How Can Every Rollout Count for Skill Seekers? Compass for Skill Proposers

Anonymous authors
Paper under double-blind review

Abstract

Textual-program optimization must balance proposing and evaluating candidates under a fixed rollout budget. Evaluating each candidate more extensively leaves fewer rollouts for proposing new candidates. We introduce Compass, which turns accumulated evaluation results into relative credit to guide subsequent proposals. On each shared instance, attaining the highest observed score earns more credit when a smaller fraction of the other candidates attain it. Compass averages these per-instance credits to select candidates for further revision and updates the credits as new results arrive. For revision, it assigns batches of instances using the selected candidates' score gaps to the best score recorded among them on each instance. Each new candidate is evaluated on instances that were not used to generate it or its ancestors. Across six benchmarks, Compass achieves the highest average score among the compared methods for both Qwen3-8B and GPT-5.4-nano. On Qwen3-8B, it improves over GEPA by 10.3 pp on average and 31.3 pp on AIME-2025. Ablations on Qwen3-8B show that both relative credit and matching improve the average score across the six benchmarks.

1 Introduction

Recent work has shown that textual-program optimization can outperform reinforcement learning with substantially fewer rollouts while keeping model weights fixed (Agrawal et al., 2025). This offers an efficient way to adapt language models when task executions are expensive. Evaluation cost has motivated methods for selecting candidates under fixed budgets and for improving performance per evaluation (Shi et al., 2024, Kang & Yoganarasimhan, 2026). The goal is to maximize the expected benchmark utility of the final program under a limited rollout budget shared by candidate generation and evaluation.

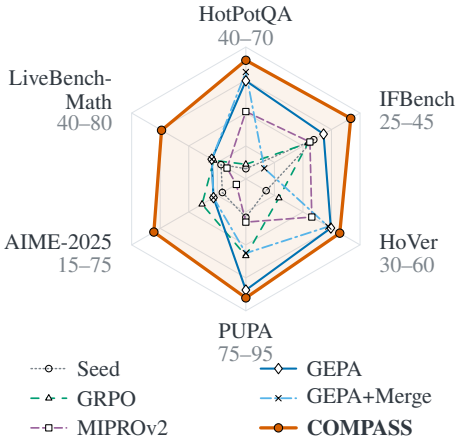

Figure 1. **COMPASS leads on all six tasks.** Qwen3-8B scores (%) use labeled center-to-rim ranges (Table 2).

Evaluating candidates more extensively leaves fewer rollouts for proposing new ones. The challenge is to use limited evaluations both to judge candidates and to guide further proposals. Rollouts provide traces for proposing revisions (Cheng et al., 2024) and scores for comparing candidates. The recorded scores can guide which candidates to revise and which instances to use without additional evaluations for these decisions. Comparisons on shared instances show where a candidate outperforms its peers or falls behind. New results can change these comparisons even when earlier scores remain fixed.

Compass (**C**redit-Aware **O**ptimization with **M**ini-batch **P**roposal and **A**dmission for **S**kill **S**earch) selects the candidates with the highest relative credit for further revision. On each shared instance, attaining the highest score earns more per-instance credit when a smaller fraction of the other can-

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

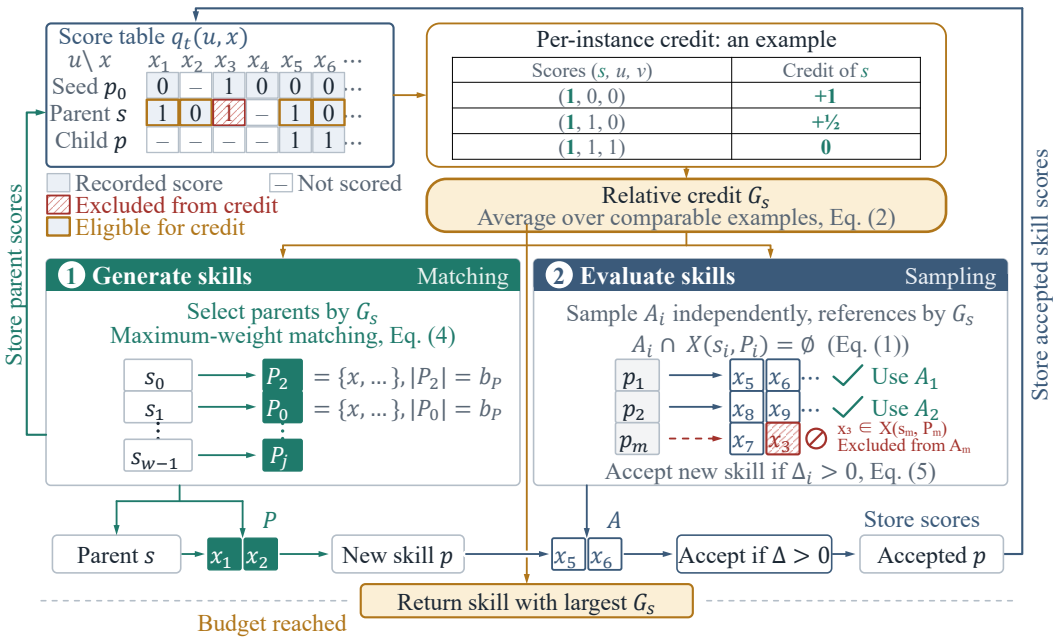

Figure 2. **Candidate comparisons guide the next search wave in COMPASS.** The example illustrates three binary-score configurations for three eligible candidates: s receives less credit as more peers also succeed. Averaging per-instance credits r_{sx} over comparable instances gives G_s for parent selection, admission references, and final selection. Maximum-weight matching assigns proposal batches using H. A child proposed from s on P is evaluated on an independently sampled batch A outside $X(s, P)$. Red hatching marks lineage-excluded observations.

didates attain it. COMPASS averages these per-instance credits and updates relative credit as new results arrive. Proposal matching assigns batches that address each selected candidate's weaknesses, using recorded score gaps between candidates on the same instances. Several candidates may favor the same batch, so the batches are assigned jointly by maximum-weight matching.

Each proposed candidate is evaluated on a separate admission batch to decide whether it enters the candidate pool. This batch excludes the instances used to generate the candidate or its ancestors, because the candidate can inherit changes made using those instances. Accepted candidates join the pool, and their admission results inform subsequent comparisons and proposal matching. Figure 2 follows this process from candidate generation to final selection.

We evaluate COMPASS on six benchmarks with Qwen3-8B and GPT-5.4-nano. COMPASS achieves the highest six-task aggregate among the compared methods for both models. On Qwen3-8B, the programs selected by COMPASS for each task reach 63.33% on AIME-2025, 43.37% on IFBench, 69.53% on LiveBench-Math, and 54.67% on HoVer. These scores exceed GEPA by 31.33, 4.76, 17.58, and 2.34 pp, respectively. Qwen3-8B ablations show that relative credit and proposal matching both improve the six-task aggregate.

This paper makes three contributions.

1. We introduce COMPASS for textual-program optimization under a fixed rollout budget. It combines relative credit for candidate selection, maximum-weight proposal matching, and paired admission outside each candidate's proposal history.
2. We analyze paired admission and characterize when relative credit preserves sample-mean rankings. We also derive a finite-pool regret bound for candidates selected by relative credit.
3. Across six benchmarks, COMPASS achieves the highest aggregate for both Qwen3-8B and GPT-5.4-nano. Ablations on Qwen3-8B show aggregate gains from both relative credit and proposal matching.

2 PROBLEM FORMULATION

2.1 BUDGETED TEXTUAL-PROGRAM OPTIMIZATION

Given a seed textual program p_0, a benchmark evaluator, and a rollout budget B, COMPASS generates and evaluates candidates, then returns one final program. Each attempt uses a proposal batch P of b_P instances to generate a candidate and a separate admission batch A of b_A instances to evaluate it. Proposal generation and admission evaluation share the rollout budget. Proposal attempts are grouped into waves. COMPASS starts a new wave while the cumulative proposal and evaluation cost remains below B, then completes that wave before stopping.

Let π denote a complete run of proposal and admission decisions, and let $\Pi(B, b_P, b_A)$ contain such runs with budget B and batch sizes b_P and b_A. Let $p_{\text{out}}(\pi)$ denote the final program returned by run π. Let $R(p)$ denote the expected benchmark utility of program p, and let $R^\star$ denote the best utility reachable under the search model. The budgeted optimization problem is

$$\pi^\star \in \arg\min_{\pi \in \Pi(B, b_P, b_A)} \mathbb{E}[R^\star - R(p_{\text{out}}(\pi))] \,.$$

After selection, the returned program is evaluated on an independent frozen test.

2.2 SEPARATING PROPOSAL AND ADMISSION EVIDENCE

Admission excludes the proposal batches used to generate a candidate or its ancestors, because the candidate can inherit changes made using those batches.

Each accepted candidate after p_0 records its parent and proposal batch. Let P_a^{birth} be the proposal batch that created candidate a, with $P_0^{\text{birth}} = \varnothing$, and let $\text{Anc}(s)$ contain all ancestors of s. The proposal history of candidate s is

$$\mathcal{B}_s = \bigcup_{a \in \text{Anc}(s) \cup \{s\}} P_a^{\text{birth}}.$$

A child proposed from parent s on batch P inherits $\mathcal{B}_s$ and adds P, giving

$$X(s, P) = \mathcal{B}_s \cup P. \tag{1}$$

Its admission batch satisfies $A \cap X(s, P) = \varnothing$. If accepted, the child records $X(s, P)$ as its proposal history.

2.3 REUSING EXECUTION RESULTS

COMPASS uses earlier execution results to select proposal parents and assign proposal batches. Executing program p_s on an optimization instance x yields a score in $[0, 1]$, an output, and an execution trace. At step t, the cache retains one record for each observed candidate–instance pair. The score $q_t(s, x)$ is the highest valid, finite score observed for that pair, and unevaluated pairs remain missing.

Relative credit also uses only observations outside a candidate's proposal history. The instances with eligible observations form the set

$$\Omega_t(s) = \{x \mid q_t(s, x) \text{ exists and } x \notin \mathcal{B}_s\}.$$

Relative credit guides parent selection. Proposal matching uses cached scores, including those on instances in a parent's proposal history, to identify where that parent falls behind other candidates.

3 CANDIDATE SELECTION, MATCHING, AND ADMISSION IN COMPASS

3.1 RELATIVE CREDIT FROM SHARED OBSERVATIONS

On a shared instance, a candidate receives more per-instance credit for attaining the highest score when a smaller fraction of the other candidates attain it. For example, if three candidates all succeed or all fail on a binary-scored instance, each receives zero credit. A single success gives credits $(1, -\frac{1}{2}, -\frac{1}{2})$. Two successes give $(\frac{1}{2}, \frac{1}{2}, -1)$, with successful candidates listed first.

Let $\mathcal{F}_t(x)$ be the *frontier*. It contains the candidates with the highest cached score among those for which $x \in \Omega_t(s)$, including all ties. In the all-failure example above, every candidate is on the frontier and each receives zero per-instance credit. Define the comparison set, its size, frontier membership, and the frontier size by

$$\mathcal{C}_t(x) = \{u \mid x \in \Omega_t(u)\}, \qquad N_x = |\mathcal{C}_t(x)|, \qquad y_{sx} = \mathbf{1}\{s \in \mathcal{F}_t(x)\}, \qquad K_x = \sum_{u \in \mathcal{C}_t(x)} y_{ux}.$$

Per-instance credit subtracts the fraction of other candidates on the frontier from the candidate's own frontier indicator. When $N_x \geq 2$, this gives

$$r_{sx} = y_{sx} - \frac{K_x - y_{sx}}{N_x - 1} = \frac{N_x y_{sx} - K_x}{N_x - 1}.$$

COMPASS averages per-instance credit over instances with at least two eligible observations to obtain relative credit G_s.

$$\Omega_t^{\mathrm{cmp}}(s) = \{x \in \Omega_t(s) \mid N_x \geq 2\}, \quad E_s^{\mathrm{cmp}} = |\Omega_t^{\mathrm{cmp}}(s)|,$$
$$R_s = \sum_{x \in \Omega_t^{\mathrm{cmp}}(s)} r_{sx}, \qquad G_s = \frac{R_s}{E_s^{\mathrm{cmp}}}. \tag{2}$$

An observation with $N_x = 1$ remains in the cache and begins contributing relative credit when another candidate has an eligible observation on that instance. The relative credit G_s is undefined when $E_s^{\mathrm{cmp}} = 0$.

New comparisons can change relative credit even when existing cached scores do not change. Whenever a cache update changes an observation or a frontier, COMPASS recomputes N_x, K_x, all affected per-instance credits, and the resulting G_s values.

3.2 A CREDIT-RANKED ACTIVE SET

At each wave, COMPASS ranks all accepted candidates with defined relative credit to select proposal parents. The active set $\mathcal{A}_t$ contains the highest-credit candidates up to the active-rank cutoff ρ, including every candidate tied at the cutoff. Parents and descendants are considered together. If fewer than ρ candidates have defined credit, all of them remain active. The tie rule can therefore leave more than ρ active programs, and a parent and its descendant can remain active at the same time. If no candidate has defined credit, COMPASS uses the seed.

Proposal parents are selected from the active set in decreasing G_s order, with candidate index breaking exact ties. The order is traversed cyclically across waves, and each proposal slot defines one revision attempt.

3.3 PROPOSAL MATCHING AND ADMISSION

COMPASS assigns proposal batches to address each parent's weaknesses. Higher cached scores from other active candidates on the same instances indicate where to focus revision. The matching weight $H(s, P)$ averages s's gap to the highest cached score among active candidates, computed separately for each instance in P.

$$H(s,P) = \frac{1}{|P|} \sum_{x \in P} \begin{cases} \left[\max_{\substack{r \in \mathcal{A}_t \\ q_t(r,x) \text{ finite}}} q_t(r,x) - q_t(s,x) \right]_+, & q_t(s,x) \text{ finite}, \\ 0, & \text{otherwise}. \end{cases} \tag{3}$$

Here $[v]_+ = \max\{v, 0\}$. Missing parent observations contribute zero to this matching weight and remain missing in the cache.

COMPASS draws w proposal batches for the w proposal slots. Several slots may favor the same batch, so COMPASS assigns the batches jointly. For example, two proposal slots may have the following matching weights.

	P_1	P_2
a	0.40	0.30
b	0.35	0.05

Both prefer P_1. Assigning P_1 to a leaves P_2 for b, for a total weight of 0.45. Assigning P_2 to a and P_1 to b gives 0.65.

COMPASS therefore solves the maximum-weight matching problem over all proposal slots and batches. Let $\mathcal{I}_w = \{0, \ldots, w-1\}$, with z_i denoting the parent of slot i and P_j the j-th proposal batch.

$$\begin{aligned} Y^\star \in \underset{Y \in \{0,1\}^{w \times w}}{\arg\max} \quad & \sum_{i,j \in \mathcal{I}_w} H(z_i, P_j) y_{ij} \\ \text{s.t.} \quad & \sum_{j \in \mathcal{I}_w} y_{ij} = 1 \ \ \forall i, \qquad \sum_{i \in \mathcal{I}_w} y_{ij} = 1 \ \ \forall j. \end{aligned} \tag{4}$$

Distinct slots receive distinct proposal batches even when they have the same parent.

Each assigned parent is evaluated on its proposal batch, updating its cached observations and providing execution traces for revision. Proposal tasks that proceed to child generation receive an admission batch and fixed references for comparison.

For such a task with parent s_i and proposal batch P_i, the evaluator independently samples its admission batch A_i outside $X(s_i, P_i)$. On each admission instance x, it selects a current frontier candidate with the largest defined relative credit as reference, using the parent if no frontier candidate has defined credit. Denote this reference by $r_i(x)$. Missing reference observations are evaluated and stored in the cache. The evaluator fixes the admission batch, references, and reference scores before generation and keeps them hidden from the generator.

Once the generator fixes child p_i, the evaluator scores it on A_i. The child is admitted when its total score exceeds that of the fixed references. Let $q(p_i, x)$ be the child's score and $q(r_i(x), x)$ the fixed reference score on the same instance. The admission rule is

$$\Delta_i = \sum_{x \in A_i} \left[q(p_i, x) - q(r_i(x), x) \right], \qquad \text{admit}(p_i) \iff \Delta_i > 0. \tag{5}$$

An accepted child enters the candidate pool with its proposal history, and its admission observations enter the cache. COMPASS then recomputes the affected G_s values and updates the active set and queue. At the budget boundary, COMPASS returns the program with the largest defined G_s, with earliest index breaking an exact tie. Expanded execution details appear in Appendix B.

4 Analysis of admission, credit, and final selection

4.1 Proposal history and paired admission

A candidate that succeeds on every instance in its proposal batch and fails elsewhere will score perfectly when evaluated on that batch again. Its expected benchmark utility can still be arbitrarily small.

Proposition 1 (Reusing proposal examples can overstate performance)**.** *For every admission size $n \geq 1$ and $\epsilon > 0$, there is a task distribution and a proposal rule that observes n examples, constructs a program, and scores one when those examples are reused for evaluation, although the program's population utility is at most ϵ.*

The construction in Appendix C uses memorization. A descendant can inherit this behavior, which explains why admission excludes the full proposal history.

Admission in Equation (5) uses the total paired score difference between a fixed child and its references. Averaging these differences over the admission batch preserves the sign and hence the

admission decision. Theorem 1 bounds the deviation of this mean paired difference from the average conditional mean. At a fixed confidence level, the bound decreases with admission batch size.

4.2 RELATIVE CREDIT

Paired admission decides whether a child enters the candidate pool. Relative credit ranks the accepted candidates for further revision and final selection. Per-instance credit rewards candidates that reach the frontier when others do not. The positive credit they receive is exactly balanced by the negative credit assigned to the remaining candidates.

Proposition 2 (Instance-wise credit conservation)**.** *For every instance x with $N_x \geq 2$, the per-instance credits defined immediately before Equation* (2) *satisfy*

$$\sum_{s \in \mathcal{C}_t(x)} r_{sx} = 0.$$

With complete shared binary observations and one score per candidate–instance pair, relative credit preserves the sample-mean ranking and ties. On each instance, per-instance credit equals the candidate's score minus the mean score of the other candidates. Averaging over the shared instances gives its sample mean minus the average sample mean of the other candidates. Equivalently, every candidate's sample mean undergoes the same centering and positive scaling (Equation (11)).

4.3 SEARCH AND FINAL SELECTION

A search can miss a strong program or discover one and return a weaker candidate. The search gap and selection gap separate these two sources of regret. Let $\mathcal{P}_B$ be the candidates available when search stops and let $\hat{p}_B$ be the program selected by the largest defined G_s. Its regret has the exact decomposition

$$R^\star - R(\hat{p}_B) = \underbrace{R^\star - \max_{p \in \mathcal{P}_B} R(p)}_{\text{search gap}} + \underbrace{\max_{p \in \mathcal{P}_B} R(p) - R(\hat{p}_B)}_{\text{selection gap}}. \tag{6}$$

Relative credit and proposal matching guide candidate generation, while paired admission determines which candidates enter the pool. The resulting pool determines the search gap, and the final choice within it determines the selection gap.

To bound the selection gap, consider M candidates, each fixed before receiving n conditionally independent and identically distributed rewards under a common benchmark distribution, with mean $R(p_i)$. Let $\widehat{R}_n(p_i)$ be the arithmetic mean of these rewards and let $\widehat{p}_G$ maximize defined relative credit in this pool.

The proof of Proposition 3 bounds all sample-mean errors simultaneously. The error bounds for the candidate with the highest expected utility and the selected candidate give the selection-noise term. For selection by relative credit, the bound also includes the difference between the highest sample mean and that of $\widehat{p}_G$,

$$\xi_G = \max_{i \leq M} \widehat{R}_n(p_i) - \widehat{R}_n(\widehat{p}_G). \tag{7}$$

Under these evaluation conditions, Corollary 1 gives, with probability at least $1 - \delta$,

$$R^\star - R(\widehat{p}_G) \leq \underbrace{R^\star - \max_{i \leq M} R(p_i)}_{\text{search gap}} + \underbrace{2\sqrt{\frac{\log(2M/\delta)}{2n}} + \xi_G}_{\text{selection gap bound}}. \tag{8}$$

With M and the confidence level fixed, increasing n reduces the selection-noise term. With the shared binary observations considered above, $\xi_G = 0$.

5 EXPERIMENTS

We evaluate whether COMPASS improves held-out benchmark utility under fixed rollout budgets across models and tasks. We also examine the contributions of relative credit and proposal matching.

5.1 TASKS, MODELS, AND EVALUATION

We evaluate COMPASS on six benchmarks with Qwen3-8B (Yang et al., 2025) and GPT-5.4-nano. The tasks, test sizes, and rollout budgets are summarized in Table 1. For each task, we use the rollout budget reported for GEPA (Agrawal et al., 2025).

Table 1. **Experimental setup.**

Benchmark	Task	Test instances	Budget
HotPotQA (Yang et al., 2018)	Question answering	300	6,871
IFBench (Pyatkin et al., 2025)	Instruction following	294	3,593
HoVer (Jiang et al., 2020)	Multi-hop fact verification	300	7,051
PUPA (Li et al., 2025)	Privacy-conscious delegation	221	2,426
AIME (Balunović et al., 2025)	Mathematical reasoning	150	1,839
LiveBench (White et al., 2025)	Mathematical reasoning	126	1,839

HoVer uses 150 training and 300 validation instances. The AIME test includes all 150 problems, and LiveBench uses its frozen 126-problem test.

In the GPT-5.4-nano experiments, GPT-5.4-nano performs both task execution and reflection. PUPA uses its official GPT-4.1-mini judge to assess response quality and leakage of personally identifiable information (PII) throughout optimization and testing.

We use each benchmark's official data and evaluation procedure, including its scoring and aggregation rules. Candidates are selected using optimization data, and the selected program is then evaluated once on the held-out test split.

5.2 COMPARISON ACROSS SIX TASKS WITH QWEN3-8B

Table 2 compares Qwen3-8B evaluations on all six shared tasks. Seed, GRPO, MIPROv2, GEPA, and GEPA+Merge reuse published GEPA-study task scores. The COMPASS row reports the test scores of the program selected for each task. Appendix D lists the selected candidates.

We compare two COMPASS ablations. *w/o Credit & Matching* selects top-ranked candidates using frontier frequency over legal exposures (F/E) and omits proposal matching. *w/o Matching* retains relative credit ranking and omits proposal matching.

Table 2. **Qwen3-8B evaluation across all shared tasks (%).** Each cell reports the official task aggregate. Aggregate is the unweighted mean over the six tasks, and Δ Seed is the aggregate improvement in pp. Both are computed from the displayed task scores before rounding to two decimal places. The first five rows reuse GEPA-study task scores. Ablation names indicate the removed components. Credit denotes G_s ranking, and Matching denotes proposal matching. TextGrad and Trace were not reported for Qwen3-8B.

Method	HotPotQA	IFBench	HoVer	PUPA	AIME	LiveBench	Aggregate	Δ Seed
Seed	42.33	36.90	35.33	80.82	27.33	48.70	45.24	–
GRPO	43.33	35.88	38.67	86.66	38.00	51.26	48.97	+3.73
MIPROv2	55.33	36.22	47.33	81.55	20.00	46.60	47.84	+2.60
GEPA	62.33	38.61	52.33	91.85	32.00	51.95	54.85	+9.61
GEPA+Merge	64.33	28.23	51.67	86.26	32.00	51.95	52.41	+7.17
w/o Credit & Matching	64.67	40.31	51.67	86.91	57.33	66.67	61.26	+16.03
w/o Matching	65.33	40.82	50.67	88.85	60.67	67.68	62.34	+17.10
COMPASS	**67.00**	**43.37**	**54.67**	**93.04**	**63.33**	**69.53**	**65.16**	**+19.92**

COMPASS has the highest six-task aggregate and the highest score on all six tasks, including PUPA, where it exceeds GEPA by 1.19 pp.

The ablations demonstrate the contributions of both relative credit and proposal matching. Replacing frontier-frequency ranking with relative credit raises the aggregate score from 61.26% to 62.34%, improving five of the six tasks. The largest gain occurs on AIME at 3.34 pp, followed by PUPA at 1.94 pp. Adding proposal matching further increases the aggregate to 65.16%, a gain of 2.82 pp over credit ranking alone. This improvement extends across all six tasks, with gains of 4.19 pp on PUPA and 4.00 pp on HoVer. The full method therefore exceeds the variant without either component by 3.90 pp on average. These results show that both candidate ranking and the allocation of proposal evidence contribute to the performance of COMPASS.

Program changes during search. Figure 3 traces how the selected AIME program develops from the seed. The revisions introduce a solving checklist, checks on substitutions, and explicit output requirements. They also remove earlier instructions. Candidate 5 drops the integer-output requirement introduced by candidate 3. Candidate 9 reintroduces it with examples, and candidate 11 keeps the requirement while removing the two conversion examples and the list of incorrect outputs.

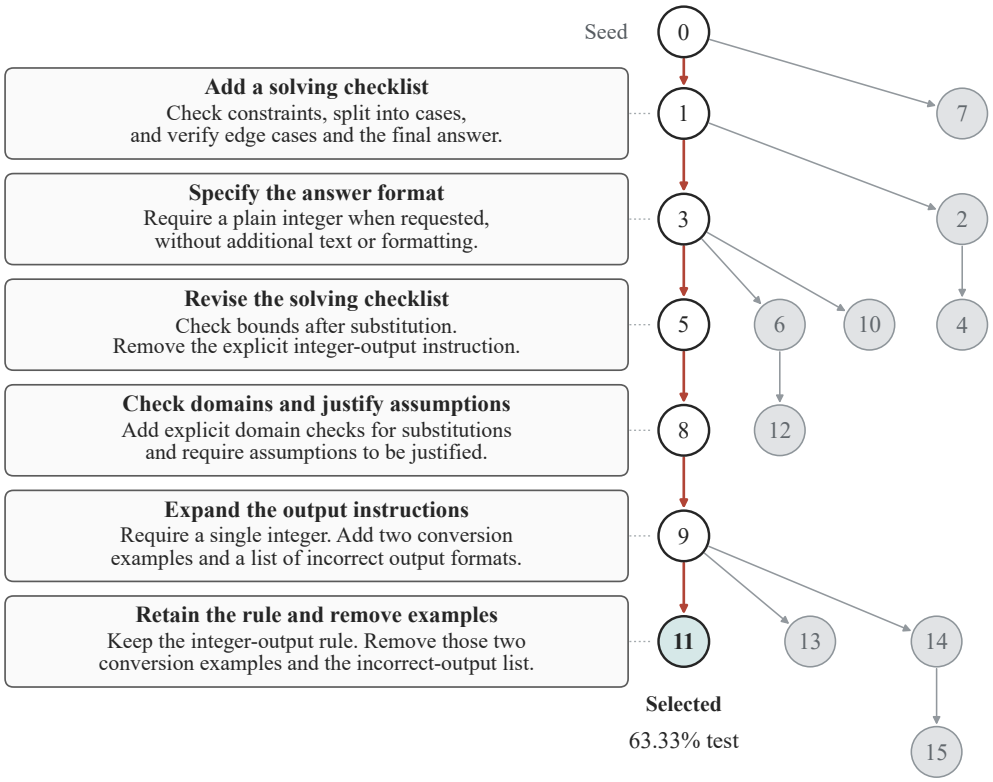

Figure 3. **Program evolution on AIME with Qwen3-8B.** Red arrows trace the path from the seed to the selected program, which achieves a test score of 63.33%. Gray nodes show other candidates. Annotations summarize changes to general instructions at each step.

5.3 SAME-MODEL COMPARISON WITH GPT-5.4-NANO

With GPT-5.4-nano, COMPASS has the highest six-task aggregate and the highest score on four tasks (Table 3). MIPROv2 is higher on HotPotQA, and GEPA is higher on HoVer.

Table 3. **GPT-5.4-nano evaluation across all shared tasks (%).** Each cell reports the official task aggregate. Aggregate is the unweighted mean over the six tasks, and Δ Seed is the aggregate improvement in pp. The table includes methods with completed same-model evaluations.

Method	HotPotQA	IFBench	HoVer	PUPA	AIME	LiveBench	Aggregate	Δ Seed
Seed	31.33	48.98	45.33	74.86	15.33	35.09	41.82	–
MIPROv2	**61.67**	47.79	53.00	75.45	30.00	43.66	51.93	+10.11
GEPA	57.33	48.30	**61.33**	81.09	34.00	35.03	52.85	+11.03
COMPASS	60.00	**51.36**	60.33	**84.14**	**42.00**	**45.03**	**57.14**	**+15.32**

6 RELATED WORK

Reflective textual-program optimization. Reflexion and Self-Refine use language feedback to revise behavior, while TextGrad, Trace/OptoPrime, MIPROv2, and GEPA optimize textual components from execution evidence (Shinn et al., 2023, Madaan et al., 2023, Yuksekgonul et al., 2025, Cheng et al., 2024, Opsahl-Ong et al., 2024, Agrawal et al., 2025). COMPASS uses the resulting scores to choose which candidates to revise next and which batches to assign them.

Evolution, illumination, and skill search. Evolutionary prompt and skill methods maintain multiple candidates to avoid a single greedy trajectory (Agrawal et al., 2025, Mi et al., 2026, Moll et al., 2026, Zhang et al., 2026, Wang et al., 2026b). COMPASS keeps parents and descendants in one credit-ranked candidate pool. Its active set contains the highest-credit candidates, including all ties at the cutoff.

Adaptive evaluation and budget allocation. TRIPLE, POES, and Bayesian-Agent allocate evaluation or retrieval effort under limited budgets (Shi et al., 2024, Ma et al., 2026, Wu et al., 2026). TextBO and APEX optimize evaluation efficiency and development-example choice (Kang & Yoganarasimhan, 2026, Wang et al., 2026a). COMPASS allocates proposal attempts by relative credit and assigns their batches jointly by maximum-weight matching. Both decisions use the existing execution records.

Generalization after adaptive search. Repeated adaptive use of optimization data can widen the gap between selected validation performance and held-out test performance (Dwork et al., 2015). GEPA addresses this empirically through reusable rules and branch diversity (Agrawal et al., 2025). COMPASS carries proposal history through the lineage. Examples used to create an ancestor remain excluded from a descendant's admission. Held-out benchmark tests evaluate the selected program.

7 CONCLUSION

COMPASS uses execution results to decide which candidates to develop and which batches they should work on. Proposal history determines the eligible comparisons, and new results update the relative credit of candidates already in the pool. Relative credit selects proposal parents, and maximum-weight matching assigns their batches. Paired admission evaluates the resulting revisions on separate examples. Under the reported budgets, COMPASS achieves the highest six-task aggregate for both Qwen3-8B and GPT-5.4-nano, and leads on all six Qwen3-8B tasks.

AI USE STATEMENT

Language models assisted with literature review, editing, code, and figure design. The authors verified every citation, algorithm, and artifact. Model output was neither experimental evidence nor used to choose test candidates.

Ethics statement

COMPASS can reduce the number of expensive environment executions needed to discover a useful textual program and can make search decisions easier to audit. The same efficiency can also optimize undesirable procedures or exploit imperfect rewards. Responsible deployment calls for secure tool policies, sandboxing, red-team evaluation, and human review.

Reproducibility statement

Algorithmic details and proofs are provided in Appendices B and C, and the experimental protocol is described in Appendix D.

References

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning, 2025. URL https://arxiv.org/abs/2507.19457. ICLR 2026 Oral.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Aime-2025, February 2025. URL https://huggingface.co/datasets/MathArena/aime_2025. Dataset from MathArena: Evaluating LLMs on Uncontaminated Math Competitions.

Ching-An Cheng, Allen Nie, and Adith Swaminathan. Trace is the next AutoDiff: Generative optimization with rich feedback, execution traces, and LLMs. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/83ba7056bce2c3c3c27e17397cf3e1f0-Abstract-Conference.html.

Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/bad5f33780c42f2588878a9d07405083-Abstract.html.

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. HoVer: A dataset for many-hop fact extraction and claim verification. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3441–3460, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.309. URL https://aclanthology.org/2020.findings-emnlp.309/.

Enoch Hyunwook Kang and Hema Yoganarasimhan. TextBO: Bayesian optimization in language space for eval-efficient self-improving AI. In *ICLR Workshop on AI with Recursive Self-Improvement*, 2026. URL https://openreview.net/forum?id=h0XYERyeL6.

Siyan Li, Vethavikashini Chithrra Raghuram, Omar Khattab, Julia Hirschberg, and Zhou Yu. PAPILLON: Privacy preservation from Internet-based and local language model ensembles. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3371–3390, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.173. URL https://aclanthology.org/2025.naacl-long.173/.

Xiaoyu Ma, Yiwen Li, Haoyue Liu, Zhichao Wang, Ye Chen, Yongxin Guo, and Xiaoying Tang. Select smarter, not more: Prompt-aware evaluation scheduling with submodular guarantees, 2026. URL https://arxiv.org/abs/2604.11328.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and

Peter Clark. Self-refine: Iterative refinement with self-feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-Pro: Learning reusable skills from experience via non-parametric PPO for LLM agents, 2026. URL https://arxiv.org/abs/2602.01869. ICML 2026 Spotlight.

Johannes Moll, Jean-Philippe Corbeil, Jiazhen Pan, Martin Hadamitzky, Daniel Rueckert, Lisa Adams, and Keno Bressem. GRASP: Gated regression-aware skill proposer for self-improving LLM agents, 2026. URL https://arxiv.org/abs/2605.29668.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.525. URL https://aclanthology.org/2024.emnlp-main.525/.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following, 2025. URL https://arxiv.org/abs/2507.02833.

Chengshuai Shi, Kun Yang, Zihan Chen, Jundong Li, Jing Yang, and Cong Shen. Efficient prompt optimization through the lens of best arm identification. In *Advances in Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=FLNnlfBGMo.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Fei Wang, Si Si, Cho-Jui Hsieh, and Inderjit S. Dhillon. APEX: Automated prompt engineering expert with dynamic data selection, 2026a. URL https://arxiv.org/abs/2606.11459.

Zefeng Wang, Minxi Yan, Jinhe Bi, Sikuan Yan, Volker Tresp, and Yunpu Ma. MetaSkill-Evolve: Recursive self-improvement of LLM agents via two-timescale meta-skill evolution, 2026b. URL https://arxiv.org/abs/2607.05297.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-limited llm benchmark, 2025. URL https://arxiv.org/abs/2406.19314.

Xiaojun Wu, Cehao Yang, Honghao Liu, Xueyuan Lin, Wenjie Zhang, Zhichao Shi, Xuhui Jiang, Chengjin Xu, Jia Li, and Jian Guo. Bayesian-Agent: Posterior-guided skill evolution for LLM agent harnesses, 2026. URL https://arxiv.org/abs/2606.08348.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–616, 2025.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. CoEvoSkills: Self-evolving agent skills via co-evolutionary verification, 2026. URL https://arxiv.org/abs/2604.01687.

A Notation and Exact State

A.1 Eligible exposure and frontiers

At optimization step t, reuse the credit-eligible set $\Omega_t(s) = \{x \mid q_t(s,x) \text{ exists and } x \notin \mathcal{B}_s\}$ from Section 2. The frontier is

$$\mathcal{F}_t(x) = \arg\max_{\substack{s \\ x \in \Omega_t(s)}} q_t(s,x),$$

including all ties, even when all observed scores are zero. For each instance, define

$$\mathcal{C}_t(x) = \{u \mid x \in \Omega_t(u)\}, \quad N_x = |\mathcal{C}_t(x)|, \quad y_{sx} = \mathbf{1}\{s \in \mathcal{F}_t(x)\}, \quad K_x = \sum_{u \in \mathcal{C}_t(x)} y_{ux}.$$

When $N_x \geq 2$, the per-instance credit is

$$r_{sx} = y_{sx} - \frac{K_x - y_{sx}}{N_x - 1}.$$

The relative credit G_s averages these per-instance credits.

$$\Omega_t^{\mathrm{cmp}}(s) = \{x \in \Omega_t(s) \mid N_x \geq 2\}, \qquad E_s^{\mathrm{cmp}} = |\Omega_t^{\mathrm{cmp}}(s)|,$$
$$R_s = \sum_{x \in \Omega_t^{\mathrm{cmp}}(s)} r_{sx}, \qquad G_s = R_s / E_s^{\mathrm{cmp}}.$$

An $N_x = 1$ observation remains in the cache but does not enter the aggregate. Whenever a new finite observation changes $\mathcal{C}_t(x)$ or $\mathcal{F}_t(x)$, COMPASS recomputes N_x, K_x, and every affected candidate contribution on that instance. Candidates with $E_s^{\mathrm{cmp}} = 0$ have undefined credit.

A.2 Inherited proposal history

If parent s is proposed on batch P, the prospective child cannot use its current proposal instances or any ancestor birth-proposal instances as admission evidence. Let P_a^{birth} be the recorded proposal batch that created accepted candidate a, with $P_0^{\mathrm{birth}} = \varnothing$ for the seed. The child's proposal history is

$$X(s,P) = P \cup \bigcup_{a \in \mathrm{Anc}(s) \cup \{s\}} P_a^{\mathrm{birth}}.$$

Proposal history records the instances used in a candidate's own and its ancestors' proposal batches. Admission observations retain their role as evaluation evidence. The proposal history $\mathcal{B}_s$ governs which cached observations may contribute to candidate s's relative credit, and $X(s,P)$ extends that exclusion to the prospective child's admission.

B Algorithmic Details

B.1 Proposal matching

Equations (3) and (4) define the proposal objective. COMPASS builds the complete slot–batch weight matrix from the cache and solves the resulting maximum-weight assignment exactly. Duplicate parent labels remain distinct left-side slot vertices. Missing cache entries remain unchanged during assignment.

B.2 Admission

Proposal tasks with usable execution traces and reflection data proceed to child generation. An optional step skips tasks whose parent attains the maximum score on every proposal instance. Let $m \leq w$ be the number of remaining tasks. For each task independently, the benchmark evaluator samples one batch A_i from that task's admission sampler after removing every instance in $X(s_i, P_i)$. For each instance in the sampled batch, the evaluator selects the frontier candidate with defined, maximal G as reference, using seeded randomness for an exact tie and the sampled parent when no frontier candidate has defined credit. Missing reference observations are evaluated before child generation and stored using the highest-score cache rule. The admission instances, references, and outcomes remain hidden from the generator. The fixed child is admitted exactly when the paired aggregate in Equation (5) is positive.

B.3 WAVE EXECUTION

Let $\mathcal{H}_t$ contain the lineage, proposal batches, cache, remaining budget, and position in the active-set order. A new observation replaces an existing score and its associated output only when its score is strictly higher. Each observed candidate–instance pair therefore retains one record.

Algorithm 1 One COMPASS wave

Require search state $\mathcal{H}_t$, wave width w, active cutoff ρ
Ensure updated search state $\mathcal{H}_{t+1}$
Recompute every defined G_s from the cache
$\mathcal{A}_t \leftarrow$ tie-inclusive Top-ρ candidates, sorted by G_s and candidate index
If $\mathcal{A}_t = \varnothing$, set $\mathcal{A}_t \leftarrow \{0\}$ ▷ Seed fallback
$(z_i)_{i \in \mathcal{I}_w} \leftarrow$ next w parents from $\mathcal{A}_t$, continuing from the saved position
Draw $(P_j)_{j \in \mathcal{I}_w}$ and form $M_{ij} \leftarrow H(z_i, P_j)$
$Y^\star \leftarrow \textsc{MaxWeightAssignment}(M)$
for all assigned parent–proposal pairs (s_i, P_i) **in parallel do**
    Evaluate s_i on P_i and construct its reflection data
    Update the cache with new pairs or strictly higher parent scores
end for
Optionally skip tasks with maximal parent scores on every proposal instance
$\mathcal{T} \leftarrow$ remaining tasks with usable traces and reflection data
$m \leftarrow |\mathcal{T}| \le w$
for all $(s_i, P_i) \in \mathcal{T}$ **in parallel do**
    $A_i \leftarrow \textsc{SampleAdmission}(s_i, P_i)$ subject to $A_i \cap X(s_i, P_i) = \varnothing$
    $r_i \leftarrow \textsc{BindReferences}(A_i)$, then freeze and hide (A_i, r_i) from the generator
    Evaluate missing references and store their scores and outputs
    $p_i \leftarrow \textsc{Reflect}(s_i, P_i)$, then evaluate p_i on A_i
    $\Delta_i \leftarrow \sum_{x \in A_i} [q(p_i, x) - q(r_i(x), x)]$
    if $\Delta_i > 0$ **then**
        Add p_i to the candidate pool and its eligible admission evidence to the cache
    end if
end for
Update the remaining budget, frontiers, credits, and position in the active-set order

Repeat Algorithm 1 while cumulative proposal and evaluation cost remains below B, and complete the final wave before stopping.

C PROPERTIES OF ADMISSION, CREDIT, AND SELECTION

We prove the exclusion and paired-concentration properties of admission, instance-wise credit conservation, finite-pool selection bounds, coverage of the active set, and optimality of the proposal assignment.

C.1 WHY PROPOSAL AND ADMISSION EVIDENCE ARE SEPARATE

Proof of Proposition 1. Let the task distribution be uniform on a domain of size $K \ge n/\epsilon$. After observing $S = (x_1, \dots, x_n)$, let the proposal rule construct a program whose reward is one exactly on the distinct elements appearing in S and zero elsewhere. Every reused evaluation example belongs to that set, so the empirical reward on S is one. The population reward is the fraction of the domain covered by the observed set, at most $n/K \le \epsilon$. □

The construction motivates admission on examples outside those that shaped the candidate. Recording proposal history extends this separation through the lineage, because a descendant can retain behavior adapted to an ancestor's proposal batch.

C.2 PAIRED ADMISSION

Theorem 1 (Conditional concentration for paired admission). *Let $\mathcal{V}$ be the information visible to the generator when child p is fixed from parent s and proposal batch P, and suppose that p is $\mathcal{V}$-measurable. Before generation, the evaluator fixes an admission batch $A = (X_1, \dots, X_n)$, with $n \geq 1$, outside $X(s, P)$ and binds each reference $r(X_j)$ from the committed pre-child state and seeded tie-breaking randomness. Admission identities, references, and outcomes are not visible in $\mathcal{V}$. Then*

$$A \cap P = \varnothing, \qquad A \cap P_a^{\mathrm{birth}} = \varnothing \quad \textit{for every } a \in \mathrm{Anc}(s) \cup \{s\}.$$

After p is fixed, reveal the paired outcomes sequentially. Let $(\mathcal{H}_j)_{j=0}^n$ be the filtration generated by the fixed child, the frozen admission plan, and the first j revealed paired outcomes. Define

$$Z_j = q(p, X_j) - q(r(X_j), X_j), \qquad \mu_j = \mathbb{E}[Z_j \mid \mathcal{H}_{j-1}],$$

$$\widehat{\mu}_A(p, r) = \frac{1}{n} \sum_{j=1}^{n} Z_j, \qquad \overline{\mu}_A(p, r) = \frac{1}{n} \sum_{j=1}^{n} \mu_j.$$

Because $Z_j \in [-1, 1]$, for every $\delta \in (0, 1)$, with probability at least $1 - \delta$,

$$|\widehat{\mu}_A(p, r) - \overline{\mu}_A(p, r)| \leq \sqrt{\frac{2 \log(2/\delta)}{n}}. \tag{9}$$

For the ith task, the implemented gate is equivalently $\mathrm{admit}(p_i) \iff \widehat{\mu}_{A_i}(p_i, r_i) > 0$.

Proof of Theorem 1. Because $X(s, P)$ contains both P and every P_a^{birth} for $a \in \mathrm{Anc}(s) \cup \{s\}$, legality $A \cap X(s, P) = \varnothing$ gives the displayed disjointness relations. After the child is fixed, each paired difference Z_j lies in an interval of length two. Conditional Hoeffding–Azuma applied to $Z_j - \mathbb{E}[Z_j \mid \mathcal{H}_{j-1}]$ gives

$$\Pr(|\widehat{\mu}_A(p, r) - \overline{\mu}_A(p, r)| \geq \epsilon) \leq 2 \exp\left(-\frac{n\epsilon^2}{2}\right).$$

Setting the right-hand side to δ proves Equation (9). □

C.3 PER-INSTANCE CREDIT SUMS TO ZERO

Proof of Proposition 2. For a fixed comparable instance x, use $\sum_{s \in \mathcal{C}_t(x)} y_{sx} = K_x$ to obtain

$$\begin{aligned} \sum_{s \in \mathcal{C}_t(x)} r_{sx} &= K_x - \frac{\sum_{s \in \mathcal{C}_t(x)} (K_x - y_{sx})}{N_x - 1} \\ &= K_x - \frac{N_x K_x - K_x}{N_x - 1} = 0. \end{aligned}$$

The statement is per instance, and the candidate averages G_s need not sum to zero because their comparable exposure sets and denominators can differ. □

C.4 FINITE-POOL SELECTION

Proposition 3 (Finite-pool mean selection). *For any $M, n \in \mathbb{N}_+$ and $\delta \in (0, 1)$, let $p_1, \dots, p_M$ be adaptively generated candidates. Let $\mathcal{J}_{i-1}$ contain the history before candidate p_i is evaluated, with p_i measurable with respect to that history. Conditional on $\mathcal{J}_{i-1}$, suppose p_i receives n independent and identically distributed rewards in $[0, 1]$ under a common benchmark evaluation distribution, with mean $R(p_i)$. Let $\widehat{R}_n(p_i)$ be their arithmetic mean and define*

$$\widetilde{p}_{M,n} \in \operatorname*{arg\,max}_{p \in \{p_1, \dots, p_M\}} \widehat{R}_n(p).$$

Then, with probability at least $1 - \delta$,

$$R^\star - R(\widetilde{p}_{M,n}) \leq \underbrace{R^\star - \max_{i \leq M} R(p_i)}_{\textit{search gap}} + \underbrace{2\sqrt{\frac{\log(2M/\delta)}{2n}}}_{\textit{selection noise}}. \tag{10}$$

Proof of Proposition 3. Conditional on $\mathcal{J}_{i-1}$, Hoeffding's inequality gives

$$\Pr\Big(|\widehat{R}_n(p_i) - R(p_i)| > \epsilon \mid \mathcal{J}_{i-1}\Big) \le 2e^{-2n\epsilon^2}.$$

Taking expectations over the history and a union bound over the M candidates gives, with probability at least $1-\delta$,

$$\max_{i\le M} |\widehat{R}_n(p_i) - R(p_i)| \le \epsilon, \qquad \epsilon = \sqrt{\frac{\log(2M/\delta)}{2n}}.$$

Let $p_M^\star$ maximize population utility in the pool. On this event,

$$R(p_M^\star) - R(\widetilde{p}_{M,n}) \le \widehat{R}_n(p_M^\star) - \widehat{R}_n(\widetilde{p}_{M,n}) + 2\epsilon \le 2\epsilon.$$

Adding the search gap proves Equation (10). □

Taking expectations and bounding regret on the failure event by one also gives

$$\mathbb{E}[R^\star - R(\widetilde{p}_{M,n})] \le \mathbb{E}\left[R^\star - \max_{i\le M} R(p_i)\right] + 2\sqrt{\frac{\log(2M/\delta)}{2n}} + \delta.$$

Along a nested candidate sequence, the search gap is nonincreasing in M, while the selection-noise term decreases with n for fixed M.

Selection by relative credit.

Corollary 1 (Credit-based selection)**.** *Under the conditions of Proposition 3, let $\widehat{p}_G$ be a candidate with the largest defined G in the pool and let ξ_G be as in Equation* (7)*. Then Equation* (8) *holds with probability at least $1-\delta$. Moreover,*

$$\mathbb{E}[R^\star - R(\widehat{p}_G)] \le \mathbb{E}\left[R^\star - \max_{i\le M} R(p_i)\right] + 2\sqrt{\frac{\log(2M/\delta)}{2n}} + \mathbb{E}[\xi_G] + \delta.$$

Proof of Corollary 1. On the simultaneous concentration event used above,

$$\begin{aligned} R(p_M^\star) - R(\widehat{p}_G) &\le \widehat{R}_n(p_M^\star) - \widehat{R}_n(\widehat{p}_G) + 2\epsilon \\ &\le \xi_G + 2\epsilon. \end{aligned}$$

Adding the search gap yields Equation (8). The expectation bound follows because regret is at most one on the failure event and the search gap and ξ_G are nonnegative. □

Coinciding rankings for binary scores. Suppose the $M \ge 2$ candidates share the same n legal instances, each candidate–instance pair has one binary score, and these are all their comparable observations. Thus $N_x = M$ and $\widehat{R}_n(p_s) = n^{-1}\sum_x q(s,x)$. On each instance, either at least one score is one and $y_{sx} = q(s,x)$, or every score is zero and every candidate is on the frontier. In both cases,

$$r_{sx} = q(s,x) - \frac{1}{M-1}\sum_{u\ne s} q(u,x).$$

Averaging over the common instances gives

$$G_s = \frac{M}{M-1}\left(\widehat{R}_n(p_s) - \frac{1}{M}\sum_{u=1}^{M}\widehat{R}_n(p_u)\right). \tag{11}$$

This is a shared positive affine transformation of the sample means, so their rankings and ties coincide and $\xi_G = 0$.

C.5 Properties of parent selection and proposal matching

Proposition 4 (Proposal turns for a fixed active set)**.** *For a fixed ordered active set of size m, every block of m consecutive queue draws contains each active candidate exactly once, including when a width-w wave wraps around the queue.*

Proof. Index the fixed active order by $0, \ldots, m-1$ and let the persistent cursor be c. Consecutive draws are $(c+r) \bmod m$. For $r = 0, \ldots, m-1$, these residues are pairwise distinct and cover every index. Grouping the same sequence into blocks of width w does not alter the argument, including a block that wraps around the end of the queue. □

Proposition 5 (Maximum-weight proposal matching)**.** *Conditional on the w drawn parent slots and w proposal batches, Equation* (4) *maximizes total matching weight over all one-to-one slot–batch assignments.*

Proof of maximum-weight proposal matching. The w parent slots and w proposal batches define a complete bipartite graph with edge weight $H(z_i, P_j)$. A feasible binary matrix in Equation (4) is exactly a perfect matching, or equivalently a permutation matrix. Selecting the maximum-weight feasible matrix therefore returns the maximum-weight matching. Repeated parent labels do not change the graph because their slot vertices remain distinct. □

D Detailed Experimental Protocol

D.1 Method configuration

The method in Section 3 uses relative credit, applies the tie-inclusive Top-ρ cutoff to all candidates with defined credit, uses the exact proposal assignment in Equation (4), and samples admission batches independently for each remaining proposal task. The selected program is evaluated once.

The Qwen3-8B runs select candidates 55, 78, 34, and 28 for HotPotQA, IFBench, PUPA, and LiveBench-Math, respectively.

D.2 Official evaluation protocol

The selected candidate is fixed at the end of optimization, before its test scores are observed, and evaluated on the complete test split. IFBench reports the mean score over 294 instances. AIME uses all 150 benchmark problems. HoVer uses 300 instances with binary scores, and LiveBench-Math uses its frozen 126-problem test with the official aggregation rule.