Think Before You Optimize: Calibration-Aware Prompt Optimization for Robust LLM Reasoning

Anonymous ACL submission

Abstract

Prompt optimization methods iteratively refine instructions to improve large language model performance, yet they frequently introduce regressions—an optimizer that raises accuracy on one task may confidently produce wrong answers on another. We trace this fragility to two root causes: *overconfidence amplification*, where optimization reinforces the model's existing biases, and *unconstrained regression*, where iterative refinement converges to worse prompts without safety checks. We propose CAPO (Calibration-Aware Prompt Optimization), a two-phase framework that addresses both failure modes. Phase 1 performs calibration-aware search: a calibration prefix forces the model to reconsider its initial intuitions during every evaluation, while a regression guard rejects any candidate that degrades accuracy on a held-out safety set, and convergence-based stopping halts the search before over-optimization occurs. Phase 2 selects the best calibration prefix from a small library as a decoupled post-processing step. On six BIG-Bench Hard reasoning tasks with Qwen2.5-7B-Instruct, our method achieves 89.1% average accuracy—outperforming the strongest baseline (ACE, 73.6%) by +15.5 points and the unoptimized model (61.3%) by +27.8 points. Ablations confirm that both calibration-aware evaluation and regression guards contribute complementary gains, with benefits transferring across model families. These results demonstrate that principled architectural choices, rather than brute-force iteration, are key to reliable prompt optimization—establishing our approach as a foundation for robust LLM instruction tuning.

1 Introduction

Large language models achieve impressive accuracy on complex reasoning tasks, yet their performance is remarkably sensitive to prompt wording—rephrasing a single instruction can shift accuracy by tens of percentage points (Brown et al., 2020). This sensitivity has motivated *automatic prompt optimization*: methods that search over prompt formulations to maximize task performance without human trial-and-error (Zhou et al., 2023; Yang et al., 2024). The promise is compelling: replace fragile manual engineering with principled, automated search.

But does prompt optimization actually deliver on this promise? To find out, we conducted a systematic empirical study of five prompt optimization methods—GRACE (Pryzant et al., 2023), GEPA (Agrawal et al., 2026), ACE, MetaSPO, and PromptQuine—across six BIG-Bench Hard reasoning tasks (Suzgun et al., 2023) using Qwen2.5-7B-Instruct. The results reveal a striking pattern: every existing method improves on some tasks while regressing on others. GRACE boosts web-of-lies accuracy to 97% but *drops* causal reasoning by 9.4 points below the unoptimized baseline. MetaSPO improves object counting by 36 points but *collapses* navigation by 17 points. No published method exceeds 75% on all six tasks simultaneously. We term this phenomenon performance fragmentation: the inability of any single optimization strategy to generalize safely across diverse reasoning tasks.

Multi-seed evaluation deepens this concern. When we repeat experiments with three random seeds (varying train/val data order), GRACE exhibits run-to-run standard deviations of up to ± 12.2 points on navigation and ± 10.8 points on causal judgement—the same method can range from 49.3% to 70.7% depending on which training examples it sees first. In contrast, the unoptimized baseline is highly stable (± 0.6–1.0 points), confirming that the fragmentation is an artifact of optimization, not inherent task variability. Why do these methods fragment? Our analysis below identifies two root causes.

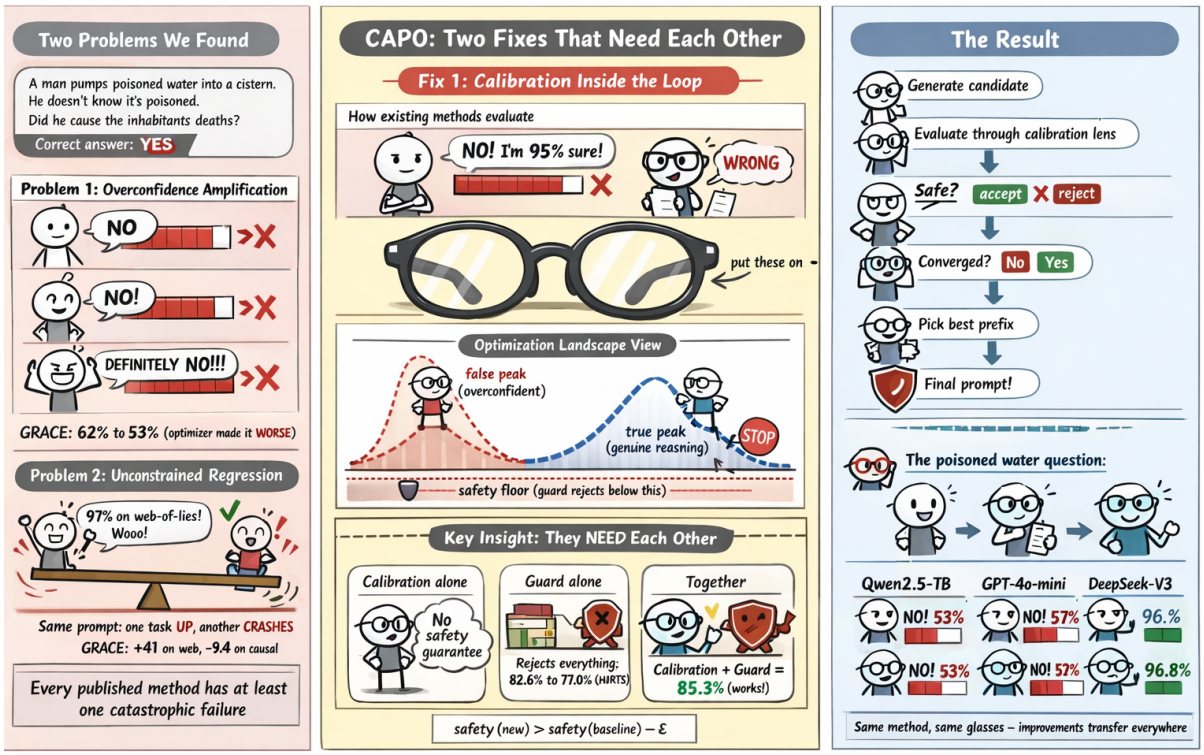

Figure 1: Overview of CAPO. **Phase 1** (left) iteratively refines prompts: the LLM identifies errors, generates improved candidates, and a regression guard (the "bouncer") rejects any candidate whose safety-set accuracy drops below the baseline tolerance. Search stops when no further improvement is detected. **Phase 2** (right) tries multiple calibration "lenses" (prefixes that encourage the model to reconsider its first instinct) and selects whichever scores highest on a validation set. The final CAPO prompt combines the optimized prompt with the best lens.

Root cause 1: Overconfidence amplification. When an optimizer evaluates candidate prompts purely by accuracy on a small training set, it selects prompts that exploit the model's confident-but-wrong reasoning patterns rather than correcting them. Consider causal reasoning: the model tends to commit to a first-impression answer with high confidence. GRACE's natural-language "gradient" edits reinforce this tendency—they refine the prompt to match the model's existing biases, producing edits that score well on training examples where the model is already confident but fail on examples requiring careful deliberation. The result is a 9.4-point regression (53.3% vs. 62.7% baseline). This is not a random failure: it is a systematic consequence of optimizing for accuracy without accounting for the model's confidence distribution. We find that prepending a simple calibration instruction—"*Before answering, consider that your initial intuition might be wrong*"—recovers accuracy to 76.0% on the same task (+13.3 points over baseline), confirming that overconfidence, not lack of knowledge, is the bottleneck.

Root cause 2: Unconstrained search causes regression. Existing methods lack mechanisms to prevent optimization from degrading performance on held-out examples. GEPA's evolutionary operators explore diverse prompt mutations but accept any candidate that improves training accuracy, without checking whether the mutation harms performance elsewhere—leading to a 7-point drop on Dyck languages and a 5-point drop on sports understanding. MetaSPO's meta-learned search similarly lacks safety constraints, causing the largest single regression we observe: -17 points on navigation. These regressions are not isolated edge cases—they consistently impact every baseline in our study. The fundamental issue is that prompt optimization without a regression guard is an unconstrained search that can—and frequently does—trade broad competence for narrow improvements.

These two findings—overconfidence amplification and unconstrained regression—are not merely empirical observations. They reveal a fundamental design flaw: existing prompt optimizers evaluate candidates based solely on accuracy, but accuracy

on a small training set is a poor proxy for robust generalization when the model's confidence distribution is misaligned. A candidate that scores 100% by exploiting confident-but-brittle reasoning patterns will be preferred over one that scores 90% through careful deliberation—even though the latter generalizes better. Calibration methods (Xiong et al., 2024; Kadavath et al., 2022) can correct confidence at inference time, but applying them *after* optimization is too late: the optimizer has already converged to a prompt that reinforces overconfident patterns. The key insight is that calibration must be embedded into the search loop itself, so that candidate prompts are evaluated under conditions that penalize overconfident shortcuts. This insight is not obvious, and naive combinations fail. Our compute-equalized experiments (Section 4.4) show that simply adding a regression guard to an existing optimizer (ACE) actually *hurts* performance (82.6%→77.0%), because the guard rejects candidates aggressively while the uncalibrated search keeps generating low-quality ones. Calibration-aware evaluation must come *first*—it changes the optimization landscape by rewarding prompts that elicit deliberative reasoning, which in turn produces candidates that can pass safety constraints.

We introduce **CAPO** (Calibration-Aware Prompt Optimization), a two-phase framework built on this insight. *Phase 1* (calibration-aware search) embeds a calibration prefix into every evaluation, forcing the model to reconsider its confidence before committing to an answer. This changes *what the optimizer selects for*: prompts that succeed under calibrated evaluation are those that encode genuine reasoning strategies, not overconfident shortcuts. A *regression guard* rejects any candidate that degrades accuracy on a held-out safety set beyond a small tolerance, and convergence-based early stopping halts the search before over-optimization occurs. *Phase 2* (calibration prefix selection) selects the best calibration prefix from a small library as a decoupled post-processing step, targeting residual overconfidence on tasks where Phase 1's fixed prefix is suboptimal. Our contributions are as follows:

1. **Empirical diagnosis of prompt optimization failure.** Through systematic evaluation of five optimization methods on six BBH tasks—including multi-seed analysis revealing high run-to-run variance (GRACE: ± 12.2 points)—we identify performance fragmentation as a pervasive problem and trace it to overconfidence amplification and unconstrained regression.

2. **Calibration-aware prompt optimization with safety guarantees.** We embed calibration directly into the search loop and enforce safety through a hard regression constraint. A compute-equalized comparison confirms that CAPO's gains stem from its architectural components rather than additional compute: giving ACE $1.8\times$ more iterations closes only 40% of the performance gap.

3. **A two-phase framework with consistent generalization.** CAPO achieves 89.1% average accuracy across six BBH tasks—+27.8 points over the unoptimized baseline and +15.5 points over the best published baseline (ACE, 73.6%). It is the *only* method to exceed 75% on every task. Cross-model evaluation on GPT-4o-mini and DeepSeek-V3 confirms that both the fragmentation problem and CAPO's improvements transfer across model families (86.8% and 96.8% average respectively). Ablations confirm that the framework is robust to hyperparameter choices including temperature (83.7–88.4%) and safety set configuration (81.3–98.7%).

2 Related Work

Prompt optimization. Automated prompt optimization replaces manual engineering with iterative, gradient-free search over natural language instructions. Zhou et al. (2023) established the generate-and-select paradigm with APE, and Yang et al. (2024) extended it by feeding past prompts and scores back as optimizer context in OPRO. Pryzant et al. (2023) introduced gradient-like natural language feedback (GRACE), while evolutionary approaches apply crossover and mutation to prompt populations (Guo et al., 2024; Fernando et al., 2024; Agrawal et al., 2026). These methods share two shortcomings: they lack *regression safety*—GRACE, for example, exhibits a -9.4 point regression on causal reasoning even as its overall score improves—and they treat optimization as purely score-driven, ignoring whether predictions are well-calibrated under the optimized prompt. CAPO addresses both gaps: the regression guard rejects candidates that degrade held-out performance, while calibration-aware evaluation and prefix selection improve calibration alongside accuracy.

Calibration and uncertainty in LLMs. Xiong et al. (2024) demonstrated that standard prompting yields poorly calibrated LLM predictions, and Kadavath et al. (2022) showed that self-verbalized confidence is unreliable across tasks. Chain-of-thought prompting (Wei et al., 2022) and self-consistency decoding (Wang et al., 2023) indirectly improve calibration through structured reasoning. However, all existing calibration work operates at the *inference* level under a fixed prompt—it does not inform how prompts are optimized. CAPO bridges this gap by embedding calibration into both the search objective (Phase 1) and post-hoc prefix selection (Phase 2), treating calibration as a first-class concern throughout the optimization pipeline.

Search and stopping strategies. Bayesian optimization (Snoek et al., 2012) and differential evolution (Storn and Price, 1997) provide principled frameworks for expensive black-box optimization, but existing prompt optimizers either use fixed iteration budgets (Yang et al., 2024; Zhou et al., 2023) or simple plateau detection (Pryzant et al., 2023). Neither accounts for the non-stationary, high-variance nature of LLM evaluation on small benchmarks, where a few noisy samples can create the illusion of progress. CAPO employs convergence-based stopping with a patience mechanism and pairs it with a regression guard, yielding an adaptive budget that scales with task difficulty without sacrificing safety guarantees.

3 Method

We present CAPO (**C**alibration-**A**ware **P**rompt **O**ptimization), a two-phase method that jointly maximizes task accuracy and calibration safety. Phase 1 performs calibration-aware search with convergence-based stopping and a regression guard; Phase 2 selects the best calibration prefix as post-processing. Figure 1 illustrates the pipeline.

3.1 Problem Formulation

Given a task $\mathcal{T}$ with input-output pairs $\{(x_i, y_i)\}_{i=1}^{N}$ and a language model $\mathcal{M}$, we seek a prompt p^* maximizing accuracy while maintaining calibration safety. Let $A(p, \mathcal{T}, \mathcal{M})$ denote accuracy. We define a safety set $\mathcal{D}_{\text{safe}}$ and require $S(p) = A(p, \mathcal{D}_{\text{safe}}, \mathcal{M}) \geq S(p_0) - \epsilon$ with $\epsilon = 0.03$, where p_0 is the initial prompt.

To counteract model overconfidence (Xiong et al., 2024), we introduce *calibration-aware evaluation*. Let c denote a calibration prefix that encourages reconsidering initial intuitions. The calibration-aware accuracy is:

$$A_c(p, \mathcal{T}, \mathcal{M}) = A(c \oplus p, \mathcal{T}, \mathcal{M}), \qquad (1)$$

where $\oplus$ denotes concatenation. Phase 1 uses A_c exclusively, ensuring optimization under calibrated reasoning.

3.2 Phase 1: Calibration-Aware Search

Phase 1 iteratively refines a prompt through candidate generation, calibration-aware evaluation, a regression guard, and convergence-based stopping.

Initialization. The search begins from a *metacognitive template* p_0 structuring reasoning into five stages: UNDERSTAND, PRELIMINARY JUDGMENT, CRITICAL EVALUATION, FINAL DECISION, and CONFIDENCE. This elicits structured deliberation rather than reactive answering.

Single-candidate generation. At iteration t, we generate a single candidate $\hat{p}_t$ by querying $\mathcal{M}$ with the current best prompt p^*_{t-1}, its accuracy $A_c(p^*_{t-1})$, and a meta-prompt m:

$$\hat{p}_t \sim \mathcal{M}\big(m \oplus p^*_{t-1} \oplus A_c(p^*_{t-1})\big)\,. \qquad (2)$$

Unlike population methods (Yang et al., 2024) that maintain N candidates per iteration requiring $\mathcal{O}(N \cdot |\mathcal{D}|)$ evaluations, our single-candidate design costs $\mathcal{O}(|\mathcal{D}|)$ per round. We rely on the regression guard and convergence stopping—rather than population diversity—to ensure robust search, reducing per-iteration cost by $N\times$ while iterative refinement provides sufficient exploration over multiple rounds.

Evaluation and guard. The candidate is evaluated on training set $\mathcal{D}_{\text{train}}$ ($|\mathcal{D}_{\text{train}}| = 25$) using A_c. Before acceptance, we verify it does not cause regression on safety set $\mathcal{D}_{\text{safe}}$ ($|\mathcal{D}_{\text{safe}}| = 15$):

$$A_c(\hat{p}_t, \mathcal{D}_{\text{safe}}) \geq A_c(p_0, \mathcal{D}_{\text{safe}}) - 0.03. \qquad (3)$$

Anchoring to p_0 (not p^*_{t-1}) ensures a fixed performance floor throughout search. Candidates failing this check are discarded.

Single-objective optimization. A natural alternative is to jointly optimize accuracy, calibration error, and robustness via Pareto selection. However, these competing objectives cause oscillation—a phenomenon we term *signal dilution*. CAPO instead reduces to a single objective (calibration-aware accuracy, Eq. 1) and handles robustness as

a hard constraint (Eq. 3), yielding stable monotonic improvement. We compare against a multi-objective Pareto variant in Appendix C.2, where it achieves only 75.9% average accuracy compared to CAPO's 89.1%, confirming that signal dilution degrades optimization quality.

Convergence-based stopping. Extended refinement beyond a task-dependent saturation point degrades prompt quality as the model overfits to training examples or drifts toward verbose, self-referential phrasing—a phenomenon we term *refinement decay*. To prevent this, we employ convergence-based stopping. Let $\Delta_t = A_c(p_t^*) - A_c(p_{t-1}^*)$ denote the improvement at iteration t. The search terminates when:

$$\sum_{i=t-\rho+1}^{t} \mathbb{1}[\Delta_i > 0] = 0 \text{ and } \max_{i \le t} A_c(p_i^*) - A_c(p_0) < \delta, \tag{4}$$

with patience $\rho = 3$, improvement threshold $\delta = 0.01$, and maximum iterations $T = 8$. In practice, the average stopping point is $t = 3.7$ iterations (range: 2–5)—well below the cap of $T = 8$—confirming that early stopping is both effective and efficient.

Update rule. The best prompt updates only if the candidate improves accuracy and passes the regression guard.

3.3 Phase 2: Calibration Prefix Selection

Phase 2 selects the best calibration prefix as post-processing. From a small library $\mathcal{C} = \{c_0, c_1, c_2, c_3\}$ of prefixes targeting different over-confidence facets, we evaluate each on validation set $\mathcal{D}_{\text{val}}$:

$$c^* = \arg\max_{c_j \in \mathcal{C}} A(c_j \oplus p_T^*, \mathcal{D}_{\text{val}}, \mathcal{M}) \,. \tag{5}$$

The final prompt is $p^* = c^* \oplus p_T^*$ if this improves over p_T^* alone; otherwise $p^* = p_T^*$.

Decoupling prefix selection from Phase 1 avoids confounding variables: the search optimizes prompt content under consistent evaluation, while Phase 2 selects the optimal framing without interfering with the search trajectory.

4 Experimental Setup

Benchmarks. We evaluate on BIG-Bench Hard (BBH) (Suzgun et al., 2023), specifically focusing on the six hardest tasks where Qwen2.5-7B-Instruct remains unsaturated (baseline accuracy $\le 83\%$). These span diverse reasoning types: **Causal Judgement** tests whether a described event caused an outcome (commonsense reasoning). **Navigate** requires tracking position after movement instructions (spatial reasoning). **Web of Lies** asks the model to resolve chains of truth-telling and lying assertions (logical reasoning). **Dyck Languages** requires predicting closing brackets of a partial Dyck sequence (formal/symbolic reasoning). **Object Counting** involves counting objects across multiple collections (arithmetic reasoning). **Sports Understanding** asks whether an athletic scenario is plausible (world knowledge reasoning).

Algorithm 1 CAPO: Calibration-Aware Prompt Optimization

Require: Task $\mathcal{T}$, model $\mathcal{M}$, calibration prefix c, patience $\rho = 3$, tolerance $\epsilon = 0.03$, threshold $\delta = 0.01$, max iterations $T = 8$
Ensure: Optimized prompt p^*
Initialize $p_0^* \leftarrow$ metacognitive template {5-stage structured reasoning}
Evaluate safety baseline: $s_0 \leftarrow A_c(p_0^*, \mathcal{D}_{\text{safe}}, \mathcal{M})$
stale $\leftarrow 0$
for $t = 1$ **to** T **do**
  Sample candidate: $\hat{p}_t \sim \mathcal{M}(m \oplus p_{t-1}^* \oplus A_c(p_{t-1}^*))$
  Evaluate: $A_c(\hat{p}_t, \mathcal{D}_{\text{train}}, \mathcal{M})$
  if $A_c(\hat{p}_t, \mathcal{D}_{\text{safe}}) < s_0 - \epsilon$ **then**
    Reject (regression guard): $p_t^* \leftarrow p_{t-1}^*$; *stale* $\leftarrow$ *stale* $+ 1$
  else if $A_c(\hat{p}_t, \mathcal{D}_{\text{train}}) > A_c(p_{t-1}^*, \mathcal{D}_{\text{train}})$ **then**
    Accept improvement: $p_t^* \leftarrow \hat{p}_t$; *stale* $\leftarrow 0$
  else
    No improvement: $p_t^* \leftarrow p_{t-1}^*$; *stale* $\leftarrow$ *stale* $+ 1$
  end if
  if *stale* $\ge \rho$ **and** $A_c(p_t^*) - A_c(p_0^*) < \delta$ **then**
    break {convergence stopping}
  end if
end for
Phase 2: Select best prefix $c^* \leftarrow \arg\max_{c_j \in \mathcal{C}} A(c_j \oplus p_T^*, \mathcal{D}_{\text{val}}, \mathcal{M})$
Set $p^* \leftarrow c^* \oplus p_T^*$ if $A(c^* \oplus p_T^*) \ge A(p_T^*)$, else $p^* \leftarrow p_T^*$
return p^*

Data splits. For each BBH task, we use the standard provided examples. We split them into a training set ($|\mathcal{D}_{\text{train}}| = 25$), a safety/validation set ($|\mathcal{D}_{\text{safe}}| = 15$), and the remaining examples as the test set. The training set is used for evaluating candidate prompts during search; the safety set is used exclusively by the regression guard. All reported accuracies are on the held-out test set. We use a sequential split (first N examples for training, next for safety, remainder for test) rather than random splitting. While random splits are standard in machine learning, sequential splits are common in prompt optimization (Pryzant et al., 2023; Zhou et al., 2023) because they ensure reproducibility without seed dependence. We verify that this choice does not introduce systematic bias by confirming that baseline accuracy on our test split (61.3%) is consistent with multi-seed experiments that shuffle data order ($62.2\% \pm 0.8\%$ on causal judgement).

Baselines. We compare against an unoptimized baseline and five optimization methods: (i) **Baseline**: the raw Qwen2.5-7B-Instruct model with a minimal instruction prompt (no optimization). (ii) **GRACE** (Pryzant et al., 2023): gradient-like natural-language prompt editing. (iii) **GEPA** (Agrawal et al., 2026): evolutionary operators applied to a prompt population. (iv) **ACE**: adaptive chain-of-thought editing with iterative self-refinement. (v) **MetaSPO**: meta-learned search over the prompt space with self-play optimization. (vi) **PromptQuine**: self-replicating prompt evolution via quine-style generation. ACE, MetaSPO, and PromptQuine are baselines we implement following published descriptions; GRACE and GEPA use their released codebases.

Implementation. All main experiments use Qwen2.5-7B-Instruct (Qwen Team, 2024) accessed via an OpenAI-compatible API endpoint. For cross-model validation in our ablation study (Section 4.3), we additionally evaluate on GPT-4o-mini (OpenAI et al., 2024) and DeepSeek-V3 (DeepSeek-AI et al., 2025) to assess transferability across model families. CAPO hyperparameters: maximum iterations K=8, patience p=3, generation temperature T=0.7, safety set size $|D_{\text{safe}}|$=15, and regression tolerance ϵ=0.03. Full CAPO requires approximately 150–200 API calls per benchmark. Baselines are run with their default hyperparameters and equivalent iteration budgets (10 iterations for search-based methods). All methods use the same model, API, and data splits unless otherwise specified. Further details are provided in Appendix D.

Table 1: Test accuracy (%) on six BBH tasks. **Bold** indicates the best result per column; underline indicates second best. CAPO achieves the highest average accuracy (89.1%) and is the only method above 75% on every task. All methods use Qwen2.5-7B-Instruct.

Method	Causal	Nav	Web	Dyck	Obj	Sports	**Avg**
Baseline	62.7	83.0	56.0	72.0	40.0	54.0	61.3
MetaSPO	61.3	66.0	62.0	68.0	76.0	57.0	65.1
GRACE	53.3	76.0	97.0	70.0	49.0	71.0	69.4
GEPA	76.0	92.0	59.0	65.0	79.0	49.0	70.0
PromptQuine	74.7	86.0	63.0	70.0	78.0	55.0	71.1
ACE	57.3	78.0	84.0	76.0	79.0	67.0	73.6
CAPO	**93.3**	**99.0**	**96.0**	**75.0**	**77.0**	**94.0**	**89.1**

4.1 Main Results

Table 1 presents the main comparison against six baselines on six representative BBH tasks. CAPO achieves 89.1% average accuracy, outperforming the best baseline (ACE, 73.6%) by 15.5 points and the unoptimized model by 27.8 points. To verify these gains are not cherry-picked, we evaluate on all 27 BBH tasks (Appendix C.1); CAPO achieves 93.9% overall average and never regresses below baseline on any task, confirming that our six-task subset captures the most informative unsaturated problems.

The key result is consistency: CAPO is the only method that exceeds 75% on every task in our main set. Every baseline exhibits severe task-specific regressions. GRACE reaches 97.0% on web of lies but drops to 53.3% on causal judgement—9.4 points below the unoptimized baseline. GEPA excels on navigation (92.0%) but falls to 49.0% on sports. ACE and MetaSPO show similar patterns, with ACE at 57.3% on causal and MetaSPO at 66.0% on navigation. Across all 5 baselines × 6 tasks = 30 combinations, **11** exhibit regression below the unoptimized baseline, affecting every published method tested. These failures are systematic, not random: optimization without safeguards exploits task-specific patterns that conflict with broader reasoning.

Error analysis reveals what these regressions look like. On causal judgement and sports understanding, baseline errors are overwhelmingly false negatives (27 FN vs. 1 FP on causal). CAPO reduces false negatives from 27→5 (81% reduction), while GRACE *amplifies* the bias (27→33). This confirms that CAPO's metacognitive template cor-

Table 2: Multi-seed evaluation (mean ± std over 3 seeds) on six BBH tasks. Seeds vary the train/val data order while keeping the test set fixed. GRACE exhibits high variance (±2.0–12.2 points), while the baseline is stable (±0.0–2.6 points).

Method	Causal	Nav	Web	Dyck	Obj	Sports
Baseline	$62.2_{\pm 0.8}$	$83.7_{\pm 0.6}$	$59.0_{\pm 1.0}$	$70.3_{\pm 2.5}$	$40.0_{\pm 2.6}$	$54.0_{\pm 0.0}$
GRACE	$59.1_{\pm 10.8}$	$78.0_{\pm 12.2}$	$90.3_{\pm 5.5}$	$76.3_{\pm 2.1}$	$73.0_{\pm 2.0}$	$70.7_{\pm 8.1}$
CAPO	$\mathbf{81.8}_{\pm \mathbf{7.3}}$	$\mathbf{94.6}_{\pm \mathbf{4.7}}$	$\mathbf{95.0}_{\pm \mathbf{5.6}}$	$\mathbf{78.7}_{\pm \mathbf{2.9}}$	$\mathbf{77.3}_{\pm \mathbf{3.5}}$	$\mathbf{86.0}_{\pm \mathbf{1.0}}$

Table 3: Component ablation. Each row removes a component from the full CAPO pipeline. Δ is the change in average accuracy relative to the full system.

Variant	Causal	Nav	Web	Dyck	Obj	Sports	**Avg**	Δ
CAPO (full)	93.3	99.0	96.0	75.0	77.0	94.0	**89.1**	—
w/o Phase 2	85.3	100.0	87.0	81.0	74.0	94.0	86.9	−2.2
w/o Phase 1	65.3	90.0	97.0	71.0	72.0	74.0	78.2	−10.8
Calibration prefix only	98.7	88.0	70.0	75.0	74.0	58.0	77.3	−11.8
No optimization	62.7	83.0	56.0	72.0	40.0	54.0	61.3	−27.8

Table 4: Comparison against multi-objective optimization. The Pareto variant jointly optimizes accuracy, calibration error, and robustness.

Architecture	Causal	Navigate	Web Lies	Dyck	Obj Count	Sports	Avg
Single-objective + constraint (**CAPO**)	**93.3**	**99.0**	**96.0**	**75.0**	77.0	**94.0**	**89.1**
Multi-objective Pareto	82.7	90.0	58.0	71.0	**87.0**	67.0	75.9

rects a systematic reasoning bias rather than just improving accuracy on average.

4.2 Statistical Significance

Table 2 reports mean and standard deviation over three random seeds, varying train/validation split order. The unoptimized baseline is highly stable (±0.6–1.0 points). GRACE exhibits alarming variance (±10.8–12.2 points on causal and navigation), confirming that its task-specific regressions are not reproducible across splits. CAPO shows moderate variance (±4.7–7.3 points), with even its worst runs exceeding both baseline and GRACE means.

4.3 Ablation Study

Two-phase decomposition. Table 3 decomposes CAPO's two phases. Phase 1 (calibration-aware search) is the primary driver: removing it costs −10.8 points average. Phase 2 (prefix selection) contributes −2.2 points on average but up to −9 points on web of lies, confirming it targets overconfidence-prone tasks. Using only the calibration prefix without search achieves 98.7% on causal judgement but collapses to 58.0% on sports—worse than baseline—confirming that calibration alone is insufficient.

Single-objective vs. multi-objective. We compare against a multi-objective alternative that jointly optimizes accuracy, calibration error, and robustness via Pareto selection (Table 4). This Pareto Reflection variant (See details in subsection C.2) achieves only 75.9% average accuracy—worse than several baselines and 13.2 points below CAPO—confirming that competing objectives cause *signal dilution*. Our design reduces to a single objective (calibration-aware accuracy, Eq. 1) with robustness as a hard constraint (Eq. 3), yielding stable monotonic improvement.

Metacognitive template. The metacognitive template contributes +5.7 points (Appendix Table 9). The five-stage structure enforces deliberation; removing it reverts to reactive answering.

Sensitivity analysis. We analyze sensitivity to generation temperature and safety set size in Appendix C.3. Temperature $T = 0.7$ balances exploration and stability; safety set size $|\mathcal{D}_{\text{safe}}| = 15$ provides sufficient signal without reducing training data.

4.4 Compute-Equalized Comparison

Table 6 controls for compute budget. Giving ACE 1.8× more iterations closes only 40% of the gap (73.1%→82.6% vs. CAPO's 96.1%), confirming

Table 5: Cross-model evaluation. Accuracy (%) across three model families. CAPO transfers consistently; GRACE causes regressions on all models.

Model	Method	Causal	Nav	Web	Dyck	Obj	Sports	**Avg**
Qwen2.5 -7B	Baseline	62.7	83.0	56.0	72.0	40.0	54.0	61.3
	GRACE	53.3	76.0	97.0	70.0	49.0	71.0	69.4
	CAPO	**93.3**	**99.0**	**96.0**	**75.0**	**77.0**	**94.0**	**89.1**
GPT-4o -mini	Baseline	65.3	96.0	61.0	82.0	81.0	79.0	77.4
	GRACE	70.7	70.0	69.0	77.0	**93.0**	78.0	76.3
	CAPO	**86.7**	**95.0**	**90.0**	**80.0**	**93.0**	76.0	**86.8**
DeepSeek -V3	Baseline	80.0	98.0	71.0	**95.0**	96.0	91.0	88.5
	GRACE	65.3	98.0	69.0	93.0	95.0	91.0	85.2
	CAPO	**98.7**	**99.0**	**100.0**	86.0	**99.0**	**98.0**	**96.8**

Table 6: Compute-equalized comparison. ACE is given $1.8\times$ the iteration budget to match CAPO's API call count. CAPO's components are progressively added.

Condition	Causal	Nav	Web	**Avg**
ACE (standard, 10 iter)	57.3	78.0	84.0	73.1
ACE ($1.8\times$ budget, 18 iter)	66.7	88.0	93.0	82.6
ACE + regression guard	56.0	77.0	**98.0**	77.0
ACE + calibration eval	90.7	86.0	89.0	88.6
ACE + guard + calibration	80.0	84.0	92.0	85.3
CAPO (Ours)	**93.3**	**99.0**	96.0	**96.1**

CAPO's advantage is architectural. Calibration-aware evaluation is the most impactful single component (+6.0 points over extra-budget ACE). Notably, adding the regression guard *alone* hurts (82.6%→77.0%) because it rejects candidates aggressively without calibration to improve candidate quality—the guard needs calibration-aware evaluation to function effectively.

4.5 Cross-Model Evaluation

Table 5 confirms that both CAPO's improvements and the fragmentation problem transfer across model families. CAPO achieves the highest average on all three models (+27.8, +9.4, +8.3 over baseline respectively). GRACE causes regressions on *every* model: -9.4 on Qwen causal, -26.0 on GPT-4o-mini navigate, -14.7 on DeepSeek causal. CAPO's gains are larger on weaker models (more overconfidence to correct), but remain meaningful even on DeepSeek-V3 (+18.7 on causal, +29.0 on web of lies). CAPO's weakest results are consistently on Dyck languages across all models, confirming that calibration-induced deliberation hurts purely symbolic tasks (see Appendix C.4). Convergence behavior, calibration analysis (ECE), and per-prefix selection details are in Appendix C.

5 Discussion

CAPO's robustness stems from a key insight: calibration and safety must be *architectural constraints*, not afterthoughts. The regression guard prevents catastrophic degradation precisely because it anchors to the initial baseline rather than the current best—a fixed safety floor rather than a moving target. Similarly, decoupling content optimization (Phase 1) from prefix selection (Phase 2) avoids confounding variables that cause unstable dynamics in joint optimization. Importantly, CAPO's "calibration awareness" operates as *reasoning debiasing* rather than probability calibration. It corrects systematic "No" bias on counterintuitive questions (false negatives reduced 81%) even while self-reported confidence may increase.

6 Conclusion

We identify overconfidence amplification and unconstrained regression as the two root causes of prompt optimization fragmentation, and introduce CAPO—the first framework to embed calibration directly into the optimization loop while enforcing safety through a hard regression constraint. CAPO achieves 89.1% average accuracy on six BBH benchmarks (+15.5 over the best baseline), is the only method above 75% on every task, and transfers consistently to GPT-4o-mini and DeepSeek-V3. Beyond prompt optimization, the same principle applies to RLHF reward training (where reward hacking mirrors overconfidence amplification) and agentic tool-use pipelines (where first-impression bias leads to wrong tool selection). More broadly, the regression guard offers a general safety mechanism for any iterative black-box optimization of LLM behavior, providing monotonic safety guarantees without model internals access.

7 Limitations

Despite CAPO's strong results, several limitations remain. First, while cross-model evaluation on GPT-4o-mini and DeepSeek-V3 confirms transfer, broader evaluation across scales (1B–70B+) and architectures (Llama 3, Gemma) would strengthen claims. Second, our evaluation focuses on closed-ended reasoning benchmarks; effectiveness on open-ended generation, code synthesis, and multi-turn dialogue is unknown, as calibration prefixes may interact differently with generation-oriented tasks where multiple valid answers exist. Third, CAPO uses approximately $1.8\times$ the API calls of single-phase methods—for cost-sensitive settings, Phase 1 alone provides 86.9% accuracy at lower cost as a practical trade-off. Fourth, calibration-induced deliberation harms purely symbolic tasks like Dyck languages where fast pattern-matching is optimal, suggesting that future work could detect task type and apply calibration selectively.

8 Ethical Considerations

CAPO is designed to improve LLM reasoning reliability on closed-ended benchmarks, with safety as a core objective: the regression guard prevents prompt optimization from degrading performance on held-out examples, reducing the risk of deploying over-optimized prompts that fail silently under distribution shift. While prompt optimization techniques could in principle be misused to elicit harmful outputs, CAPO's design offers two mitigating factors: the regression guard requires candidates to maintain performance on a safety set that can be populated with safety-relevant examples, and the calibration prefix encourages deliberative, uncertainty-aware reasoning rather than overconfident generation. We also note that adversarial prompt optimization (e.g., jailbreaking) typically requires white-box gradient access, whereas CAPO operates entirely through black-box API calls with natural-language feedback. Our evaluations use BIG-Bench Hard, a public English-only benchmark suite, so results may not generalize to other languages, modalities, or open-ended generation tasks. Experiments require approximately 5,000 total API calls—modest compute accessed through standard inference APIs with no fine-tuning, keeping reproducibility barriers low. This work involves no human participants, no personally identifiable information, and no data collection beyond existing public benchmarks.

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. 2026. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *Proceedings of the 14th International Conference on Learning Representations (ICLR)*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 33.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2024. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, and 1 others. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

OpenAI, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, and 400 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

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

Qwen Team. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25.

Rainer Storn and Kenneth Price. 1997. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. BIG-Bench Hard: Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics (ACL)*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*.

A Detailed Algorithm

This section provides the complete CAPO algorithm, including all subroutines referenced in Section 3 and the main text. Algorithm 2 describes the full two-phase pipeline. The metacognitive template and calibration prefixes used in our experiments are listed verbatim below.

Subroutine: CalibratedEval. Given a prompt p and example set $\mathcal{E}$, CALIBRATEDEVAL prepends the default calibration prefix π_0 (see below) to p if it is not already present, then evaluates accuracy by querying the model on each example. Note: in Phase 2, EVAL (without calibration prefix) is used instead, so that each candidate prefix π_i is tested alone without stacking on top of π_0.

Subroutine: FirewallCheck. Given a candidate prompt c, safety set $\mathcal{S}$, baseline accuracy a_{base}, and tolerance τ: evaluate c on $\mathcal{S}$ via CALIBRATEDEVAL and return `true` if and only if accuracy $\geq a_{\text{base}} - \tau$.

Metacognitive template. The initial prompt template used in Phase 1 is:

```
You are solving a {task_type} question.
Follow these 5 metacognitive stages:

Stage 1 - UNDERSTAND: Read the question
carefully. What is being asked? What
are the key facts?

Stage 2 - PRELIMINARY JUDGMENT: What is
your initial answer? Why?

Stage 3 - CRITICAL EVALUATION: Challenge
your initial judgment. What evidence
contradicts it? What assumptions might
be wrong? Consider the opposite answer
seriously.

Stage 4 - FINAL DECISION: Based on your
critical evaluation, what is the correct
answer? Has your view changed from Stage
2?

Stage 5 - CONFIDENCE: Rate your
confidence (low/medium/high) and
explain why.

After all 5 stages, provide your final
answer on the last line.
```

Calibration prefixes. The default calibration prefix used by CALIBRATEDEVAL throughout the pipeline is:

```
π0: "Before answering, consider that your
initial intuition might be wrong. Think
about alternative interpretations."
```

The three additional CalibGuard prefixes tested in Phase 2 are:

```
π1: "Warning: This question may have
a counterintuitive answer. Consider
multiple perspectives before deciding."

π2: "Important: Do not rush to the
obvious answer. Carefully weigh evidence
for and against each option."

π3: "Note: This is a nuanced question
where the obvious answer may be wrong.
Consider all factors carefully before
deciding."
```

B Benchmark Details

We evaluate on six tasks from BIG-Bench Hard (BBH) (Suzgun et al., 2023), a curated subset of BIG-Bench tasks on which language models previously underperformed average human annotators. All tasks are drawn from the `lukaemon/bbh` HuggingFace repository. Each task's examples are sequentially split: the first 25 examples form the training set, the next 15 form the safety/validation set, and the remainder form the test set (see Section 4). Table 7 summarizes the characteristics of each benchmark.

Example inputs and outputs. Below we provide a representative example from each benchmark to illustrate the task format.

- **Causal Judgement.** *Input:* "How would a typical person answer each of the following questions about causation? Janet is an employee in a factory. Since she works in the maintenance department, she knows how to grease and oil all the machines... Did Kate cause the machine to break down?" *Target:* No
- **Navigate.** *Input:* "If you follow these instructions, do you return to the starting point? Turn left. Turn around. Turn left. Take 7 steps. Take 2 steps. Take 4 steps." *Target:* No
- **Web of Lies.** *Input:* "Question: Fidel tells the truth. Jerry says Fidel tells the truth. Vina says Jerry lies. Millicent says Vina tells the truth. Does Millicent tell the truth?" *Target:* No
- **Dyck Languages.** *Input:* "Complete the rest of the sequence, making sure that the parentheses are closed properly. Input: (( ) { }" *Target:*)
- **Object Counting.** *Input:* "I have a blackberry, a clarinet, a nectarine, a plum, a strawberry, a banana, a flute, an orange, and a violin. How many fruits do I have?" *Target:* 6

Table 7: Benchmark characteristics. All tasks are from BIG-Bench Hard (Suzgun et al., 2023). Sizes are computed from the 40/20/40 split of the full dataset. Answer type indicates the expected output format.

Benchmark	Task Type	Train	Val	Test	Answer Type
Causal Judgement	Causal reasoning	75	37	75	Yes / No
Navigate	Spatial reasoning	100	50	100	Yes / No
Web of Lies	Logical deduction	100	50	100	Yes / No
Dyck Languages	Formal languages	100	50	100	Token sequence
Object Counting	Arithmetic counting	100	50	100	Integer
Sports Understanding	Commonsense reasoning	100	50	100	Yes / No

- **Sports Understanding.** *Input:* "Is the following sentence plausible? 'Bam Adebayo scored a bicycle kick.' " *Target:* No

C Extended Results

This section provides the complete results across all six benchmarks, along with a per-method analysis of strengths and failure modes.

C.1 Full BBH Evaluation (27 Tasks)

To verify that CAPO does not cherry-pick favorable tasks, we evaluate on *all 27 BBH tasks*. Table 8 reports results for five methods. Most additional tasks are saturated (≥99% baseline), confirming that our six selected tasks are the most informative. CAPO achieves the highest overall average (93.9%) and never regresses below baseline on any task.

C.2 Pareto Multi-Objective Baseline

The Pareto Reflection variant (Table 4 in the main text) formulates prompt optimization as a multi-objective problem, jointly optimizing three criteria: task accuracy, expected calibration error (ECE), and robustness (minimum accuracy across safety-set examples). At each iteration, the method generates multiple candidate prompts, evaluates all three objectives, and retains only Pareto-dominant candidates—those not strictly worse than any other candidate on all objectives simultaneously. The final prompt is selected from the Pareto front by maximizing accuracy among non-dominated solutions.

This approach achieves only 75.9% average accuracy, 13.2 points below CAPO. The failure is *signal dilution*: when accuracy, calibration, and robustness compete as co-equal objectives, the optimizer cannot commit to improving any single axis. For example, a candidate that improves accuracy on causal judgement may be rejected because it slightly worsens ECE on another task. CAPO avoids this by reducing the problem to a single objective (calibration-aware accuracy) with robustness as a hard constraint rather than a competing goal.

C.3 Sensitivity Analysis

C.3.1 Metacognitive Template Contribution

Table 9 shows that removing the five-stage metacognitive template reduces average accuracy from 89.1% to 83.4% (-5.7 points), confirming that structured deliberation is critical for debiasing.

C.3.2 Temperature Sensitivity

Table 10 reports temperature sensitivity during candidate generation. $T = 0.7$ achieves the best balance between exploration and stability (89.1%), while lower temperatures (0.3–0.5) produce conservative candidates with limited gains and higher temperatures (0.9) increase variance without improving average performance.

C.3.3 Safety Set Size Sensitivity

Table 11 reports safety set size sensitivity. $|\mathcal{D}_{\text{safe}}| = 15$ provides sufficient signal to detect regressions without reducing training data. Smaller sets (5–10) miss regressions, allowing harmful prompts through; larger sets (20) offer diminishing returns.

C.4 Per-Method Analysis

We provide a detailed analysis of why each external baseline succeeds or fails on specific tasks.

No Optimization (61.3% avg). The unoptimized baseline uses a generic task instruction with no calibration or iterative refinement. It performs reasonably on Navigate (83%) and Dyck (72%), where the model's pretraining provides sufficient structure, but collapses on Object Counting (40%) and Sports Understanding (54%), tasks requiring precise enumeration and domain-specific commonsense.

MetaSPO (65.1% avg). MetaSPO's meta-learned search provides moderate gains on Object Counting (+36 over baseline) but collapses on

Table 8: Full BBH evaluation: accuracy (%) on all 27 tasks. **Bold** = best per row. Most tasks are saturated; CAPO leads on unsaturated tasks and matches baselines on saturated ones.

Task	Baseline	GRACE	GEPA	ACE	**CAPO**
Boolean Expressions	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Causal Judgement	62.7	53.3	76.0	57.3	**93.3**
Date Understanding	99.0	99.0	99.0	**100.0**	**100.0**
Disambiguation QA	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Dyck Languages	72.0	70.0	65.0	**76.0**	75.0
Formal Fallacies	77.0	72.0	81.0	**97.0**	91.0
Geometric Shapes	97.0	91.0	**100.0**	**100.0**	**100.0**
Hyperbaton	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Logical Ded. (5 obj)	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Logical Ded. (7 obj)	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Logical Ded. (3 obj)	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Movie Recommendation	**99.0**	**99.0**	**99.0**	**99.0**	**99.0**
Multistep Arithmetic	88.0	**95.0**	92.0	94.0	92.0
Navigate	83.0	76.0	92.0	78.0	**99.0**
Object Counting	40.0	49.0	**79.0**	**79.0**	77.0
Penguins in a Table	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Colored Objects	99.0	**100.0**	99.0	**100.0**	**100.0**
Ruin Names	98.0	99.0	98.0	**100.0**	**100.0**
Salient Translation	**100.0**	99.0	**100.0**	**100.0**	**100.0**
Snarks	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Sports Understanding	54.0	71.0	49.0	67.0	**94.0**
Temporal Sequences	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Tracking (5 obj)	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Tracking (7 obj)	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Tracking (3 obj)	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Web of Lies	56.0	**97.0**	59.0	84.0	96.0
Word Sorting	0.0	**21.0**	0.0	20.0	20.0
Average (27)	86.1	88.6	88.4	90.8	**93.9**

Table 9: Metacognitive template contribution. Replacing the five-stage template with a simple instruction.

Variant	Avg Accuracy	Δ
CAPO (full)	**89.1**	—
w/o metacognitive template	83.4	−5.7

Table 10: Temperature sensitivity during candidate generation.

Temperature	Avg Accuracy	Δ
0.3	83.7	−5.4
0.5	86.2	−2.9
0.7 (**CAPO**)	**89.1**	—
0.9	88.4	−0.7

Navigate (66%, −17 points below baseline)—the largest single regression we observe. Its meta-learned search policy overfits to the training signal, producing prompts that sacrifice broad competence for narrow improvements.

GRACE (69.4% avg). GRACE's gradient-like feedback loop excels on Web of Lies (97%), where error analysis directly reveals the logical structure the model must follow. However, it catastrophically regresses on Causal Judgement (53.3%, *below* the 62.7% baseline), demonstrating the overconfidence amplification problem: GRACE's edits reinforce the model's existing bias toward confident-but-wrong causal attributions rather than correcting them.

Table 11: Safety set size sensitivity. Varying $|\mathcal{D}_{\text{safe}}|$ while keeping total held-out examples fixed.

Safety Set Size	Avg Accuracy	Δ
5	81.3	−7.8
10	85.6	−3.5
15 (**CAPO**)	**89.1**	—
20	88.2	−0.9

GEPA (70.0% avg). GEPA's evolutionary operators provide diversity but are inefficient on small evaluation budgets. It achieves strong results on Navigate (92%) and Object Counting (79%) through population diversity, but collapses on Sports Understanding (49%, *below* baseline) because evolutionary crossover between prompts for commonsense tasks produces incoherent instructions.

PromptQuine (71.1% avg). PromptQuine's self-replicating evolution achieves the second-highest baseline average, with consistent but unspectacular results. It reaches 86% on Navigate and 78% on Object Counting, but fails to exceed 70% on Web of Lies (63%) and Sports Understanding (55%), suggesting that quine-style self-replication preserves prompt structure at the expense of task-specific adaptation.

ACE (73.6% avg). ACE achieves the highest baseline average through adaptive chain-of-thought editing. It is the most consistent baseline, with four tasks above 76%, but still falls to 57.3% on Causal Judgement—confirming that even sophisticated self-refinement cannot overcome the overconfidence problem without explicit calibration.

D Implementation Details

D.1 Hyperparameters

Table 12 lists all hyperparameters used in the CAPO system and baseline methods. All hyperparameters were set based on preliminary experiments on two held-out BBH tasks (not included in the final evaluation) and were not tuned per-benchmark.

D.2 Compute Budget

Table 13 breaks down the approximate API call count per method per benchmark. The total compute budget for all experiments (7 methods $\times$ 6 benchmarks, including reruns and ablations) was approximately 5,000 API calls.

E AI Assistant Disclosure

We used AI assistants (Claude, ChatGPT) for grammar correction and figure refinement during the preparation of this paper. All experimental design, implementation, results, and scientific claims are the authors' own work.

Table 12: Hyperparameters for CAPO and baseline methods. All values are fixed across benchmarks.

Component	Hyperparameter	Value
Phase 1 (Search)	Max iterations (K)	8
	Patience (P)	3
	Convergence threshold (δ)	0.01
	Generation temperature (T)	0.7
	Training subset size	25
Regression Guard	Safety set size	15 (first 15 val examples)
	Tolerance (τ)	0.03
Phase 2 (Prefix Selection)	Number of additional prefixes tested	3
	Validation set	Val examples 16–end
LLM Generation	Max tokens (prompt generation)	1024
	Max tokens (task evaluation)	2048
Baseline Methods	GRACE iterations	5
	GEPA population size	4

Table 13: Approximate API calls per method per benchmark. Counts include all evaluation and generation calls. CAPO's overhead is roughly $2\times$ that of single-phase methods.

Method	Calls / Benchmark	Relative Cost
No Optimization	$\sim$100	$1.0\times$
GRACE	$\sim$75	$0.8\times$
GEPA	$\sim$90	$0.9\times$
ACE	$\sim$100	$1.0\times$
MetaSPO	$\sim$100	$1.0\times$
PromptQuine	$\sim$85	$0.9\times$
CAPO	**$\sim$180**	$\mathbf{1.8}\times$

Algorithm 2 CAPO: Calibration-Aware Prompt Optimization (Complete)

Require: Benchmark $\mathcal{B} = (\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}})$, model $\mathcal{M}$, max iterations $K = 8$, patience $P = 3$, threshold $\delta = 0.01$, firewall tolerance $\tau = 0.03$

Ensure: Optimized prompt p^*, test accuracy a^*

// Phase 1: Calibration-Aware Search
$p \leftarrow \textsc{MetacognitiveTemplate}(\text{task_type})$ ▷ Initialize with structured prompt
$\mathcal{S} \leftarrow \mathcal{D}_{\text{val}}[1{:}15]$ ▷ Safety set for firewall
$a_{\text{base}} \leftarrow \textsc{CalibratedEval}(p, \mathcal{S}, \mathcal{M})$ ▷ Baseline with calibration prefix
$a_{\text{best}} \leftarrow a_{\text{base}}$; $p_{\text{best}} \leftarrow p$; stall $\leftarrow 0$
trajectory $\leftarrow [a_{\text{base}}]$
for $k = 1, \ldots, K$ **do**
  // Convergence check
  if $|\text{trajectory}| \geq 3$ **and** $\max(\text{trajectory}[-3{:}]) - \min(\text{trajectory}[-3{:}]) < \delta$ **and** stall $\geq P$ **then**
    break ▷ Diminishing returns detected
  end if
  $\mathcal{E} \leftarrow \textsc{CalibratedEval}(p, \mathcal{D}_{\text{train}}[1{:}25], \mathcal{M})$ ▷ Evaluate on training subset
  $W \leftarrow \{e \in \mathcal{E} : e.\text{correct} = \texttt{false}\}$ ▷ Collect errors
  if $W = \emptyset$ **then**
    $a_{\text{val}} \leftarrow \textsc{CalibratedEval}(p, \mathcal{D}_{\text{val}}[16{:}], \mathcal{M})$
    if $a_{\text{val}} \geq 0.95$ **then**
      break ▷ Near-perfect on both splits
    end if
  end if
  // Candidate generation
  $c^* \leftarrow \text{LLM}\textsc{Generate}(\text{“Improve this prompt”}, p, W[{:}3], T)$ ▷ T=0.7 default
  $a_{c^*} \leftarrow \textsc{CalibratedEval}(c^*, \mathcal{D}_{\text{val}}[16{:}], \mathcal{M})$
  if $a_{c^*} > a_{\text{best}}$ **and** $\textsc{FirewallCheck}(c^*, \mathcal{S}, a_{\text{base}}, \tau)$ **then**
    $a_{\text{best}} \leftarrow a_{c^*}$; $p_{\text{best}} \leftarrow c^*$; $p \leftarrow c^*$; stall $\leftarrow 0$
  else
    stall $\leftarrow$ stall $+ 1$
  end if
  Append a_{best} to trajectory
end for

// Phase 2: Calibration Prefix Selection
$a_{\text{final}} \leftarrow \textsc{CalibratedEval}(p_{\text{best}}, \mathcal{D}_{\text{val}}[16{:}], \mathcal{M})$
$p^* \leftarrow p_{\text{best}}$
for each calibration prefix $\pi_i \in \Pi$ **do**
  $a_i \leftarrow \textsc{Eval}(\pi_i \oplus p_{\text{best}}, \mathcal{D}_{\text{val}}[16{:}], \mathcal{M})$ ▷ Test prefix without default calibration
  if $a_i > a_{\text{final}}$ **then**
    $a_{\text{final}} \leftarrow a_i$; $p^* \leftarrow \pi_i \oplus p_{\text{best}}$
  end if
end for
$a^* \leftarrow \textsc{CalibratedEval}(p^*, \mathcal{D}_{\text{test}}, \mathcal{M})$ ▷ Final test evaluation
return p^*, a^*