SHOULD I OPTIMIZE PROMPTS FOR THIS TASK? MEASURING TASK SUITABILITY WITH CARAT

Anonymous authors
Paper under double-blind review

ABSTRACT

Reflective prompt optimization, in which a reflector model revises instructions from task feedback, improves a language model on specific tasks efficiently and without updating its weights, but its outcomes are unstable. Repeated runs of the same setup with different random seeds can shift held-out accuracy considerably, and prompts with nearly the same accuracy can still solve different cases, so a single run's returned prompt says little about the task itself. We introduce CARAT, two task signals for a fixed answerer, reflector and dataset-represented task: CARAT-A measures whether prompt preferences persist across disjoint case samples, and CARAT-C measures whether the observed successes are compatible with a shared prompt. We also propose Multi-Root Full-Retention Sampling (MRFS), which explores the prompt space from independent roots under a fixed proposal budget and retains every valid proposal. Across ten tasks with Qwen3-8B as answerer and reflector, we validate the signals against collections from five prompt optimization methods over multiple seeds: both signals persist across the entire task distribution and are consistent across methods, and MRFS yields the task ordering the optimizers' collections share most. CARAT thus quantifies the task-suitability question that run-level evaluations leave open, putting prompt optimization results on a firmer footing. Code is available at `https://anonymous.4open.science/r/carat-1B00`.

1 INTRODUCTION

Reflective prompt optimization uses task examples and execution feedback to generate and revise instructions, aiming to improve a language model's performance on unseen instances of the same task (Pryzant et al., 2023; Agrawal et al., 2026). Its outcomes can vary substantially across random seeds, even with the method and data split fixed. On HotpotQA (Table 13, Appendix C.7), the final prompts of one method's seeds disagree on up to 22.6% of non-trivial held-out cases and differ by up to 10 points in held-out accuracy, comparable to the 6–13-point gains reported for GEPA, MIPROv2 and ACE (Agrawal et al., 2026; Opsahl-Ong et al., 2024; Zhang et al., 2026a).

An unsuccessful optimization run may reflect limited exploration, while a successful held-out result need not recur under another seed. Different prompts may also solve different instances without showing whether their successes can be combined. These outcomes leave two questions about task suitability: how reliably does available evidence support prompt selection, and how compatible are the instances' requirements?

We characterize these two aspects through *accessibility* and *coherence*. Accessibility concerns locating an effective prompt using the evidence at hand. We measure it through the persistence of pairwise prompt comparisons across disjoint case samples: a preference supported by one set of cases may reverse on another. Coherence concerns meeting instance requirements with a shared prompt. Two prompts can have equal accuracy while solving different cases, and their collection may contain no prompt that combines their successes. Figure 1 shows observed prompt pools with contrasting combinations of these characteristics. Accessibility concerns the reliability of comparisons; coherence concerns the structure of the solved sets, which average accuracy alone does not preserve.

We introduce **CARAT—Coherence and Accessibility of Reflector–Answerer–Task configurations**—to measure these characteristics for a fixed experimental condition: an answerer, a reflector,

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

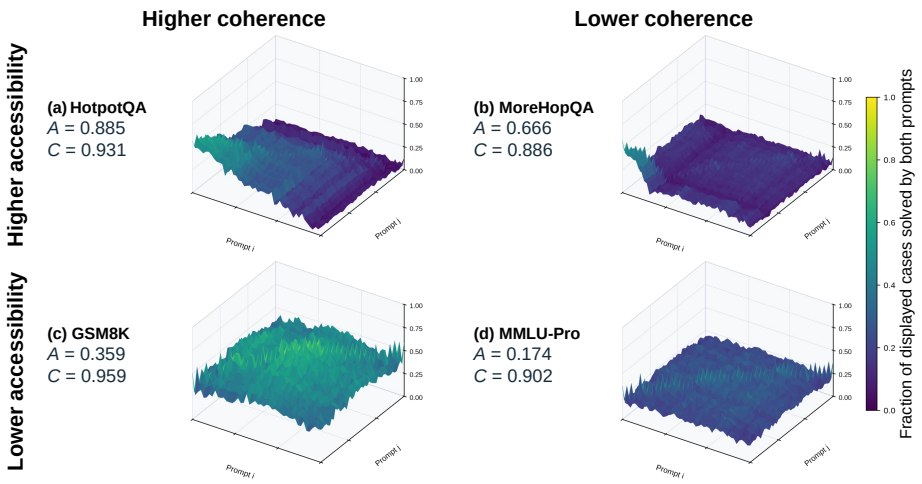

Figure 1: **Prompt-pair overlap across tasks with contrasting CARAT-A and CARAT-C.** Each panel shows a representative prompt pool. Both horizontal axes index prompts; height and color indicate the fraction of displayed cases solved by both prompts. Each terrain shows the cases on which prompts differ (display rule in Appendix D); the readings A (CARAT-A) and C (CARAT-C) use the full held-out case set. Higher A means fewer prompt-preference reversals across disjoint case samples; higher C means observed successes are more compatible with a shared prompt (§4).

and a dataset-represented task. Its separately reported metrics are **CARAT-A** (accessibility) and **CARAT-C** (coherence), with readings denoted $A(n)$ and C. CARAT-C uses weighted containment among distinct maximal observed solved sets. Our analyses reveal recurring task differences across held-out cases from the task distribution, accessibility measurements on individual optimizers' collections, and coherence comparisons with collections pooled across methods and seeds. These patterns characterize collections even when their final selected prompts vary.

We make two contributions. First, we specify the experimental condition and the reflective proposal process through the reflector, exposed evidence, and proposal rules. We introduce Multi-Root Full-Retention Sampling (MRFS), a sampling procedure that generates initial prompts with the reflector from randomized task examples and explores bounded revisions, avoiding dependence on a manually chosen starting instruction. Retaining all valid proposals, including revisions that do not improve performance, yields a common prompt-by-instance response matrix. Second, we introduce accessibility and coherence and their extraction from these outcomes, evaluating their repeatability and agreement with observations from other exploration procedures.

The explicit task definition and proposal specification establish when these measurements can inform research and practice. For research, they inform task selection, evaluation design, and interpretation of optimization results. In practice, CARAT-A shows where candidate preferences persist across cases at the declared call size and where stronger selection evidence is needed. CARAT-C shows whether the observed successes can be served together by a shared prompt, or whether specialized prompting and routing (Wang et al., 2024a) deserve investigation. We also find CARAT-A useful for measuring prompt optimization methods: read on each method's generated prompts, it gauges how reliably they can be compared, a second measure beside accuracy (Section 6).

2 Related Work

Reflective prompt optimization uses task feedback to guide instruction search. ProTeGi combines natural-language critiques with candidate editing and beam search (Pryzant et al., 2023). GEPA extends this approach through reflection on execution traces and Pareto-based exploration, reporting stronger average performance than GRPO (Shao et al., 2024) across six benchmarks with Qwen3-8B (Agrawal et al., 2026). RLMOpt places search decisions under language-model control and reports higher held-out scores than GEPA-light in nine of eleven matched benchmark–seed compar-

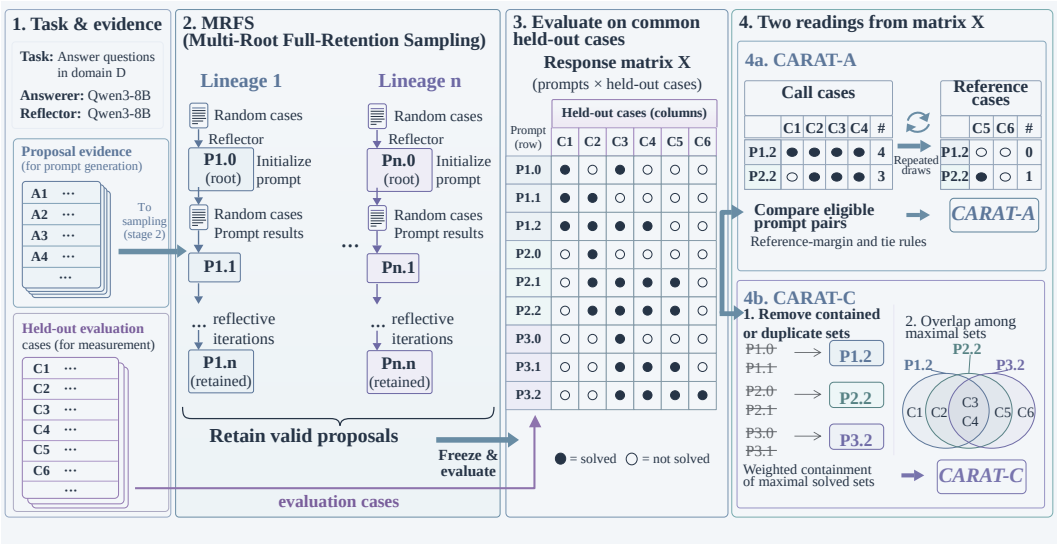

Figure 2: **From task specification to the two CARAT readings.** Every retained MRFS prompt is evaluated on common held-out cases; CARAT-A and CARAT-C are read from the matrix X.

isons (Satheesha et al., 2026). These results demonstrate the potential of prompt optimization while showing that the performance reached depends on the exploration procedure.

Interpreting these outcomes requires distinguishing starting instructions, random seeds, search budgets, and evaluation splits. In *Reflection in the Dark*, GEPA fails to repair an output-field ordering defect in a GSM8K starting prompt, whereas VISTA recovers through a framework combining explicit hypothesis generation, verification, and exploration (Liu et al., 2026). Its candidate traces (Liu et al., 2026, p. 19) also contain revisions that score below their parents on evaluation minibatches, illustrating information omitted by best-so-far summaries. Changing the starting prompt examines a different source of variation from repeating an optimizer with that prompt fixed. Likewise, RL-MOpt's repeated comparisons (Satheesha et al., 2026) vary data splits with the random seed, whereas our Table 13 in Appendix C.7 examines final-selection variation with method and split fixed. Held-out evaluation measures the selected prompt's performance; repeated runs assess the repeatability of producing it. Failure under one search procedure therefore need not establish a limitation of the task. Other diagnostics measure prompts or optimization outcomes directly. ProSA quantifies how much a model's instance-level results change across prompt variants (Zhuo et al., 2024), and Zhang et al. (2026b) predict whether optimization is worthwhile from how far the best of 10–20 candidate prompts improves on the zero-shot prompt. Neither characterizes whether a task's evidence supports prompt selection or whether its instances' requirements are compatible.

CARAT instead examines which properties recur across prompt collections and exploration procedures, reading CARAT-A and CARAT-C from response matrices of sampled prompts; Section 3 defines the experimental condition and proposal process on which these comparisons depend.

3 Specifying and Sampling Reflective Prompt Optimization

3.1 Specifying the task and reflective proposal process

We study reflective prompt optimization for a fixed answerer, reflector, and dataset-represented task. Denote this experimental condition by $\mathcal{T} = (A, R, \tau)$, with A the answering model, R the reflecting model, and τ the task: the instance distribution represented by the data. The objective is to improve the answerer's performance on unseen instances of the same task using a shared prompt that the reflector proposes from the information available to it. The reflector is bounded by its own knowledge and by the evidence exposed to it, not only by which instances it has seen (Liu et al., 2026). Both models matter: the answerer determines how instructions affect task performance, while the reflector's capabilities and available evidence determine which instructions are proposed. An instruction

acts as a bias on the answerer's responses toward the behaviour a case requires (Xie et al., 2022); the cases of a task may require biases that can coexist in one instruction, or biases that conflict.

Let E_t denote the task evidence exposed at step t, including examples, outputs, scores or textual feedback, and let H_t denote the available proposal history (Zhou et al., 2023; Yang et al., 2024). Under elicitation rules η, a proposal is drawn as

$$p_{t+1} \sim q_{R,\eta}(\cdot \mid E_t, H_t). \tag{1}$$

A complete exploration procedure π specifies these rules, how evidence and histories are obtained, and when generation stops. Together with a rule for selecting an emission from a trajectory, it induces a marginal prompt distribution $q_{\mathcal{T},\pi}$. We define the associated prompt space by $\mathcal{P}_{\mathcal{T},\pi} = \mathrm{supp}(q_{\mathcal{T},\pi})$. An optimizer's starting prompt, evidence selection and continuation rules shape this distribution and may also change its support (Liu et al., 2026). Outside guidance, such as supplied scientific knowledge, changes the information supplied to the proposal process, so our characterization is conditional on both the reflector and its available information; the ability to recognize a useful instruction does not establish the ability to generate it.

This specification also locates prompt optimizers. A method that proposes with the same reflector from the same kinds of task evidence proposes within the same prompt space; a method that changes the reflector or the evidence available to it, such as one whose proposer never observes a task case, changes the space itself. Within a space, an optimizer's starting prompt, evidence selection, parent selection, acceptance rule and stopping condition concentrate its proposals on a small region, and one run yields a finite collection from that region under that policy. Its outcome therefore describes the region visited, not the task, and a second seed can move within the same concentration.

3.2 Sampling the prompt space

The response patterns available for measurement depend on where exploration concentrates within the prompt space. An optimizer's collection concentrates where its policy leads from its starting prompt, so measurements taken on that collection describe the policy's region. Measuring the task instead requires an observation procedure whose concentration is declared in advance and tied to no starting prompt or search policy. We therefore distribute exploration opportunities across randomized task evidence, independently initiated roots (Honovich et al., 2023) and bounded revisions. Random case draws vary the evidence, independent roots provide separate starting points, a common revision bound limits concentration on any one lineage, and retaining every valid proposal, including revisions that do not improve, preserves unsuccessful behaviours (Figure 2). These choices define Multi-Root Full-Retention Sampling (MRFS), the exploration procedure from which CARAT obtains its response observations. The sample remains conditional on the reflector, elicitation and sampling rules, and its rows are dependent observations (Appendix B.1).

A single optimization outcome can also depend strongly on the particular run. Across ten tasks and five optimizers, each run with three seeds on a fixed data split, the final prompts of two seeds differ in correctness on a median 18.9% of non-trivial cases, and held-out accuracy spans at least 5 points in 15 of the 50 method–task groups (Appendix C.7). On HotpotQA, for example, MIPROv2's seeds span 10.0 points and disagree on 22.6% of these cases.

3.3 One retained sample and its response matrix

One retained sample is the full collection of valid prompts from one execution of MRFS (Section 3.2). Let $p_1, \ldots, p_m$ denote these prompts and $z_1, \ldots, z_N$ the common held-out instances. Write $A_\tau(p, z)$ for the answer produced using prompt p on instance z under the declared task settings, and $s_\tau(y, z) \in \{0, 1\}$ for the task's correctness score of answer y. Evaluating every prompt on every instance gives the binary response matrix (Maia Polo et al., 2024)

$$X_{j,i} = s_\tau(A_\tau(p_j, z_i), z_i), \qquad X \in \{0,1\}^{m \times N}. \tag{2}$$

Under stochastic answering, an entry is one observed outcome. Only proposed prompts enter the matrix; both metrics are origin-free. The prompt collection is frozen before held-out evaluation, and these outcomes do not guide generation or retention (Algorithm 3, Appendix B.1).

Section 4 reads two properties from the same matrix (Figure 2): whether prompt comparisons persist across cases and whether different instances agree on the prompt they need.

4 CARAT: EXTRACTING TASK-SUITABILITY SIGNALS

One execution of MRFS (§3) supplies a response matrix $X \in \{0,1\}^{m\times N}$ and a complete pair of measurements: CARAT-A (accessibility) asks whether evidence for preferring one prompt persists across cases, and CARAT-C (coherence) asks whether the successes observed across prompts can be served by a shared prompt. Both readings are conditional on the experimental condition and exploration procedure, and we report them separately.

4.1 CARAT-A: LOCATING A PROMPT WHOSE ADVANTAGE SURVIVES

A prompt preferred on the cases available for selection may lose that advantage on other instances of the same task (Card et al., 2020; Perez et al., 2021). Accessibility measures whether the evidence available at a call size n supports prompt preferences that persist on separate cases. It characterizes the comparison of available candidates; discovering those candidates also depends on the exploration procedure in §3.

In repetition r, draw a reference subset R_r of size h uniformly without replacement from the held-out columns of X, then a call subset L_r of size n from the remaining columns. For each prompt pair $j < k$, let $d_{jk,i} = X_{j,i} - X_{k,i}$ and write $\bar{d}_{jk,L_r}$ and $\bar{d}_{jk,R_r}$ for its subset means. The reference orders a pair when $|\bar{d}_{jk,R_r}| > 1.96\,\widehat{\mathrm{se}}(\bar{d}_{jk,R_r})$, using the sample standard error of the paired differences. Let $\mathcal{Q}_r$ contain these pairs after excluding call-side ties. Define

$$\widehat{D}_r(n) = \frac{1}{|\mathcal{Q}_r|} \sum_{(j,k)\in\mathcal{Q}_r} \mathbf{1}\big\{\mathrm{sign}(\bar{d}_{jk,L_r}) \neq \mathrm{sign}(\bar{d}_{jk,R_r})\big\},$$

$$\widehat{D}(n) = \frac{1}{|\mathcal{V}_n|} \sum_{r\in\mathcal{V}_n} \widehat{D}_r(n),$$

where $\mathcal{V}_n$ contains repetitions with at least one usable pair. Thus each evaluable repetition receives equal weight. We report CARAT-A as

$$A(n) = -\tfrac{1}{2}\log_{10}\widehat{D}(n),$$

applying the logarithm once, after averaging within the run, together with its comparison support: the reference-ordered share, usable comparisons and evaluable repetitions (Appendix D).

4.2 CARAT-C: WHETHER INSTANCE REQUIREMENTS CAN BE SHARED

Different instances can favour different instructions even when prompts have equal average accuracy. Coherence measures how far the observed successes can be served by a shared prompt. Let $S_j = \{i : X_{j,i} = 1\}$. Collapse identical solved sets and discard sets strictly contained in another, as GEPA prunes dominated candidates (Agrawal et al., 2026), leaving the distinct maximal behaviours $\mathcal{B}$, with $K = |\mathcal{B}|$. A duplicate then receives no extra weight, and a fully covered behaviour adds no separate requirement. This reduction applies only to coherence; accessibility uses all retained proposal rows.

Let $M = |\bigcup_{a\in\mathcal{B}} S_a|$ be the number of observed solved cases and $n_i = |\{a \in \mathcal{B} : i \in S_a\}|$. Each solved case distributes one unit of weight across the behaviours that solve it:

$$v_i = \frac{1}{n_i} \quad (n_i > 0), \qquad W_a = \sum_{i\in S_a} v_i, \qquad \sum_{a\in\mathcal{B}} W_a = M.$$

Never-solved cases receive zero weight. For $M > 0$, define containment and CARAT-C by

$$s_{ab} = \frac{|S_a \cap S_b|}{\min(|S_a|, |S_b|)}, \qquad C = \frac{1}{M^2} \sum_{a,b\in\mathcal{B}} W_a W_b s_{ab}.$$

The pair score is the fraction of the smaller solved set also solved by the other behaviour; unlike the disagreement measure for classifier ensembles (Kuncheva & Whitaker, 2003), it treats a nested pair as fully compatible. Behaviours that differ on only a few cases show no observed conflict between the biases they carry, so containment reads them as compatible; solved sets that stay apart indicate biases that exclude one another. The sum includes ordered pairs and self-pairs. We report CARAT-C (C) with M: compatibility over observed successes can be high even when task accuracy is low. If $M = 0$, coherence is undefined. For example, $S_a = \{1, 2, 3\}$ and $S_b = \{1, 2, 4\}$ give $W_a = W_b = 2$, $M = 4$ and $C = 5/6$; adding a prompt that solves all four cases leaves one maximal behaviour and gives $C = 1$.

4.3 Metric properties

To measure accessibility, we require three properties. *Origin-freeness:* the reading must not depend on a starting or reference prompt; subtracting any fixed prompt's outcomes from every row cancels in $d_{jk,i}$ and leaves every ordering decision unchanged. *Proportional response:* halving the reversal rate must raise the reading by the same amount at any rate; with the declared units, this fixes the logarithm in $A(n)$, so halving either a 20% or a 2% rate adds $\frac{1}{2}\log_{10} 2 \approx 0.151$. *Declared boundaries:* $A(n) = 1$ at a 1% reversal rate and can exceed 1; with no clipping or added epsilon, zero observed reversals give $+\infty$, and no evaluable repetition leaves the reading undefined. Appendix A.2 proves these properties (Propositions A.2–A.4) and gives the boundary conventions.

To measure coherence, we require four properties. *Origin-freeness:* no reference prompt's outcomes enter the weights, the containment scores or M. *Redundancy invariance:* adding a prompt whose solved set duplicates or is contained in a retained one, adding cases no prompt solves, or reordering prompts or cases leaves C unchanged. *Interpretable endpoints:* for $M > 0$, $1/M \leq C \leq 1$, with $C = 1$ exactly when one retained prompt solves every case solved anywhere in the collection; equivalently, $1/C$, the order-2 similarity-sensitive diversity of the behaviours (Leinster & Cobbold, 2012), is an effective number of distinct behaviours between 1 and M. *Locality:* when every behaviour lies within one of several disjoint case blocks, changing the behaviours inside one block leaves the other blocks' contributions unchanged, provided that block's solved cases stay the same. The endpoints concern the finite collection and make no claim about unseen cases or undiscovered prompts. Appendix A.3 proves these properties (Lemma A.6 and Propositions A.7–A.12), including why locality needs both of its conditions.

4.4 Computing and reporting the readings

Freeze the matrix and its identities, estimate reversal rates at the declared call sizes and apply the fixed log transform, then compute coherence by the exact weighted pair sum; Algorithms 1 and 2 in Appendix A.4 give both procedures. Report CARAT-A with n, reference size h, the reversal rate and comparison support, and CARAT-C with M, identifying the answerer, reflector, task cases and prompt collection. Higher scores mean fewer conditional reversals or greater compatibility of observed successes, respectively; the two scales should not be compared numerically. Both calculations reuse recorded outcomes without further model inference, and each source of randomness is assessed by its own repetition (Appendices B.3 and D.2).

5 Experimental Design

5.1 Task conditions and prompt optimization method selection

We evaluate CARAT on ten dataset-represented tasks with Qwen3-8B fp16 (Yang et al., 2025) as both answerer and reflector, with thinking disabled (decoding settings in Appendix C): HotpotQA (Yang et al., 2018), MMLU-Pro (Wang et al., 2024b), ARC-Challenge (Clark et al., 2018), GSM8K (Cobbe et al., 2021), an interleaved HotpotQA+MMLU-Pro mixture, MoreHopQA (Schnitzler et al., 2024), CondaQA (Ravichander et al., 2022), TAT-QA (Zhu et al., 2021), CaseHOLD (Zheng et al., 2021) and Iris (Fisher, 1936). They span multi-hop reading, multiple-choice reasoning, arithmetic, negation, table-and-text numerical reasoning, legal holding selection and tabular classification, each scored by a binary answer-matching rule; the mixture draws half its cases from each component under one shared program. Each setup has 150 proposal cases and 300 evaluation cases, fixed as 150 select and 150 measure cases for optimizer comparisons. Iris uses a 50/50/50

frame with 100 evaluation cases, and its reference and call sizes are scaled to it. Appendix C gives the per-task programs, scoring rules, model roles and runtime setup.

We execute MRFS three times per setup, yielding thirty draws with regenerated roots, revisions and answers. Each execution initiates 16 roots from ten random proposal cases apiece and the program's introspected structure. Each root permits two revisions, using fresh 16-case feedback batches and displaying at most ten failures. A valid revision becomes the next parent without requiring improvement; a lineage stops on a failure-free batch or an unobtainable revision. Every retained proposal is evaluated on all evaluation cases. Appendix B gives generation, parsing and cost details.

We include five prompt optimization methods—GEPA (Agrawal et al., 2026), COPRO (Khattab et al., 2024), MIPROv2 (Opsahl-Ong et al., 2024), SIMBA (DSPy, 2025) and ACE (Zhang et al., 2026a)—to test whether the task patterns persist under different proposal and selection policies. Each method uses three seeds per task, with the same answerer–reflector pair and proposal/select/measure split. Their policies differ: trace reflection with minibatch acceptance (GEPA), rewriting without observing cases (COPRO), proposals from summaries and bootstrapped demonstrations (MIPROv2), rules from contrasting trajectories (SIMBA) and an evolving playbook (ACE); they concentrate exploration differently (§3.1; Appendix C.4).

Collections contain evaluated instructions, kept or discarded, except that GEPA contributes accepted candidates only. Its frequent, inexpensive minibatch proposals make evaluating every rejected proposal on the measure cases more costly than the search itself; comparisons using its collection retain this disclosure. Appendix D defines the cell statuses (no pool, undefined, missing). SIMBA uses DSPy's native rule policy (Khattab et al., 2024) on six tasks; on MoreHopQA, CondaQA, TAT-QA and CaseHOLD it uses a binary-contrast variant that draws rules from pairs of trajectories scored one and zero (Appendix C). Appendix C records per-cell reflection allowances, guard versions, retries and failure accounting.

5.2 Persistence across the task distribution

CARAT is computed on a finite sample of cases, but it is meant to describe the entire case distribution the task represents. We test this directly. Each task's 300 evaluation cases are split at random into 200 cases on which both metrics are computed and 100 held-out cases, drawn from the same distribution but never used by the metrics, on which they are recomputed. If a reading describes the distribution rather than the cases scored, the two sides agree in value. Source groups stay whole and the mixture stays balanced by source. The split is fixed before its results are inspected and shared by a task's three MRFS draws, and 100 further random splits show whether it is typical. Both sides use 300 accessibility partitions at reference/call sizes 50/25, the full-set proportions scaled to 100 cases, and coherence uses every case on its side. Iris, with 100 evaluation cases, is excluded.

5.3 Consistency across prompt optimization methods

The held-out check keeps the search procedure fixed, so we also compare the MRFS collections with those of the five optimizers in two ways: whether MRFS represents the task orderings and values those collections share, and whether CARAT-A and CARAT-C persist when the search changes. Both use the common measure cases, which were not consulted during optimization, with accessibility read at settings matched to each optimizer (Appendix D).

For the first comparison, we order tasks by each procedure's three-seed mean readings. Each pairwise comparison includes only tasks with three usable seeds under both procedures and counts the share of task pairs their orderings place alike. Observation 2 compares each procedure's agreement with the other five, pooled over its pairings; because the pairings share tasks, these averages are descriptive.

For the second comparison, accessibility describes comparisons available within a search, so we keep each collection separate: for MRFS and each optimizer, we read CARAT-A per task in each of its three runs and test the numerical distribution alignment of these readings with MRFS's by Pearson correlation over tasks, with placements in the upper and lower halves of each run's task ranking as a summary of the ordering. For coherence we broaden exploration by pooling each task's collections across methods and seeds: 300 archive draws of 48 prompts (the MRFS budget), balanced across method–seed cells and without replacement within cells, each scored by CARAT-C

on the same cases as the MRFS samples and aligned with the MRFS values in the same way. All ten tasks enter this pooled analysis; missing and no-pool cells contribute no prompts, while all-zero collections remain eligible. Observation 3 reports these comparisons, including the mixture against its components; the archive draws measure sensitivity to the recorded collection rather than independent optimization runs.

To test whether CARAT-A also characterizes the methods, we read every optimizer's collections at a common call of 37 cases (12 on Iris), rank the five optimizers by three-seed mean CARAT-A on each task where all five have three usable seeds, and measure how consistently this method order repeats across tasks by Kendall's W with a permutation test; the same statistic on the held-out accuracy of each method's final prompt tests whether the order follows accuracy; Observation 4 reports this test.

6 Validity Checks

Observation 1: CARAT readings persist across the entire task distribution. Readings computed on 200 cases hold on the 100 cases the metrics never used. On the fixed split, CARAT-A and CARAT-C differ between the two sides by 0.024 and 0.017 on average across tasks, against between-task SDs of 0.312 and 0.069; across 100 further splits, every task's mean difference lies within 0.009 of zero (Appendix D.2). A CARAT reading therefore describes the entire case distribution the task represents, not only the cases on which it was computed.

$\lvert\Delta\rvert$	HotpotQA	MMLU-Pro	ARC-Challenge	GSM8K	Mixture	MoreHopQA	TAT-QA	CaseHOLD	CondaQA	Mean	Between-task SD
CARAT-A	0.035	0.034	0.028	0.017	0.008	0.004	0.017	0.041	0.035	0.024	0.312
CARAT-C	0.015	0.014	0.011	0.012	0.007	0.009	0.036	0.017	0.036	0.017	0.069

Table 1: **CARAT readings persist across the task distribution.** $\lvert\Delta\rvert$: absolute difference between three-draw mean readings on the 100 held-out and 200 computed cases (fixed split; Appendix D.2).

Observation 2: MRFS is the most representative across prompt optimization methods. An optimizer's collection can order tasks by where its search policy leads as well as by the task. MRFS is tied to no starting prompt or search policy, so its ordering should be the one that differently directed collections share. Table 2 reports, for each procedure, the average agreement of its task ordering with the orderings of the other five. On both metrics, the MRFS ordering agrees most with the others: 80.4% of task pairs for accessibility, against 70.3% to 77.1% for the optimizers, and 92.0% for coherence, against 85.7% to 88.9%.

Metric	MRFS	GEPA	COPRO	MIPROv2	SIMBA	ACE
CARAT-A	**80.4%**	73.4%	70.3%	74.4%	77.1%	73.9%
CARAT-C	**92.0%**	86.9%	86.0%	88.9%	85.7%	87.9%

Table 2: **Average agreement of each procedure's task ordering with the other five**: shares of task pairs ordered alike by three-seed mean readings (not metric scores), pooled over each procedure's five pairings at matched accessibility read points. Tasks need three usable seeds: COPRO excludes CondaQA; SIMBA excludes ARC-Challenge and CondaQA. These comparisons were added after inspecting per-method results; pairwise counts in Appendix D.

Observation 3: CARAT-A and CARAT-C show consistency over prompt optimization methods. Each optimizer's collection carries its own search characteristics, yet its CARAT-A readings stay tied to the MRFS values (Figure 3a): per-task means correlate with MRFS at Pearson $r = 0.64$ to 0.84, and the average of the five optimizers at $r = 0.93$. Task rankings stay relatively stable: HotpotQA, TAT-QA and MoreHopQA fall in the upper half of their runs' rankings in 46 of 54 placements, and MMLU-Pro, ARC-Challenge and CaseHOLD in the lower half in 47 of 51. CARAT-C shows the same alignment under pooling: prompts pooled across optimizers and seeds reproduce the MRFS values ($r = 0.98$) and preserve the ordering on all 45 task pairs (Figure 3b). The value correlation therefore holds despite each method's characteristic search.

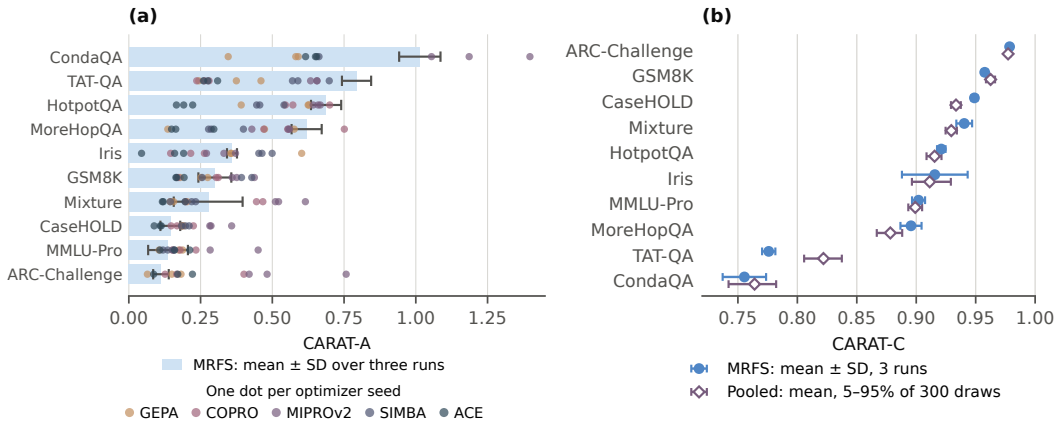

Figure 3: **Task-level consistency of CARAT-A and CARAT-C.** (a) Seeds at their method's read point; levels between procedures are descriptive. (b) Pooled draws take 48 recorded optimizer prompts, balanced over method–seed cells; tasks in MRFS-mean order. Both analyses were completed after inspecting per-method results; rank counts and archive details are in Appendix D.

Pooled exploration also reproduces the constructed mixture's coherence relation to its components on the common measure cases.

Observation 4: Prompt optimization methods display consistent divergence in CARAT-A signal across datasets. CARAT-A measures the accessibility of the biased prompt space each method searches, independently of the accuracy the method reaches. With every method evaluated on the same cases at a common call size, the five optimizers keep nearly the same accessibility order across the eight tasks all of them cover (Kendall's $W = 0.54$, permutation $p < 0.001$; Table 3), whereas the order of their final prompts' held-out accuracy does not repeat ($W = 0.07$, $p = 0.71$; Appendix D.6). We find that MIPROv2's collections read the highest accessibility on most datasets (six of eight; Table 3) while still ordering the tasks like MRFS ($r = 0.81$). In its search space, the evidence at hand therefore locates the truly better prompt more often, signalling a possibly more reliable reflective strategy.

	MIPROv2	COPRO	SIMBA	GEPA	ACE
Mean rank (1 = most accessible)	1.4	2.9	2.9	3.3	4.6
Mean CARAT-A	0.467	0.333	0.353	0.311	0.174

Table 3: **Accessibility of each optimizer's generated prompts.** Three-seed means at a common 37-case call (Iris 12) on the eight tasks where all five have three usable seeds; ranks are per task.

7 Conclusion

We introduced CARAT, two task signals for reflective prompt optimization, and Multi-Root Full-Retention Sampling (MRFS). CARAT-A measures whether prompt preferences persist across disjoint case samples; CARAT-C, whether observed successes are compatible with a shared prompt. Both persist across the entire task distribution and are consistent across prompt optimization methods on the same task, and MRFS represents the prompt space well: differently directed optimizers share its task ordering. Read on each optimizer's own prompts, CARAT-A also measures the accessibility of the prompt space the method searches, independently of the accuracy it reaches, so it can complement accuracy in evaluating prompt optimization methods. By reading a sampled prompt collection rather than one optimizer's returned prompt, CARAT quantifies the task suitability that run-level evaluations leave open, putting the field on a firmer footing. We look forward to seeing future work build a prompt optimization metric on top of the CARAT-A signal.

AI USE STATEMENT

Generative AI tools assisted with developing the conceptual framework and hypotheses, feedback on methodology and experiments, code implementation and review, data processing and result interpretation, mathematical checks and proof drafting, literature review, figures, and drafting and editing this manuscript. The authors take responsibility for the final claims, text, code, and figures. Appendix A gives proofs and numerical checks; Appendices B–D document the experimental procedures and results.

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. In *International Conference on Learning Representations*, 2026.

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, Samira Ebrahimi Kahou, Vincent Michalski, Tal Arbel, Chris Pal, Gaël Varoquaux, and Pascal Vincent. Accounting for variance in machine learning benchmarks. In *Proceedings of Machine Learning and Systems*, volume 3, 2021.

Dallas Card, Peter Henderson, Urvashi Khandelwal, Robin Jia, Kyle Mahowald, and Dan Jurafsky. With little power comes great responsibility. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 9263–9274, 2020.

Gavin C. Cawley and Nicola L. C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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. *arXiv preprint arXiv:2110.14168*, 2021.

DSPy. SIMBA: Stochastic introspective mini-batch ascent. DSPy optimizer documentation, 2025. URL `https://dspy.ai/api/optimizers/SIMBA/`.

Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7 (2):179–188, 1936.

Or Honovich, Uri Shaham, Samuel R. Bowman, and Omer Levy. Instruction induction: From few examples to natural language task descriptions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1935–1952, 2023.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. In *International Conference on Learning Representations*, 2024.

Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207, 2003.

Tom Leinster and Christina A. Cobbold. Measuring diversity: The importance of species similarity. *Ecology*, 93(3):477–489, 2012.

Shiyan Liu, Qifeng Xia, Qiyun Xia, Yisheng Liu, Xinyu Yu, and Rui Qu. Reflection in the dark: Exposing and escaping the black box in reflective prompt optimization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pp. 76–109, 2026.

Felipe Maia Polo, Ronald Xu, Lucas Weber, Mírian Silva, Onkar Bhardwaj, Leshem Choshen, Allysson Flavio Melo de Oliveira, Yuekai Sun, and Mikhail Yurochkin. Efficient multi-prompt evaluation of LLMs. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

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 *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, 2024.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Abhilasha Ravichander, Matt Gardner, and Ana Marasović. CONDAQA: A contrastive reading comprehension dataset for reasoning about negation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 8729–8755, 2022.

Subhash Bangalore Satheesha, Nirvik Pande, Deepthi Duddempudi, and Bharath Dandala. RLMOpt: Adaptive prompt optimization via recursive language models. *arXiv preprint arXiv:2608.10471*, 2026.

Julian Schnitzler, Xanh Ho, Jiahao Huang, Florian Boudin, Saku Sugawara, and Akiko Aizawa. MoreHopQA: More than multi-hop reasoning. *arXiv preprint arXiv:2406.13397*, 2024.

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. *arXiv preprint arXiv:2402.03300*, 2024.

Ruochen Wang, Sohyun An, Minhao Cheng, Tianyi Zhou, Sung Ju Hwang, and Cho-Jui Hsieh. One prompt is not enough: Automated construction of a mixture-of-expert prompts. In *Proceedings of the 41st International Conference on Machine Learning*, 2024a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2024b.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2022.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, 2026a.

Xing Zhang, Guanghui Wang, Yanwei Cui, Wei Qiu, Ziyuan Li, Bing Zhu, and Peiyang He. Prompt optimization is a coin flip: Diagnosing when it helps in compound AI systems. *arXiv preprint arXiv:2604.14585*, 2026b.

Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. When does pretraining help? Assessing self-supervised learning for law and the CaseHOLD dataset of 53,000+ legal holdings. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, pp. 159–168, 2021.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations*, 2023.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pp. 3277–3287, 2021.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. ProSA: Assessing and understanding the prompt sensitivity of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.

A DEFINITIONS AND JUSTIFICATION

This appendix supports Section 4. Appendix A.1 explains why CARAT does not estimate attainable improvement. Appendices A.2 and A.3 state and prove the properties of CARAT-A and CARAT-C used in Section 4.3 and fix the boundary conventions of Algorithms 1 and 2. Appendix A.5 records the numerical checks run before the proofs were written. Throughout, $X \in \{0,1\}^{m\times N}$ is one frozen response matrix with m retained prompts and N held-out cases, and every statement is conditional on X.

A.1 WHY CARAT DOES NOT MEASURE TASK HEADROOM

Remaining room below a score ceiling and improvement attainable through prompt optimization are different quantities. For a declared prompt family $\mathcal{P}$, population performance $J(p) \in [0,1]$ of prompt p on the task, and a reference prompt $p_0 \in \mathcal{P}$, the attainable improvement is

$$H(p_0, \mathcal{P}) = \sup_{p\in\mathcal{P}} J(p) - J(p_0).$$

The distance to the formal ceiling, $1 - J(p_0)$, is not the same quantity, because no admissible prompt need reach the ceiling. The experimental condition $\mathcal{T} = (A, R, \tau)$ of Section 3.1 fixes the answerer, the reflector and the task, and it does not designate p_0. A starting prompt belongs to an optimization procedure or to a chosen comparison; it is not an origin supplied by the task. Task headroom relative to an unspecified starting prompt is therefore not a defined task-level quantity. Choosing a reference prompt makes the comparison well defined and changes the question to improvement over that prompt.

Finite exploration does not resolve the attainable optimum either. A search observes a finite collection $\mathcal{S} \subsetneq \mathcal{P}$, and its best observed performance does not identify $\sup_{p\in\mathcal{P}} J(p)$ without further coverage assumptions or a valid bound.

Proposition A.1 (non-identification). Suppose the observations are the exact values $\{J(p) : p \in \mathcal{S}\}$, that $\max_{p\in\mathcal{S}} J(p) < 1$, and that J is otherwise unrestricted on $\mathcal{P} \setminus \mathcal{S} \neq \varnothing$. Then $\sup_{\mathcal{P}} J$ is not identified, and neither is $H(p_0, \mathcal{P})$ for any $p_0 \in \mathcal{S}$.

Proof. Let $J^\star = \max_{\mathcal{S}} J$ and fix $p^* \in \mathcal{P} \setminus \mathcal{S}$. Define $J_1 = J$ on $\mathcal{S}$ and $J_1 = J^\star$ elsewhere, and $J_2 = J_1$ except $J_2(p^*) = 1$. Both functions reproduce every observation, yet $\sup_{\mathcal{P}} J_1 = J^\star < 1 = \sup_{\mathcal{P}} J_2$. Since $J_1(p_0) = J_2(p_0)$ for $p_0 \in \mathcal{S}$, the two values of $H(p_0, \mathcal{P})$ differ by $1 - J^\star > 0$. □

The condition is that admissible unevaluated prompts remain; the argument does not concern exhaustive search of a known finite family, and it does not exclude estimation under added assumptions. A search that finds no improvement shows only that the procedure did not find one under its evidence and budget, and repeated failures can share the same exploration limits (Section 3.1). A successful search establishes an achieved improvement over a declared reference, not how much improvement remains. With exact population values and $p_0 \in \mathcal{S}$, $\max_{\mathcal{S}} J - J(p_0)$ is a lower bound on $H(p_0, \mathcal{P})$; held-out estimates of these values, especially after selection, require uncertainty treatment and are not exact bounds.

We therefore use observed performance only with its evaluation conditions explicit, as for the held-out accuracies of the optimizer runs in Table 13; an improvement over a reference prompt remains a valid experimental summary only where that reference is declared. CARAT-A and CARAT-C are origin-free (Proposition A.2 and Appendix A.3) and conditional on the declared observation process, and neither is an estimate of attainable improvement. We do not use a chosen reference prompt's observed improvement as an entry condition for task suitability, and a low observed improvement does not rule out useful future exploration.

A.2 CARAT-A: PROPERTIES AND CONVENTIONS

Fix X, sizes $h \geq 2$ and $n \geq 1$ with $h + n \leq N$, and T repetitions whose partitions (R_r, L_r) are drawn as in Algorithm 1. For a pair $j < k$ in repetition r, write $\rho = d_{jk,R_r}$, $\ell = d_{jk,L_r}$ and $\widehat{\mathrm{se}}$ for the sample standard error of $\{d_{jk,i} : i \in R_r\}$, whose variance uses divisor $h - 1$. The pair is *ordered* when $|\rho| > 1.96\,\widehat{\mathrm{se}}$, *usable* when it is ordered and $\ell \neq 0$, and *reversed* when it is

usable and $\ell\rho < 0$. O_r, U_r and E_r count ordered, usable and reversed pairs; $\widehat{D}_r(n) = E_r/U_r$ for $r \in \mathcal{V}_n = \{r : U_r > 0\}$, and $\widehat{D}(n)$ is their mean.

Proposition A.2 (origin-freeness). For any $c \in \mathbb{R}^N$, replacing every row X_j by $X_j - c$ leaves every ordering, tie and reversal decision, the counts O_r, U_r and E_r, the set $\mathcal{V}_n$, $\widehat{D}(n)$ and $A(n)$ unchanged, given the same partitions. In particular, subtracting the outcomes of any reference prompt, such as a shipped instruction, changes no reading.

Proof. $(X_{j,i} - c_i) - (X_{k,i} - c_i) = X_{j,i} - X_{k,i} = d_{jk,i}$ for every case, and every quantity above is a function of the paired differences alone. □

The tie tests compare means with zero exactly. With binary outcomes and integer shifts the cancellation is exact in floating-point arithmetic; an arbitrary real shift can replace an exact zero by a rounding residue, so shifts are applied in exact arithmetic.

Proposition A.3 (range). If $\mathcal{V}_n \neq \varnothing$, then $0 \leq \widehat{D}(n) \leq 1$ and $0 \leq A(n) \leq +\infty$. $A(n) = 0$ exactly when every usable pair reverses in every evaluable repetition; $A(n) = 1$ exactly when $\widehat{D}(n) = 0.01$, and $A(n) > 1$ exactly when $\widehat{D}(n) < 0.01$; $A(n) = +\infty$ exactly when no usable pair reverses in any evaluable repetition.

Proof. Each $\widehat{D}_r(n)$ is a proportion, so their mean lies in $[0, 1]$ and equals 0 only if every term does. The map $D \mapsto -\frac{1}{2}\log_{10} D$ is continuous and strictly decreasing from $(0, 1]$ onto $[0, \infty)$, and the definition assigns $+\infty$ to $D = 0$. □

Proposition A.4 (proportional changes and the fixed logarithm). For $D, cD \in (0, 1]$, $A(cD) - A(D) = -\frac{1}{2}\log_{10} c$, whatever D. Conversely, let $f : (0, 1] \to \mathbb{R}$ be differentiable with $f(cD) - f(D)$ depending on c alone whenever $D, cD \in (0, 1]$. Then $f(D) = a \ln D + b$, and the declared units $f(1) = 0$ and $f(0.01) = 1$ give $f(D) = -\frac{1}{2}\log_{10} D$.

Proof. The first statement is direct. For the converse, write $g(c) = f(cD) - f(D)$, differentiable because f is. Differentiating in c at $c = 1$ gives $Df'(D) = g'(1) =: a$ for every $D \in (0, 1]$, so $f(D) = a \ln D + b$. Then $f(1) = 0$ gives $b = 0$, and $f(0.01) = 1$ gives $a = 1/\ln 0.01 = -1/(2 \ln 10)$. □

Halving any positive rate therefore adds $\frac{1}{2}\log_{10} 2 = 0.1505$ to $A(n)$; on the complement $1 - D$, halving 0.20 or 0.02 adds 0.10 or 0.01. The anchors are declared units, not fitted parameters, and the uniqueness holds for the proportional-change requirement. The transformation preserves order and adds no comparison support. It also does not stabilize variance by itself. If a rate were estimated from q independent comparisons, the delta method would give $\operatorname{Var} A \approx (1 - D)/(4(\ln 10)^2 qD)$, which grows as D falls; a multiplicative noise law shared across tasks would instead give a constant variance on the log scale. CARAT's overlapping pairs satisfy neither model by construction, and three draws per task do not establish either.

Boundary conventions. Algorithm 1 and the readings in Sections 5–6 use the following conventions. (i) *Reference ties.* A pair with $\rho = 0$ is never ordered, since the margin inequality is strict. A pair with $\rho \neq 0$ and $\widehat{\mathrm{se}} = 0$, meaning all h reference differences are equal, is ordered. (ii) *Call ties.* An ordered pair with $\ell = 0$ makes no call claim and leaves both U_r and E_r. (iii) *Evaluable repetitions.* Repetitions with $U_r = 0$ are dropped and the rest receive equal weight. This differs from the pooled ratio $\sum_r E_r / \sum_r U_r$ whenever the U_r vary. (iv) *Undefined readings.* If $\mathcal{V}_n = \varnothing$, the reading is undefined and is reported with its counts. This occurs when $m < 2$, when all rows are identical, or when no pair is both ordered and untied in any repetition. (v) *Zero reversals.* $\widehat{D}(n) = 0$ gives $A(n) = +\infty$, with no clipping and no added constant, and is reported with O_r, U_r and $|\mathcal{V}_n|$. (vi) *Sizes.* $h \geq 2$ is required for the sample standard error, and $h + n \leq N$ keeps the subsets disjoint. Section 5 uses $h = \lfloor N/2 \rfloor$ and $n \leq \lfloor N/4 \rfloor$: 150 and 75 on 300 cases, 75 and 37 on 150 measure cases, 50 and 25 on Iris's 100 cases, and 25 and 12 on its 50 measure cases. (vii) *Reproducibility.* Given X, a reading is a deterministic function of h, n, T and the generator seed, which are retained with it.

A worked reversal. Suppose p_a solves three of four call cases and p_b solves two, and that on 150 reference cases their solved sets are nested, with p_a solving 75 and p_b solving 105. The paired reference differences are -1 on 30 cases and 0 on 120, so $\rho = -0.2$, the sample variance is $24/149 \approx 0.161$ and $\widehat{\mathrm{se}} = \sqrt{24/(149 \cdot 150)} \approx 0.0328$. The margin $1.96\,\widehat{\mathrm{se}} \approx 0.0642$ is below 0.2, so the pair is ordered in favour of p_b. On the call, $\ell = 3/4 - 2/4 = 0.25$ has the opposite sign, and the pair contributes one reversal; equal call accuracies would give $\ell = 0$ and exclude it.

Remark A.5 (dependence). Two dependences qualify the average. First, pairs sharing a prompt are not separate evidence. If all $\binom{m}{2}$ pairs are usable and the call agrees with the reference except that one prompt moves from the top of the ordering to the bottom, $m - 1$ pairs reverse and $\widehat{D}_r = 2/m$. Second, conditional on X the call and reference means of a pair are negatively correlated. For disjoint uniform subsets of sizes n and h drawn without replacement from N fixed values with variance S^2 (divisor $N - 1$), $\mathrm{Var}\,\bar{y}_L = (1/n - 1/N)S^2$ and $\mathrm{Cov}(\bar{y}_L, \bar{y}_R) = -S^2/N$, so

$$\mathrm{corr}(\ell, \rho) = -\sqrt{\frac{nh}{(N-n)(N-h)}}.$$

At $N = 300$, $h = 150$ and $n = 75$ this is -0.577. A call equal to the reference's complement ($n = N - h$) would have correlation -1 and would be determined by the reference; drawing the call from the remainder without exhausting it avoids that. Repeated partitions of one matrix therefore measure Monte Carlo variation given X (Appendix B.3), not sampling of new cases.

A.3 CARAT-C: PROPERTIES AND CONVENTIONS

Let $S_1, \ldots, S_m$ be the retained solved sets and $\mathcal{B}$ the distinct maximal behaviours left by the reduction of Section 4.2, with $K = |\mathcal{B}|$, $\mathcal{U} = \bigcup_{a \in \mathcal{B}} S_a$ and $M = |\mathcal{U}|$. For $M > 0$, n_i, W_a and s_{ab} are as defined there, and $p_a = W_a/M$.

Lemma A.6 (reduction). $\mathcal{U} = \bigcup_j S_j$, and every S_j is contained in some S_a with $a \in \mathcal{B}$. If $M > 0$, every S_a with $a \in \mathcal{B}$ is nonempty, and no two distinct behaviours are nested, so $s_{ab} < 1$ for $a \neq b$.

Proof. Under inclusion, every set of a finite family lies in a maximal one, obtained by following a strictly increasing chain. Collapsing identical sets makes the maximal sets distinct, and distinct maximal sets cannot be nested. If $M > 0$, an empty set is strictly contained in a nonempty one and is discarded. For $a \neq b$, $|S_a \cap S_b| = \min(|S_a|, |S_b|)$ would place the smaller set inside the other. □

Proposition A.7 (weights). For $M > 0$, every $W_a > 0$ and $\sum_{a \in \mathcal{B}} W_a = M$. Hence p is a probability vector and $C = \sum_{a,b} p_a p_b s_{ab} = \mathbb{E}[s_{AB}]$ for independent behaviours A and B drawn with probabilities p.

Proof. $\sum_a W_a = \sum_a \sum_{i \in S_a} 1/n_i = \sum_{i \in \mathcal{U}} \sum_{a \ni i} 1/n_i = \sum_{i \in \mathcal{U}} 1 = M$. Since $n_i \leq K$ and $S_a \neq \varnothing$, $W_a \geq |S_a|/K > 0$. □

Proposition A.8 (upper endpoint). For $M > 0$, $C \leq 1$, with equality exactly when $K = 1$, equivalently when some retained prompt solves every case in $\mathcal{U}$.

Proof. Since $s_{ab} \leq 1$, $C \leq \sum_{a,b} p_a p_b = 1$. If $K \geq 2$, each off-diagonal term has $p_a p_b > 0$ and $s_{ab} < 1$ (Lemma A.6), so $C < 1$; if $K = 1$, $C = s_{aa} = 1$. By Lemma A.6, $K = 1$ means the single behaviour's solved set is $\mathcal{U}$, and that set is some retained prompt's solved set. Conversely, if $S_j = \mathcal{U}$, every other set is contained in S_j, so the reduction leaves S_j alone. □

Proposition A.9 (lower bounds). For $M > 0$, $C \geq 1/M$, with equality exactly when every behaviour solves a single case; the behaviours are then the M singletons. In addition, $C \geq \sum_a p_a^2 \geq 1/K$, so $C \geq 1/\min(K, M)$.

Proof. Let $u_a = \mathbf{1}_{S_a}/\sqrt{|S_a|} \in \mathbb{R}^{\mathcal{U}}$. Because $\min(x, y) \le \sqrt{xy}$, $s_{ab} \ge \langle u_a, u_b \rangle \ge 0$, and the weights are positive, so

$$M^2 C \ge \Big\| \sum_a W_a u_a \Big\|_2^2 \ge \frac{1}{M} \Big(\sum_a W_a \sqrt{|S_a|} \Big)^2 \ge \frac{M^2}{M} = M.$$

The middle step is the Cauchy–Schwarz inequality over the M coordinates of $\mathcal{U}$, and the last uses $|S_a| \ge 1$ with $\sum_a W_a = M$. Equality in the last step forces $|S_a| = 1$ for every a, because every $W_a > 0$. Conversely, distinct singletons are disjoint and each has weight one, so $C = M/M^2$. For the second bound, dropping the nonnegative off-diagonal terms gives $C \ge \sum_a p_a^2$, and the Cauchy–Schwarz inequality gives $\sum_a p_a^2 \ge (\sum_a p_a)^2/K = 1/K$. □

K can exceed M: the six two-case subsets of four cases are distinct maximal behaviours. The bound $1/M$ is then the stronger one.

Corollary A.10 (effective number of behaviours). Let $Z = (s_{ab})$, whose entries lie in $[0, 1]$ with unit diagonal. For relative abundances p and similarity matrix Z, the similarity-sensitive diversity of order $q \ne 1$ is ${}^qD^Z(p) = \big(\sum_{a:p_a>0} p_a (Zp)_a^{q-1} \big)^{1/(1-q)}$ (Leinster & Cobbold, 2012). At $q = 2$ with $p = W/M$, ${}^2D^Z(p) = (p^\top Z p)^{-1} = 1/C$. By Propositions A.8 and A.9, $1 \le 1/C \le \min(K, M)$, with $1/C = 1$ exactly for one behaviour and $1/C = M$ exactly for M singleton behaviours.

Proposition A.11 (invariances). (a) Adding a prompt whose solved set equals or is strictly contained in a retained solved set leaves $\mathcal{B}$, M and C unchanged. (b) Adding cases that no retained prompt solves leaves M and C unchanged. (c) C and M do not depend on the order of prompts or cases.

Proof. (a) The added set is collapsed or discarded, and it cannot contain a maximal set, so the maximal sets and $\mathcal{U}$ are unchanged. (b) Such cases have $n_i = 0$ and lie in no S_a, so they enter neither the weights, the containment counts nor M. (c) The definitions are symmetric in prompts and cases. □

By (b), task accuracy can fall without moving C: one prompt that solves one of 300 cases has $C = 1$ and $M = 1$. This is why CARAT-C is reported with M.

Proposition A.12 (locality). Partition the cases into disjoint blocks $T_1, \dots, T_q$ and suppose every retained solved set lies within a single block. Let $\mathcal{B}_t$ be the behaviours in T_t, M_t the number of solved cases in T_t, and C_t the coherence of block t's prompts alone when $M_t > 0$. Then

$$\mathcal{B} = \bigcup_t \mathcal{B}_t, \qquad C = \sum_{t:M_t>0} \Big(\frac{M_t}{M} \Big)^2 C_t.$$

If block t's prompts are replaced by prompts whose solved sets lie in T_t and M_t is unchanged, as when the block's solved-case union is unchanged, then M, every weight W_a and pair score s_{ab} with $a, b \notin \mathcal{B}_t$, and every other block's term $(M_u/M)^2 C_u$ are unchanged.

Proof. A nonempty set within T_t cannot be contained in a set within T_u for $u \ne t$, so identity and containment act within blocks and $\mathcal{B} = \bigcup_t \mathcal{B}_t$. Cross-block pairs are disjoint, so their $s_{ab} = 0$. For $i \in T_t$, n_i counts only behaviours in $\mathcal{B}_t$, so each W_a with $a \in \mathcal{B}_t$ equals its weight in block t's collection, and $\sum_{a \in \mathcal{B}_t} W_a = M_t$. Hence $M^2 C = \sum_t \sum_{a,b \in \mathcal{B}_t} W_a W_b s_{ab} = \sum_t M_t^2 C_t$. Replacing block t's prompts alters only $\mathcal{B}_t$ and its weights, and $M = \sum_u M_u$ is unchanged. □

Both conditions are needed. *Separation:* with blocks $\{1, 2\}$ and $\{3\}$, the prompts $\{1, 2, 3\}$, $\{1, 2\}$ and $\{3\}$ leave one behaviour and $C = 1$. Changing only the first prompt's outcomes on the first block, to $\{3\}$, keeps $M = 3$ but leaves behaviours $\{1, 2\}$ and $\{3\}$: $C = 5/9$, and the term of the behaviours in $\{3\}$ rises from 0 to $1/9$. *Fixed union:* with blocks $\{1, 2, 5\}$ and $\{3, 4\}$, the prompts $\{1\}$, $\{2\}$ and $\{3, 4\}$ give $M = 4$ and a second-block term of $(2/4)^2 = 1/4$. If the first prompt also solves case 5, $M = 5$ and that term becomes $4/25$, although nothing in the second block changed.

Examples and conventions. If the behaviours are pairwise disjoint, Proposition A.12 with one behaviour per block gives $C = \sum_a (|S_a|/M)^2$: two disjoint behaviours of sizes u and v give $(u^2 +$

Algorithm 1 CARAT-A computation

Require: Matrix X; sizes h, n; repetitions T
$h \geq 2$, $n \geq 1$, $h + n \leq N$
for $r = 1, \ldots, T$ **do**
  Draw h reference cases R_r uniformly
  Draw L_r: a uniform n-subset of the rest
  Both draws are without replacement
  $O_r \leftarrow 0$; $U_r \leftarrow 0$; $E_r \leftarrow 0$
  for each prompt pair $j < k$ **do**
    $d_i \leftarrow X_{j,i} - X_{k,i}$
    $\rho \leftarrow \mathrm{mean}_{i \in R_r} d_i$
    $\ell \leftarrow \mathrm{mean}_{i \in L_r} d_i$
    if $|\rho| > 1.96\, \widehat{\mathrm{se}}_{R_r}(d)$ **then**
      $O_r \leftarrow O_r + 1$
      if $\ell \neq 0$ **then**
        $U_r \leftarrow U_r + 1$
        $E_r \leftarrow E_r + \mathbf{1}\{\ell\rho < 0\}$
      end if
    end if
  end for
end for
$\mathcal{V} \leftarrow \{r : U_r > 0\}$
if $\mathcal{V} = \varnothing$ **then**
  return undefined, $\{O_r, U_r\}$, 0
end if
$\widehat{D} \leftarrow \mathrm{mean}_{r \in \mathcal{V}}(E_r/U_r)$
$A \leftarrow -\frac{1}{2} \log_{10} \widehat{D}$
return A, $\widehat{D}$, $\{O_r, U_r\}$, $|\mathcal{V}|$

Algorithm 2 CARAT-C computation

Require: The same response matrix X
$S_j \leftarrow \{i : X_{j,i} = 1\}$ for each prompt j
Collapse identical solved sets
Discard sets strictly contained in another
$\mathcal{B} \leftarrow$ indices of the remaining sets
$\mathcal{U} \leftarrow \bigcup_{a \in \mathcal{B}} S_a$; $M \leftarrow |\mathcal{U}|$
if $M = 0$ **then**
  return undefined, M
end if
for each solved case $i \in \mathcal{U}$ **do**
  $n_i \leftarrow |\{a \in \mathcal{B} : i \in S_a\}|$
  $v_i \leftarrow 1/n_i$
end for
for each behaviour $a \in \mathcal{B}$ **do**
  $W_a \leftarrow \sum_{i \in S_a} v_i$
end for
$C \leftarrow 0$
for each ordered pair $(a, b) \in \mathcal{B}^2$ **do**
  $s_{ab} \leftarrow \dfrac{|S_a \cap S_b|}{\min(|S_a|, |S_b|)}$
  $C \leftarrow C + W_a W_b s_{ab} / M^2$
end for
return C, M

Computing CARAT from one response matrix. Algorithm 1: CARAT-A averages reversal fractions equally across evaluable partitions, then applies the fixed log transform. O_r counts reference-ordered pairs before call-tie exclusion; U_r counts usable pairs. The reading is undefined if $\mathcal{V}$ is empty and is $+\infty$ if $\widehat{D} = 0$. Algorithm 2: CARAT-C reduces solved sets to distinct maximal behaviours, allocates each solved case's weight among them, and sums containment over ordered pairs including self-pairs. Both procedures reuse recorded outcomes; repeated sampling runs are reported with Table 14.

$v^2)/(u+v)^2$, which is $1/2$ for equal sizes and $5/8$ for $\{1\}$ and $\{2,3,4\}$. In Section 4.2's example, $\{1,2,3\}$ and $\{1,2,4\}$ have $n_1 = n_2 = 2$ and $n_3 = n_4 = 1$, so $W = (2,2)$, $M = 4$, $s_{ab} = 2/3$ and $C = (4 + 4 + 2 \cdot 4 \cdot 2/3)/16 = 5/6$; adding $\{1,2,3,4\}$ leaves one behaviour and $C = 1$. CARAT-C depends only on the retained prompts' solved sets: no reference prompt's outcomes enter the weights, the containment scores or M. It is undefined when $M = 0$, which occurs when no retained prompt solves any case. A collection with one nonempty behaviour, including a single retained prompt, has $C = 1$. The pair sum is exact, over ordered pairs with self-pairs, so no pair sampling enters. Weights reflect rarity, while containment uses ordinary case counts.

A.4 Extraction algorithms

Algorithms 1 and 2 give the two extraction procedures of Section 4 side by side; the boundary conventions of Appendix A.2 apply to both.

A.5 Numerical checks

Propositions A.2–A.12, the boundary conventions, the worked example and Remark A.5 were checked numerically before the proofs were written, against an independent implementation of Section 4 written from the definitions and against the analysis code used for all readings. The

independent implementation reproduces the analysis code on 3,000 random matrices for CARAT-C (1–25 prompts, 1–40 cases, solve rates 0.03–0.97; largest difference 1.6×10^{-15}, identical M and undefined cases) and on 250 random matrices for CARAT-A under identical partitions (identical reversal estimates and undefined cases). Coherence properties were then tested with exact rational arithmetic. Table 4 lists the checks; all passed, as did the thirteen fixtures of the analysis code.

Statement	Check	Result
Origin-freeness (A.2)	150 random matrices; binary, integer and exactly representable real shifts; every pair decision compared	No decision or reading changed
Range and scale (A.3, A.4)	106 finite readings; 10,000 pairs (D, c); anchors	Holds; halving adds 0.1505
Conventions (i)–(v)	Constructed cases for each	As stated
Mean of rates, convention (iii)	59 random matrices	Pooled ratio differs by up to 0.159
Worked example	Exact	$\widehat{\mathrm{se}} = 0.0328$; margin 0.0642; one reversal
Correlation (Remark A.5)	40,000 partitions at $(N, n, h) = (300, 75, 150)$, $(150, 37, 75)$, $(100, 25, 50)$, $(300, 150, 150)$	-0.577, -0.571, -0.583, -1.000 against -0.577, -0.572, -0.577, -1.000
Displaced prompt (Remark A.5)	$m = 4, 10, 48, 100$	$\widehat{D}_r = 2/m$ exactly
Weights, endpoints, bounds, diversity (A.6–A.10)	2,957 collections with $M > 0$, 73 with $K > M$	All exact
Expectation form (A.7)	40 collections, 20,000 sampled pairs each	Largest $\lvert z \rvert = 1.88$
Invariances (A.11)	1,464 random matrices	All exact
Locality (A.12)	1,499 block collections; the two counterexamples	Decomposition exact; $1 \to 5/9$; $1/4 \to 4/25$
Examples	Exact	$5/6$, 1, $1/2$, $5/8$, $(u^2 + v^2)/(u + v)^2$, $1/M$

Table 4: **Numerical checks of Appendix A.** Random matrices use varied numbers of prompts and cases and solve rates from 0.03 to 0.97; structured families add singletons, nested chains, all k-subsets and near-nested collections. Coherence checks use exact rational arithmetic.

B SAMPLING AND REPRODUCIBILITY

This appendix specifies MRFS, the sampling procedure of Section 3.2, as executed in Section 5: how proposals are generated and parsed, the decoding settings, seeds and case splits, the recorded hashes and the cost. Appendix C gives the task programs and runtime conditions.

B.1 GENERATION

The instrument is static. The same code, prompt templates and settings apply to every task; a task adapter supplies only the cases and the program. The program's stages are introspected and rendered as one sentence per stage, such as "Stage `answer` receives the fields question, context and must produce the fields reasoning, answer." No task description is authored. The 150 proposal cases (Iris: 50) are split in half: the first 75 supply root examples and the last 75 supply feedback minibatches (Iris: 25 each). Only proposal cases are shown to the reflector.

Each draw initiates 16 roots. For each root, the instrument draws ten cases from the root half and gives the reflector their questions and correct answers together with the stage sentences. For multiple-choice tasks the question includes its options; case contexts are not shown. The root prompt states the goal positively and prohibits nothing.

Each root may undergo up to two revisions. At each step, the current instruction runs on a fresh minibatch of 16 cases from the feedback half. The reflector receives the current instruction, the stage sentences, the numbers of correct and incorrect answers, and up to ten failures, each with its question, correct answer and produced answer cut at 200 characters. It returns a diagnosis and a revised instruction; the diagnosis is discarded. A valid revision becomes the next parent without requiring a higher score. A lineage stops early when its minibatch contains no failure or when a

revision cannot be obtained under the parsing rules of Appendix B.2; a root that cannot be obtained contributes no lineage. The design therefore yields at most 48 proposals.

Every retained proposal is then evaluated on all evaluation cases (300; Iris 100). The collection is frozen before this evaluation, and its outcomes guide no generation step. Across the thirty draws the instrument retained 1,341 proposals, 475 roots and 866 revisions, between 35 and 48 per draw (Table 6). Five root slots produced no root, all on TAT-QA; every other missing proposal comes from a lineage that stopped early.

The retained sample is conditional on the reflector, elicitation and sampling rules. Equal maximum opportunities per root do not imply uniform probabilities over prompt strings: roots can share evidence, revisions share their lineage's history, and longer realized lineages contribute more entries, so the retained rows are dependent observations. Algorithm 3 summarizes the evaluation; prompt and instance identities accompany the matrix so that subsequent comparisons use the same observed cases.

Algorithm 3 Construct the response matrix from a retained prompt sample.

Require: Frozen prompts $(p_1, \ldots, p_m)$; held-out instances $(z_1, \ldots, z_N)$; answerer A and task settings τ.
Ensure: Binary matrix X with ordered prompt and instance identities.

```
Allocate X ∈ {0,1}^{m×N}.
for j = 1, ..., m do
    for i = 1, ..., N do
        y_{j,i} ← A_τ(p_j, z_i)
        X_{j,i} ← s_τ(y_{j,i}, z_i)
    end for
end for
return X and the ordered prompt and instance identities.
```

Prompt templates. The two templates below are used verbatim for every task. Braces mark filled slots: examples are numbered "`Q:`/`A:`" pairs; failures add "`correct:`" and "`produced:`" lines; the key list is `["answer"]` for roots and `["diagnosis", "answer"]` for revisions on these single-stage programs. The style slot is empty for the elicitation used throughout.

Root prompt.

```
You are writing the instructions for a program that answers questions from a dataset.

Here are {k} example questions from that dataset, with their correct answers:

{examples}

The program's stages, with their input and output fields:

{spec}

Write one instruction for each stage. These instructions will be used to answer
OTHER questions from this same dataset, which you will not see. Write them so
they solve that dataset.{style}

Return STRICT JSON only, no prose, no code fence, with exactly these keys:
{keys}
```

Revision prompt.

```
These are the current instructions for a program that answers questions from a dataset.

{current}

The program's stages, with their input and output fields:

{spec}

Running these instructions on {n} questions produced {n_ok} correct and {n_bad}
incorrect answers. Here are the failures - the question, the correct answer, and
what the program actually produced:
```

```
{failures}

First write a diagnosis of what the current strategy gets wrong. Then revise the
instructions so they would handle failures like these without breaking what
already works. The diagnosis goes in its own field; the instructions are what
the program will run.

Return STRICT JSON only, no prose, no code fence, with exactly these keys:
{keys}
```

B.2 Parsing, redraw and integrity rules

Replies are requested as strict JSON keyed by stage name. The parser takes the first balanced JSON object in the reply and requires every requested key to hold a nonempty string. A reply that fails is requested again, up to five tries per slot; a slot that still fails ends its root or lineage, and the failure is logged.

Unwrap rule. Under instruments `aabb7490` and `ae0da5df`, a reply that wraps the instruction is accepted by a declared rule only: a list holding one string, or a string stored under the stage's non-final output field (`reasoning`) or under an explicit `instruction` or `instructions` key. A value under an input field, the mimicked answer or the longest string is never taken. Every unwrap is logged with the path taken and the first 300 characters of the raw reply. In the 18 draws under this rule, 162 of 816 retained prompts come from unwrapped replies, 2 to 17 per draw; Appendix C.3 reports the effect of removing them.

Specification echo. A reply containing its stage's rendered specification sentence, or the older signature line, is the pipeline's own text rather than an instruction. It is redrawn up to five times; if the echo persists, the last draw is retained and flagged, so no lineage disappears silently. Across the thirty draws, 61 redraws occurred and seven proposals were retained with the flag, six on CondaQA and one on TAT-QA.

Split integrity. Before a proposal is retained, the instrument searches it for gold answers of evaluation cases that the reflector could not have seen. A distinctive answer string, with at least eight characters and two words and containing a capital letter or digit, that occurs on word boundaries in the proposal and in none of the shown material voids the run. Other proposal content is admissible; how much shown material a proposal repeats is recorded, not filtered. The CondaQA instrument `ae0da5df` exempts the fixed labels YES, NO and DON'T KNOW from this check, because the labels occur in every split; nothing else changes.

Instrument	Parsing and checks	Draws
`a334bb70`	Rules above without the unwrap rule	HotpotQA, ARC-Challenge, GSM8K
`5c5e9b8a`	`a334bb70` with a serving-backend switch whose branch for our server is unchanged	MoreHopQA
`aabb7490`	Adds the unwrap rule and logs failed revisions	MMLU-Pro, Mixture, TAT-QA, CaseHOLD, Iris
`ae0da5df`	`aabb7490` with the CondaQA label exemption	CondaQA

Table 5: **Instrument versions.** Each hash covers the instrument code and its readout module; `ae0da5df` also covers the two repair files. Mixture denotes HotpotQA + MMLU-Pro.

B.3 Decoding, caching, seeds and splits

The reflector is Qwen3-8B fp16 with thinking disabled, temperature 1.0 and up to 4,000 output tokens, and each reflector call is a fresh request. The answerer runs the task program under DSPy 3.2.1 with thinking disabled, temperature 0.6, top-p 0.95 and up to 2,048 output tokens. DSPy's response cache is enabled, so within one run an identical request returns the stored answer: all 42 groups of duplicate instructions in the thirty draws have identical rows. Guarded runs start with an empty cache directory.

One seed, 20260827, fixes every task's case frame for all draws and optimizer runs. Draw seeds 0, 1 and 2 seed the generator that draws root examples and feedback minibatches. Model sampling is not seeded per request, so repeating a draw seed produces a new draw. Accessibility partitions and item subsamples use seeded generators whose seeds are recorded with each reading.

Every task uses 150 proposal, 150 select and 150 measure cases, except Iris (50 each). The select and measure cases form the evaluation cases. (a) *HotpotQA, MMLU-Pro, ARC-Challenge and GSM8K.* The task takes 450 cases as the prefix of a seeded shuffle of the source split, which is nested across sizes: the first 300 are evaluation cases and the next 150 proposal cases. A second seeded shuffle splits the evaluation cases into select and measure halves. (b) *Mixture.* Each component loads exactly as its single task. The mixture takes the first 150 evaluation cases and 75 proposal cases of each, interleaves the two sources case by case, and splits the evaluation cases into halves stratified by source. (c) *MoreHopQA, CondaQA, TAT-QA and CaseHOLD.* Source groups (source question, passage, table and citing context, respectively) are shuffled and allocated so that no group crosses the proposal, select and measure boundaries. Each case comes from a different group where enough groups exist. MoreHopQA has 307 source questions, so its groups are allocated in proportion and variants are taken in rounds within each side; its 300 evaluation cases span 204 groups, which item resampling keeps whole. (d) *Iris.* Each of the 150 rows is used once.

Units of uncertainty. Uncertainty depends on what is repeated (Bouthillier et al., 2021). Repeated call/reference draws quantify Monte Carlo variation within a fixed matrix; coherence's exact pair sum has no corresponding pair-sampling error. Item subsampling without replacement assesses sensitivity to cases conditional on the recorded prompts, with task groups preserved and absolute call/reference sizes held fixed where feasible. Balanced archive draws assess sensitivity to the composition of the observed optimizer collection. Each replicate produces its own reversal estimate before the fixed log transform is applied. Independent MRFS runs regenerate prompts and recorded answers and assess the stability of the complete measurement procedure; their per-run log readings are summarized by means and sample standard deviations when all readings are finite. Three runs provide descriptive spread without precisely separating proposal and answering randomness. Each reading retains its resampling unit, model and generation settings, seeds, proposal lineage and analysis version.

B.4 Provenance and hashes

Recorded provenance identifies how each proposal arose; identical text may arise under another procedure. Each matrix file stores the retained prompts with their instruction texts, parents and depths, the outcome of every prompt on every evaluation case, the select and measure positions, the rendered program specification, the redraw log and the instrument hash. Guarded runs also record the server identity, including the model digest `0b358e6e` and the context length, and a ledger of every request with its verified input fit; the unguarded historical draws predate this check. Table 6 lists the thirty draws with SHA-256 prefixes of their matrix files. The prepared case files of the four added tasks and Iris are pinned by SHA-256 (Appendix C.2); the other four tasks load fixed dataset splits from a local cache. Each reading records the version of the analysis code; the full-set readings of Table 14 used version `7234365c`.

Task	Ans.	Refl.	Instrument	Context	K	Matrix SHA-256
HotpotQA	Q3	Q3	a334bb70	8,192	48/48/48	0b64f1fc/985b119b/b832c2dc
MMLU-Pro	Q3	Q3	aabb7490	40,960	48/48/48	aa8ba11b/c1c2cd96/5fd68de9
ARC-Challenge	Q3	Q3	a334bb70	8,192	37/39/39	ea9c67a9/17719f0f/e9fa1367
GSM8K	Q3	Q3	a334bb70	8,192	46/36/40	b25bbc46/5dc3dd7e/80117a3a
Mixture	Q3	Q3	aabb7490	40,960	46/48/48	2f208fcc/57ee8115/a35bccb4
MoreHopQA	Q3	Q3	5c5e9b8a	8,192	48/48/48	dcd17b08/6312daef/50e7f6c5
TAT-QA	Q3	Q3	aabb7490	40,960	37/45/44	81eaf1fd/64f297cd/8591095a
CaseHOLD	Q3	Q3	aabb7490	40,960	48/48/48	c1d962b2/aeb3c368/0bd88475
CondaQA	Q3	Q3	ae0da5df	40,960	48/48/48	4d90c94a/40bae385/c908b99d
Iris	Q3	Q3	aabb7490	40,960	42/35/39	ed3fbb88/b43184b4/b5c27404

Table 6: **The thirty MRFS draws.** Q3 denotes Qwen3-8B fp16 in the separate answerer (Ans.) and reflector (Refl.) columns; Mixture is HotpotQA + MMLU-Pro. Context is the serving window in tokens. K is the number of retained proposals in draws 0/1/2, and hashes follow the same order; every draw evaluates them on 300 cases (Iris 100). Hashes are the first eight hexadecimal digits of each matrix file's SHA-256.

B.5 Compute and cost

All experiments ran on eight NVIDIA RTX 3090 GPUs (24 GB each), with Qwen3-8B fp16 served by Ollama, and used about 4,000 GPU-hours in total. One draw evaluates up to $48 \times 300 = 14{,}400$ prompt–case pairs for its matrix ($48 \times 100 = 4{,}800$ on Iris), plus up to $16 \times 2 \times 16 = 512$ feedback evaluations and up to 48 generation calls with their retries and redraws. Network requests differ from these nominal counts because of caching, duplicate instructions, early-stopped lineages and retries. For two audited draws, CaseHOLD and CondaQA at draw 0, the servers recorded 13,162 and 14,366 model requests over 14.72 and 12.14 hours, about 96% of it spent evaluating the matrix, with 15.52 million input and 4.61 million output tokens combined. The optimizer runs used for validation record about 6,300 to 7,500 task evaluations per search on the five core tasks: 6,871 configured for GEPA, and means of 6,750 for COPRO, 7,455 for MIPROv2, 6,874 for SIMBA and 6,286 for ACE. These counts exclude reflection calls and the later scoring of each collection on the measure cases. Once a matrix exists, both metrics reuse it without model inference: recomputing both on 200 item subsamples of each of the thirty draws, with 300 accessibility partitions per subsample, took 602 seconds on four worker processes.

C Task and replication details

This appendix gives the model roles and serving conditions, the per-task programs, fields and scoring rules, the instrument cohorts behind the thirty draws, the optimizer conditions, the retries and failure accounting behind the 150 optimizer cells, and the seed variation of the optimizers' final selections. Appendix B gives MRFS; Appendix D.4 gives every cell's reading and marks the cells without one.

C.1 Model roles and serving

Qwen3-8B fp16 (Yang et al., 2025) is the answerer and the reflector in every experimental condition, for MRFS and for all five optimizers, served by Ollama on RTX 3090 GPUs. Guarded runs verify the served model digest (0b358e6e) and context length on every request. Table 7 lists the decoding settings of each role. Answering disables thinking and uses temperature 0.6, slightly below the recommended 0.7 for non-thinking mode, with top-p 0.95 rather than the recommended 0.8. The lower temperature sharpens token probabilities, while the higher top-p retains a less restrictive probability-mass cutoff, preserving stochastic decoding.

The draws and cells ran under two context windows. The HotpotQA, ARC-Challenge, GSM8K and MoreHopQA draws and the 75 optimizer cells on the five core tasks used servers with an 8,192-token context and no per-request context override. The other seventeen draws, the three scoring passes of the mixture's GEPA collections, and the Iris and added-task cells used a 40,960-token context.

Role	Decoding	Output allowance (tokens)
Answerer, every procedure	Temperature 0.6, top-p 0.95, thinking off	2,048
MRFS reflector	Temperature 1.0, thinking off	4,000
GEPA, COPRO, MIPROv2 and SIMBA proposal	Temperature 0.6, top-p 0.95, thinking off	2,048 (core tasks, Iris); 8,192 (added tasks)
ACE reflector and curator	Temperature 0.0, thinking off	4,096 (core tasks, Iris); 8,192 (added tasks)
SIMBA trajectory sampling	Answerer copies at temperature 1.0	2,048

Table 7: **Model roles.** Every role uses Qwen3-8B fp16. The core tasks are HotpotQA, MMLU-Pro, ARC-Challenge, GSM8K and the mixture; the added tasks are MoreHopQA, CondaQA, TAT-QA and CaseHOLD. SIMBA's temperature-1.0 trajectory samples are recorded and excluded from its collection.

C.2 Programs, fields and scoring

Every task uses one program: a single DSPy ChainOfThought module whose signature takes the fields the cases carry and produces `reasoning` and `answer`. Only its instruction varies; the output format, case rendering and scorer are fixed by the harness, and the same program class serves MRFS and every optimizer. The mixture's program takes question, context and options, and each case leaves its component's absent field empty.

The answer field is scored against the gold answer with a binary rule. The harness first removes text after a stop marker or a hallucinated new turn, a trailing thinking-control token, emphasis markup and a `\boxed{}` wrapper. *Span* answers must match exactly after SQuAD normalization: lower case, punctuation and articles removed, whitespace collapsed. *Numeric* answers must equal the gold number within 10^{-6} after commas, currency and percent signs are removed. *Multiple-choice* answers are read as the leading option letter or, failing that, matched to an option's text after normalization. *TAT-QA* answers must be a JSON object with keys `answer` and `scale`, with the scale drawn from the dataset's five values, and are scored by the official scale-aware exact-match metric; each question carries this answer contract. A program error scores zero. Tables 8 and 9 give the per-task sources, fields and rules.

Task	Ans.	Refl.	Source and split	Preparation
HotpotQA	Q3	Q3	HotpotQA distractor setting, validation	Gold and distractor paragraphs rendered as "title: text"
MMLU-Pro	Q3	Q3	MMLU-Pro, test	Up to ten lettered options
ARC-Challenge	Q3	Q3	ARC-Challenge, test	Numeric option labels relettered
GSM8K	Q3	Q3	GSM8K main, test	Gold is the number after "####"
Mixture	Q3	Q3	HotpotQA and MMLU-Pro frames	Components as in their own rows
MoreHopQA	Q3	Q3	Release file at revision `27c72b1`; 1,118 cases, 307 source questions	Official open-answer task; decomposition labels not shown
CondaQA	Q3	Q3	Test file at revision `bd4857f`; 7,240 cases, 700 passages	Passage as context; label or free-text answer
TAT-QA	Q3	Q3	Training file at revision `870accc`; 13,215 questions, 2,201 tables	Table rows joined by "|", then paragraphs; answer contract appended
CaseHOLD	Q3	Q3	Rows 48,900–52,499 of the full CaseHOLD release, LexGLUE's test positions; 3,600 cases	Fixed question; citing text as context; five holdings as options
Iris	Q3	Q3	Bundled UCI copy in scikit-learn; 150 rows	Four measurements in the question; three species as options

Table 8: **Task sources and preparation.** Q3 denotes Qwen3-8B fp16 in the separate answerer (Ans.) and reflector (Refl.) columns; Mixture is HotpotQA + MMLU-Pro. Revisions are those of the dataset authors' repositories. The prepared files of the four added tasks and Iris are pinned by SHA-256: MoreHopQA `570f1825`, CondaQA `0ded7cd5`, TAT-QA `670b7090`, CaseHOLD `5a93e577` and Iris `29e45770`.

Task	Ans.	Refl.	Program fields	Answer form	Scoring rule
HotpotQA	Q3	Q3	question, context	Span	Normalized exact match
MMLU-Pro	Q3	Q3	question, options	Letter	Multiple choice
ARC-Challenge	Q3	Q3	question, options	Letter	Multiple choice
GSM8K	Q3	Q3	question	Number	Numeric equality
Mixture	Q3	Q3	question, context, options	Span or letter	Component's rule
MoreHopQA	Q3	Q3	question, context	Span	Normalized exact match
CondaQA	Q3	Q3	question, context	Label or span	Normalized exact match
TAT-QA	Q3	Q3	question, context	JSON answer and scale	Official scale-aware exact match
CaseHOLD	Q3	Q3	question, context, options	Letter	Multiple choice
Iris	Q3	Q3	question, options	Letter	Multiple choice

Table 9: **Programs and scoring rules.** Q3 denotes Qwen3-8B fp16 in the separate answerer (Ans.) and reflector (Refl.) columns; Mixture is HotpotQA + MMLU-Pro. Every program also outputs `reasoning`, which is not scored.

Before sampling, the four added tasks' raw files were verified against pinned checksums, and their preparation passed exact frame reconstruction, source-group separation and gold self-scoring checks. The longest added-task cases were tokenized to confirm that every case fits with its full output allowance: MoreHopQA 1,294, CaseHOLD 811, CondaQA 357 and TAT-QA 2,849 tokens.

C.3 Instrument cohorts, parser versions and context checks

The thirty draws come from four instrument versions (Table 5) and two context windows. Table 10 groups them.

Tasks	Ans.	Refl.	Instrument	Context	Draws
HotpotQA, ARC-Challenge, GSM8K	Q3	Q3	`a334bb70`	8,192	Draw 0 historical; draws 1–2 guarded repeats
MoreHopQA	Q3	Q3	`5c5e9b8a`	8,192	Three historical draws; context recovered from server logs
MMLU-Pro, Mixture, TAT-QA, CaseHOLD, Iris	Q3	Q3	`aabb7490`	40,960	Guarded draws with the unwrap rule
CondaQA	Q3	Q3	`ae0da5df`	40,960	Guarded draws with the unwrap rule and label exemption

Table 10: **Instrument cohorts of the thirty draws.** Q3 denotes Qwen3-8B fp16 in the separate answerer (Ans.) and reflector (Refl.) columns; Mixture is HotpotQA + MMLU-Pro. Earlier 8,192-token pilot draws of MMLU-Pro and the mixture are not among the thirty.

Parser version. Within each of the 18 draws produced under the unwrap rule, removing the prompts that came from unwrapped replies changes CARAT-C by at most 0.026, and by less than 0.005 on 13 draws; CARAT-A changes by at most 0.068. Against 200 removals of the same number of randomly chosen prompts, the observed change lies in the outer 5% only for TAT-QA draw 0 (CARAT-C), CaseHOLD draw 1 (CARAT-A) and CondaQA draw 0 (the reversal rate).

Context window. HotpotQA, ARC-Challenge and GSM8K have no 40,960-token draw. In their guarded 8,192-token repeats, every request fitted the window with its full output allowance; the largest input plus allowance was 4,976 to 5,102 tokens. The historical draw 0 of these tasks has no request ledger. The 8,192-token pilot draws of MMLU-Pro and the mixture were redrawn at 40,960 tokens because their largest root batch plus the 4,000-token reflection allowance exceeded 8,192 tokens.

C.4 Optimizer conditions

Every optimizer runs the task program with the same answerer, reflector and case frame. GEPA, MIPROv2 and ACE draw training and reflection cases from the 150 proposal cases and select on the 150 select cases; COPRO and SIMBA take a single case set and receive the select cases. No method sees the measure cases. Every program call is recorded. After a run, every distinct instruction the method evaluated on select cases is scored on the measure cases and forms its collection; GEPA's collection holds its accepted candidates. Instructions seen only on proposal cases and SIMBA's temperature-1.0 trajectory samples are excluded. Table 11 gives the settings. Budgets target GEPA's 6,871 metric calls per seed, and the realized counts are recorded.

Method	Proposer input	Selection	Settings	Metric calls
GEPA	Execution traces with generic feedback	Child beats parent on a minibatch	Reflection minibatch 3	6,871
COPRO	Earlier instructions and their scores; no cases	Top of the breadth on the full select set	Breadth 15, depth 3	6,750
MIPROv2	Data summary and bootstrapped demonstrations	Trials on 35-case minibatches with periodic full evaluation	72 candidates, 108 trials, zero demonstrations in the prompt	≈7,400
SIMBA	Contrasting trajectories on a 32-case batch	Best of six rule-appended candidates on the batch; seven winners validated on the full set	14 steps, no demonstrations	6,874
ACE	Reflections on rollouts, curated into a playbook	New best at a full-select checkpoint every 25 cases	5 epochs, batch 5, 3 rounds, 3,000-token playbook	≈6,300

Table 11: **Optimizer settings.** Every method uses Qwen3-8B fp16 as answerer and reflector. GEPA's feedback states only whether an answer is correct and, if not, the expected answer. Metric calls count task evaluations during search on the five core tasks; GEPA's is the configured budget.

Runners by campaign. The 75 core-task cells ran on the core runner with the allowances in Table 7; the mixture's three GEPA collections were scored on the measure cases in separate passes that score accepted prompts only. The 15 Iris cells used the core runner's settings. The 60 added-task cells used a runner that raises every reflection allowance to 8,192 tokens and rejects a reflection that reaches its allowance instead of passing truncated text on. SIMBA used DSPy's native rule on the core tasks and Iris, which skips a contrast when the better trajectory scores at or below the batch's 10th percentile or the worse at or above its 90th; with binary scores on ARC-Challenge every rule attempt was skipped, so its three runs kept only the shipped instruction. On MoreHopQA, CondaQA, TAT-QA and CaseHOLD, SIMBA ran the binary-contrast shared-input variant: rules come from pairs of trajectories scored one and zero, equal-score pairs are skipped, and trajectories reach the reflector as shared strings, with SIMBA's search otherwise unchanged.

Core-task truncation. The 8,192-token servers of the core-task runs truncated over-long prompts at the head. Server logs record 673 such events between 7 and 10 September, all in optimizer reflection or proposal calls on HotpotQA and none in candidate scoring. Among the five methods, the ACE runs account for 169, 44 and 194 events per seed, SIMBA's for 22, 23 and 30 and MIPROv2's for 10 each; GEPA's runs have none. The recorded measure-case outcomes are unaffected; those HotpotQA searches ran with truncated reflection prompts.

C.5 Retries and failure accounting

All thirty draws and all 150 optimizer cells (ten tasks, five methods, three seeds) completed. Twelve draw attempts did not complete (HotpotQA 2, MMLU-Pro 2, mixture 3, CaseHOLD 2, CondaQA 3); the three CondaQA attempts stopped at the split-integrity check on the label DON'T KNOW, before the exemption of Appendix B.2. Each retry is a fresh draw under the same seed, and failed attempts contribute nothing to any matrix. Table 12 accounts for the optimizer campaigns run under the request guard. Every attempt is a fresh run of the cell's command; failed attempts contribute nothing to any collection, and their records are retained.

Campaign	Cells	Attempts	Failed	Completing attempts
Iris	15	19	5	12 first, 1 third (1 superseded); ACE seeds 1–2 completed in the re-run
Added tasks	60	77	32	43 first, 2 second; 15 cells lost both attempts (GEPA 5, ACE 10)
Re-run	17	33	16	9 first, 4 second, 2 third, 2 fifth

Table 12: **Optimizer campaigns under the request guard.** Every cell uses Qwen3-8B fp16 as answerer and reflector. The re-run covered the 15 lost added-task cells and Iris ACE seeds 1–2. Its failed attempts include three aborted before completion. The searches of the 75 core-task cells are historical runs made without this guard.

Four mechanisms caused the lost attempts. (1) The server's sampler panicked on non-finite logits and returned an HTTP 500 error; in these campaigns this happened on long optimizer requests. (2) Guard version 1 stopped a run when its tokenizer estimate of a request's input differed from the server's count by more than 256 tokens. The estimate under-counts runs of repeated whitespace or punctuation, which ACE carries from reflector outputs into later prompts. From version 2 on, the guard stops only when the server counts fewer tokens than estimated, which signals truncation, or when a request could not fit the window. (3) The server returned aborted generations as empty successful responses, repeatedly on one cache slot. Version 2.3 answers these and HTTP 500 errors with a retryable status. Version 2.4 also reloads the model after an aborted generation and returns an empty completion after three aborts of one request. (4) A reflection that ran to the 8,192-token allowance ended its run; in GEPA's runs this happened six times in about 940 reflections, each a repetition loop. Guard version 2.5 re-samples such a reflection, but no recorded attempt ran under it.

The first added-task campaign allowed one automatic fresh attempt per cell. The re-run allowed up to three attempts after a server-side loss, an HTTP 500 error or a reflection-cap stop, with a hard limit of six. Guard failures that recur deterministically stopped after one attempt. The re-run's completing attempts ran under guard versions 2 (one cell), 2.2 (one), 2.3 (two) and 2.4 (thirteen). Their ledgers record 55 transient server errors, all retried by the clients, and no empty completion issued by the guard.

C.6 Iris frame

Iris has 150 rows, so it uses its own 50/50/50 proposal/select/measure frame, with 100 evaluation cases. The proposal halves hold 25 cases each. Accessibility uses reference and call sizes 50 and 25 on the evaluation cases and 25 and 12 on the measure cases. Two Iris rows are identical in measurements and species, one among the proposal cases and one among the select cases. The optimizer runner indexes cases by question text, so the Iris cells run with a patch that assigns the duplicate to its select-case copy; the case frame and scoring are unchanged. COPRO seed 0 failed on the unpatched runner, and its third attempt completed.

C.7 Seed variation of optimizer final selections

Table 13 compares, for every task, the final prompts that three seeds of each optimizer return on the same training, selection and measurement split (Section 3.2).

D Full validation results and limitations

D.1 Comparison and reporting rules

MRFS readings use all evaluation cases; optimizer comparisons use only the common measure cases because optimizers consulted the select cases. Figure 1 display rule: each terrain omits cases solved by at least 90% of prompts on average across the task's three seeds, and cases no prompt solves; its panel readings use the full held-out case set. Cell statuses: a run accepting only the shipped instruction is marked "no pool"; an all-zero measure matrix has undefined accessibility; a cell without a completed attempt at freeze is missing. Accessibility is reported on §4's fixed log scale

Task	Ans.	Refl.	Retained	Optimizer	Held-out accuracy (%)	Range (pp)	Case disagreement (%)
HotpotQA	Q3	Q3	115/150	GEPA	53.3 / 55.3 / 52.0	3.3	12.2
				COPRO	38.7 / 41.3† / 43.3	4.7	13.9
				MIPROv2	43.3 / 53.3 / 44.0	10.0	22.6
				SIMBA	57.3 / 53.3 / 49.3	8.0	12.2
				ACE	58.7 / 55.3 / 57.3	3.3	13.3
MMLU-Pro	Q3	Q3	86/150	GEPA	56.7 / 59.3 / 59.3	2.7	24.8
				COPRO	59.3 / 61.3 / 62.0	2.7	22.5
				MIPROv2	60.7 / 63.3 / 59.3	4.0	19.4
				SIMBA	62.0 / 61.3 / 58.7	3.3	21.7
				ACE	55.3 / 61.3 / 61.3	6.0	27.9
ARC-Challenge	Q3	Q3	24/150	GEPA	90.7† / 90.7† / 90.7†	0.0	0.0
				COPRO	90.7† / 90.7† / 90.7†	0.0	0.0
				MIPROv2	90.7† / 90.7† / 90.7†	0.0	0.0
				SIMBA	90.7† / 90.7† / 90.7†	0.0	0.0
				ACE	92.0 / 90.7 / 92.0	1.3	22.2
GSM8K	Q3	Q3	34/150	GEPA	93.3 / 92.7 / 90.0	3.3	27.5
				COPRO	92.7† / 92.7† / 92.7†	0.0	0.0
				MIPROv2	94.0 / 92.7 / 94.7	2.0	17.6
				SIMBA	89.3 / 91.3 / 92.7†	3.3	29.4
				ACE	89.3 / 90.7 / 91.3	2.0	29.4
Mixture	Q3	Q3	77/150	GEPA	57.3† / 60.0 / 56.0	4.0	17.3
				COPRO	64.7 / 63.3 / 61.3	3.3	15.6
				MIPROv2	62.0 / 57.3 / 60.7	4.7	17.3
				SIMBA	62.0† / 62.0† / 56.7	5.3	7.8
				ACE	58.7 / 58.0 / 60.0	2.0	18.2
MoreHopQA	Q3	Q3	121/150	GEPA	58.0 / 43.3 / 56.0	14.7	28.1
				COPRO	35.3 / 37.3† / 34.7	2.7	9.4
				MIPROv2	43.3 / 37.3 / 40.0	6.0	15.4
				SIMBA*	35.3 / 40.0 / 36.0	4.7	10.5
				ACE	56.7 / 52.7 / 52.7	4.0	20.4
TAT-QA	Q3	Q3	150/150	GEPA	58.7 / 62.0 / 59.3	3.3	19.1
				COPRO	48.0 / 51.3 / 52.0	4.0	13.3
				MIPROv2	55.3 / 62.7 / 62.7	7.3	17.3
				SIMBA*	50.7 / 51.3 / 48.7	2.7	28.4
				ACE	16.7 / 26.7 / 16.0	10.7	25.3
CaseHOLD	Q3	Q3	71/150	GEPA	66.0 / 64.7 / 64.7†	1.3	24.4
				COPRO	64.7 / 63.3 / 65.3	2.0	16.0
				MIPROv2	63.3† / 67.3 / 63.3	4.0	17.8
				SIMBA*	64.7 / 63.3 / 64.0	1.3	21.6
				ACE	66.0 / 62.0 / 64.7	4.0	17.8
CondaQA‡	Q3	Q3	150/150	GEPA	57.3 / 30.7 / 61.3	30.7	28.4
				COPRO	0.0 / 0.0 / 0.0	0.0	0.0
				MIPROv2	46.0 / 45.3 / 51.3	6.0	19.1
				SIMBA*	0.0† / 54.0 / 0.0†	54.0	36.0
				ACE	45.3 / 60.7 / 56.7	15.3	31.1
Iris	Q3	Q3	32/50	GEPA	90.0 / 88.0 / 94.0	6.0	18.8
				COPRO	72.0 / 70.0 / 82.0	12.0	29.2
				MIPROv2	78.0 / 82.0 / 80.0	4.0	35.4
				SIMBA	94.0 / 90.0 / 90.0	4.0	20.8
				ACE	76.0 / 80.0 / 86.0	10.0	35.4

Table 13: **Final optimization outcomes vary across seeds under a fixed method and data split.** Q3 denotes Qwen3-8B fp16 in the separate answerer (Ans.) and reflector (Refl.) columns; Mixture denotes HotpotQA + MMLU-Pro. Each row compares three seeds of the same optimizer (methods in Section 5.1) on the same training, selection and measurement split. Accuracy is measured on all held-out cases (150; 50 for Iris) and listed for seeds 0 / 1 / 2; range is maximum minus minimum across seeds. Case disagreement is the mean fraction of cases on which two final selected prompts differ in correctness, averaged over seed pairs 0–1, 0–2 and 1–2. It uses each task's retained cases, those remaining after excluding cases solved by at least 90% of the common method/seed-balanced candidate pool; Retained gives their number out of the held-out cases. Each task's mask is fixed across methods and is not derived from the three final choices. †The seed returned its starting instruction unchanged. *SIMBA's binary-contrast shared-input variant (Appendix C.4). ‡CondaQA retains the recorded output-format artifact. The table describes observed run variation under the recorded stochastic evaluation conditions.

with the reference-ordered share, usable comparison counts and evaluable repetitions; zero observed reversals yield $+\infty$ with their supporting counts, without clipping. Coherence is untransformed and accompanied by its solved-case count. Accessibility comparisons stay within methods; cross-method level differences are descriptive. Tables retain model roles, provenance, actual task and seed counts, and reasons for unavailable readings; seeds do not multiply the number of tasks.

Matched accessibility settings for the optimizer comparisons. Accessibility applies §4's estimator on the common measure cases with 300 partitions and reference size 75, using the call size of each optimizer's read point: 37 for GEPA, COPRO and ACE, 35 for MIPROv2 and 32 for SIMBA. MRFS readings are recomputed at each partner's setting, and Iris uses reference 25 and call 12 throughout. Accessibility orderings average per-run log readings.

Exploratory analyses are identified separately from tests specified before their outcomes were examined.

D.2 Held-out cases and repeated sampling

Table 1 (Observation 1) gives the fixed-split differences; the check was added after the main results were available. Each reading uses the task's three MRFS draws, and CARAT-A uses reference/call sizes 50/25 on both sides, so its levels are not comparable with Table 14. Each task's split is drawn once for its three MRFS draws; MoreHopQA source-question groups stay whole and the mixture splits 100/100 and 50/50 by source. Across the 100 further splits, the held-out side keeps 93.2% of CARAT-A and 94.8% of CARAT-C task-pair orders on average, and 35 and 34 of 36 on the fixed split. A single 100-case side varies more than the 200-case side because it has half the cases: across the 100 further splits, each task's mean signed difference lies within 0.009 of zero, while the 5th–95th percentiles of one split's difference stay within ± 0.18 for CARAT-A and ± 0.07 for CARAT-C. On the held-out side, ARC-Challenge has a usable prompt pair in only 39 to 60 of 300 partitions per draw.

Table 14 reports the full-set readings across three complete MRFS draws per task. CARAT-A draw SDs have a median of 0.049 (range 0.002 to 0.146) against an SD of 0.336 between task means; CARAT-C draw SDs have a median of 0.005 (range 0.000 to 0.020) against 0.067. Task identity accounts for 97.1% of the variance among the thirty accessibility readings and 98.6% among the coherence readings. Resampling 80% of the evaluation cases within each draw gives item-subset SDs of 0.013 to 0.046 for accessibility and 0.001 to 0.009 for coherence (Table 16). CARAT-A varies more when few prompt pairs can be ordered: the mixture's draws supply 56 to 183 usable comparisons per partition, and its readings range from 0.172 to 0.457. CondaQA's reversal rates differ by at most 0.005, a difference the log scale expands. Across paired MRFS draws on the common measure panel, CARAT preserves 119 of 135 (88.1%) accessibility and 127 of 135 (94.1%) coherence task-pair orders.

Task	CARAT-A ↑	CARAT-C ↑
HotpotQA	0.866 ± 0.044	0.931 ± 0.004
MMLU-Pro	0.154 ± 0.048	0.903 ± 0.001
ARC-Challenge	0.171 ± 0.071	0.985 ± 0.000
GSM8K	0.396 ± 0.064	0.962 ± 0.004
Mixture	0.295 ± 0.146	0.926 ± 0.006
MoreHopQA	0.693 ± 0.039	0.884 ± 0.013
TAT-QA	0.838 ± 0.051	0.778 ± 0.006
CaseHOLD	0.156 ± 0.002	0.942 ± 0.002
CondaQA†	1.072 ± 0.074	0.794 ± 0.020
Iris	0.440 ± 0.036	0.900 ± 0.014

Table 14: **CARAT-A and CARAT-C across repeated MRFS draws.** Higher scores indicate fewer conditional pairwise reversals and more compatible observed solved sets, respectively. Qwen3-8B fp16 is both answerer and reflector; Mixture denotes HotpotQA + MMLU-Pro. Entries are means and sample SDs across three complete draws per task on all evaluation cases. Accessibility uses reference/call sizes 150/75 and up to 300 evaluable partitions per draw. Item-subset SDs, solved-case counts and estimator support are in Appendix D.3. †CondaQA retains the recorded output-format artifact.

D.3 LIMITATIONS OF THE VALIDATION

The validation covers ten tasks with Qwen3-8B fp16 as both answerer and reflector, using finite prompt collections and recorded responses on dataset cases. The cases represent the declared task setting, and repeated answers are observations under the declared decoding settings. Accessibility depends on which candidate pairs the collection makes orderable, and a reading that is stable across runs can still be sensitive to the evaluated cases. The task-pair agreements reuse tasks and runs, and the pooled-coherence agreement concerns mean order: neighboring archive bands can overlap. The archive draws reuse recorded candidates, and the pooled analysis was completed after per-method results had been inspected. CondaQA's accessibility partly reflects its required answer format. These results establish recurring task profiles under the tested procedures, not coverage of every prompt the reflector or an optimizer could produce. Changing the answerer, reflector, proposal information, scoring rule or serving condition requires a new measurement, and gains from acting on the readings require separate outcome evaluation.

Accessibility is conditional on prompt pairs ordered by the reference subset. Its support counts delimit the evidence behind a high reading; they are not stability measurements. The underlying $\widehat{D}(n)$ measures conditional pairwise reversals, and its relation to winner-selection error requires separate validation (Cawley & Talbot, 2010). The reference margin is a per-pair comparison rule with no simultaneous error guarantee, population interpretation requires representative held-out cases, and zero observed reversals do not establish a zero population rate. Only 4.5% of CaseHOLD pairs, 5.4% of MMLU-Pro pairs and 5.5% of ARC-Challenge pairs receive such an order in the full-set readings. HotpotQA and TAT-QA differ in mean accessibility by 0.028, less than either task's draw SD, while their coherence gap is 0.153. ACE's within-method accessibility order agreement is 65.9% (Table 20). The ARC-Challenge shift from last under MRFS to fourth under MIPROv2 establishes collection dependence but does not identify which proposal or selection mechanism caused it. CondaQA's high accessibility partly tracks compliance with the required answer form and does not, by itself, establish reasoning quality.

Task	Ans.	Refl.	Ordered share	Usable pairs	Evaluable partitions
HotpotQA	Q3	Q3	66.9%	734.2	300/300/300
MMLU-Pro	Q3	Q3	5.4%	52.9	299/300/300
ARC-Challenge	Q3	Q3	5.5%	28.6	300/298/299
GSM8K	Q3	Q3	16.5%	122.4	300/300/300
Mixture	Q3	Q3	11.1%	108.1	300/300/300
MoreHopQA	Q3	Q3	42.2%	469.2	300/300/300
TAT-QA	Q3	Q3	67.9%	587.8	300/300/300
CaseHOLD	Q3	Q3	4.5%	41.7	300/300/300
CondaQA	Q3	Q3	75.6%	849.1	300/300/300
Iris	Q3	Q3	20.4%	139.0	300/300/300

Table 15: **Full-set CARAT-A support.** Q3 denotes Qwen3-8B fp16 in the separate answerer (Ans.) and reflector (Refl.) columns; Mixture is HotpotQA + MMLU-Pro. Ordered shares are before call-tie exclusion; usable pairs are mean counts per evaluable partition. Partitions are shown for each of three complete draws.

Task	Ans.	Refl.	s_{item}(CARAT-A)	s_{item}(CARAT-C)	M
HotpotQA	Q3	Q3	0.020	0.005	212.7
MMLU-Pro	Q3	Q3	0.029	0.005	260.3
ARC-Challenge	Q3	Q3	0.046	0.001	294.0
GSM8K	Q3	Q3	0.025	0.002	294.0
Mixture	Q3	Q3	0.028	0.004	229.7
MoreHopQA	Q3	Q3	0.017	0.007	225.3
TAT-QA	Q3	Q3	0.013	0.008	232.3
CaseHOLD	Q3	Q3	0.034	0.004	258.0
CondaQA	Q3	Q3	0.015	0.009	260.7
Iris	Q3	Q3	0.025	0.007	99.7

Table 16: **Item-subset spread and solved cases.** Q3 denotes Qwen3-8B fp16 in the separate answerer (Ans.) and reflector (Refl.) columns; Mixture is HotpotQA + MMLU-Pro. Item SD is the within-draw SD over 200 group-preserving 80% case subsets, averaged across the three draws; M is the mean number of solved cases. These columns accompanied Table 14 before it was reduced to the draw readings.

Task	Ans.	Refl.	Runs	1	2	3	4	5	6	7	8	9	10	Mean
CondaQA	Q3	Q3	13	56	12	6	0	0	0	0	0	0	0	0.08
HotpotQA	Q3	Q3	18	22	17	33	21	6	6	0	0	0	0	0.24
TAT-QA	Q3	Q3	18	11	35	18	26	6	6	4	0	0	0	0.26
MoreHopQA	Q3	Q3	18	6	16	19	13	25	9	7	6	6	0	0.39
Iris	Q3	Q3	18	6	14	6	4	31	6	11	11	1	17	0.52
GSM8K	Q3	Q3	18	0	0	4	7	19	44	25	6	0	0	0.54
Mixture	Q3	Q3	18	0	0	9	20	11	8	23	17	1	17	0.59
ARC-Challenge	Q3	Q3	15	0	6	6	4	2	17	0	11	12	28	0.69
MMLU-Pro	Q3	Q3	18	0	0	0	6	0	0	17	25	30	28	0.79
CaseHOLD	Q3	Q3	18	0	0	0	0	0	6	13	25	51	11	0.80

Table 17: **CARAT-A position shares behind the rank counts of Observation 3.** Q3 denotes Qwen3-8B fp16 in the answerer (Ans.) and reflector (Refl.) columns; Mixture is HotpotQA + MMLU-Pro. Each of eighteen runs, three for MRFS and for each optimizer, ranks the tasks it covers by CARAT-A on the common measure cases; each placement becomes a position from 0 (top) to 1 (bottom) and is spread over ten equal bins. Columns 1–10 are the position bins, from the top tenth of the ranking (1) to the bottom tenth (10); entries are the percentage of each bin's placements held by the task, and each bin sums to 100. Runs counts the runs that rank the task; Mean is its mean position. ARC-Challenge and CondaQA enter fewer runs because some collections leave CARAT-A undefined: SIMBA kept no instruction beyond the shipped one on ARC-Challenge, and COPRO's three CondaQA collections and two of SIMBA's solve no measure case, since none of their prompts produces the required answer form. Expressing each placement as a position within its run's ranking lets the remaining runs still place these tasks.

The task-pair agreement counts reuse tasks and runs, so their percentages are descriptive rather than independent-trial estimates. Excluding CondaQA gives 134/172 matched accessibility order agreements (77.9%) across methods; those comparisons remain dependent. COPRO's CondaQA accessibility is undefined, SIMBA has no ARC-Challenge pool, and SIMBA's CondaQA mean uses one usable run; other included task-method means use three. Tables 2 and 20 require three usable seeds and exclude those cells as stated in their captions.

The 45/45 pooled-coherence result concerns the ordering of task means, not separation of every pair across archive resamples; some neighboring 5th–95th percentile bands overlap in Figure 3b. The 300 balanced draws quantify sensitivity to the recorded archive, not variation over independent optimizer runs. The analysis was completed after inspection of per-method results. GEPA contributes accepted candidates only; the five no-pool cells are excluded; the three all-zero CondaQA/COPRO cells are included. Forty-eight prompts match the maximum proposal budget of MRFS, while realized MRFS collections can be smaller. TAT-QA's 0.046 level shift shows that absolute coherence depends on the explored collection even when its rank recurs. The mixture's full-set and common-measure relations differ, so the mixture comparison supplies no monotonicity rule for combining tasks.

D.4 PER-TASK READINGS FOR EVERY PROCEDURE

Reversal rates behind the CARAT-A readings. The recurring pattern separates tasks by how reliably observed prompt preferences carry to unseen cases. On HotpotQA, TAT-QA and MoreHopQA, only 2% to 4% of the prompt pairs ordered on reference cases reverse on separate call cases in the full-set MRFS readings, so observed candidate preferences are more likely to persist on separate cases. On MMLU-Pro, ARC-Challenge and CaseHOLD, 47% to 50% reverse, signalling that the prompt optimization signal may be unreliable without more data. CondaQA's high accessibility partly reflects whether prompts produce the required answer form.

Tables 18 and 19 give every task's CARAT-A and CARAT-C reading under MRFS and under each of the five optimizers, with the cells that have no usable reading marked. They summarise the readings behind Table 2, the CARAT-A ranks in Figure 3a and the MRFS CARAT-C means in Figure 3b. The comparisons use each procedure's task order; differences in level between columns are descriptive.

			CARAT-A ↑					
Task	Ans.	Refl.	MRFS	GEPA	COPRO	MIPROv2	SIMBA	ACE
HotpotQA	Q3	Q3	0.687	0.548	0.606	0.656	0.481	0.193
			(0.053)	(0.135)	(0.083)	(0.013)	(0.052)	(0.028)
MMLU-Pro	Q3	Q3	0.137	0.156	0.189	0.295	0.138	0.160
			(0.069)	(0.044)	(0.041)	(0.150)	(0.019)	(0.052)
ARC-Challenge	Q3	Q3	0.112	0.132	0.231	0.553	no pool	0.160
			(0.027)	(0.060)	(0.148)	(0.180)		(0.068)
GSM8K	Q3	Q3	0.300	0.235	0.263	0.391	0.359	0.174
			(0.057)	(0.050)	(0.079)	(0.041)	(0.091)	(0.016)
Mixture	Q3	Q3	0.277	0.168	0.370	0.549	0.199	0.144
			(0.120)	(0.031)	(0.149)	(0.057)	(0.047)	(0.045)
MoreHopQA	Q3	Q3	0.620	0.395	0.592	0.515	0.321	0.203
			(0.052)	(0.230)	(0.144)	(0.075)	(0.068)	(0.081)
TAT-QA	Q3	Q3	0.793	0.360	0.260	0.648	0.619	0.282
			(0.051)	(0.109)	(0.022)	(0.013)	(0.069)	(0.025)
CaseHOLD	Q3	Q3	0.145	0.187	0.180	0.309	0.197	0.104
			(0.035)	(0.004)	(0.040)	(0.043)	(0.013)	(0.013)
CondaQA	Q3	Q3	1.014	0.506	undefined	1.213	0.663‡	0.640
			(0.072)	(0.139)		(0.173)		(0.021)
Iris	Q3	Q3	0.359	0.438	0.208	0.325	0.472	0.132
			(0.017)	(0.142)	(0.059)	(0.051)	(0.024)	(0.077)

Table 18: **CARAT-A for every task and procedure.** Source readings behind Table 2 and Figure 3a. Q3 denotes Qwen3-8B fp16, both answerer (Ans.) and reflector (Refl.) for every procedure; Mixture is HotpotQA + MMLU-Pro. Readings use the common measure cases (150; Iris 50), not the full evaluation set of Table 14. Each cell gives the mean over three runs (MRFS draws or optimizer seeds) and, below it, the sample SD. Level differences between procedures are descriptive. Call sizes: 37 for GEPA, COPRO and ACE, 35 for MIPROv2, 32 for SIMBA and 37 for MRFS, whose task order is the same at 35 and 32; reference 75 and up to 300 evaluable partitions; Iris uses reference 25 and call 12 throughout. No pool: all three SIMBA runs on ARC-Challenge accepted only the shipped instruction. Undefined: every COPRO prompt scores zero on every CondaQA measure case. ‡One usable seed; no SD.

			CARAT-C ↑					
Task	Ans.	Refl.	MRFS	GEPA	COPRO	MIPROv2	SIMBA	ACE
HotpotQA	Q3	Q3	0.921	0.928	0.915	0.904	0.926	0.940
			(0.004)	(0.010)	(0.002)	(0.011)	(0.009)	(0.005)
MMLU-Pro	Q3	Q3	0.902	0.907	0.914	0.910	0.900	0.896
			(0.005)	(0.009)	(0.003)	(0.004)	(0.003)	(0.004)
ARC-Challenge	Q3	Q3	0.979	0.979	0.979	0.981	no pool	0.981
			(0.000)	(0.001)	(0.000)	(0.001)		(0.001)
GSM8K	Q3	Q3	0.958	0.971	0.958	0.973	0.962	0.959
			(0.002)	(0.002)	(0.002)	(0.003)	(0.005)	(0.007)
Mixture	Q3	Q3	0.940	0.940	0.939	0.938	0.926	0.929
			(0.007)	(0.001)	(0.004)	(0.001)	(0.003)	(0.008)
MoreHopQA	Q3	Q3	0.896	0.891	0.909	0.884	0.890	0.896
			(0.009)	(0.004)	(0.004)	(0.006)	(0.004)	(0.008)
TAT-QA	Q3	Q3	0.776	0.879	0.883	0.829	0.764	0.502
			(0.006)	(0.006)	(0.010)	(0.009)	(0.015)	(0.071)
CaseHOLD	Q3	Q3	0.949	0.938	0.945	0.951	0.931	0.944
			(0.002)	(0.004)	(0.004)	(0.003)	(0.003)	(0.002)
CondaQA	Q3	Q3	0.755	0.847	undefined	0.834	0.906‡	0.808
			(0.018)	(0.013)		(0.024)		(0.011)
Iris	Q3	Q3	0.916	0.943	0.861	0.884	0.945	0.923
			(0.028)	(0.016)	(0.013)	(0.005)	(0.020)	(0.017)

Table 19: **CARAT-C for every task and procedure.** Source readings behind Table 2 and Figure 3b. Q3 denotes Qwen3-8B fp16, both answerer (Ans.) and reflector (Refl.) for every procedure; Mixture is HotpotQA + MMLU-Pro. Measure cases, runs, cell format and markers are as in Table 18, and no read point applies. Level differences between procedures are descriptive.

D.5 Supporting analyses for the MRFS comparison

Table 20 gives each optimizer's own task-order repeatability, the check reported for MRFS with Table 14. Table 21 gives every pairwise agreement among the six task orderings and the pooled counts behind Table 2. Agreement between two optimizers can reflect exploration concentrated in the same direction, so the pairwise values are not interpreted separately; they are the components of the pooled column.

		CARAT-A		CARAT-C	
Optimizer	Tasks	Seed pairs	Stable pairs	Seed pairs	Stable pairs
GEPA	10	101/135	20/45	123/135	39/45
COPRO	9	82/108	16/36	104/108	32/36
MIPROv2	10	117/135	30/45	129/135	42/45
SIMBA	8	76/84	24/28	70/84	21/28
ACE	10	89/135	20/45	129/135	40/45

Table 20: **Task-order repeatability within each optimizer's collection.** Qwen3-8B fp16 is both answerer and reflector. Seed pairs count task-pair orders preserved between seeds 0–1, 0–2 and 1–2; stable pairs keep their order across all nine combinations of the two tasks' seeds. Readings use the common measure cases (150; Iris 50) at each optimizer's accessibility read point; only tasks with three usable seeds enter (COPRO excludes CondaQA; SIMBA excludes ARC-Challenge and CondaQA). The MRFS values accompany Table 14.

Procedure	MRFS	GEPA	COPRO	MIPROv2	SIMBA	ACE	Other five
CARAT-A							
MRFS	–	39/45	25/36	34/45	26/28	36/45	160/199
GEPA	39/45	–	24/36	30/45	23/28	30/45	146/199
COPRO	25/36	24/36	–	28/36	18/28	26/36	121/172
MIPROv2	34/45	30/45	28/36	–	21/28	35/45	148/199
SIMBA	26/28	23/28	18/28	21/28	–	20/28	108/140
ACE	36/45	30/45	26/36	35/45	20/28	–	147/199
CARAT-C							
MRFS	–	41/45	33/36	42/45	25/28	42/45	183/199
GEPA	41/45	–	29/36	38/45	27/28	38/45	173/199
COPRO	33/36	29/36	–	33/36	22/28	31/36	148/172
MIPROv2	42/45	38/45	33/36	–	23/28	41/45	177/199
SIMBA	25/28	27/28	22/28	23/28	–	23/28	120/140
ACE	42/45	38/45	31/36	41/45	23/28	–	175/199

Table 21: **Pairwise agreement of task orderings.** Qwen3-8B fp16 is both answerer and reflector. Each entry counts task pairs ordered alike by the row procedure's three-seed mean readings and by the column's, on the tasks both cover; MRFS is read at its partner's accessibility read point. Other five pools each row over its five pairings and gives the counts behind Table 2. Measure cases and task eligibility are as in Table 20.

Against each single optimizer on the tasks both cover, the MRFS ordering is the closer partner in 17 of 20 coherence comparisons, with one tie, and in 15 of 20 accessibility comparisons, with one tie. Excluding CondaQA, the MRFS ordering still has the highest average agreement with the other five: 77.9% for accessibility, against 70.3% to 77.1% for the optimizers, and 91.9% for coherence, against 85.7% to 88.4%. The task-order audit also admits SIMBA's single usable CondaQA run; on that set, SIMBA agrees with the MRFS ordering on 94.4% of accessibility pairs and 83.3% of coherence pairs.

Run-to-run noise does not account for the MRFS ordering's agreement. Optimizers whose own orderings repeat more reliably than the MRFS ordering agree less with the other collections. On coherence, COPRO, MIPROv2 and ACE repeat at 96.3%, 95.6% and 95.6%, against 94.1% for MRFS, and average 86.0%, 88.9% and 87.9% agreement with the other five collections, against 92.0% for MRFS (Table 2). On accessibility, SIMBA repeats at 90.5% against 88.1% and averages 77.1% against 80.4%. The pattern holds when single runs replace three-run means (MRFS 77.4% and 90.4%; at most 72.5% and 89.2% for any optimizer) and on the eight tasks available to every collection. On those eight tasks, the MRFS ordering contradicts none of the task orders shared by the other five collections (12 accessibility and 18 coherence pairs); each optimizer contradicts at most three. These comparisons were conceived after the per-method results had been inspected; every reading is frozen and hash-referenced in the evidence record.

D.6 Method-level accessibility and final accuracy

Table 3 ranks the five optimizers by CARAT-A read on their own collections at a common call of 37 cases (12 on Iris), on the eight tasks where every optimizer has three usable seeds (ARC-Challenge and CondaQA drop out). The method order is concordant across tasks (Kendall's $W = 0.54$, Friedman $\chi^2 = 17.2$ with 4 degrees of freedom, permutation $p < 0.001$) and holds when each seed is ranked separately ($W = 0.51$). Table 22 gives the held-out accuracy of each optimizer's final selected prompt, the runs of Table 13. Its method order does not repeat across tasks ($W = 0.06$ over ten tasks, permutation $p = 0.68$; $W = 0.07$, $p = 0.71$ on the eight tasks above), so the accessibility order is not carried by accuracy. SIMBA uses its binary-contrast variant on the four added tasks, and COPRO's CondaQA prompts score zero because of the answer form.

	Mean accuracy, 10 tasks	Mean rank (1 = best)	Best on
GEPA	67.0%	2.75	1 task
MIPROv2	64.5%	2.55	3
ACE	63.0%	2.90	4
SIMBA	61.7%	3.35	1
COPRO	57.4%	3.45	1

Table 22: **Held-out accuracy of each optimizer's final prompt.** Mean over three seeds on the common measure cases, averaged over the ten tasks; ranks are within each task, with tied accuracies sharing their average rank (four methods tie on ARC-Challenge). The order does not repeat across tasks ($W = 0.06$), unlike the accessibility order of Table 3 ($W = 0.54$).