rearrangement and a sign-error variant receive the same adjustment after entering one group: both displayed coefficients rise from $-5/8$ to $-1/8$ and remain negative. The rule therefore also weakens the negative coefficient on the erroneous equation.

Predictive value of selected sources. We test whether correspondence selects informative outcome sources by withholding each target response's reward and predicting it from its cluster peers. We compare their squared error with the expected error of an equally sized subset sampled uniformly from the other responses to the same question. Across three windows of 1.7B SCS run 1, cluster peers reduce mean squared error by 0.01367–0.01865 relative to this reference, with all three 95% question-bootstrap intervals above zero. Using all 15 other responses gives the lowest prediction error. Appendix F reports the complete comparison.

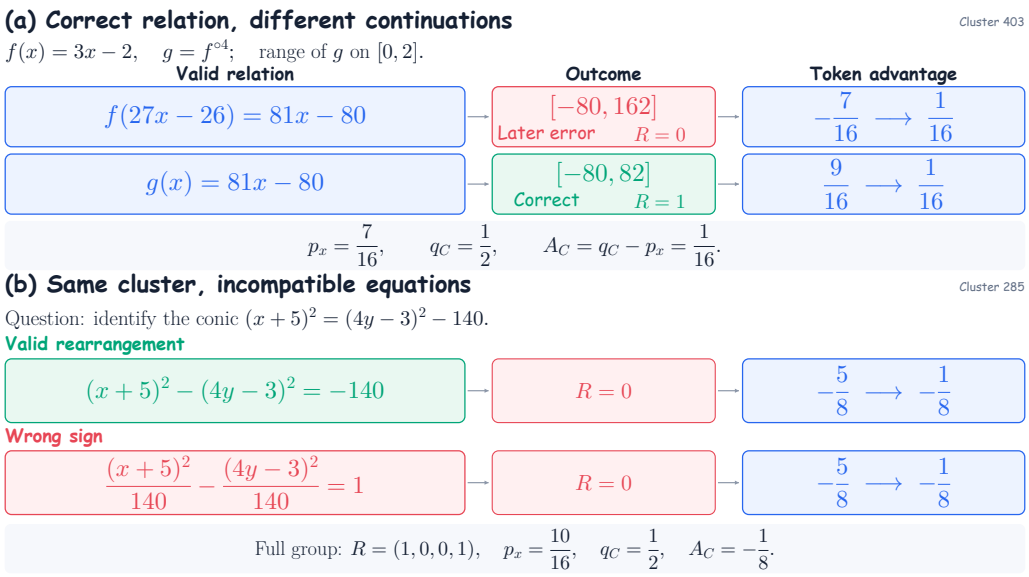

Figure 4: Advantage changes in recorded correspondence groups. (a) Both responses derive $g(x) = 81x - 80$, but one later uses the wrong upper endpoint. Its coefficient changes from $-7/16$ to $+1/16$. (b) A valid rearrangement and a sign-error variant receive the same upward adjustment. Both displayed responses have zero terminal reward; the full group has two successes and two failures. Additional case details and an example of the encoder input are provided in Appendix D.

4.4 Compute Cost

At 1.7B, SCS uses 12.94% more recorded training GPU-hours than Dr.GRPO and 28.40% more than DAPO w/o DS; its cost is close to SCS-shuffle. Total generated tokens and compute can differ under the matched response and update counts. Appendix B reports the complete costs and timing scope.

5 Related Work

Related methods differ in both how they obtain local feedback and how they define the units that share it.

Outcome-based optimization. GRPO and RLOO construct policy updates from outcomes within response groups (Shao et al., 2024; Ahmadian et al., 2024). Dr.GRPO revisits length and reward normalization, while DAPO combines changes to sampling and optimization (Liu et al., 2025; Yu et al., 2025). We use Dr.GRPO as the direct background for isolating the sharing rule. High-entropy token selection and HICRA instead emphasize particular reasoning roles or token subsets (Wang et al., 2025; 2026), whereas SCS assigns coefficients through correspondences between solutions.

Local feedback from conditional continuations. VinePPO estimates intermediate values using additional continuations from sampled prefixes (Kazemnejad et al., 2025). SPO-chain partitions trajectories using token probabilities and estimates boundary values through Monte Carlo continuations; SPO-tree uses fixed-token segments, estimates values from descendant outcomes, and reuses the sampled tree for policy optimization (Guo et al., 2025). PPPO accumulates the terminal scores of the original response and new continuations into feedback for its retained prefix (Sun et al., 2026). These approaches organize outcome evidence around a sampled prefix or segment boundary.

Feedback organized across trajectories. SALT (Li et al., 2026) shares a similar starting point: using recurring steps across trajectories to refine outcome-based feedback. It averages advantages over matched state–action–result transitions after rollout collection, uses embedding-based matching in AppWorld, and discusses mathematical-segment matching as a future extension. SCS focuses on

correspondence between local mathematical segments and their preceding context, and applies the shared coefficients to the original token spans.

SCPO aligns failed steps with a successful sibling reference and adds nonnegative semantic-progress credit to their returns (Xu et al., 2026). SCS pools all represented terminal rewards into shared advantages. GraphPO couples correspondence with generation: it constructs semantic-state equivalence classes during graph rollout, shares suffix evidence, and reallocates expansion budgets (Zhan et al., 2026). These methods differ in the signal being shared, the units being matched, and whether correspondence is established during or after rollout collection. Table 3 summarizes these mechanisms.

Table 3: Sources and units of local credit. "Post-hoc" means after rollout collection.

Method	Credit construction	Sampling / correspondence
VinePPO	Prefix values from terminal outcomes	Additional prefix continuations
SPO-chain	Differences between segment-boundary values	Monte Carlo continuation estimates
SPO-tree	Segment credit in a sampled tree	Branching during generation
SALT	Average existing step advantages	Post-hoc approximate state–action–result matching
SCPO	Progress credit added to failed-step returns	Alignment to a successful sibling reference
GraphPO	Shared suffix evidence and state scores	Graph rollout and expansion-budget reallocation
SCS	Average existing advantages; one vote per response	Post-hoc local mathematical-segment matching

6 Discussion and Conclusion

We introduced Semantic Credit Sharing (SCS), which uses local mathematical correspondences to pool outcome-based advantages across independently sampled solutions. On Qwen3-1.7B-Base, SCS improves the five-benchmark mean over Dr.GRPO in all three paired training runs. An additional paired comparison on Qwen3-4B-Base improves the five-benchmark mean by 1.49 points over Dr.GRPO. At 1.7B, the assignment control further shows that retaining the original credit-to-segment assignments yields better performance at a comparable intervention scale. Complementing these training results, leave-one-out analysis finds that cluster peers predict terminal outcomes more accurately than equal-sized random subsets in all three sampled windows of 1.7B SCS run 1. This provides evidence that the correspondence rule selects informative outcome sources at a fixed source count.

Recorded mismatches suggest a practical direction for improving correspondence: augment embedding-based retrieval with a lightweight compatibility check before sharing credit. Recently introduced decision-oriented models such as Jev (Almeida, 2026), which are designed for fast, typed probabilistic judgments rather than free-form generation, could serve as such a filter. Restricting this check to retrieved candidate pairs could improve correspondence precision while limiting additional computation.

More broadly, a rollout batch offers two complementary forms of structure: responses share a question, and local segments can correspond across otherwise different solutions. SCS uses the former to establish a common reference and the latter to organize local feedback. Its results highlight feedback organization as a design choice alongside rollout generation: solutions can be sampled independently while their terminal outcomes jointly inform updates to recurring local content.

AI Use Statement

Generative AI tools assisted with literature retrieval and comparison, research ideation and experimental design, implementation support, data organization and interpretation, figure and table preparation, translation, and manuscript drafting. The authors are responsible for the final content and claims.

REFERENCES

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024. doi: 10.18653/v1/2024.acl-long.662.

Diogo Almeida. Introducing System One Models & Jev. TypeSafe AI Blog, September 15, 2026. Accessed September 26, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in RL for large language models. In *Advances in Neural Information Processing Systems*, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024. doi: 10.18653/v1/2024.acl-long.211.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Amirhossein Kazemnejad et al. VinePPO: Refining credit assignment in RL training of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

Jiazheng Li et al. SALT: Step-level advantage assignment for long-horizon agents via trajectory graph. In *Findings of the Association for Computational Linguistics: EACL 2026*, 2026.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective, 2025. URL https://arxiv.org/abs/2503.20783.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Yiliu Sun, Zicheng Zhao, Yang Wei, Yanfang Zhang, and Chen Gong. Well begun, half done: Reinforcement learning with prefix optimization for LLM reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(39):33144–33152, 2026. doi: 10.1609/aaai.v40i39.40598. URL https://ojs.aaai.org/index.php/AAAI/article/view/40598.

Haozhe Wang et al. Emergent hierarchical reasoning in LLMs through reinforcement learning. In *International Conference on Learning Representations*, 2026.

Shenzhi Wang et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In *Advances in Neural Information Processing Systems*, 2025.

Peng Xu, Sijia Chen, Junzhuo Li, and Xuming Hu. Semantic consistency policy optimization for reinforcement learning of llm agents, 2026. URL https://arxiv.org/abs/2606.25852. Preprint.

Qiying Yu et al. DAPO: An open-source LLM reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Yuliang Zhan, Xinyu Tang, Jian Li, Dandan Zheng, Weilong Chai, Jingdong Chen, Jun Zhou, Ge Wu, Wenyue Tang, and Hao Sun. GraphPO: Graph-based policy optimization for reasoning models, 2026. URL https://arxiv.org/abs/2606.18954.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

A Complete Per-Run Results

Table 4: Complete per-run test results for Qwen3-1.7B-Base. Every run has 400 updates; the development-selected checkpoint is evaluated on every test set. Earlier checkpoints break development-score ties.

Method	Run	Step	MATH	GSM8K	OCW	Olympiad	AMC83	Mean
GRPO	1	320	67.20	84.91	30.15	32.20	31.33	49.16
Dr.GRPO	1	160	70.40	84.08	30.15	31.45	32.53	49.72
SCS-shuffle	1	280	70.20	84.00	29.41	32.34	36.14	50.42
DAPO w/o DS	1	360	70.40	84.99	29.78	31.45	39.76	51.28
SCS	1	280	70.40	83.93	28.31	34.57	40.96	51.63
GRPO	2	160	69.00	82.26	32.72	32.05	26.51	48.51
Dr.GRPO	2	320	71.80	84.00	27.94	33.53	31.33	49.72
SCS-shuffle	2	360	71.20	83.93	29.78	33.23	38.55	51.34
DAPO w/o DS	2	280	71.40	84.69	28.68	31.16	39.76	51.14
SCS	2	400	72.40	84.15	30.88	34.27	40.96	52.53
GRPO	3	400	69.40	84.15	27.57	32.79	34.94	49.77
Dr.GRPO	3	400	70.40	83.32	30.88	32.64	36.14	50.68
SCS-shuffle	3	320	70.60	84.15	29.78	32.94	36.14	50.72
DAPO w/o DS	3	400	70.40	85.22	30.88	34.57	37.35	51.68
SCS	3	320	71.20	84.61	31.62	33.23	37.35	51.60

Standard deviations use denominator $n-1$ and describe variation across the 1.7B runs. Matched run indices pair the randomization settings; the learned policies can generate different responses. The single 4B results appear in Table 1; the selected checkpoints are update 280 for Dr.GRPO and 320 for SCS, after 400 total updates in each run.

Table 5: Sensitivity to benchmark aggregation. Mean and sample SD across three 1.7B runs; Δ is SCS minus the row method (pp). Checkpoints and evaluations are unchanged. The five-task mean remains primary.

Method	Five-task mean	Without AMC83	Δ_{five}	Δ_{four}
GRPO	49.14 ± 0.63	53.70 ± 0.27	+2.78	+1.26
Dr.GRPO	50.04 ± 0.55	54.22 ± 0.17	+1.88	+0.75
SCS-shuffle	50.83 ± 0.47	54.30 ± 0.28	+1.10	+0.67
DAPO w/o DS	51.37 ± 0.28	54.47 ± 0.70	+0.56	+0.50
SCS	51.92 ± 0.53	54.96 ± 0.59	–	–

B Training Curves and Accounting

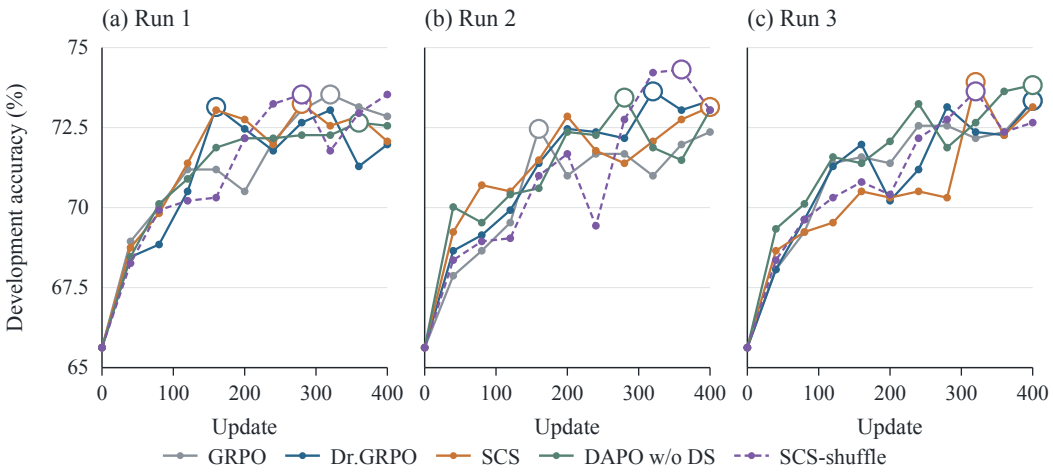

Figure 5: Development accuracy on Qwen3-1.7B-Base. Each panel includes GRPO, Dr.GRPO, SCS, DAPO w/o DS, and SCS-shuffle at all 11 evaluation checkpoints, without smoothing. Open circles mark the development-selected checkpoints.

Selected steps are 320/160/400 for GRPO, 160/320/400 for Dr.GRPO, 280/400/320 for SCS, 360/280/400 for DAPO w/o DS, and 280/360/320 for SCS-shuffle. All runs still incur 400 updates.

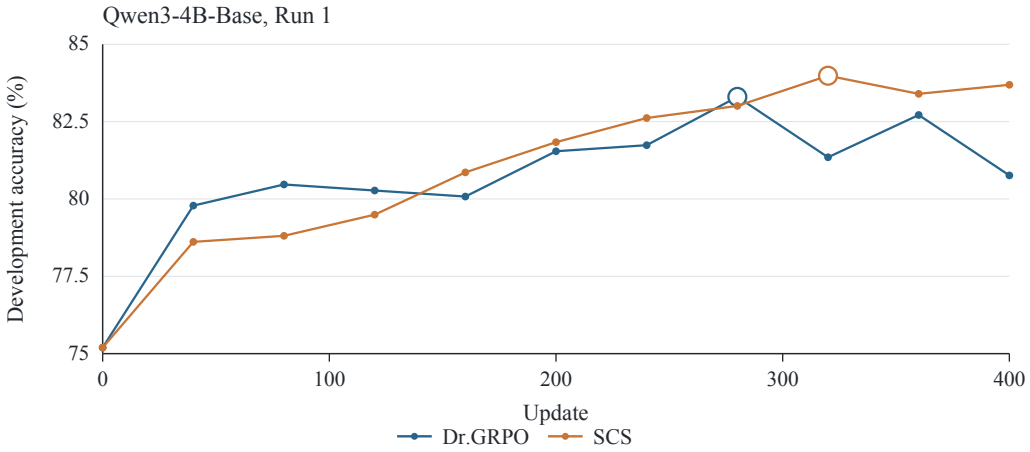

Figure 6: Development accuracy on Qwen3-4B-Base. One paired run of Dr.GRPO and SCS, evaluated on 1,024 questions every 40 updates. Lines connect the evaluation points without smoothing; open circles mark the selected checkpoints.

Table 6: Development-set correct counts for the added trajectories. Every denominator is 1,024; plotted percentages equal $100c/1{,}024$. Bold marks the selected checkpoint, choosing the earliest maximum.

	1.7B SCS-shuffle			4B	
Update	Run 1	Run 2	Run 3	Dr.GRPO	SCS
0	672	672	672	770	770
40	699	700	700	817	805
80	716	706	713	824	807
120	719	707	720	822	814
160	720	727	725	820	828
200	739	734	721	835	838
240	750	711	739	837	846
280	**753**	745	745	**853**	850
320	735	760	**754**	833	**860**
360	747	**761**	741	847	854
400	753	748	744	827	857

The correct counts in Table 6 determine every point of the added curves. For the 4B comparison, Dr.GRPO selects update 280 and SCS selects update 320.

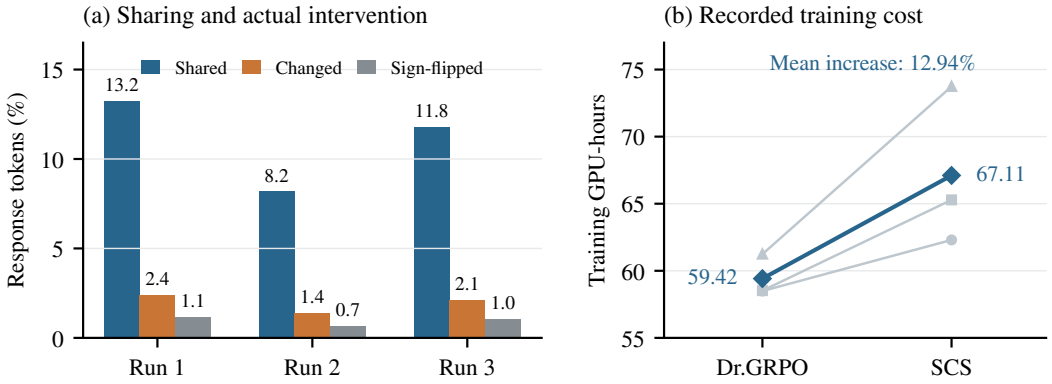

Figure 7: SCS interventions and training costs at 1.7B. Shared tokens, changed advantages, and strict sign flips are distinct quantities. The right panel compares SCS and Dr.GRPO; Table 7 includes all completed recipe runs. Costs include the full recorded training interval.

Table 7: Recorded 1.7B training GPU-hours for 400 updates, including development evaluation and checkpoint handling. Final test evaluation, preparation, failed attempts, and paused runs are excluded.

Method	Run 1	Run 2	Run 3	Mean	n
GRPO	68.01	58.75	69.09	65.28	3
Dr.GRPO	58.49	61.24	58.52	59.42	3
SCS	62.30	73.74	65.28	67.11	3
DAPO w/o DS	48.23	54.77	53.79	52.26	3
SCS-shuffle	62.55	74.10	65.55	67.40	3

Table 8: Per-run sharing statistics at 1.7B. Percentages use all valid generated response tokens in each full 400-update run.

Method	Run	Shared (%)	Changed (%)	Sign-flipped (%)
SCS	1	13.20	2.36	1.15
SCS-shuffle	1	13.35	2.41	1.09
SCS	2	8.16	1.38	0.67
SCS-shuffle	2	8.44	1.53	0.74
SCS	3	11.77	2.12	1.02
SCS-shuffle	3	11.65	2.16	1.06

C Implementation and Data Identity

Table 9: Common settings and the SCS configuration. Recipe-specific loss differences are described in the text.

Setting	Value
1.7B reference hardware	4 NVIDIA GeForce RTX 5090 GPUs
4B training hardware (Dr.GRPO / SCS)	4 NVIDIA RTX 5880 Ada GPUs
Policy	`Qwen3-1.7B-Base`
Frozen encoder	`Qwen3-4B-Base`
Training / development questions	`Qwen3-Embedding-0.6B`
Updates / questions per update / responses per question	6,439 / 1,024
Generation limit / learning rate	400 / 64 / 16
Training temperature / top-p / top-k	$4,096$ / 2×10^{-6}
SCS candidates / threshold / mixture	1 / 1 / disabled
SCS candidate token range	$K = 4$ / $\tau = 0.90$ / $\lambda = 1$
SCS / Dr.GRPO clipping / KL coefficient	1–512
Optimization epochs / global minibatches per update	0.2 (both sides) / 0
Development frequency / selection	1 / 1
Test decoding	Every 40 updates / best, earliest tie
	Greedy, 4,096-token limit

C.1 Candidate construction

Candidate selection and boundaries. For $n > K$ segments, zero-based candidates are $j_k = \lfloor (2k+1)n/(2K) \rfloor$, $k = 0, \ldots, K-1$; otherwise all segments are considered. Selection precedes eligibility checks and does not refill rejected positions. Formula-aware splitting protects recognized inline/display mathematics and supported environments. Eligible candidates have a complete mathematical region, a verified mapping to the original generated tokens, and 1–512 updateable tokens. The preceding segment supplies context only; uncertain token boundaries retain baseline advantages.

C.2 Representations and correspondence

Mathematical representations. The restricted normalizer supports rational scalar expressions, finite sets of up to 16 rational members, real intervals with rational or infinite endpoints, and single affine equations with rational coefficients. For affine equations, it moves both sides to one expression, sorts variable names, and divides coefficients and the constant by the first nonzero variable coefficient. It does not rename variables or resolve their roles. Nonlinear expressions, function calls, and equality chains outside the supported grammar remain unknown. Parsing uses a 240-character outer limit and a 0.25-second symbolic-operation limit; unsupported or timed-out objects retain raw source text. Auxiliary representations are ordered as in the source, with explicit unknown fields.

Encoder and clustering. The encoder receives three fields: previous raw segment, current raw segment, and ordered mathematical representations. Inputs are limited to 8,192 tokens, with an embedding batch size of eight per rank. Embeddings are normalized to unit length; complete-link similarity is the minimum cosine across two groups. Groups are rebuilt inside each question and update. Distinct-response voting prevents repeated occurrences from multiplying one trajectory's reward. Homogeneous-outcome clusters leave coefficients unchanged; a cluster spanning all responses has zero centered advantage.

C.3 Policy update

Optimizer. AdamW uses learning rate 2×10^{-6}, betas $(0.9, 0.999)$, weight decay 0.01, constant learning rate, no warmup, and gradient clipping at 1.

Policy-gradient implementation. The implementation uses the clipped surrogate below, with $\rho_{i,t} = \pi_\theta(y_{i,t} \mid x_i, y_{i,<t}) / \pi_{\text{old}}(y_{i,t} \mid x_i, y_{i,<t})$ evaluated in the original response context:

$$\mathcal{L}(\theta) = -\frac{1}{NH} \sum_{i,t} M_{i,t} \min\Big\{ \rho_{i,t} \widetilde{A}_{i,t},\ \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon) \widetilde{A}_{i,t} \Big\}. \tag{5}$$

SCS and Dr.GRPO use $\epsilon = 0.2$ and no KL penalty. There is one global minibatch and one optimizer step per rollout update. In this branch, the denominator's log probability is the detached current log probability. Writing $\ell_{i,t}(\theta) = \log \pi_\theta(y_{i,t} \mid x_i, y_{i,<t})$ gives $\rho_{i,t} = \exp(\ell_{i,t} - \text{stopgrad}(\ell_{i,t})) = 1$ at loss evaluation, while $\nabla_\theta \rho_{i,t} = \nabla_\theta \ell_{i,t}$. Clipping is inactive, so $-\nabla_\theta \mathcal{L} = g_{\text{SCS}}$ before gradient clipping and the AdamW parameter update. The configured dual-clip coefficient is three for SCS/Dr.GRPO and ten for DAPO w/o DS; it is also inactive on this update path. Encoding and advantage construction are detached, and their cost is included in the recorded training interval.

DAPO without dynamic sampling. Dynamic group filtering is disabled; each update retains its single batch of 64×16 sampled responses. DAPO retains reward-standard-deviation normalization, token-mean loss, lower/upper clip settings 0.2/0.28, and an enabled 1,024-token overlength buffer with penalty factor one. Actor KL and entropy coefficients are zero.

C.4 Model, data, and evaluation

Model identities. The 1.7B policy is `Qwen/Qwen3-1.7B-Base`, revision `ea980cb0a6c2ae4b936e82123acc929f1cec04c1`. The 4B policy is `Qwen/Qwen3-4B-Base`, revision `906bfd4b4dc7f14ee4320094d8b41684abff8539`. The frozen encoder is `Qwen/Qwen3-Embedding-0.6B`, revision `97b0c614be4d77ee51c0cef4e5f07c00f9eb65b3`.

Training and evaluation data. We use 6,439 training questions and 1,024 development questions from `DigitalLearningGmbH/MATH-lighteval`, revision `0530c78699ea5e8eb5530600900e1f328b48acad`. The splits are disjoint by source ID and exact problem hashes. These checks do not exclude paraphrases or pretraining contamination.

MATH500 uses `HuggingFaceH4/MATH-500`; GSM8K uses the test split of `openai/gsm8k`. OCW uses the frozen 272-question `minerva_math/test.jsonl` from Qwen2.5-Math resources. OlympiadBench uses the complete `OE_TO_maths_en_COMP` subset from its official-linked resource. AMC83 aggregates the 2022 and 2023 subsets.

Evaluation records. Accuracies are computed from correct-answer counts under fixed denominators.

D Actual Matching Cases and Encoder Inputs

Both cases are taken from update 1 of the 1.7B SCS run 2, with recorded cluster IDs 403 and 285. Membership, rewards, source segments, and token offsets are retained; exact pairwise cosines were not recorded.

Case 403. Candidates 1641 and 1645 come from response rows 24 and 714. Seven of sixteen question-group responses succeed; one of two cluster members succeeds. The first derives $81x - 80$ correctly but gives $[-80, 162]$; the second correctly evaluates $g(2) = 82$. Their advantages change from $-7/16$ and $9/16$ to $1/16$. The first encoder input follows.

```
Previous segment (context only):
Continuing this pattern, g(x) = f(f(f(f(x)))) = f(27x − 26).
Current segment:
Applying f to 27x − 26 gives
f(27x − 26) = 3(27x − 26) − 2 = 81x − 78 − 2 = 81x − 80.
Ordered mathematical representations:
orders=0,1,2:  each has normalization_status=unknown,
kind=null, canonical=null.
```

All three expressions are marked unknown by the normalizer; the encoder input retains the original segment text.

Case 285. Candidates 311, 317, 328, and 343 come from rows 723, 840, 135, and 551, with rewards $1, 0, 0, 1$. Candidate 317 gives the correct rearrangement; 328 gives the incompatible positive-one equation. Candidate 311 discusses the hyperbola form after a sign error, while 343 also describes the conic. All receive $2/4 - 10/16 = -1/8$.

E Within-Response Assignment Control

SCS-shuffle constructs SCS's candidates and initial clusters on its own on-policy batch, retaining the encoding step. It visits each original response once in a fixed order. Let $\mathcal{S}_i$ be the candidate set of response i, and $C(s)$ the current cluster of segment s. Its eligible partners are

$$\mathcal{P}_i(s) = \{u \in \mathcal{S}_i : C(u) \neq C(s),\ \max(|C(s)|, |C(u)|) > 1\}. \tag{6}$$

Here $|C|$ counts candidate segments, not distinct responses. Choose s uniformly among candidates with $\mathcal{P}_i(s) \neq \varnothing$, then choose u uniformly from $\mathcal{P}_i(s)$. Exchange their cluster assignments and move to the next response. If no such s exists, skip the response. Every proposed eligible swap is accepted, including swaps with equal cluster scores; there is no reward, similarity, or coefficient-change test and no retry. Thus each response undergoes at most one swap, and pairs of singleton clusters are excluded. This two-stage choice is not uniform sampling of all eligible unordered pairs.

Only cluster assignments are exchanged; each segment retains its source response, original token positions, and generation context. Because both segments come from the same response, every affected cluster retains both its segment count and its entire source-response set I_C. Therefore q_C, p_x, $A_C = q_C - p_x$, and eligibility $|I_C| \geq 2$ are unchanged within the batch. The ordinary SCS scoring and token-assignment rule is then applied to the modified clusters. Structural checks verify these invariants; they are not rejection conditions.

The perturbation changes which local segments receive each cluster's credit. It is partial, can leave coefficients unchanged, and does not erase all semantic associations. Token coverage and intervention rates can vary with the reassigned spans (Tables 2 and 8). These invariants compare the same batch before and after its swaps; independently trained policies need not generate identical batches or scores.

F LOCAL OUTCOME-PREDICTION DIAGNOSTIC

This analysis tests whether semantic correspondence selects more informative sources than uniform sampling at the same source count. We also compare both with all 15 other responses to the same question. Each predictor excludes the target response's reward, measuring held-out outcome prediction rather than training performance.

We use the 1.7B SCS run 1 at updates 78–81, 198–201, and 397–400, corresponding to anchors 80, 200, and 400. All eligible clusters are included: 7,050 clusters and 28,058 response–cluster pairs across 768 question groups. Eligibility depends on distinct-source membership, including clusters that do not change training advantages.

For target response i in cluster C, let $p_{x,-i}$ be the mean reward of all 15 other responses to question x, and $q_{C,-i}$ the mean reward of the $k = |I_C| - 1$ distinct other responses in C. Their squared errors are $E_i^{\text{all}} = (R_i - p_{x,-i})^2$ and $E_{i,C}^{\text{sem}} = (R_i - q_{C,-i})^2$. The local gain relative to all peers is

$$g_{i,C} = E_i^{\text{all}} - E_{i,C}^{\text{sem}}. \tag{7}$$

Let $\mathcal{C}_i$ contain the distinct eligible clusters of response i. Each cluster contributes once, regardless of repeated segments from that response. For $|\mathcal{C}_i| > 0$, the response-level local gain is

$$g_i^{\text{local}} = \frac{1}{|\mathcal{C}_i|} \sum_{C \in \mathcal{C}_i} g_{i,C}. \tag{8}$$

The random reference uniformly samples k responses without replacement from the same 15 peers. With binary rewards, its expected squared error is

$$\mathbb{E}[E_{i,C}^{\text{rand}}] = (R_i - p_{x,-i})^2 + \frac{p_{x,-i}(1 - p_{x,-i})}{k} \frac{15 - k}{14}. \tag{9}$$

The second term is the finite-population variance of the random mean and vanishes at $k = 15$. The difference $D_{i,C} = \mathbb{E}[E_{i,C}^{\text{rand}}] - E_{i,C}^{\text{sem}}$ favors semantic sources when positive. We compute the random expectation analytically from the recorded rewards.

All three errors and their differences use equal cluster weights within a response, without token-length or cluster-size weighting. For responses without an eligible cluster, all predictors use $p_{x,-i}$ and contribute zero gain. We average over all responses in each question group, over repeated appearances of the same question within a window, and then equally over distinct questions. The 95% intervals use 2,000 bootstrap resamples of these distinct questions after averaging repeated appearances; they describe question-sampling uncertainty within each window of one training run.

Table 10: Local prediction error at matched source counts. Lower MSE is better. Random-k is an exact expectation, and D is Random-k minus Semantic. Intervals resample questions within each window.

Anchor	Questions	All 15	Semantic	Random-k	D [95% interval]
80	256	0.08967	0.10414	0.12279	+0.01865 [0.01178, 0.02608]
200	254	0.06747	0.07762	0.09332	+0.01570 [0.00895, 0.02243]
400	256	0.06457	0.07594	0.08961	+0.01367 [0.00786, 0.01986]

Semantic sources have lower error than same-count random sources in all three windows, with positive difference intervals. The all-15 predictor has the lowest error. Thus semantic correspondence provides useful source selection at a fixed count, while pooling all available peers remains more accurate for this terminal-outcome prediction task. This comparison does not measure local mathematical correctness or the causal effect of sharing on training performance.