Mixture of Self-Improving Branches for Agent Harness Optimization

Paper discussion: adaptive search, routing, and evidence

Paper authors: Haoyu Dong, Yuhang Zhou, Zihao Lin, Yifan Wu, Bo Peng, Mingyi Wang, Xiangjun Fan, Lizhu Zhang, Zhuokai Zhao

arXiv:2609.37834v1 23 October 2026

Adapt the search process, then route among its products

Core idea

Let branches change both the development cases they optimize and the guidance used to propose their next harness; deploy their best heads through an input-level router.

- Math–Gemini held-out accuracy: **46.0%** $\rightarrow$ **62.0%** vs Meta-Harness.
- Relative gains: 34.8% Math–Gemini, 11.6% Terminal-Bench 2.0, 3.8% SWE-bench Lite.
- Selection and router configuration use development data; test-best analysis is reported separately.

Paper, Abstract and Table 1, PDF pp. 1, 6; Sec. 4.6, PDF p. 9.

A fixed development set can hide complementary harnesses

Harness

Executable code around the action model: retrieval, prompts, tool interfaces, verification, and control flow.

Meta-Harness iteratively proposes code and scores it on a fixed development set.

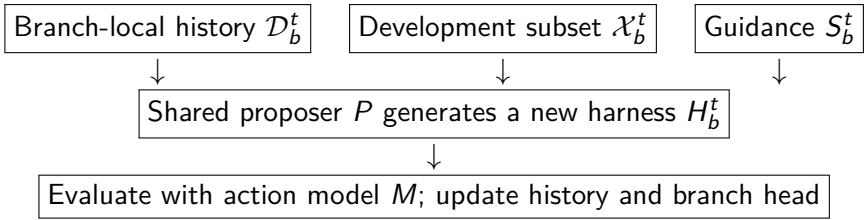
Paper, Sec. 1, PDF pp. 1–2.

Failure mode motivating branches

A candidate that solves cases missed by the leading harness can rank lower overall, so later proposals may never build on it.

The paper also changes proposal instructions as each branch learns from its own history.

Each branch maintains an objective and a proposal policy



- Two branches start from the same seeds, full development set, and initial guidance.
- One candidate per branch per iteration; branches run in parallel.

Paper, Sec. 3 and Algorithm 1, PDF pp. 4–5. Diagram is an explanatory redrawing.

Changing the development subset changes the preferred parent

$$J_b^t(H) = \frac{1}{|\mathcal{X}_b^t|} \sum_{x \in \mathcal{X}_b^t} \mathbb{E}[r(f_M(H, x), x)]$$

- H: candidate harness; M: action model; $f_M(H, x)$: execution trajectory; r: task reward.
- The branch head $\widehat{H}_b^t$ is the candidate with highest observed mean reward on its current subset.
- A subset update can change the ranking and head even when candidate code stays the same.

Paper, Sec. 3, Eq. (1), PDF p. 4.

Frontier coverage assigns each development case

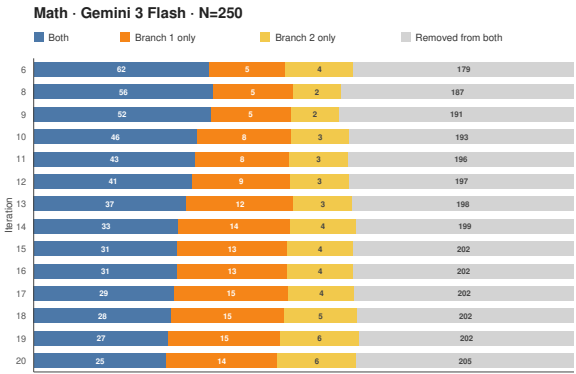
$$\mathcal{F}_b^t = \mathrm{Top}_q(\{H\}_b^t), \qquad c_b^t(x) = \sum_{H \in \mathcal{F}_b^t} \mathbf{1}[H \text{ solves } x]$$

Coverage condition for case x	Subset update
Every frontier harness solves x	Remove from all branches
One branch leads by at least Δ votes	Keep only in that branch
Otherwise	Preserve current membership

- Default $q = 5$, $\Delta = 2$: votes 4:1 assign ownership; 5:5 remove; 3:2 leave unchanged.

Paper, Sec. 3, Eq. (2), PDF p. 4. Vote examples expand the stated rule.

Math–Gemini's development objective changes substantially



At iteration 20, of 250 cases:

- 25 shared.
- 14 Branch 1 only.
- 6 Branch 2 only.
- 205 removed from both.

Final head selection uses each branch's retained subset.

Original paper Fig. 3, cropped Math–Gemini panel, PDF p. 14. Bar widths use a log scale.

Proposal guidance also learns from branch-local history

- Every $C_s = 5$ iterations, the same proposer P revises each branch's `SKILL.md` guidance.
- The update prompt compares candidates within the same development stage and distinguishes repeated evidence from isolated outcomes.
- At iteration 15 on Math–Gemini, Branch 1 emphasizes concrete information from retrieved solutions; Branch 2 emphasizes free-form derivation and two-sided closure.

Different levers

Development updates change which harness scores well; guidance updates change what the proposer is encouraged to try next.

Paper, Sec. 3, PDF pp. 4–5; Sec. 4.3, PDF p. 7; Appendix C, PDF pp. 14–16.

A head replacement redirects subsequent proposals

Iteration 13	New head	14–15	17–19
Branch 2 drops four cases: 44 → 40.	Free-form derivation replaces a geometry-aware dual solver.	Targeted retrieval and two-sided closure extend that head.	Rehearsal, agreement challenge, and gap-directed construction follow.

Recorded dependencies support a search-redirection mechanism; they do not isolate the cause of the eventual test gain.

Paper, Sec. 4.3, PDF p. 7; Appendix B.2, PDF pp. 13–14.

The router chooses a branch head before execution

1. Select one final head per branch using its own final development subset.
2. Label development cases solved by exactly one head; provide expert code, these examples, and both heads' *development-example* outputs.
3. Optimize the router instruction with GEPA on those development labels.
4. For a new input, select one head *before* running it: no test answer or expert output on that input is available.

Paper, Sec. 3, PDF p. 5; Appendix D.1–D.2, PDF pp. 17–19.

Four held-out settings test deployment performance

Setting	Action model	Dev / test	Metric
Olympiad math	Gemini 3 Flash	250 / 200	Accuracy
Olympiad math	Sonnet 4.5	250 / 200	Accuracy
Terminal-Bench 2.0	Sonnet 4.5	30 / 58	Pass rate
SWE-bench Lite	Sonnet 4.5	50 / 250	Resolution

- $B = 2$ branches, $N = 20$ iterations, one candidate per branch per iteration.
- Meta-Harness uses fixed development cases and proposal guidance; both systems select from development data.

Paper, Sec. 4.1, PDF p. 6.

Held-out gains are largest for Math–Gemini

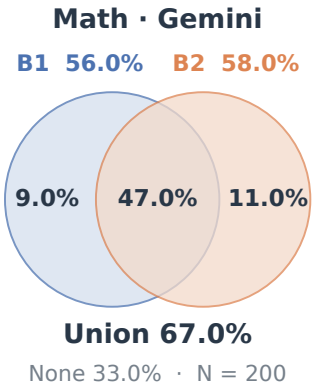
Setting	Meta-Harness	Routed system	Absolute gain	Relative gain
Math–Gemini	46.0%	**62.0%**	**16.0 pp**	**34.8%**
Math–Sonnet	29.0%	30.5%	1.5 pp	5.2%
Terminal-Bench 2.0	44.8%	50.0%	5.2 pp	11.6%
SWE-bench Lite	63.6%	66.0%	2.4 pp	3.8%

Read the percentage correctly

46.0% → 62.0% is a 16.0 percentage-point increase, or a 34.8% relative increase.

Adapted from paper Table 1, PDF p. 6.

The heads solve complementary held-out cases



Math–Gemini, 200 test cases

Head 1 solves 56.0%; Head 2 solves 58.0%. Their union reaches **67.0%**, a 9.0-point opportunity above the stronger head.

Other union rates: Math–Sonnet 36.0%, Terminal-Bench 55.2%, SWE-bench Lite 70.4%.

Original paper Fig. 2(b), cropped Math–Gemini panel, PDF p. 7. Union is an oracle upper bound.

The router uses some, but not all, complementarity

Setting	Head 1	Head 2	Router	Union
Math–Gemini	56.0%	58.0%	**62.0%**	67.0%
Math–Sonnet	26.0%	**31.0%**	30.5%	36.0%
Terminal-Bench 2.0	44.8%	48.3%	**50.0%**	55.2%
SWE-bench Lite	65.6%	**66.4%**	66.0%	70.4%

- Router exceeds the stronger head by 4.0 pp on Math–Gemini and 1.7 pp on Terminal-Bench.
- It falls short by one test case in each of Math–Sonnet and SWE-bench Lite.

Paper, Table 2, PDF p. 8; union coverage in Fig. 2(b), PDF p. 7.

The two branches develop different math mechanisms

Branch 1: answer verification

- Dual-path adjudication.
- Concrete answer checks.
- Proof-obligation routing.

Branch 2: solution construction

- Free-form derivation.
- Two-sided closure.
- Challenge an agreed answer.

Fig. 2(a) plots *retrospective test-best* progress; it was not the criterion used to propose or deploy a harness.

Paper, Sec. 4.3 and Fig. 2(a), PDF p. 7; Appendix E, PDF pp. 19–20.

Both adaptive components outperform either one alone

Update interval

Dev update every	Accuracy
5 iterations	53.0%
3 iterations	56.0%
1 iteration	**62.0%**

Component combination

Enabled components	Accuracy
Neither	51.0%
Guidance only	50.0%
Dev pruning only	54.0%
Both	**62.0%**

Paper, Table 3 and Sec. 4.5, PDF pp. 8–9. Math–Gemini; default router.

Router prompt optimization has setting-dependent effects

Full router context	Math–Gemini	Math–Sonnet	Terminal-Bench	SWE-bench
Without GEPA	59.5%	29.0%	46.6%	**66.4%**
With GEPA	**62.0%**	**30.5%**	**50.0%**	66.0%

- Full context: expert source code, head-exclusive development examples, and their outputs.
- Adding example context alone is not consistently beneficial across settings.

Paper, Appendix D.2, Table 6, PDF pp. 18–19.

Retrospective test-best results answer a different question

Method	Math–Gemini	Math–Sonnet	TB2	SWE Lite
Meta-Harness	49.0%	32.5%	46.6%	63.6%
Branching search	**58.0%**	32.5%	**50.0%**	**66.4%**

- Each pool contains ten candidates chosen by development performance; the maximum shown here uses test outcomes *retrospectively*.
- The deployable Math–Gemini router scores 62.0%, above the 58.0% best individual harness in this pool.

Paper, Sec. 4.6 and Table 4, PDF p. 9.

Search gains come with more total token use

Sonnet 4.5 action-model setting	Meta-Harness	Branching	Ratio
Math	79.92 M	172.90 M	**2.16**×
Terminal-Bench 2.0	2,366.80 M	4,001.39 M	**1.69**×
SWE-bench Lite	1,907.17 M	2,951.48 M	**1.55**×

- Totals combine task-solving and harness-authoring tokens; cached input is counted once.
- The math cost row is for Sonnet 4.5, not the largest-gain Gemini 3 Flash setting.

Paper, Appendix A, Table 5, PDF pp. 12–13. Ratios calculated from reported totals.

What the evidence supports, and what remains open

1. Dynamic development subsets and branch-local guidance produce distinct harness histories and stronger development-selected systems in four tested settings.
2. Routing uses development data only and sometimes exceeds the stronger individual head; it still leaves 4.4–5.5 pp of union coverage unrealized.
3. Full search token use is higher, and independent repeated-search variability is not reported; performance per unit search budget remains open.
4. Exact coverage-vote handling for newly proposed harnesses on cases outside their current subset is not specified in the algorithm description.

Paper, Secs. 3–5, PDF pp. 4–10; Appendix A, PDF pp. 12–13. Last item is a reproducibility observation from Eqs. (1)–(2) and Algorithm 1.

Take-home messages

Search mechanism

Relative frontier coverage changes each branch's scoring cases; local guidance turns branch experience into future proposal priorities.

Deployment mechanism

A development-trained router combines two final heads without test labels or new-input expert outputs.

Measured outcome

The strongest gain is Math–Gemini: 46.0% $\rightarrow$ 62.0% held-out accuracy, with higher search token use in the costed Sonnet settings.

Paper, Tables 1–2, PDF pp. 6, 8; Table 5, PDF p. 13.

Backup: full search schedule and ownership cases

- $B = 2$ branches; $N = 20$ iterations; top-q frontier with $q = 5$; ownership margin $\Delta = 2$.
- Development pruning interval $C_p = 1$ after both branches have at least five candidates.
- Guidance update interval $C_s = 5$; when both updates are due, prune first, then revise guidance.
- If votes are 5:5, remove a case from both; if 4:1, assign to the leading branch; if 3:2, preserve existing assignment.

Paper, Secs. 3 and 4.1, PDF pp. 4–6.

Backup: example mechanisms in discovered harnesses

- **Math–Gemini:** construct an attainable answer, derive necessary conditions, then test gaps or challenge agreement.
- **Math–Sonnet:** clean retrieved solutions, append reference answers, and conditionally critique disagreements.
- **Terminal-Bench:** request verification on the first completion signal, then allow a later completion.
- **SWE-bench Lite:** find issue identifiers and relevant source/test files before editing.

Paper, Appendix E, PDF pp. 19–20. These are descriptions of test-best harnesses, not a claim that each component alone caused the final gain.