Select Smarter, Not More? Prompt-Aware Evaluation Scheduling with Submodular Guarantees

Haoyue Liu[1,2,†] Xiaoyu Ma[1,†] Yiwen Li[1] Zhichao Wang[1] Shuguang Cui[1,2,3] Xiaoying Tang[1,2,3,*]

[1]School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen
[2]Shenzhen Future Network of Intelligence Institute (FNii-Shenzhen)
[3]Guangdong Provincial Key Laboratory of Future Networks of Intelligence, CUHK(SZ)
[†]Equal contribution. [*]Corresponding author.
Emails: haoyueliu@link.cuhk.edu.cn, tangxiaoying@cuhk.edu.cn

Abstract

Prompt optimizers invest heavily in *how* to generate better prompts, yet pay almost no attention to *which examples* they use to judge them. This evaluation subset directly shapes every feedback signal that the optimizer receives, while existing methods either fix it before optimization begins (principled but agnostic to the evolving prompt population) or adapt it heuristically (flexible but unstable). We bridge this gap by constructing an *online adaptive testing* problem: Prompts are examinees, training examples are test items, and the scheduler selects items that best discriminate among the strongest candidates. We introduce POES (Prompt-Aware Online Evaluation Scheduling), which pairs a monotone submodular objective over discrimination and coverage with bounded subset updates, yielding a $(1-1/e)$ cold-start guarantee and a warm-start tracking bound under bounded inter-round drift. Across a 35-task APO suite plus 57 MMLU subjects and 6 optimizer–model configurations, POES achieves the highest mean accuracy in *every* configuration, improving over the best baseline by $+3.0$ pp to $+7.8$ pp. These results show that adapting the evaluation subset to the evolving prompt population can improve APO across optimizers and worker models. Our code is publicly available at https://github.com/T-Lab-CUHKSZ/POES.

1 Introduction

Automatic prompt optimization is only as reliable as the feedback used to guide it. Despite the well-documented sensitivity of LLMs to prompt wording [1, 2] and extensive work on prompt-proposal mechanisms since in-context learning [3] and chain-of-thought prompting [4], the evaluation side of automatic prompt optimization (APO) remains comparatively underexplored. Yet evaluation is not merely a bookkeeping step: The subset on which candidate prompts are judged defines the learning signal that the optimizer follows.

In each APO round, a *prompt optimizer* proposes rewritten *candidate prompts*, which a fixed *worker model* evaluates on the same subset of labeled examples. The resulting scores and errors guide the next round. This *evaluation subset* is drawn from a larger training pool, separate from the held-out evaluation split excluded from optimization. Its composition determines which differences among candidates are visible: Examples that all candidates solve or fail leave them tied, while narrow coverage favors prompts that work only on part of the task. An effective subset must therefore distinguish candidates, represent the task as a whole, and remain stable enough across rounds for comparable scores. We term this per-round selection *evaluation-subset scheduling*, a problem orthogonal to candidate generation.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

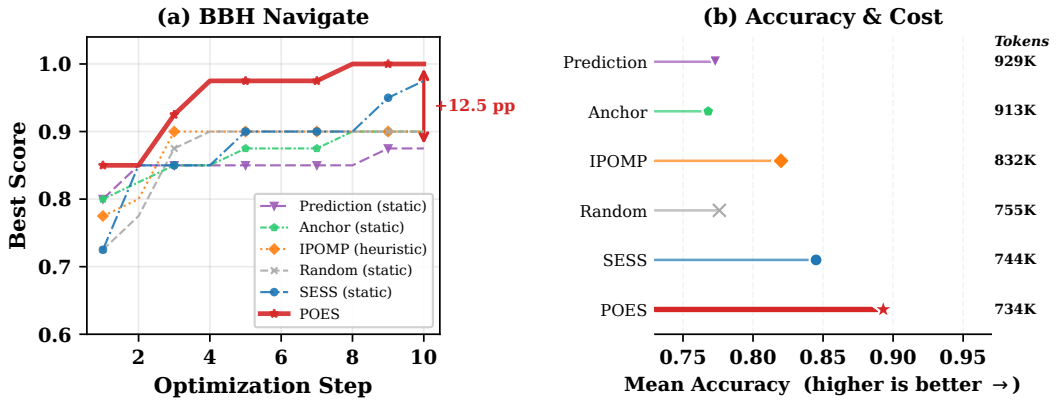


Figure 1. **(a)** BBH Navigate per-round training-subset score: POES keeps improving while baselines plateau (arrow: gap vs. Prediction). **(b)** Held-out test accuracy vs. token cost across 4 tasks (Dyck, GSM-Hard, MultiArith, Navigate) at k=20: POES achieves the highest mean accuracy with the lowest token cost among all schedulers.

Evaluation-subset scheduling is, however, a practical bottleneck in current APO pipelines. Most optimizers [5, 6, 7, 8, 9] fix or hand-tune the evaluation subset, often sampling many examples to stabilize scores. SESS [10] and IPOMP [11] reduce this cost, but static selection cannot track evolving candidates, while heuristic adaptation lacks a stable optimization objective. The difficulty is that informativeness is prompt-dependent: An example separating two weak prompts may become uninformative once both solve it, whereas another becomes decisive as the leaders converge. This calls for online, prompt-aware scheduling that preserves coverage and temporal stability—a natural item response theory (IRT) [12] problem.

We propose POES (Prompt-Aware Online Evaluation Scheduling), which treats APO evaluation as *online adaptive testing*: Prompts are examinees, and training examples are test items selected to distinguish the strongest candidates. Its monotone submodular objective combines IRT-based discrimination with facility-location coverage; bounded warm-start swaps prevent abrupt subset changes. Unlike static selection, the scheduler adapts to the prompt population; unlike unconstrained heuristic adaptation, it follows an explicit optimization objective.

Empirically, POES improves both optimization quality and evaluation efficiency. Figure 1a shows the optimization trajectory on BBH Navigate: Static and heuristic baselines plateau after the early rounds, whereas POES continues to improve by updating the subset around the current prompt competition, reaching $+12.5$ pp over the prediction-based baseline. Figure 1b illustrates the resulting accuracy–cost tradeoff (mean over 4 representative tasks): At k=20, POES achieves the best mean accuracy among all schedulers while using the fewest tokens.

POES *advances evaluation-subset scheduling for APO by making it prompt-aware, theoretically grounded, and temporally stable.* Our contributions are three-fold:

- **(i)** We formalize APO evaluation-subset scheduling as an *online adaptive testing* problem and identify three desiderata—informativeness, coverage, and temporal stability—that prior APO subset schedulers do not jointly satisfy.
- **(ii)** We introduce POES, which combines IRT-based discrimination and facility-location coverage in a single monotone submodular objective and updates the subset across rounds through bounded warm-start swaps, yielding Fisher-information motivation, a $(1-1/e)$ cold-start guarantee, warm-start drift/improvement/local-optimality guarantees, and a tracking bound under bounded inter-round drift.
- **(iii)** Across a 35-task APO suite plus 57 MMLU subjects and a 3×2 optimizer–model matrix, POES achieves the highest mean accuracy in all six configurations ($+3.0$ pp to $+7.8$ pp over the best baseline).[1]

[1]Throughout, accuracy improvements are reported in absolute percentage points ("pp", i.e., differences between accuracy values in $[0, 1]$); token-cost figures are reported as relative percentages.

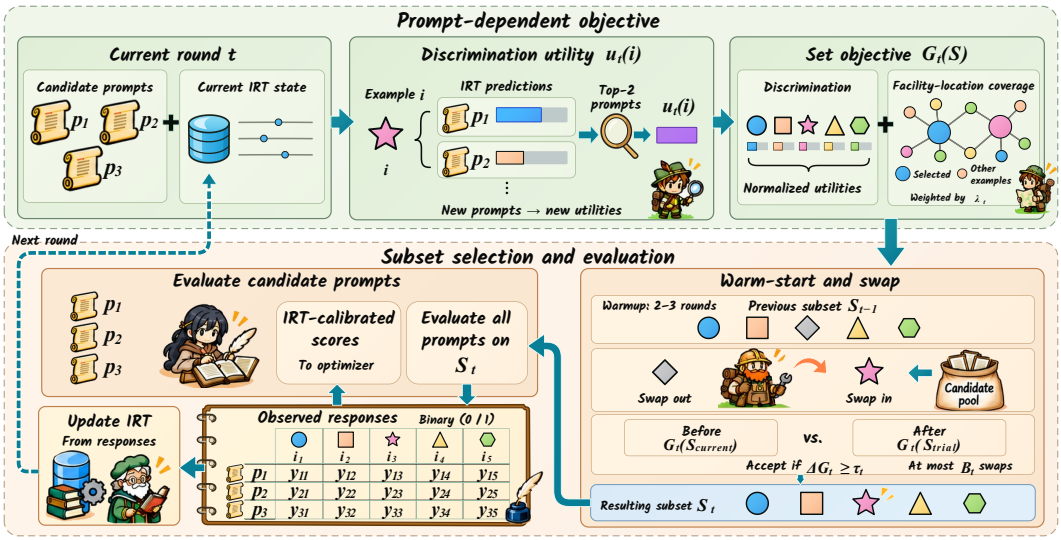


Figure 2. Overview of the POES workflow. Given the current candidate prompts and IRT state, POES estimates prompt-dependent discrimination utilities, combines them with full-pool coverage into the per-round objective G_t, and constructs S_t through bounded warm-start swaps from S_{t-1}. All candidate prompts are evaluated on the shared subset S_t. The resulting binary outcomes update the scheduler's IRT state for the next round, while IRT-calibrated scores are returned to the optimizer. The first two to three rounds use random warmup subsets before active scheduling begins; see Algorithm 1.

2 Method

2.1 Problem Formulation

Let $V=\{1,\dots,N\}$ index the labeled training examples and P_t the candidate prompts at optimization round t. The scheduler selects $S_t \subseteq V$ with $|S_t|=k$ and evaluates every candidate on this same subset. Binary outcomes $y_{p,i} \in \{0,1\}$ record correctness, and the raw score of p is its subset accuracy $\hat{a}_t(p) = k^{-1}\sum_{i\in S_t} y_{p,i}$. The subset should be *informative* about leading prompts, *representative* of the pool, and *stable* across rounds; when scores from different rounds are compared, the optimizer uses the IRT-calibrated scores in Section 2.6.

Definition 1 (Online Evaluation Scheduling)**.** *Given pool V of size N, budget k, and T optimization rounds, produce subsets $S_1,\dots,S_T$ with $|S_t|=k$ that maximize the quality of the prompt returned by the optimizer after T rounds, as measured on a held-out test set.*

With no test-set feedback available online, the scheduler selects each S_t by a per-round surrogate built from the training pool and observed outcomes: Discrimination and coverage make the feedback informative and representative, and bounded swaps keep it consistent across rounds. Its link to held-out prompt quality is validated empirically (Section 3, Section F).

Figure 2 shows where POES sits in the optimization loop. The rest of this section addresses four questions: how to measure prompt-discriminating power (Section 2.2), how to trade it against coverage (Section 2.3), how to evolve S_t stably across rounds (Sections 2.4 and 2.5), and how to compare prompts scored on different subsets (Section 2.6).

2.2 Online Discrimination Utility

Static methods (e.g., SESS) select examples from data-side features alone, ignoring which are informative for the *current* prompt candidates, while the useful examples shift as the population evolves.

IRT-based discrimination. We use one-parameter logistic (1PL) IRT: Prompt p has ability θ_p, example i has difficulty b_i, and $\Pr(y_{p,i}{=}1) = \sigma(\theta_p{-}b_i)$, where $\sigma(x) = (1+e^{-x})^{-1}$ is the sigmoid. A two-parameter logistic (2PL) model additionally fits an item-specific slope. Per-observation maximum-a-posteriori (MAP) gradient ascent updates the parameters with step size $\alpha{=}0.05$, L2 prior strength $\kappa{=}0.01$, and values clipped to $[-6, 6]$ (update rules in Section C; identification discussion in Section D). With only 8 candidate prompts per round, each scored on $k{=}20$ items, 1PL is less prone to overfitting; it wins 3/4 tasks against 2PL (Figure 3), with a $+21$ pp gap on the noisiest task.

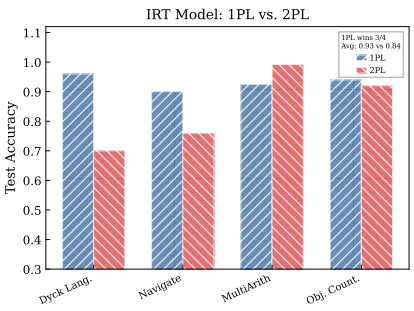


Figure 3. 1PL vs. 2PL: 1PL wins 3/4 tasks (mean 0.93 vs. 0.84).

Let $p_t^{(1)}, p_t^{(2)}$ be the top-two prompts at round t. We compare their predicted binary-response distributions using symmetric Kullback–Leibler (KL) divergence, $D_{\mathrm{KL}}^{\mathrm{sym}}(p,q) := D_{\mathrm{KL}}(p\|q) + D_{\mathrm{KL}}(q\|p)$. Here $\mathrm{Bern}(q)$ denotes a Bernoulli distribution with success probability q. The discrimination utility is

$$u_t(i) = D_{\mathrm{KL}}^{\mathrm{sym}}\Big(\mathrm{Bern}\big(\sigma(\theta_{p_t^{(1)}} - b_i)\big),\ \mathrm{Bern}\big(\sigma(\theta_{p_t^{(2)}} - b_i)\big)\Big). \tag{1}$$

Intuitively, $u_t(i)$ is large when example i is expected to produce *different outcomes* for the two top prompts, peaking when b_i lies between $\theta_{p_t^{(1)}}$ and $\theta_{p_t^{(2)}}$ (the "decision boundary" in ability space), and small when both almost always solve or fail it. It is *prompt-dependent*: An example that separates the current leaders may become uninformative once both solve it. The following proposition formalizes the connection to classical adaptive testing theory:

Proposition 1 (Fisher information connection)**.** *Under the 1PL model, let* $\Delta\theta_t = \theta_{p_t^{(1)}} - \theta_{p_t^{(2)}}$ *and* $\bar{\theta}_t = (\theta_{p_t^{(1)}} + \theta_{p_t^{(2)}})/2$. *Then:*

$$u_t(i) = (\Delta\theta_t)^2 \cdot \mathcal{I}(\bar{\theta}_t;\, b_i) + O\big((\Delta\theta_t)^4\big), \tag{2}$$

where $\mathcal{I}(\theta; b) = \sigma(\theta{-}b)(1{-}\sigma(\theta{-}b))$ *is the Fisher information of item* i *at ability* θ.

Thus, selecting high-u_t items maximizes Fisher information at the ability midpoint—the classical computerized adaptive testing (CAT) criterion [12]—adapted to two *specific* examinees. We write $\tilde{u}_t(i) \in [0,1]$ for the min-max-normalized utility over the pool ($\tilde{u}_t \equiv 0$ if the range is zero). We use the top two prompts because separating the current leaders matters most for the returned prompt; the IRT estimates still use all observed outcomes, and the leading pair changes across rounds (10–15 distinct leaders over $T{=}10$; Section D).

2.3 Coverage and the Per-Round Objective

Discrimination alone may concentrate the subset on a narrow region (e.g., only date-format questions if the top two prompts differ only there), so we add facility-location coverage of the full training pool:

$$\mathrm{Cov}(S) = \frac{1}{|V|}\sum_{j\in V} \max_{i\in S} s(i,j), \qquad s(i,j) = \max\big(0, \cos(\mathbf{v}_i, \mathbf{v}_j)\big), \tag{3}$$

where $\max_{i\in S} s(i,j)$ measures how well S represents pool example j (with $\max_{i\in\emptyset} s(i,j) = 0$), and $\mathbf{v}_i$ is example i's term-frequency–inverse-document-frequency (TF-IDF) vector, chosen for zero external dependencies and structural-form robustness (Section D).

Per-round objective. We therefore optimize

$$G_t(S) \;=\; \underbrace{\sum_{i\in S} \tilde{u}_t(i)}_{\text{discrimination}} \;+\; \lambda_t \underbrace{\mathrm{Cov}(S)}_{\text{coverage}} \qquad \text{subject to } |S| = k, \tag{4}$$

where $\lambda_t \geq 0$ balances discrimination (Section 2.2) and coverage (Equation (3)). Besides λ_t, only $\tilde{u}_t$ changes across rounds; since $G_t(S) \in [0, k{+}\lambda_t]$, one weight schedule transfers across pool sizes.

Temporal stability is enforced separately, not by a third objective term: We warm-start from S_{t-1} with bounded one-for-one swaps (Section 2.4), so k limits the evaluation size while the swap budget B_t limits how much the subset may change. The weight follows a linear schedule,

$$\lambda_t \;=\; \lambda_{\text{start}} + (\lambda_{\text{end}} - \lambda_{\text{start}}) \cdot \pi_t, \tag{5}$$

where $\pi_t \in [0, 1]$ is the normalized optimization progress (fraction of rounds completed), with λ_{start}=0.8 and λ_{end}=0.4. Keeping $\lambda_t \in [0.4, 0.8]$ strictly positive prevents the scheduler from collapsing onto pure discrimination.

2.4 POES: Warm-Start Swap Updates

Re-solving greedily each round can produce entirely different subsets, creating discontinuous evaluation signals (a prompt that scored 0.9 on one subset might score 0.6 on another). We instead start from S_{t-1} and perform at most B_t one-for-one swaps (the *swap budget* at round t), accepting a swap only when

$$G_t(S') - G_t(S) \geq \tau_t, \tag{6}$$

where $\tau_t \geq 0$ is an adaptive threshold and $S' = S \setminus \{i\} \cup \{j\}$ denotes a one-for-one swap neighbor of S (with $i \in S$, $j \notin S$). The incoming side is restricted to a candidate pool $\mathcal{C}_t$ of top-M pool elements ranked by discrimination utility $u_t(\cdot)$, and each step picks the best neighbor, $(i^*, j^*) = \arg\max_{i \in S,\, j \in \mathcal{C}_t \setminus S} G_t(S')$. Swaps are applied sequentially: After each accepted swap, S is updated and the marginal gains are recomputed before selecting the next (i^*, j^*), until either B_t swaps have been accepted or no remaining swap meets the threshold τ_t. This guarantees bounded drift while tracking the evolving landscape.

2.5 Adaptive Controller and Auto Warmup

Auto warmup. Early IRT estimates are noisy; applying discrimination-based scheduling prematurely can mislead. The scheduler uses random subsets for the first $t_w^{\min}$=2 rounds and continues to do so while the *top-k discrimination ratio* ρ_t stays below ρ_{exit}=1.05, but for at most $t_w^{\max}$=3 rounds in total, where

$$\rho_t = \text{mean}_{i \in \text{TopK}_k(u_t)}\, u_t(i) \;/\; \big(\text{mean}_{i \in V}\, u_t(i) + \varepsilon\big), \tag{7}$$

where $\text{TopK}_k(u_t)$ selects the k highest-utility pool items and ε=10^{-6} stabilizes the denominator. Warmup thus lasts 2 or 3 rounds; the empirical mean is 2.9±0.3 (Section 3.6).

Adaptive controller. Once active, the controller coordinates (λ_t, τ_t, B_t) each round: λ_t follows the progress schedule in Equation (5), while (τ_t, B_t) are recomputed from progress $\pi_t \in [0, 1]$, score-history standard deviation σ_t (of per-round aggregate subset scores over the last 5 rounds), and a stagnation indicator. The switch cost τ_t rises with progress (raising the acceptance bar near convergence) and with noise; the swap budget B_t shrinks linearly with progress from $0.25k$ to $0.10k$ (for k=20, this is B_t: 5→2), encoding an *evolve-early-stabilize-late* inductive bias. Exact update rules and all constants (fixed across all 35 tasks, no per-task tuning; robustness in Section E.4) are in Algorithm 1 (Section A).

2.6 Calibrated Scoring

Swaps bound how much S_t changes but do not freeze it, so the optimizer ranks prompts scored on different subsets. Raw accuracies $\hat{a}_t$ then favor prompts that drew easy items, and changes in subset difficulty can distort comparisons across optimization rounds. POES therefore refits the 1PL model on all observed outcomes (MAP, prior κ_c=0.5) and returns $\bar{a}_t(p) = \text{mean}_{i \in R_t}\, \sigma(\hat{\theta}_p - \hat{b}_i)$, the predicted accuracy of p on all examples R_t evaluated so far; the difficulties $\hat{b}_i$ absorb how hard each subset was. Retained prompts are rescored after every step. Calibration adds no LLM calls and leaves G_t and the guarantees below unchanged.

2.7 Theoretical Guarantees

POES admits four formal guarantees (full statements and proofs in Section B). Below, monotonicity means adding items cannot lower G_t, submodularity means diminishing marginal gains, and the symmetric difference $\triangle$ counts items entering or leaving the subset.

- **(P1)** *Fisher connection* (Proposition 1): $u_t(i) = (\Delta\theta_t)^2 \mathcal{I}(\bar{\theta}_t; b_i) + O\big((\Delta\theta_t)^4\big)$, so high-$u_t$ items maximize Fisher information at the top-two ability midpoint.
- **(P2)** *Submodularity* (Proposition 2): G_t is monotone submodular for $\tilde{u}_t, s, \lambda_t \geq 0$, giving a $(1-1/e)$ cold-start greedy guarantee [13].
- **(P3)** *Warm-start guarantees* (Proposition 3): For $m_t \leq B_t$ accepted swaps, (i) $|S_t \triangle S_{t-1}| \leq 2B_t$; (ii) $G_t(S_t) \geq G_t(S_{t-1}) + m_t\tau_t$; (iii) τ_t-local optimality holds when stopping because no swap from $\mathcal{C}_t$ gains at least τ_t.
- **(P4)** *Tracking* (Proposition 4): Let $\mathrm{OPT}_t = \max_{|S|=k} G_t(S)$. If the prior subset achieves ratio $\gamma \in (0, 1]$ of the prior optimum and δ bounds objective drift for every size-k subset, then $G_t(S_t) \geq \gamma\,\mathrm{OPT}_t - 2\delta + m_t\tau_t$. The bound is conditional and complements local-optimality analysis [14, 15]: $\gamma{=}1{-}1/e$ holds when S_{t-1} is a greedy cold-start solution, but swaps only raise G_t above $G_t(S_{t-1})$ and do not by themselves preserve this ratio in later rounds; observed mean drift does not certify δ (Sections B.6 and F).

Scope. These guarantees concern the scheduling objective for a given candidate pool, not end-to-end APO convergence: The quality of the returned prompt also depends on the optimizer's proposals, the worker model, and the match between training pool and test distribution. Diagnostics in Section 3.6 assess observed behavior; they do not verify every theorem assumption.

3 Experiments

We organize our experimental evaluation around the following research questions:

- **Q1:** Does POES outperform existing scheduling methods? (Section 3.2)
- **Q2:** Does the advantage generalize across optimizers and worker LLMs? (Section 3.3)
- **Q3:** How do the scheduling design variants behave? (Section 3.4)
- **Q4:** What is the evaluation cost and budget tradeoff? (Section 3.5)
- **Q5:** How does the internal scheduling mechanism behave? (Section 3.6)

3.1 Experimental Setup

Setup. We evaluate on **35 tasks** across three families—BBH (26 reasoning) [16], BigBench-IPOMP (5) [17], Math (4: GSM8K, GSM-Hard, MultiArith, MATH)—plus MMLU (57 subjects) [18] (full task lists in Section C.3), evaluated across all configurations in Table 1, with $80/20$ splits and $N{\approx}200$. Baselines: Random, SESS [10], IPOMP [11], Anchor (k-medoids), Prediction (model-based); POES is reported with/without auto warmup. Three optimizers (OPRO [6], EvoPrompt-GA/DE [7]) $\times$ two worker LLMs (Llama-3.1-8B-Instruct, Qwen-2.5-7B-Instruct) yield $3{\times}2{=}6$ configurations; each run uses $T{=}10$, $k{=}20$, 2 seeds (12 runs/method). Meta model: GPT-OSS-120B. All scheduler constants are shared across tasks without per-task retuning. The held-out split is excluded from the APO optimization loop, while offline parameter sensitivity is evaluated on representative tasks using held-out accuracy (Sections C and E.4).

3.2 Q1: POES Outperforms Existing Scheduling Methods

Result 1: Best overall performance. Across the 35-task suite plus 57 MMLU subjects, POES attains the highest BBH family average (0.893, +6.0 pp over IPOMP) and the highest MMLU average (0.565, +11.0 pp over IPOMP); Figure 4 shows per-task behavior on 10 representative tasks across BBH, MMLU, and BigBench.

Result 2: Improvements across task families. POES attains the highest Math family average (0.888), with gains on GSM-Hard (+5.3 pp) and BigBench (+5.8 pp). Together with the BBH and MMLU results above, these aggregate results indicate benefits across different task families; individual failure cases are examined in Section H.4.

Result 3: Per-task comparisons. Figures 5a and 5b summarize the 35-task POES–baseline comparison and the relationship between baseline accuracy and observed improvement. Representative task-level exceptions are examined in Section H.4.

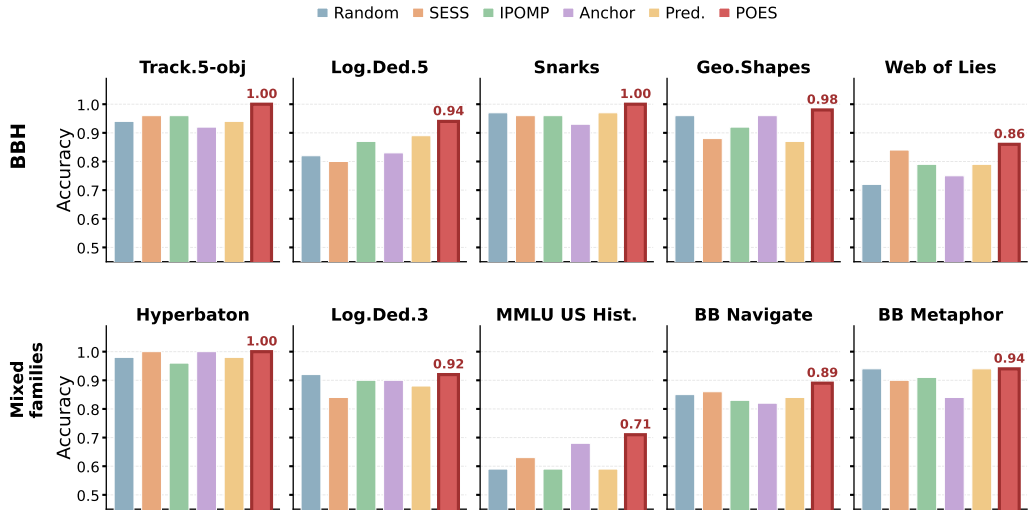
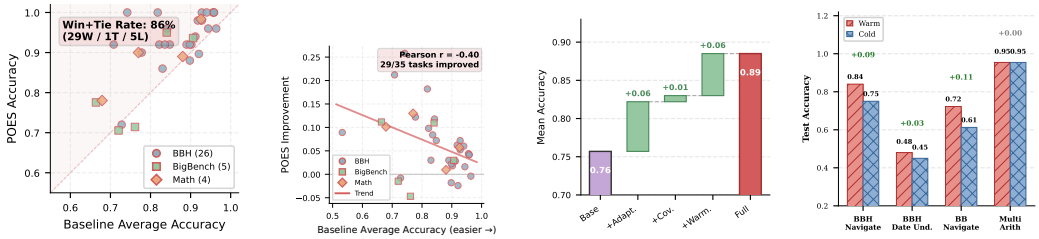


Figure 4. Per-task accuracy on 10 representative tasks across BBH, MMLU, and BigBench families (OPRO × Llama-3.1-8B). **Top row:** BBH reasoning; **bottom row:** BBH, MMLU, and BigBench. Red annotations show the POES score when best or tied-best. Task-name abbreviations are spelled out in Section C.3.



(a) POES vs. baseline. **(b)** Gain vs. difficulty. **(c)** Waterfall across configurations. **(d)** Warm vs. cold start.

Figure 5. Analysis and ablation. (a) Per-task POES accuracy versus the baseline average over 35 tasks (29 wins, 1 tie, and 5 losses). **(b)** Observed improvement versus baseline accuracy ($r = -0.40$, $p = 0.016$); larger gains are associated with lower baseline accuracy. **(c)** Mean accuracies across the scheduling configurations shown in the waterfall. Consecutive differences describe changes between these configurations rather than isolated effects of individual components. **(d)** Warm-start versus cold-start test accuracy on four tasks; warm-start matches or exceeds cold-start on all four, including gains of $+9$ pp and $+11$ pp on BBH and BigBench Navigate.

3.3 Q2: The Advantage Generalizes Across Optimizers and Worker LLMs

Across the 3×2 matrix (Table 1), POES achieves rank-1 mean in all six settings with consistent BBH ($+4.0$–9.9 pp), BigBench ($+5.8$–19.2 pp), and MMLU ($+6.9$–11.0 pp) gains. The advantage is not OPRO-specific: POES retains the highest mean accuracy with both worker models under all three optimizers, although the improvement margin varies across configurations.

3.4 Q3: Behavior of Scheduling Design Variants

We examine the scheduling design variants on five tasks: Dyck Languages, Navigate, GSM-Hard, MATH, and MultiArith. Table 2 reports the held-out scores of two specified configurations on all five tasks; the appendix describes the remaining controls (Section E.1). All entries are absolute test accuracies, averaged over two seeds. These results describe the behavior of the configurations themselves, without attributing their differences to a single component. The configuration sequence in Figure 5c is likewise descriptive rather than a controlled component-by-component decomposition.

Result 4: Scheduling variants exhibit task-dependent behavior. No-Coverage and No-AutoWarmup have five-task means of 0.822 and 0.831, respectively. Their relative task performance varies: No-AutoWarmup is higher on Dyck Languages and GSM-Hard; No-Coverage is higher on

Table 1. Cross-optimizer/model generalization across benchmark families. Panel A reports the full comparison for OPRO × Llama-3.1-8B; Panels B–F report the other five settings. **Bold**: best per column; underline: second best. Δ after POES in the Mean column denotes improvement over the best baseline, in absolute percentage points (avg. **+4.9 pp**).

	Reasoning & Knowledge			Mathematics				
Method	BBH (26)	BigBench (5)	MMLU (57)	GSM8K	GSM-Hard	MATH	MultiArith	Mean
Panel A: OPRO × Llama-3.1-8B-Instruct								
Random	0.809	0.748	0.440	0.853	0.689	0.775	0.933	0.750
SESS	0.817	0.719	0.435	0.728	0.686	0.720	0.850	0.708
IPOMP	0.833	0.758	0.455	**0.907**	0.667	0.767	0.892	0.754
Anchor	0.798	0.709	0.442	0.892	0.727	0.735	0.783	0.727
Prediction	0.815	0.746	0.437	**0.907**	0.481	0.720	0.942	0.721
POES	**0.893**	**0.816**	**0.565**	0.890	**0.780**	**0.900**	**0.983**	**0.832** (+7.8 pp)
Panel B: OPRO × Qwen-2.5-7B-Instruct								
Random	0.756	0.482	0.688	0.915	0.795	0.907	0.917	0.780
SESS	0.753	0.465	0.698	0.935	0.761	0.905	**0.967**	0.783
IPOMP	0.776	0.520	0.701	0.952	**0.826**	0.901	0.958	0.805
Anchor	0.754	0.465	0.710	0.932	0.682	0.913	0.933	0.770
Prediction	0.758	0.467	0.687	0.940	0.807	0.908	0.933	0.786
POES	**0.830**	**0.593**	**0.779**	**0.972**	0.822	**0.915**	0.933	**0.835** (+3.0 pp)
Panel C: EvoPrompt-GA × Llama-3.1-8B-Instruct								
Random	0.675	0.512	0.446	0.930	0.686	0.690	0.925	0.695
SESS	0.685	0.538	0.481	0.927	0.664	0.695	0.925	0.702
IPOMP	0.679	0.508	0.485	0.910	0.694	0.682	0.925	0.698
Anchor	0.674	0.533	0.488	0.910	**0.705**	0.667	0.958	0.705
Prediction	0.671	0.550	0.465	0.897	0.656	0.688	0.917	0.692
POES	**0.746**	**0.612**	**0.571**	**0.950**	0.686	**0.715**	**0.967**	**0.750** (+4.5 pp)
Panel D: EvoPrompt-GA × Qwen-2.5-7B-Instruct								
Random	0.680	0.495	0.555	**0.958**	0.799	0.917	0.967	0.767
SESS	0.607	0.464	0.537	0.940	0.780	0.907	0.958	0.742
IPOMP	0.600	0.467	0.539	0.942	**0.807**	0.912	0.942	0.744
Anchor	0.697	0.425	0.546	0.950	0.754	0.919	**0.983**	0.753
Prediction	0.735	0.496	0.585	0.930	0.799	0.908	0.967	0.774
POES	**0.785**	**0.566**	**0.664**	0.948	0.803	**0.927**	0.967	**0.809** (+3.5 pp)
Panel E: EvoPrompt-DE × Llama-3.1-8B-Instruct								
Random	0.669	0.501	0.499	0.945	0.686	0.708	0.925	0.705
SESS	0.667	0.517	0.470	0.925	**0.694**	**0.713**	0.925	0.702
IPOMP	0.678	0.496	0.459	0.922	0.667	0.677	0.925	0.689
Anchor	0.664	0.521	0.481	0.948	0.675	0.685	0.917	0.699
Prediction	0.671	0.512	0.491	0.943	**0.694**	0.630	0.958	0.700
POES	**0.777**	**0.713**	**0.595**	**0.953**	0.686	0.703	**0.967**	**0.771** (+6.6 pp)
Panel F: EvoPrompt-DE × Qwen-2.5-7B-Instruct								
Random	0.711	0.464	0.552	0.940	0.811	0.903	**0.975**	0.765
SESS	0.732	0.486	0.558	0.952	0.799	0.902	0.933	0.766
IPOMP	0.732	0.558	0.549	0.950	0.811	0.894	0.958	0.779
Anchor	0.726	0.546	0.548	0.952	0.792	0.915	0.950	0.776
Prediction	0.730	0.446	0.567	0.935	0.803	0.898	**0.975**	0.765
POES	**0.772**	**0.633**	**0.666**	**0.965**	**0.818**	**0.922**	0.967	**0.820** (+4.1 pp)

Table 2. Held-out test accuracy for two specified scheduling variants on the five ablation tasks (two-seed means). The final column is the unweighted mean over these tasks.

Configuration	Dyck	Navigate	GSM-Hard	MATH	MultiArith	Mean
No-Coverage ($\lambda_t = 0$)	0.860	0.800	0.690	0.810	0.950	0.822
No-AutoWarmup	0.900	0.780	0.750	0.775	0.950	0.831

Table 3. Budget efficiency. **Top** ((i), method comparison at k=20, single seed): POES achieves rank-1 on all 4 tasks while spending fewer tokens *on average* than every baseline (-3% vs. Random, -1% vs. SESS). **Bottom** ((ii), POES budget sweep, cost deltas are relative to k=30): k=10 trades 64% fewer tokens for a $\sim$5pp accuracy drop; k=30 substantially increases the token budget without further accuracy gain. **Bold**/underline/ green : best/2nd/ours.

	Accuracy					Token		Time	
Method	Dyck	GSM-Hard	MultiArith	Navigate	Mean	K	Δ	s	Δ
(i) Method comparison at k=20									
Random	0.740	0.689	0.933	0.740	0.776	755	ref	2,564	ref
SESS	0.900	0.686	0.975	0.820	0.845	744	−1%	2,340	−9%
IPOMP	0.920	0.686	0.975	0.700	0.820	832	+10%	2,618	+2%
Anchor	0.840	**0.727**	0.783	0.720	0.768	913	+21%	2,479	−3%
Prediction	0.960	0.629	0.942	0.560	0.773	929	+23%	2,683	+5%
POES	**1.000**	**0.727**	**0.983**	**0.860**	**0.893**	734	−3%	2,718	+6%
(ii) POES *budget sweep*									
k=10	0.800	**0.799**	**0.967**	0.650	0.804	478	−64%	2,370	−27%
k=30	**0.980**	0.705	0.942	**0.800**	**0.857**	1,331	ref	3,264	ref

Navigate and MATH; and both score 0.950 on MultiArith. The fixed-parameter No-Adaptive setting has a five-task mean of 0.757. These observations motivate treating adaptive control, coverage, and warmup as separate design axes; the main cross-optimizer results for the configured POES method are reported independently in Table 1.

3.5 Q4: Efficient Evaluation and Warm-Start Validation

Result 5: Superior accuracy–cost tradeoff. At the common evaluation budget k=20, POES achieves the best or joint-best accuracy on all four tasks in Table 3, with lower mean token consumption than every baseline (-3% vs. Random, -1% vs. SESS). Figure 6 addresses the complementary question of how much evaluation is needed, rather than which scheduler is best at a fixed budget.

Evaluation budget. The budget $k = |S_t|$ counts training examples evaluated per candidate, not test examples or optimization rounds. Evaluating all candidates therefore requires $k|P_t|$ prompt–example evaluations per round. The swap budget B_t instead limits changes to S_t; it does not reduce this evaluation count.

Reading the sweep. Figure 6 plots the returned prompt's held-out accuracy against k: Light curves show six tasks and red shows their average. The mean improves from k=5 to 20, but changes little from 20 to 30. Individual curves differ, so k=20 is a practical default for these experiments, not a universal optimum.

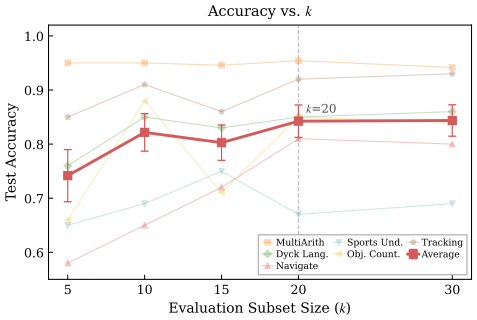


Figure 6. Evaluation-budget sensitivity (six tasks, two seeds). Red: task-average test accuracy; light curves: individual tasks. The dashed line marks the default k=20.

The token measurements in Table 3 complement this accuracy-only plot: Additional evaluations at k=30 increase cost without improving the four-task mean over k=20. Token consumption is not wall-clock time; answer lengths and scheduler overhead also matter (full-suite costs in Table 6). Together, these results support selecting more informative examples before simply increasing the subset size.

Result 6: Warm-start is about signal quality, not cost. A cold start rebuilds the subset, whereas a warm start reuses S_{t-1} and replaces only a bounded number of examples; both still evaluate k examples per candidate. Warm-start matches or exceeds cold-start on 4/4 tasks (Figure 5d; +9/+11 pp

on BBH/BigBench Navigate) at $<5\%$ token difference. Its empirical value is reducing abrupt changes in the evaluation signal, rather than reducing the number of evaluations.

3.6 Q5: Interpretable and Stable Scheduling Mechanism

Result 7: Empirical diagnostics characterize subset updates. (a) the logged per-run average subset shift is 0.287 ± 0.043 (largest 0.395); the 0.50 per-round swap bound of Proposition 3(i) applies to active warm-start updates, not the random-warmup transitions; **(b)** 96.8% of attempted swaps pass the threshold τ_t (Table 10); **(c)** warmup exits in 2.9 ± 0.3 rounds; **(d)** observed mean objective drift $\bar{\delta} \approx 0.02$ is below the typical threshold $\tau_t \approx 0.05$. Here $\bar{\delta}$ averages $|G_t(S_{t-1}) - G_{t-1}(S_{t-1})|$ across rounds (Table 10); it is not the uniform bound δ required by Proposition 4.

4 Related Work

Automatic prompt optimization (APO) [1, 6] has emerged as a lightweight alternative to fine-tuning, alongside broader LLM optimization paradigms [19, 20] and collaborative AI directions [21], with LLMs serving as optimizers that iteratively propose candidate prompts and rank them on a hand-fixed evaluation subset of $\sim$50–500 examples [9, 6, 7]. Methods span textual gradients [5, 9], evolutionary operators [7, 22], instruction editing [23], black-box optimization [24], pipeline compilation [25, 8], best-arm identification [26], and compositional discovery [27]. Most treat the evaluation subset as a fixed implementation detail rather than a learnable component; we focus on the orthogonal axis of evaluation-subset scheduling, integrable with any prompt-proposal method.

Recently, some attempts have been made to improve the evaluation step in APO: SESS [10] selects a fixed subset with a static, prompt-agnostic submodular objective, and IPOMP [11] updates its subset with past-performance heuristics that are not tied to the current candidates and carry no unified per-round or cross-round guarantee, while concurrent work GRACE [28] targets the orthogonal optimizer layer through gated refinement. In contrast, POES conditions each subset on the current prompt competition, selects it through discrimination and coverage, and updates it through bounded swaps with a $(1-1/e)$ cold-start approximation and a conditional tracking bound; to our knowledge, it is the first APO scheduler to make evaluation-subset selection online, prompt-aware, and theoretically controlled across rounds. Several adjacent literatures also inform our design: Item Response Theory in static LLM benchmarking [29, 30, 31, 32] and computerized adaptive testing [12, 33], submodular maximization [13, 34, 35] with switching-cost-aware updates [36, 37, 38, 39, 40] for fixed-arm online problems, and curriculum/active learning [41, 42, 43] or coreset pruning [44, 45, 46, 47, 48] for training-set compute. Online resource scheduling has also been studied in dynamic networks [49, 50].

5 Conclusion

POES advances evaluation-subset scheduling for APO by casting subset selection as online adaptive testing, providing four formal guarantees, and attaining the highest mean accuracy in all 6 optimizer–model configurations ($+3.0$ pp to $+7.8$ pp over the best baseline). Our theoretical guarantees concern the scheduling layer rather than end-to-end APO convergence, and the current formulation uses binary outcomes; extension to continuous-valued evaluation metrics remains future work (Section I). These results open evaluation scheduling as a research axis complementary to prompt proposal—*what* an APO system evaluates on matters as much as *how* it generates prompts.

Acknowledgments and Disclosure of Funding

This work is supported in part by the Guangdong Basic and Applied Basic Research Foundation under Grant No. 2025A1515012968, in part by the Shenzhen Science and Technology Program under Grant No. JCYJ20240813113502004, in part by the National Natural Science Foundation of China under Grant No. 62001412, in part by Shenzhen Stability Science Program 2023, in part by the Guangdong Provincial Key Laboratory of Future Networks of Intelligence (Grant No. 2022B1212010001), and in part by the Shenzhen Key Lab of Crowd Intelligence Empowered Low-Carbon Energy Network (Grant No. ZDSYS20220606100601002). The authors declare no competing interests.

References

[1] Y Zhou, AI Muresanu, Z Han, K Paster, S Pitis, H Chan, and J Ba. Large language models are human-level prompt engineers (arxiv: 2211.01910). arxiv, 2023.

[2] Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. arxiv 2310.11324 [preprint] https://arxiv. org/abs/2310.11324. published october 17, 2023. *Accessed January*, 2024.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[5] Reid Pryzant, Dan Iter, Jerry Li, Yin 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*, pages 7957–7968, 2023.

[6] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2023.

[7] Q Guo, R Wang, J Wang, B Li, K He, X Tan, J Bian, and Y Zheng. Evoprompt: Connecting large language models with evolutionary algorithms for prompt engineering. In *Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria*, pages 7–11, 2024.

[8] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

[9] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

[10] Jinming Nian, Zhiyuan Peng, Hongwei Shang, Dae Hoon Park, and Yi Fang. Submodular evaluation subset selection in automatic prompt optimization. *arXiv preprint arXiv:2601.03493*, 2026.

[11] Ximing Dong, Shaowei Wang, Dayi Lin, and Ahmed Hassan. Model performance-guided evaluation data selection for effective prompt optimization. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2844–2859, 2025.

[12] Frederic M Lord. *Applications of item response theory to practical testing problems*. Routledge, 2012.

[13] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[14] Yuval Filmus and Justin Ward. Monotone submodular maximization over a matroid via non-oblivious local search. *SIAM Journal on Computing*, 43(2):514–542, 2014.

[15] Uriel Feige, Vahab S Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[16] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, 2023.

[17] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*, 2023.

[18] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[19] Zhichao Wang, Dongyang Ma, Xinting Huang, Deng Cai, Tian Lan, Jiahao Xu, Haitao Mi, Xiaoying Tang, and Yan Wang. The end of manual decoding: Towards truly end-to-end language models. *arXiv preprint arXiv:2510.26697*, 2025.

[20] Yongxin Guo, Wenbo Deng, Zhenglin Cheng, and Xiaoying Tang. G^2rpo-a: Guided group relative policy optimization with adaptive guidance, 2025.

[21] Fei Wu, Tao Shen, Thomas Bäck, Jingyuan Chen, Gang Huang, Yaochu Jin, Kun Kuang, Mengze Li, Cewu Lu, Jiaxu Miao, et al. Knowledge-empowered, collaborative, and co-evolving ai models: The post-llm roadmap. *Engineering*, 44:87–100, 2025.

[22] Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.

[23] Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. Grips: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3845–3864, 2023.

[24] Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. Instructzero: Efficient instruction optimization for black-box large language models. *arXiv preprint arXiv:2306.03082*, 2023.

[25] Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*, 2023.

[26] Chengshuai Shi, Kun Yang, Zihan Chen, Jundong Li, Jing Yang, and Cong Shen. Efficient prompt optimization through the lens of best arm identification. *Advances in Neural Information Processing Systems*, 37:99646–99685, 2024.

[27] Haoyue Liu, Zhichao Wang, Yongxin Guo, Haoran Shou, and Xiaoying Tang. Adaptive prompt structure factorization: A framework for self-discovering and optimizing compositional prompt programs, 2026.

[28] Wenhang Shi, Yiren Chen, Shuqing Bian, Xinyi Zhang, Kai Tang, Pengfei Hu, Zhe Zhao, Wei Lu, and Xiaoyong Du. No loss, no gain: Gated refinement and adaptive compression for prompt optimization, 2025.

[29] Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: evaluating llms with fewer examples. *arXiv preprint arXiv:2402.14992*, 2024.

[30] Alex Kipnis, Konstantinos Voudouris, Luca M Schulze Buschoff, and Eric Schulz. metabench–a sparse benchmark of reasoning and knowledge in large language models. *arXiv preprint arXiv:2407.12844*, 2024.

[31] Fernando Martínez-Plumed, Ricardo BC Prudêncio, Adolfo Martínez-Usó, and José Hernández-Orallo. Item response theory in ai: Analysing machine learning classifiers at the instance level. *Artificial intelligence*, 271:18–42, 2019.

[32] Yan Zhuang, Qi Liu, Zachary Pardos, Patrick C Kyllonen, Jiyun Zu, Zhenya Huang, Shijin Wang, and Enhong Chen. Position: Ai evaluation should learn from how we test humans. In *Forty-second International Conference on Machine Learning Position Paper Track*, 2025.

[33] Wim J Van der Linden, Cees AW Glas, et al. *Computerized adaptive testing: Theory and practice*, volume 13. Springer, 2000.

[34] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3(71-104):3, 2014.

[35] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[36] Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. *Advances in Neural Information Processing Systems*, 21, 2008.

[37] Daniel Golovin, Andreas Krause, and Matthew Streeter. Online submodular maximization under a matroid constraint with application to learning assignments. *arXiv preprint arXiv:1407.1082*, 2014.

[38] Ofer Dekel, Jian Ding, Tomer Koren, and Yuval Peres. Bandits with switching costs: T 2/3 regret. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 459–467, 2014.

[39] Nikhil Bansal, Anupam Gupta, Ravishankar Krishnaswamy, Kirk Pruhs, Kevin Schewior, and Cliff Stein. A 2-competitive algorithm for online convex optimization with switching costs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*, pages 96–109. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2015.

[40] Emilie Kaufmann and Shivaram Kalyanakrishnan. Information complexity in bandit subset selection. In *Conference on Learning Theory*, pages 228–251. PMLR, 2013.

[41] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA, 2009. Association for Computing Machinery.

[42] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[43] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*, 2019.

[44] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pages 6950–6960. PMLR, 2020.

[45] Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. Glister: Generalization based data subset selection for efficient and robust learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 8110–8118, 2021.

[46] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*, 2019.

[47] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34:20596–20607, 2021.

[48] Adyasha Maharana, Prateek Yadav, and Mohit Bansal. D2 pruning: Message passing for balancing diversity and difficulty in data pruning. *arXiv preprint arXiv:2310.07931*, 2023.

[49] Jiaorui Huang, Min Cao, Chungang Yang, Zhu Han, and Tong Li. Learning-based matching game for task scheduling and resource collaboration in intent-driven task-oriented networks. *Engineering*, 2025.

[50] Xiaohan Qin, Tianqi Zhang, Kai Yu, Xin Zhang, Haibo Zhou, Weihua Zhuang, and Xuemin Shen. Dynamic time-difference qos guarantee in satellite–terrestrial integrated networks: An online learning-based resource scheduling scheme. *Engineering*, 2025.

Contents of Appendix: Section A Algorithm Pseudocode · Section B Complete Proofs · Section C Experiment Settings (incl. Dataset Details) · Section D Design Choice Justifications · Section E Additional Experiment Results (incl. Larger-Pool Scaling and Parameter Sensitivity) · Section F Scheduler Mechanism Diagnostics · Section G Comparison with Oracle Full-Set Evaluation · Section H Qualitative Case Studies · Section I Limitations and Societal Impacts

A Algorithm Pseudocode

GREEDYMAX adds highest-marginal-gain items to an empty set until size k; MINMAXNORM is defined in Section 2.2.

Algorithm 1 POES: Online Evaluation Scheduling (per-round procedure)

Require: Training pool V, budget k, similarity $\{s(i,j)\}$, candidate-pool size M, warmup bounds $t_w^{\min}, t_w^{\max}$, exit threshold ρ_{exit}

Init IRT parameters $\theta_p \leftarrow 0$, $b_i \leftarrow 0$; warmup $\leftarrow$ TRUE; $S_0 \leftarrow \emptyset$
for round $t = 1, \ldots, T$ **do**
  Obtain prompt candidates for round t {IRT (θ_p, b_i) already streaming-updated by line 22 of round $t-1$ via per-observation MAP ascent (Section 2.2); no batch update here}
  Compute $u_t(i)$ for all $i \in V$ via Equation (1); $\tilde{u}_t \leftarrow \text{MINMAXNORM}(u_t)$
  $\rho_t \leftarrow \big[\text{mean}_{i \in \text{TopK}_k(u_t)} u_t(i)\big] \big/ \big[\text{mean}_{i \in V}\, u_t(i) + \varepsilon\big]$ {Top-k discrimination ratio, Equation (7)}
  if warmup **and** $\big(t \le t_w^{\min}$ **or** $(t \le t_w^{\max}$ **and** $\rho_t < \rho_{\text{exit}})\big)$ **then**
    $S_t \leftarrow$ random subset of size k {Auto warmup: forced if $t \le t_w^{\min}$, else gated by ρ_t until $t_w^{\max}$}
  else
    warmup $\leftarrow$ FALSE
    Compute (λ_t, τ_t, B_t) from $(\pi_t, \sigma_t, \text{stagnation}_t)$ via Equations (5), (9) and (10)
    if cold start $(S_{t-1} = \emptyset)$ **then**
      $S_t \leftarrow \text{GREEDYMAX}(G_t, k)$ {$(1-1/e)$ guarantee}
    else
      $S_t \leftarrow S_{t-1}$; $\mathcal{C}_t \leftarrow$ top-M elements of V by $\tilde{u}_t$ {Warm-start + candidate pool}
      for $\ell = 1, \ldots, B_t$ **do**
        $(i^*, j^*) \leftarrow \arg\max_{i \in S_t,\, j \in \mathcal{C}_t \setminus S_t} G_t(S_t \setminus \{i\} \cup \{j\}) - G_t(S_t)$
        $\Delta^* \leftarrow G_t(S_t \setminus \{i^*\} \cup \{j^*\}) - G_t(S_t)$
        if $\Delta^* \ge \tau_t$ **then** $S_t \leftarrow S_t \setminus \{i^*\} \cup \{j^*\}$ **else break**
      end for
    end if
  end if
  Evaluate candidate prompts on S_t; for each new outcome $(p, i, y_{p,i})$, update the scheduler's IRT parameters via streaming MAP (Section 2.2)
  Refit the scoring IRT model on all observed outcomes with prior κ_c=0.5; compute calibrated accuracies $\bar{a}_t(p)$ for the evaluated and retained prompts, and return the calibrated scores to the optimizer (Section 2.6)
end for

Adaptive schedule equations (referenced by Alg. 1). The controller computes (λ_t, τ_t, B_t) from progress $\pi_t \in [0,1]$, score-history standard deviation σ_t, and a regime indicator $\zeta_t \in \{\text{stag}, \text{neutral}, \text{imp}\}$. Here $\text{clip}(x; a, b) = \min\{b, \max\{a, x\}\}$ restricts x to $[a, b]$, and round rounds to the nearest integer. Define the per-round swap-budget ratio

$$r_t^B \;=\; r_{\text{start}}^B + (r_{\text{end}}^B - r_{\text{start}}^B)\, \pi_t. \tag{8}$$

The schedule equations are then

$$\tau_t = \text{clip}\big(\tau_0 \cdot (1 + \gamma_p\, \pi_t + \gamma_\sigma\, \sigma_t) \cdot \beta_{\zeta_t}^{\tau};\;\; \tau_{\min},\; \tau_{\max}\big), \tag{9}$$

$$B_t = \text{clip}\big(\text{round}(r_t^B \cdot k) + \beta_{\zeta_t}^{B};\;\; B_{\min},\; B_{\max}\big), \tag{10}$$

with base τ_0=0.05, clip $[0.01, 0.30]$, progress/noise weights $(\gamma_p, \gamma_\sigma)$=$(0.5, 0.5)$, regime multiplier $\beta^{\tau}_{\zeta_t} \in \{0.8, 1.0, 1.1\}$; swap-budget ratios r^B_{start}=0.25, r^B_{end}=0.10, clip $[B_{\min}, B_{\max}]$=$[1, 5]$, regime offset $\beta^B_{\zeta_t} \in \{+1, 0, -1\}$. Concretely, at k=20 this gives $\text{round}(r^B_t \cdot k) = 5$ at π_t=0 and 2 at π_t=1, recovering the B_t=5 $\to$ 2 trajectory referenced in Section 2.4. Coverage schedule λ_t (Equation (5)) is given in Section 2.3. Stagnation is detected when the best evaluation-subset score has not improved by ϵ_{stag}=0.01 for two consecutive rounds.

B Complete Proofs

B.1 Proof of Submodularity

Proposition 2 (Submodularity)**.** *The composite objective $G_t(S)$ in Equation (4) is monotone submodular for any $\tilde{u}_t \geq 0$, $s \geq 0$, and $\lambda_t \geq 0$. Min-max normalization of u_t preserves nonnegativity and only rescales the modular term, and the per-element factor $1/|V|$ is a positive constant; neither affects submodularity.*[2]

Proof. We show that $G_t(S) = \tilde{U}_t(S) + \lambda_t \text{Cov}(S)$ is monotone and submodular in S, where $\tilde{U}_t(S) = \sum_{i \in S} \tilde{u}_t(i)$ and $\text{Cov}(S) = \frac{1}{|V|} \sum_{j \in V} \max_{i \in S} s(i, j)$.

Step 1: $\tilde{U}_t(S)$ is modular. The marginal gain of adding element e to any set S is $\tilde{U}_t(S \cup \{e\}) - \tilde{U}_t(S) = \tilde{u}_t(e)$, which is independent of S. Therefore $\tilde{U}_t$ is modular. A modular function is trivially both monotone (since $\tilde{u}_t(e) \geq 0$) and submodular.

Step 2: $\text{Cov}(S)$ is monotone submodular. The facility-location objective is

$$\text{Cov}(S) = \frac{1}{|V|} \sum_{j \in V} \max_{i \in S} s(i, j) = \frac{1}{|V|} \sum_{j \in V} f_j(S),$$

where $f_j(S) = \max_{i \in S} s(i, j)$. Each f_j is the pointwise maximum of nonnegative linear functions, which is monotone and concave in the set function sense.

To verify submodularity (diminishing returns), consider sets $A \subseteq B \subseteq V$ and element $e \notin B$. For any j:

$$f_j(A \cup \{e\}) - f_j(A) = \max(0, s(e, j) - \max_{i \in A} s(i, j)),$$
$$f_j(B \cup \{e\}) - f_j(B) = \max(0, s(e, j) - \max_{i \in B} s(i, j)).$$

Since $A \subseteq B$, we have $\max_{i \in A} s(i, j) \leq \max_{i \in B} s(i, j)$, hence $f_j(A \cup \{e\}) - f_j(A) \geq f_j(B \cup \{e\}) - f_j(B)$. Summing over all $j \in V$ and dividing by the positive constant $|V|$ gives $\text{Cov}(A \cup \{e\}) - \text{Cov}(A) \geq \text{Cov}(B \cup \{e\}) - \text{Cov}(B)$.

Monotonicity follows because adding an element e to S can only increase each $\max_{i \in S} s(i, j)$ term (or leave it unchanged), and the per-element factor $1/|V|$ is positive.

Step 3: Nonnegative combination preserves the properties. Since $\lambda_t \geq 0$ and both $\tilde{U}_t$ and Cov are monotone submodular, their nonnegative linear combination $G_t(S) = \tilde{U}_t(S) + \lambda_t \text{Cov}(S)$ is also monotone submodular. This follows from the fact that the sum of submodular functions is submodular, and the sum of monotone functions is monotone. □

B.2 Proof of Bounded Drift (Warm-start, part i)

Proposition 3 (Warm-start guarantees)**.** *For m_t accepted swaps at round t, the warm-start procedure ensures: **(i)** bounded drift: $|S_t \triangle S_{t-1}| \leq 2B_t$; **(ii)** monotone improvement: $G_t(S_t) \geq G_t(S_{t-1}) + m_t\tau_t$; **(iii)** local optimality within $\mathcal{C}_t$: If no swap from the candidate pool exceeds threshold τ_t, then S_t is a τ_t-local optimum w.r.t. the 1-swap neighborhood restricted to $\mathcal{C}_t$. Parts (i)–(iii) are proved in Sections B.2 to B.4.*

[2]When $\max_i u_t(i) = \min_i u_t(i)$ (degenerate case of identical discrimination), we set $\tilde{u}_t \equiv 0$ by convention, in which case G_t reduces to the pure facility-location term and the proposition still holds. In practice this case does not occur after warmup.

Proof of (i). Starting from $S_t^{(0)} = S_{t-1}$, each accepted swap removes one element $i^* \in S_t^{(\ell)}$ and adds one element $j^* \notin S_t^{(\ell)}$. After $m_t \le B_t$ accepted swaps:

$$|S_t \setminus S_{t-1}| \le m_t, \quad |S_{t-1} \setminus S_t| \le m_t.$$

Therefore $|S_t \triangle S_{t-1}| = |S_t \setminus S_{t-1}| + |S_{t-1} \setminus S_t| \le 2m_t \le 2B_t$.

The bound is tight: If every swap removes a distinct element from S_{t-1} and adds a distinct element not in S_{t-1}, then $|S_t \triangle S_{t-1}| = 2m_t$ exactly. □

B.3 Proof of Monotone Improvement

Proof. By the acceptance condition (6), each accepted swap ℓ satisfies $G_t(S_t^{(\ell+1)}) - G_t(S_t^{(\ell)}) \ge \tau_t$ for $\ell = 0, \dots, m_t - 1$. Summing (telescoping):

$$G_t(S_t) - G_t(S_{t-1}) = \sum_{\ell=0}^{m_t-1} \left[G_t(S_t^{(\ell+1)}) - G_t(S_t^{(\ell)}) \right] \ge m_t \tau_t. \qquad \square$$

B.4 Proof of τ_t-Local Optimality

Proof. If the swap procedure terminates before the B_t-th iteration (i.e., $m_t < B_t$), the loop exited via the **else break** branch (Alg. 1, line 18), so the best candidate swap (i^*, j^*) within the candidate pool $\mathcal{C}_t$ failed to meet the threshold: $G_t(S_t - \{i^*\} + \{j^*\}) - G_t(S_t) < \tau_t$. Since the procedure selects the best available swap within $\mathcal{C}_t$ at each step, every other candidate swap from the pool also has gain strictly below τ_t:

$$\forall\, i \in S_t,\ j \in \mathcal{C}_t \setminus S_t: \quad G_t(S_t - \{i\} + \{j\}) - G_t(S_t) < \tau_t.$$

This is a τ_t-approximate local optimum with respect to the 1-swap neighborhood *restricted to the candidate pool* $\mathcal{C}_t$ (the top-M elements of V by discrimination utility); when $\tau_t = 0$ and $\mathcal{C}_t = V$, this reduces to exact 1-swap local optimality on the full pool. □

B.5 Proof of Fisher Information Connection

Proof. Let $p_k = \sigma(\theta_{p_t^{(k)}} - b_i)$ for $k = 1, 2$. The symmetric KL divergence between Bernoulli distributions decomposes as $u_t(i) = D_{\mathrm{KL}}(\mathrm{Bern}(p_1) \,\|\, \mathrm{Bern}(p_2)) + D_{\mathrm{KL}}(\mathrm{Bern}(p_2) \,\|\, \mathrm{Bern}(p_1))$.

Step 1 (endpoint expansion). The Bernoulli with success probability $\sigma(\theta - b_i)$ is an exponential family in the natural parameter $\eta = \theta - b_i$, with Fisher information $\mathcal{I}(\theta; b_i) = \sigma(\theta - b_i)(1 - \sigma(\theta - b_i))$. The standard quadratic approximation of KL divergence in exponential families anchors at either endpoint: for $k = 1, 2$,

$$D_{\mathrm{KL}}\big(\mathrm{Bern}(p_k) \,\|\, \mathrm{Bern}(p_{3-k})\big) \;=\; \tfrac{1}{2}(\Delta\theta_t)^2\, \mathcal{I}\big(\theta_{p_t^{(k)}}; b_i\big) + O\big((\Delta\theta_t)^3\big).$$

Step 2 (re-anchor at the midpoint). Taylor-expanding $\mathcal{I}(\theta; b_i)$ around $\bar{\theta}_t = (\theta_{p_t^{(1)}} + \theta_{p_t^{(2)}})/2$ and substituting $\theta_{p_t^{(k)}} = \bar{\theta}_t \pm \frac{1}{2}\Delta\theta_t$ gives

$$\mathcal{I}\big(\theta_{p_t^{(k)}}; b_i\big) \;=\; \mathcal{I}(\bar{\theta}_t; b_i) + (\pm\tfrac{1}{2}\Delta\theta_t)\, \partial_\theta \mathcal{I}(\bar{\theta}_t; b_i) + O\big((\Delta\theta_t)^2\big).$$

Multiplying by the $\frac{1}{2}(\Delta\theta_t)^2$ prefactor, the $O(\Delta\theta_t)$ correction enters at order $O\big((\Delta\theta_t)^3\big)$, which is absorbed into the existing remainder:

$$D_{\mathrm{KL}}\big(\mathrm{Bern}(p_k) \,\|\, \mathrm{Bern}(p_{3-k})\big) \;=\; \tfrac{1}{2}(\Delta\theta_t)^2\, \mathcal{I}(\bar{\theta}_t; b_i) + O\big((\Delta\theta_t)^3\big), \qquad k = 1, 2.$$

Step 3 (cancellation of odd orders). $D_{\mathrm{KL}}^{\mathrm{sym}}$ is invariant under the swap $p_1 \leftrightarrow p_2$ (equivalent to $\Delta\theta_t \to -\Delta\theta_t$), so as a function of $\Delta\theta_t$ around zero it is even and its Taylor expansion contains only even-order terms. The $O((\Delta\theta_t)^3)$ contributions from the two individual KLs therefore cancel exactly in the sum, yielding

$$u_t(i) \;=\; (\Delta\theta_t)^2\, \mathcal{I}(\bar{\theta}_t; b_i) + O\big((\Delta\theta_t)^4\big). \qquad \square$$

B.6 Proof of Warm-Start Tracking

Proposition 4 (Warm-start tracking)**.** *Let* $\mathrm{OPT}_t := \max_{|S|=k} G_t(S)$ *denote the round-*t *global optimum. Assume* ***(A1)*** *prior-round approximation* $G_{t-1}(S_{t-1}) \geq \gamma\,\mathrm{OPT}_{t-1}$ *for some* $\gamma \in (0, 1]$*, and* ***(A2)*** *bounded inter-round drift* $|G_t(S) - G_{t-1}(S)| \leq \delta$ *for all* $|S| = k$*. Then after* m_t *accepted swaps:*

$$G_t(S_t) \;\geq\; \gamma\,\mathrm{OPT}_t - 2\delta + m_t\tau_t.$$

Proof. By the bounded-drift assumption applied to S_{t-1}:

$$G_t(S_{t-1}) \geq G_{t-1}(S_{t-1}) - \delta.$$

By the approximation guarantee from round $t-1$: $G_{t-1}(S_{t-1}) \geq \gamma \cdot \mathrm{OPT}_{t-1}$.

To relate OPT_{t-1} to OPT_t, let $S_t^* = \arg\max_{|S|=k} G_t(S)$. Applying bounded drift to S_t^*:

$$\mathrm{OPT}_{t-1} \geq G_{t-1}(S_t^*) \geq G_t(S_t^*) - \delta = \mathrm{OPT}_t - \delta.$$

Combining the three inequalities:

$$G_t(S_{t-1}) \geq \gamma(\mathrm{OPT}_t - \delta) - \delta = \gamma \cdot \mathrm{OPT}_t - (\gamma + 1)\delta \geq \gamma \cdot \mathrm{OPT}_t - 2\delta,$$

where the last step uses $\gamma \leq 1$. By the monotone improvement guarantee (Proposition 3 (ii)), m_t accepted swaps each contribute at least τ_t:

$$G_t(S_t) \geq G_t(S_{t-1}) + m_t\tau_t \geq \gamma \cdot \mathrm{OPT}_t - 2\delta + m_t\tau_t. \qquad \square$$

Remark (sufficient condition for small δ). Proposition 4 relies on a uniform inter-round drift bound δ (assumption A2). We contrast a worst-case scale estimate with observed mean drift, rather than use the mean to certify A2. The round-dependence of G_t enters through (i) the discrimination utilities $\{\tilde{u}_t(i)\}$ (which depend on IRT abilities $\{\theta_p\}$ and difficulties $\{b_i\}$) and (ii) the scheduled weight λ_t (the facility-location structure itself is time-invariant). Each online MAP step changes a coordinate by $\alpha\,|r_{p,i} - \kappa\theta_p|$ or $\alpha\,|r_{p,i} + \kappa b_i|$ (Section C); since $|r_{p,i}| \leq 1$ and the clip keeps $|\theta_p|, |b_i| \leq 6$, each step moves a coordinate by at most $\bar{\alpha} := \alpha(1 + 6\kappa) = 0.053$. Within one round, a prompt p receives at most k observations (it is evaluated on the k items of S_t) and an item i receives at most $|P_t|$ observations (one per candidate prompt in round t). Thus

$$\|\theta^{(t)} - \theta^{(t-1)}\|_\infty \leq \bar{\alpha}\,k, \qquad \|b^{(t)} - b^{(t-1)}\|_\infty \leq \bar{\alpha}\,|P_t|,$$

and the λ_t schedule drifts by at most $(\lambda_{\text{start}} - \lambda_{\text{end}})/T$ per round. Let L_θ, L_b, L_λ denote partial Lipschitz constants of $G_t(\cdot)$ with respect to θ, b, and λ_t, measured in $\|\cdot\|_\infty$ on the compact domain $[-6, 6]^{|P|+|V|} \times [\lambda_{\min}, \lambda_{\max}]$. These constants are finite only locally: G_t is a smooth composition of sigmoids and bounded similarities while the identity of the top-two pair is fixed and the range used for min-max normalization stays bounded away from zero, but a change of the leading pair can change $\tilde{u}_t$ discontinuously. The estimate below is therefore a scale heuristic, not a uniform bound. Note that G_t depends on θ only through the two coordinates $(\theta_{p_t^{(1)}}, \theta_{p_t^{(2)}})$ that enter u_t via Equation (1), so L_θ acts on a 2-coordinate slice of the θ-vector with bounded partial derivatives; the b_i coordinates are likewise updated only for $i \in S_t$ ($\leq k$ entries), so $\|b^{(t)} - b^{(t-1)}\|_\infty \leq \bar{\alpha}|P_t|$ is realized on at most k coordinates. Combining the per-coordinate bounds additively,

$$\delta_t \;\leq\; L_\theta\,\bar{\alpha}\,k \;+\; L_b\,\bar{\alpha}\,|P_t| \;+\; L_\lambda \cdot (\lambda_{\text{start}} - \lambda_{\text{end}})/T.$$

With $\bar{\alpha}$=0.053 (α=0.05, κ=0.01), k=20, $|P_t|\leq$8, $\lambda_{\text{start}} - \lambda_{\text{end}}$=0.4, T=10, and $L_\theta, L_b, L_\lambda \approx 1$, the worst-case sum evaluates to $0.053 \cdot 20 + 0.053 \cdot 8 + 0.4/10 = 1.06 + 0.424 + 0.04 = 1.524$. This bound assumes that all IRT coordinates move maximally in the same direction. *The observed mean drift is much smaller:* we measure $\bar{\delta} \approx 0.02$ across the main suite (Section F), below the typical threshold $\tau_t \approx 0.05$. This averages drift at the selected subsets; it does not establish a uniform δ or make the worst-case tracking bound non-vacuous by itself. Two structural reasons plausibly account for the small observed drift, though we do not formalize them here. **(i)** The coverage term does not depend on the IRT parameters at all: $\mathrm{Cov}(S)$ is computed from fixed TF-IDF similarities $s(i, j)$, so updates of (θ, b) change G_t only through the discrimination term, and the coverage term changes across rounds only through λ_t, by at most $(\lambda_{\text{start}} - \lambda_{\text{end}})/T \cdot \mathrm{Cov}(S) \leq 0.04$. **(ii)** For the discrimination term, candidate prompts in P_t shift their abilities in inconsistent directions across rounds, so contributions to $\sum_i \tilde{u}_t(i)$ partially cancel rather than summing constructively. A certified, tighter uniform bound remains future work; the reported mean is a stability diagnostic, not verification of assumption A2.

Remark (assumption A1 in the warmup-to-active transition). Proposition 4 relies on assumption A1, $G_{t-1}(S_{t-1}) \geq \gamma\,\mathrm{OPT}_{t-1}$. The value $\gamma{=}1{-}1/e$ is guaranteed only when S_{t-1} is the output of a greedy cold-start round on G_{t-1} (Proposition 2). Warm-start swaps do not propagate this ratio automatically: Proposition 3(ii) guarantees $G_t(S_t) \geq G_t(S_{t-1}) + m_t\tau_t$, an improvement relative to the starting subset, not a bound relative to OPT_t. Writing $\gamma_t := G_t(S_t)/\mathrm{OPT}_t$ and applying Proposition 4 with $\gamma = \gamma_{t-1}$ gives

$$\gamma_t \;\geq\; \gamma_{t-1} - \frac{2\delta - m_t\tau_t}{\mathrm{OPT}_t},$$

so the ratio is maintained in rounds where the accepted swaps offset the drift ($m_t\tau_t \geq 2\delta$) and may otherwise decrease by up to $2\delta/\mathrm{OPT}_t$ per round. Moreover, under the default schedule the warmup phase emits random subsets (Alg. 1, line 7), so the first active round warm-starts from a random S_{t-1} rather than a greedy one, and the cold-start branch (line 12) is not reached. A1 then holds only with the ratio γ_{rand} attained by that random subset, which we do not certify. The $(1{-}1/e)$ factor in Proposition 4 should therefore be read as the guarantee for runs that start from a greedy cold-start round (e.g., with warmup disabled); for default runs, Proposition 4 bounds each round relative to the previous round's achieved ratio. The empirical results in Section 3 are measured directly and do not rely on this constant.

C Experiment Settings

C.1 Hyperparameter Configuration

Table 4 lists the hyperparameters. All values are fixed across all tasks without per-task tuning.

Table 4. Hyperparameter settings for POES. All values fixed across all tasks.

Module	Parameter	Default
APO loop	rounds T / candidates per round / eval budget k	10 / 8 / 20
Data split	train pool N / test size	$\sim$200 / $\sim$50
Coverage	λ_t schedule (linear; Eq. 5)	$0.8 \to 0.4$
Swap budget	ratio r_t^B (linear); B_t clip $[1, 5]$, $+1$ on stag. (Eq. 10)	$0.25 \to 0.10$
Switch cost	base τ_0 (range $[0.01, 0.30]$)	0.05
Swap pool	candidate pool size M	128
Warmup	exit ratio ρ_{exit} / max rounds $t_w^{\max}$	1.05 / 3
IRT (1PL)	step size α / L2 prior κ / clip	0.05 / 0.01 / $[-6, 6]$
Similarity	TF-IDF cosine	–

The adaptive schedule constants are $(\gamma_p, \gamma_\sigma){=}(0.5, 0.5)$, $(\beta^\tau_{\text{stag}}, \beta^\tau_{\text{imp}}){=}(0.8, 1.1)$, stagnation tolerance $\epsilon_{\text{stag}}{=}0.01$, coverage range $\lambda_t \in [0.4, 0.8]$ (defensive clip $[\lambda_{\min}, \lambda_{\max}]{=}[0.4, 0.8]$ implemented but never active under the default schedule; see Section 2.3), min warmup rounds $t_w^{\min}{=}2$, and IRT initialization $\theta_p^{(0)}{=}b_i^{(0)}{=}0$. These are held fixed across all 35 tasks with no per-task tuning.

Default selection rationale. The defaults reflect the parameter sweeps in Section E.4, which are scored on held-out accuracy of the swept tasks; the defaults are therefore informed by those tasks' held-out results. Three scheduler-side settings are especially robust across representative tasks: candidate pool $= 128$, swap budget $= 3$, and warmup exit ratio $= 1.05$. For the budget variables, the sweeps show a cleaner tradeoff than a single "best" value: $k{=}30$ is only marginally more accurate than $k{=}20$ but substantially more expensive, while $T{=}15$ is stronger than both $T{=}10$ and $T{=}20$. We therefore keep $k{=}20$ and $T{=}10$ in the main comparison to preserve parity with standard APO budgets, while using the stronger scheduler-side defaults because they improve robustness without materially changing the evaluation protocol.

C.2 Implementation Details

Model serving. We use Llama-3.1-8B-Instruct and Qwen-2.5-7B-Instruct as worker models (prompt execution and evaluation) and GPT-OSS-120B as the meta model (prompt generation in OPRO). All

models are served via vLLM with OpenAI-compatible API endpoints on two dedicated NVIDIA A100 80GB GPUs to prevent resource contention.

Hardware configuration. All experiments run on a single machine with two NVIDIA A100 80GB GPUs. The full experiment suite (35 main + 57 MMLU tasks, 7 methods, multiple seeds) requires approximately 1,000 GPU-hours of compute.

Scoring protocol. Prompts are evaluated using deterministic, task-specific answer extraction followed by normalized exact-match scoring. Multiple-choice tasks (BBH, MMLU, BigBench) extract the predicted option label and compare it to the gold label after normalization (case-folding, whitespace stripping, and answer-letter parsing); numeric/math tasks (GSM8K, GSM-Hard, MultiArith, MATH, BBH numeric) extract a normalized numeric value and accept matches within a 10^{-5} absolute tolerance. No LLM is used in the loop to grade answers, ensuring that the binary outcome $y_{p,i} \in \{0, 1\}$ that drives the IRT update and scheduler is fully reproducible.

IRT parameter estimation. The IRT parameters (θ_p, b_i) are estimated by online *maximum a posteriori* (MAP) gradient ascent under an isotropic Gaussian prior $\mathcal{N}(0, 1/\kappa)$. Each evaluation observation $(p, i, y_{p,i})$ triggers a single per-coordinate gradient step. Letting $\hat{y}_{p,i} = \sigma(\theta_p - b_i)$ denote the current predicted success probability and $r_{p,i} = y_{p,i} - \hat{y}_{p,i}$ the residual, the update rules are

$$\begin{aligned} \theta_p &\leftarrow \mathrm{clip}_{[-6,6]}\big(\theta_p + \alpha\,[\,r_{p,i} - \kappa\,\theta_p\,]\big), \\ b_i &\leftarrow \mathrm{clip}_{[-6,6]}\big(b_i - \alpha\,[\,r_{p,i} + \kappa\,b_i\,]\big), \end{aligned}$$

with step sizes $\alpha_\theta{=}\alpha_b{=}\alpha{=}0.05$, L2 strengths $\kappa_\theta{=}\kappa_b{=}\kappa{=}0.01$, and a hard clip to $[-6, 6]$ applied after each step. Both θ_p and b_i are initialized to 0 on first appearance. The signs reflect the 1PL likelihood $\partial_\theta \log \Pr(y_{p,i}) = r_{p,i}$ and $\partial_b \log \Pr(y_{p,i}) = -r_{p,i}$, with the prior contributing the standard $-\kappa\theta_p$ and $-\kappa b_i$ regularization terms. This streaming update is equivalent to an online MAP estimator and only involves lightweight parameter updates over the observed prompt–example outcomes. We deliberately avoid batch L-BFGS because the per-round outcome matrix is sparse (each round adds $|P_t|\,k{=}160$ outcomes at the defaults, out of $|P_t|\,N{\approx}1{,}600$ possible prompt–example pairs); a streaming MAP update with explicit L2 prior gives equivalent stability with no bookkeeping overhead.

Reproducibility. Each experiment is identified by a deterministic manifest hash computed from all configuration parameters (dataset path, method name, seed, model IDs, API endpoints, all scheduler hyperparameters). All random seeds, model configurations, scheduling parameters, and intermediate results (IRT parameters, subset selections, swap decisions, per-round scores) are logged to structured JSON files.

Release plan. We will publicly release: (i) code for POES and all baseline schedulers (Random, SESS, IPOMP, Anchor, Prediction); (ii) complete configurations and prompts, including initial prompts and optimizer meta-prompts; (iii) task splits and experiment manifests; (iv) scheduler traces (IRT parameters, subset selections, swap decisions, and controller states); (v) per-example worker outputs and scores; and (vi) the scripts used for the statistical analyses.

C.3 Dataset Details

Table 5. Benchmark summary with task counts and types.

Family	Representative Tasks	# Tasks	N_{train} / N_{test}	Task Type
BBH	boolean_expr, dyck_lang, navigate, word_sort, logical_ded	26	∼200 / 50	Reasoning, Symbolic
BigBench	implicatures, metaphor, navigate, presuppositions, sports	5	∼200 / 50	NLU, Reasoning
MMLU	abstract_algebra, formal_logic, anatomy, college_chem	57	∼200 / 50	Knowledge QA
Math	GSM8K, GSM-Hard, MultiArith, MATH	4	∼200 / 50	Math

BBH (26 tasks). The BIG-Bench Hard tasks are a curated subset of BIG-Bench [17] selected by Suzgun et al. [16] for their difficulty and diversity. Our subset covers: *logical reasoning* (boolean expressions, causal judgement, formal fallacies, logical deduction at three scales (3-/5-/7-obj), web of lies), *spatial reasoning* (navigate, reasoning about colored objects, tracking shuffled objects at

three scales (3-/5-/7-obj)), *temporal reasoning* (date understanding, temporal sequences), *linguistic reasoning* (disambiguation QA, hyperbaton, ruin names, salient translation error detection, snarks), *world knowledge* (movie recommendation, penguins in a table, sports understanding), and *symbolic manipulation* (dyck languages, geometric shapes, object counting, word sorting).

BigBench-IPOMP (5 tasks). The 5-task subset selected by the IPOMP benchmark [11]: implicatures, metaphor understanding, navigate, presuppositions-as-NLI, and sports understanding. These tasks test diverse NLU capabilities at moderate difficulty.

MMLU (57 subjects). We use all 57 subjects of MMLU [18], spanning *STEM* (abstract algebra, college chemistry/physics, astronomy, computer security, electrical engineering), *humanities* (formal logic, high school European history), *social sciences* (business ethics, econometrics, government/politics, macroeconomics), and *applied domains* (anatomy, clinical knowledge, college medicine). After pooling the official splits, subjects contain roughly 110 to 1,700 multiple-choice questions, from which the optimization pool and held-out split are drawn as described below.

Train/test split protocol. For each MMLU subject, we pool the questions from the official `test`, `validation`, and `dev` splits of `cais/mmlu` on Hugging Face into a single subject-level question pool, and apply a deterministic random partition (`random_state=42`) in an 80/20 ratio into an optimization pool and a held-out evaluation split, with the pool capped at $N{\approx}200$ and the held-out split at $\approx$50 examples (smaller for compact subjects). We pool the official splits rather than designate one as held-out because MMLU does not provide a permissively-licensed training partition—its `auxiliary_train` consists of items aggregated from other benchmarks. The held-out split is excluded from the APO optimization loop: it is not used for prompt generation, evaluation-subset scheduling, optimizer feedback, or prompt selection. After each run, the returned prompt is evaluated on the held-out split. Parameter sensitivity is evaluated offline on representative tasks using held-out accuracy; these studies inform the shared scheduler defaults (Section E.4 and the "Default selection rationale" above). The main experiments use a shared configuration across all tasks, without per-task hyperparameter tuning. The same 80/20 partition and size caps apply to all 35 main-suite tasks (BBH, BigBench-IPOMP, Math), with the fixed `random_state=42`. Exact splits are included in the released manifests.

Math (4 tasks). GSM8K (grade-school math word problems, drawn from a 2,000-example subset of its training split), GSM-Hard (numerically harder variant), MultiArith (multi-step arithmetic), and MATH (competition-level mathematics, drawn from a 2,000-example subset). As for all tasks, the optimization pool and held-out split are sampled from these sources under the size caps of the split protocol below ($N{\approx}200$ / $\approx$50; Table 5).

Task-name abbreviations used in Figure 4. For compactness, the panel titles in Figure 4 use shortened forms. Their full task names are: Track. 5-obj = Tracking Shuffled Objects (5-obj); Log. Ded. 5/3 = Logical Deduction (5-obj / 3-obj); Geo. Shapes = Geometric Shapes; Snarks, Web of Lies, and Hyperbaton are BBH tasks of the same name; MMLU US Hist. = MMLU High School US History; BB Navigate / BB Metaphor are the BigBench-IPOMP Navigate and Metaphor Understanding subsets.

D Design Choice Justifications

Why top-two rather than top-m? We use the two leading prompts for three reasons. **(i)** The comparison between the current leaders most directly affects which prompt is returned and which prompts the optimizer builds on next, and for a pair the utility has the Fisher-information interpretation of Proposition 1 at low computational cost. **(ii)** The IRT estimates use all observed outcomes, not only those of the top two: Every $(p, i, y_{p,i})$ triple updates θ_p and b_i, so the whole candidate pool shapes the difficulty estimates b_i that enter $u_t(i)$; the top-two choice only determines which pair of abilities is read off at selection time. **(iii)** The leading pair $(p_t^{(1)}, p_t^{(2)})$ changes from round to round as IRT updates and new candidates arrive; across $T{=}10$ rounds, POES discriminates among 10–15 distinct leading prompts on average. A top-m or listwise utility is a natural alternative that costs more per round and may be preferable when several candidates are closely matched; we use top-two as the default throughout.

Why TF-IDF rather than dense embeddings? For $N{\approx}200$ pools, TF-IDF provides three practical advantages over learned embeddings. **(i)** *Zero external dependencies*: No additional encoder model is loaded or queried, keeping the scheduler self-contained and reproducible across hardware setups. **(ii)** *Domain robustness*: TF-IDF captures the *structural form* of a question (word-problem phrasing, logical-template pattern, date-format layout) rather than the numeric or lexical answer value, so it behaves sensibly even on tasks like GSM-Hard or MultiArith, where dense embeddings could confound structurally different problems that happen to share surface vocabulary. Qualitatively, on GSM-Hard the coverage term groups together word problems sharing the same reasoning template regardless of the numeric answer. **(iii)** *Computational triviality*: At $N{\approx}200$, the $O(N^2)$ similarity matrix fits in a few hundred kilobytes and is computed once per task. Dense embeddings would be a natural drop-in extension for larger pools ($N{\gtrsim}1000$), where structural-form robustness may no longer outweigh the advantages of learned semantic proximity; we leave this comparison to future work.

Why 1PL over 2PL? We deliberately choose the 1PL model over richer variants (e.g., 2PL with per-item discrimination) because it provides interpretable difficulty and ability estimates with fewer parameters to estimate from limited observations (8 candidate prompts per round, each scored on $k{=}20$ items), and the L2 prior provides additional regularization against overfitting in early rounds when the outcome matrix is sparse. An empirical comparison confirms this choice: 1PL outperforms 2PL with mean accuracy 0.93 vs. 0.84 across 4 tasks (Figure 3), with the largest single-task gap on Dyck Languages ($+21$ pp) where observations are noisiest. We also avoid batch L-BFGS for IRT estimation because the per-round outcome matrix is sparse (each round adds $|P_t|\ k{=}160$ outcomes at the defaults, out of $|P_t|\ N{\approx}1{,}600$ possible prompt–example pairs); a streaming MAP update with explicit L2 prior gives equivalent stability with no bookkeeping overhead.

Identification of the 1PL model. The 1PL model $\Pr(y{=}1){=}\sigma(\theta_p - b_i)$ is identifiable only up to a common additive shift: Translating all $\{\theta_p\}$ and $\{b_i\}$ by the same constant leaves every likelihood term unchanged. We handle this implicitly via the isotropic Gaussian prior $\mathcal{N}(0, 1/\kappa)$ on both parameter vectors ($\kappa{=}0.01$), which penalizes deviation from the origin and thus pins the shift. Our discrimination utility $u_t(i)$ depends on $\theta_{p_t^{(1)}} - b_i$ and $\theta_{p_t^{(2)}} - b_i$ (Equation (1)), so a common shift of all $\{\theta_p\}$ and $\{b_i\}$ leaves every $u_t(i)$ unchanged, and the unidentified shift does not affect selection. The prior strength κ, however, also changes the estimates in ways other than a common shift, so it can still influence selection. The hard clip $[-6, 6]$ provides a secondary safeguard against numerical drift during long runs but is rarely activated in practice.

Seeds and configurations. Each method is run with 2 random seeds in each of the $3 \times 2 = 6$ optimizer–model configurations, i.e., 12 optimizer–model–seed combinations per method. Table 1 reports each configuration separately, so the results show consistency across configurations; with two seeds per configuration, within-configuration seed variance is only coarsely estimated (Section I).

E Additional Experiment Results

Discussion. Figure 7 probes the per-round behavior underlying Result 7 and the 96.8% swap-acceptance statistic. Two patterns emerge. **(i) Late-stage gains.** On every panel, including GSM-Hard, POES (red) reaches or matches the best final score. Its curve still improves after round 6 on all 8 panels and within the last three rounds (8–10) on 6/8 (all except Prof. Medicine and GSM-Hard, where it reaches its final value at round 7), while most baseline curves plateau by rounds 4–6. The largest late jumps occur on Anatomy, College CS, Intl. Law, Formal Logic, and US Foreign Policy ($\geq$15 pp between rounds 7 and 9). We attribute these gains to the prompt-aware swaps continuing to extract signal as the leading prompt pair $(p_t^{(1)}, p_t^{(2)})$ rotates and the IRT difficulty estimates sharpen. **(ii) Monotone curves.** POES's curves are non-decreasing on all 8 panels, but so are the baseline curves, so monotonicity here is not specific to POES and does not test Proposition 3(ii): That proposition concerns the scheduling objective G_t and holds by construction of the swap rule (Section B.3), not the prompt scores plotted here. These training-subset curves are consistent with, but do not by themselves establish, the held-out results in Table 1.

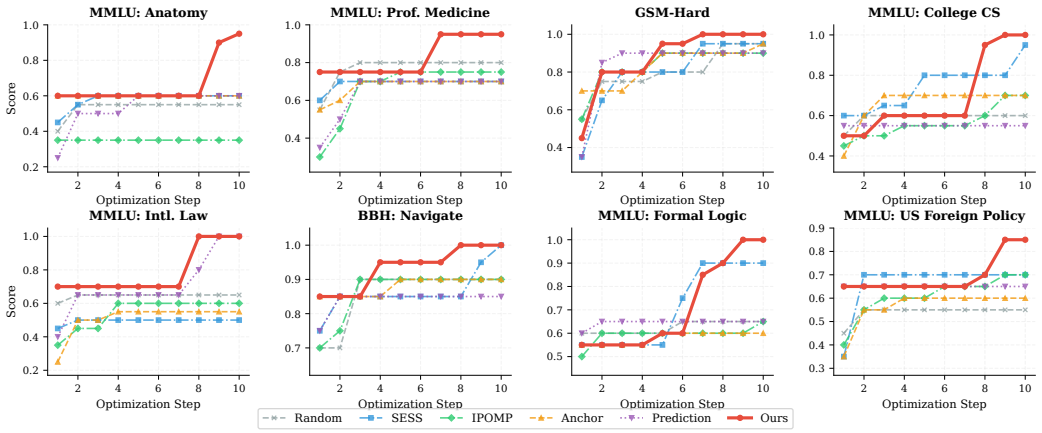


Figure 7. Optimization curves on 8 representative tasks where baselines have non-trivial headroom: BBH (Navigate), Math (GSM-Hard), and MMLU (Anatomy, Prof. Medicine, College CS, Intl. Law, Formal Logic, US Foreign Policy). Each curve is the mean over 2 seeds; POES denotes the fixed full method with the auto-warmup controller used in the main experiments.

E.1 Detailed Ablation Analysis

We report the configurations by their specified control settings and describe their observed accuracy on the five ablation tasks. The per-task values for No-Coverage and No-AutoWarmup are given together in Table 2; these values are not combined across configurations.

No-AutoWarmup. This setting does not use the automatic random-warmup controller. Its held-out accuracies are 0.900 on Dyck Languages, 0.780 on Navigate, 0.750 on GSM-Hard, 0.775 on MATH, and 0.950 on MultiArith (five-task mean 0.831).

No-Coverage ($\lambda_t = 0$). This setting omits the facility-location coverage term. Its held-out accuracies are 0.860 on Dyck Languages, 0.800 on Navigate, 0.690 on GSM-Hard, 0.810 on MATH, and 0.950 on MultiArith (five-task mean 0.822).

No-Adaptive (fixed parameters). This setting uses constant τ_t=0.01, B_t=3, and λ_t=0.5 throughout optimization. Its five-task mean is 0.757, illustrating the behavior of this fixed-parameter configuration on the same task set.

Disc+Cov (no swap updates). This setting uses IRT-based discrimination and facility-location coverage without the warm-start swap updates or adaptive controller. The recorded held-out scores include 0.790 on Dyck Languages, 0.720 on GSM-Hard, and 0.990 on MultiArith. This configuration shows that discrimination and coverage can work without adaptive swaps on some tasks, while their accuracy varies across tasks.

E.2 Efficiency Breakdown by Task Family

Table 6. Token consumption (K) and wall-clock time (s) by task family. RTI: relative token (or time) increase of POES vs. Random.

	Token (K)					Time (s)				
Family	Random	SESS	IPOMP	POES	RTI	Random	SESS	IPOMP	POES	RTI
BBH (26)	624	624	639	632	+1.2%	1,532	1,581	1,612	1,689	+10.3%
BigBench (5)	407	422	478	431	+5.9%	775	770	954	894	+15.4%
MMLU (57)	413	450	395	416	+0.7%	929	1,038	959	992	+6.8%
Overall (88)	475	500	471	481	+1.2%	1,098	1,183	1,152	1,192	+8.5%

Table 6 reports per-family token consumption and wall-clock time across Random, SESS, IPOMP, and POES (seed 44, k=20). The relative token increase is consistent across families (0.7–5.9%), with $+1.2\%$ overall vs. Random and -3.8% vs. SESS, indicating that POES's evaluation cost remains comparable to the baseline schedulers; the $+8.5\%$ wall-clock overhead reflects scheduler-internal CPU compute (IRT updates, swap optimization, adaptive control) outside the LLM-call cost.

E.3 Scaling to Larger Evaluation Pools

We examine whether POES remains effective when the optimization pool grows beyond the $N{\approx}200$ used in the main comparisons. On GSM8K, we vary the pool size over $N \in \{200, 500, 1000, 2000\}$ while keeping the per-candidate evaluation budget k=20, the number of optimization rounds T=10, the controller configuration, optimizer, worker model, initial prompt, and candidate-generation budget fixed. Only the pool available to the schedulers changes. Unlike the approximately 50-example held-out splits in the main suite, this supplementary experiment evaluates each returned prompt on the full official GSM8K test set of 1,319 examples.

Table 7. Larger-pool scaling on GSM8K. Full official test accuracy (%) for a fixed per-candidate budget k=20 and T=10 optimization rounds. **Gain** is the absolute percentage-point improvement of POES over the strongest competing scheduler in the same row; only N varies.

N	Random	SESS	IPOMP	Anchor	Prediction	POES	Gain (pp)
200	87.8	88.6	87.9	87.4	88.5	**89.4**	**+0.8**
500	88.3	88.7	86.9	87.9	88.9	**90.7**	**+1.8**
1,000	87.9	87.4	87.0	88.2	89.0	**91.1**	**+2.1**
2,000	88.4	87.8	88.0	87.6	89.0	**90.6**	**+1.6**

POES achieves the highest test accuracy at every evaluated pool size, outperforming the strongest baseline by $+0.8$ to $+2.1$ pp (Table 7). Its accuracy rises from 89.4% at N=200 to 91.1% at N=1000, then changes to 90.6% at N=2000. These results support effectiveness across the evaluated range without per-size controller retuning; they do not establish performance for larger pools or other tasks.

Table 8. Scheduler-side overhead for larger pools. Pairwise-similarity preprocessing (TF-IDF and cosine similarities) is performed once per pool; facility-location selection is measured per round. Times exclude worker-model inference, API, and network overhead.

N	Similarity (one-time)	Selection / round	Peak memory
200	0.09 s	0.02 s	0.19 GB
500	0.13 s	0.04 s	0.19 GB
1,000	0.26 s	0.04 s	0.20 GB
2,000	0.31 s	0.04 s	0.28 GB

Table 8 reports measured preprocessing and selection times for the evaluated implementation, rather than a subquadratic worst-case complexity claim. Peak memory increases from 0.19 to 0.28 GB over the evaluated range. Facility-location coverage still involves $O(N^2)$ pairwise similarities in general, so substantially larger pools require further validation.

E.4 Complete Parameter Sensitivity with Token Cost

Table 9 presents the complete parameter sensitivity analysis across all six scheduler parameters, including token consumption. The three scheduler-internal parameters (M, τ, ρ) are a subset of this table. A total of **108 experiment runs** (6 parameters $\times$ 3 values $\times$ 3 tasks $\times$ 2 seeds) underpin this analysis.

Key observations. **(1) Budget k:** Figure 6 (main text) shows the accuracy trade-off across $k \in \{5, 10, 15, 20, 30\}$ on 6 representative benchmarks (2 seeds each). The 6-task average rises from 0.74 (k=5) to 0.84 (k=20) and then changes little, while token cost substantially increases from k=20 to k=30 in the parameter sweep. This supports k=20 as a practical accuracy–cost compromise for

Table 9. Complete parameter sensitivity with computational cost. We evaluate six parameters across three tasks with two random seeds per setting. Each parameter sweep is conducted independently. Token consumption and runtime are averaged across seeds. Default parameter values (*) are highlighted in blue. Boldface indicates the highest reported accuracy within each parameter group. A total of 108 runs are included.

		ACCURACY				COST (avg)		
Param	Value	Dyck Lang.	GSM-Hard	MultiArith	Mean	Tok. (K)	Time (s)	ΔTok
k	10	0.800	**0.799**	**0.967**	0.855	478	2,370	−64%
	20*	0.900	0.742	0.950	0.864	892	2,884	−33%
	30	**0.980**	0.705	0.942	**0.876**	1,331	3,264	—
T	10*	0.820	0.826	**0.992**	0.879	999	3,091	−37%
	15	**0.980**	**0.852**	0.983	**0.938**	1,299	4,241	−18%
	20	0.920	0.720	0.975	0.872	1,589	5,357	—
M	64	0.920	0.686	0.975	0.860	916	2,615	+0%
	128*	**0.980**	**0.814**	**0.983**	**0.926**	913	2,666	—
	256	0.780	0.705	0.958	0.814	862	2,379	−6%
B_0	1	0.880	**0.909**	0.958	**0.916**	842	2,769	+1%
	3*	**0.920**	0.758	**0.983**	0.887	830	2,527	—
	5	0.840	0.674	0.958	0.824	878	2,447	+6%
τ	0.01	0.900	**0.886**	0.975	**0.920**	952	3,018	+9%
	0.05*	**0.940**	0.716	**0.983**	0.880	873	2,776	—
	0.10	0.900	0.705	0.925	0.843	818	2,364	−6%
ρ	1.02	**0.960**	0.693	**0.958**	0.870	807	2,498	−4%
	1.05*	0.940	**0.841**	**0.958**	**0.913**	842	2,537	—
	1.10	0.840	0.663	0.925	0.809	999	2,841	+19%

these experiments, rather than a task-independent optimum. **(2) Steps T:** T=15 yields the best mean accuracy (0.938) but costs 30% more tokens than T=10. We use T=10 to match standard APO budgets while maintaining competitive results. **(3) Candidate pool M:** M=128 dominates at near-identical token cost to M=64, while M=256 degrades accuracy despite similar cost—indicating that too many swap candidates introduces noise. **(4) Swap budget B_0, switch cost τ, warmup ratio ρ:** These scheduler-internal parameters affect subset selection and can also change token consumption: increasing ρ from the default 1.05 to 1.10 raises token use from 842K to 999K (approximately 19%). Default values are generally among the strongest across the representative tasks.

F Scheduler Mechanism Diagnostics

This section provides detailed diagnostics of the POES scheduler's internal behavior, extracted from the `train_scheduler_trace` logged during our POES experiments.

F.1 Subset Stability

In active warm-start rounds, each accepted swap replaces one element and the shift is bounded by $2m_t/k \leq 2B_t/k$, with increments of 0.1 at k=20 when swapped items are distinct from the starting subset. Warmup instead draws random subsets, so the swap-budget bound is not applicable to those transitions. The statistics in Table 10 average logged shifts within each run; they should not be interpreted as the shift of a particular round or as a proof of the active-round bound. Both $\bar{\delta}$ and τ_t are measured on $G_t \in [0, 20.8]$, on which a single swap changes the value by at most $1+\lambda_t \leq 1.8$.

F.2 Swap Dynamics and Adaptive Behavior

Swap dynamics: (1) Swap activity is highest in the first few active rounds after warmup, as the scheduler transitions from a random subset to a discriminative one, consistent with the schedule

Table 10. Subset-shift diagnostics for POES. Subset shift is $|S_t \triangle S_{t-1}|/k \in [0, 2]$. During active warm-start scheduling, Proposition 3(i) bounds this quantity by $2B_t/k \leq 0.50$ at $B_{\max}$=5, k=20; random subsets drawn during warmup are not subject to this bound. The reported statistics summarize per-run averages over logged subset transitions, not single-round values.

Statistic	Value
Per-run average shift: mean $\pm$ std across runs	0.287 ± 0.043
Per-run average shift: median	0.280
Per-run average shift: min / max	0.195 / 0.395
Swap acceptance rate (over all swap-loop iterations)	96.8%
Mean warmup rounds	2.9 ± 0.3
Warmup range	2–3 rounds
Mean active (non-warmup) rounds per run	$\sim$7
Inter-round drift $\bar{\delta} = \mathbb{E}_t\lvert G_t(S_{t-1}) - G_{t-1}(S_{t-1})\rvert$	≈ 0.02
Operating threshold τ_t (mean)	≈ 0.05
Absolute scale: $G_t(S) \in [0,\ k{+}\lambda_{\max}] = [0, 20.8]$	(k=20, $\lambda_{\max}$=0.8)
Change of G_t from one swap: $\lvert G_t(S') - G_t(S)\rvert \leq 1{+}\lambda_{\max} = 1.8$	($\tilde{u}_t$, Cov $\in [0, 1]$)

$r_t^B : 0.25 \to 0.10$ in Equation (10) that allocates a larger budget early. (2) The switch cost τ_t rises with progress π_t and with score-history standard deviation σ_t (Equation (9)), then is multiplied by β^τ_{stag}=0.8 during stagnation or β^τ_{imp}=1.1 during improvement; in practice τ_t gradually rises across training and tightens further when score history is noisy, exactly as designed. (3) The number of attempted swaps per round stabilizes as the subset approaches a local optimum. The swap-acceptance rate (96.8%) is computed over swap-loop iterations, each of which tests only the best available swap (i^*, j^*) within the top-M candidate pool. Because the loop stops at the first rejected swap, each round contributes at most one rejection, so this rate mainly indicates that most rounds use their full swap budget B_t. It does not mean that candidates pass τ_t by construction: A high discrimination utility of the incoming item does not by itself guarantee that the total gain $G_t(S') - G_t(S)$, which also reflects the removed item and the change in coverage, reaches τ_t.

Coverage weight evolution: The coverage weight follows the linear schedule of Equation (5): λ_t starts at λ_{start}=0.8 at π_t=0 and linearly decays toward λ_{end}=0.4 at π_t=1, so $\lambda_t \in [0.4, 0.8]$ throughout optimization. The lower endpoint λ_{end}=0.4 keeps the coverage coefficient strictly positive at all times, preventing the scheduler from collapsing onto pure discrimination once IRT estimates concentrate. Early rounds therefore receive maximum coverage weight (broad sampling across the data manifold), and late rounds shift toward discrimination-heavy selection while still preserving a non-trivial diversity floor.

Warmup-to-active transition: Warmup lasts 2–3 of the T=10 rounds (mean 2.9$\pm$0.3): The first $t_w^{\min}$=2 rounds always use random subsets, round 3 does so unless ρ_3 already reaches ρ_{exit}, and the cap $t_w^{\max}$=3 then ends warmup regardless of ρ_t. The short warmup therefore reflects this cap and does not by itself show that the discrimination signal has become informative. The cap limits warmup to at most 30% of the round budget, and the exit threshold $\rho_{\text{exit}} = 1.05$ is deliberately low—it requires only a 5% discrimination ratio above background—so that an informative signal ends warmup early.

G Comparison with Oracle Full-Set Evaluation

An important reference point is the *oracle* setting where prompts are evaluated on the full training set ($k = N$) rather than a subset. While we do not run full-set evaluation for all tasks (due to prohibitive cost), we note that the purpose of evaluation scheduling is to *approximate* full-set evaluation quality at a fraction of the cost.

Why subset evaluation can outperform full-set. Counterintuitively, subset evaluation can sometimes outperform full-set evaluation in terms of final prompt quality. This occurs because:

1. **Noise reduction:** A well-chosen subset can filter out noisy or uninformative examples that dilute the evaluation signal.
2. **Focus effect:** By concentrating evaluation on discriminative examples, the optimizer receives sharper feedback about which prompts are truly better.
3. **Budget reallocation:** The cost savings from subset evaluation can be reinvested into more optimization steps or more prompt candidates per step.

This phenomenon is analogous to the data pruning literature [48], where training on a carefully selected subset can match or exceed full-data training.

H Qualitative Case Studies

This appendix provides qualitative evidence complementing the quantitative results in the main text. We examine illustrative prompt outputs from each scheduler (Section H.1), how the evaluation subset evolves during optimization (Section H.2), and which training examples IRT identifies as most versus least discriminative (Section H.3).

H.1 Illustrative Prompt Outputs from Different Schedulers

Table 11 presents illustrative prompt outputs from the main experiments (OPRO $\times$ Llama-3.1-8B-Instruct). The examples show how task instructions differ in wording, specificity, and operational detail across schedulers. They are qualitative examples, not estimates of average behavior across seeds or evidence of comparative accuracy; numerical comparisons are reported separately in the main experimental results.

Observation: Instructions vary in operational detail. On **BBH Navigate**, the examples range from displacement summaries to explicit matrix operations and sequential position updates. On **BBH Logical Deduction**, they range from short task instructions to descriptions of arrangement enumeration. On **MultiArith**, several schedulers produce a similar "parse $\rightarrow$ compute $\rightarrow$ output" structure. These examples illustrate variation in the instructions generated during APO; they do not establish which wording is more accurate or whether a scheduler systematically produces shorter prompts.

H.2 Evaluation Subset Evolution Over Rounds

A distinguishing feature of POES is that its evaluation subset *co-evolves* with the prompt population, whereas Random fixes a single subset before optimization and SESS selects one principled subset that also remains static throughout. Table 12 illustrates the qualitative evolution of POES's selected subset on BBH Navigate (seed 44) over the course of optimization.

Several observations are worth highlighting:

1. **Warmup provides a stable foundation.** During warmup, POES uses randomly sampled subsets. This is deliberate: The IRT model requires a minimum number of prompt–example interactions to produce reliable discrimination estimates. Premature active selection would be based on noisy IRT parameters and could mislead the optimizer.
2. **Transition to active scheduling is gated by the data and capped.** The warmup-to-active transition is triggered once the top-k discrimination ratio $\rho_t = \mathrm{mean}_{i \in \mathrm{TopK}_k(u_t)} u_t(i) \,/\, \left(\mathrm{mean}_{i \in V}\, u_t(i) + \varepsilon\right)$ (Equation (7)) reaches ρ_{exit}=1.05, indicating that the items the scheduler would select are more informative than the pool average, or at the latest after round $t_w^{\max}$=3. The check is applied only after $t_w^{\min}$=2 forced warmup rounds, and the cap ends warmup even if ρ_t is still below the threshold. On BBH Navigate, warmup lasts 2–3 rounds.
3. **Bounded swaps ensure stability.** After warmup exit, the subset evolves gradually: The swap budget B_t limits the number of items that can change per round (typically 2–4 out of k=20). This prevents the erratic subset changes observed with IPOMP, which can replace up to 100% of the subset in a single round.

Table 11. Illustrative prompt outputs on three tasks. Green rows identify POES. These examples illustrate instruction wording and operational detail under OPRO × Llama-3.1-8B-Instruct. Ellipses indicate truncation of long prompts.

Task	Method	Example Prompt
BBH Navigate	RANDOM	Calculate the final coordinates after executing the given moves and output "Yes" if they equal the starting point (0,0); otherwise output "No".
	SESS	Initialize position $= (0, 0)$ and orientation matrix $R = [[0, -1], [1, 0]]$ (facing north). Process each command sequentially: "Turn left" updates R via a 90° rotation matrix; "Take N steps" updates position $\leftarrow$ position $+N \cdot (R \cdot [0, 1])$…output "Yes" if position $= (0, 0)$, else "No".
	IPOMP	Trace the path step by step, compute the final coordinates relative to the start, and output "Yes" if the coordinates are (0,0); otherwise output "No".
	ANCHOR	Translate each command into its coordinate change, sum all changes, and output "Yes" if the total displacement is (0,0); otherwise output "No".
	PREDICTION	Determine the total displacement vector; answer "Yes" if it equals zero, otherwise answer "No".
	POES (OURS)	Process the command list sequentially, updating a current (x, y) position; after the final instruction output "Yes" if the position is (0,0), otherwise output "No".
BBH Logical Deduction (5-obj)	RANDOM	Solve the puzzle by determining the single arrangement that fulfills every given condition, then choose the answer choice that correctly specifies the position of the queried item.
	SESS	Analyze the given statements, infer the correct ordering, and choose the option that logically follows.
	IPOMP	Analyze the given constraints and choose the option that logically satisfies them.
	ANCHOR	Enumerate every arrangement that meets all the given constraints, pinpoint the one ordering that remains, and select the answer choice that is inevitably true for that ordering.
	PREDICTION	Combine all given statements to infer the definitive ranking of the five items, then pick the answer option that correctly states the specified position or relationship.
	POES (OURS)	*Select the correct option based on the given logical statements.*
MultiArith	RANDOM	Read the description, list every numeric value, convert the narrative into the exact integer arithmetic expression implied (add for gains, subtract for losses, multiply for repeated groups, divide for splits), evaluate it, and output only the resulting number.
	SESS	Read the statement, pull out all relevant numbers, determine the required arithmetic steps, compute the result, and output the final integer answer only.
	IPOMP	Parse the problem, extract all quantities, infer the needed arithmetic steps, compute the result, and output only the final numeric answer.
	ANCHOR	Extract the relevant numbers, decide the needed arithmetic operation, compute the result, and output only the final numeric answer.
	PREDICTION	List all numbers present, deduce the precise sequence of integer operations needed to answer the prompt, compute the result, and output only that final number.
	POES (OURS)	Parse the word problem, identify all relevant quantities and the required arithmetic operation(s), perform the calculation, and output only the final numeric answer.

Table 12. Qualitative evolution of the POES evaluation subset on BBH Navigate (seed 44). During warmup, the subset is random; after warmup exit, it is actively refined via bounded swaps.

Round	Phase	Subset Characteristics
1	Warmup	Random subset with broad coverage across difficulty levels. Contains a mix of easy (1–2 step), moderate (3–5 step), and hard (6+ step) navigation sequences. No discrimination signal available yet.
3	Warmup exit	First actively selected subset. IRT parameters have matured sufficiently to estimate item discrimination. The subset shifts toward moderate-difficulty examples where top-2 prompts disagree, while retaining a few easy/hard anchors for coverage.
5	Active	Subset refined via 2–3 swaps from round 3. Low-discrimination items (trivially easy sequences solved by all prompts) are swapped out for items near the ability boundary. Coverage term prevents over-concentration.
10	Active (final)	Subset has stabilized: Most items are moderate-complexity navigation sequences (3–5 turns) that maximally separate the current top prompt candidates. Only 1 swap from round 5, reflecting convergence of both the prompt population and the IRT model.

4. **Contrast with static methods.** Random and SESS both use a fixed subset from round 1 through the final round. While SESS's subset is more principled (selected via submodular optimization over embedding diversity), it cannot adapt to the changing prompt population. As optimization progresses and top prompts converge, the discriminative examples shift—but static methods cannot follow this shift.

H.3 Examples with High vs. Low Discrimination Utility

The IRT-based discrimination utility is the core mechanism by which POES identifies informative evaluation examples. Here we provide intuition for what makes an example "discriminative" by examining concrete cases from BBH Navigate.

An example has **high discrimination** when it lies near the IRT difficulty boundary where the current top-2 prompts *disagree*: One prompt solves it correctly while the other fails. Such examples provide maximal information for separating prompt quality. Conversely, an example has **low discrimination** when all top prompts either succeed (trivially easy) or fail (impossibly hard)—in both cases, the example contributes no signal for distinguishing prompt quality.

Table 13. Examples with high vs. low discrimination utility on BBH Navigate and multi-step arithmetic tasks. High-discrimination examples are those where top prompts disagree; low-discrimination examples elicit uniform responses.

Discrimination	Example Type	Explanation
High	Moderate-complexity navigation (3–5 turns)	Top prompts disagree: One correctly tracks facing direction through multiple turns, while the other loses track after 2–3 direction changes. These examples are near the "ability boundary" in IRT terms.
High	Multi-step arithmetic with carries	Prompt phrasing affects whether the model explicitly shows intermediate work. Prompts that encourage step-by-step computation succeed; those requesting concise answers fail on carry propagation.
Low	Simple 2-step navigation (always correct)	All prompts succeed regardless of phrasing—the task is too easy to reveal quality differences. Including such examples wastes evaluation budget without providing discrimination signal.
Low	10+ step navigation (always wrong)	All prompts fail because the sequence exceeds the model's reliable tracking capacity. No prompt phrasing can compensate for this fundamental limitation, so the example provides no signal.

This pattern generalizes across tasks: Discrimination utility is highest for examples at the "Goldilocks zone" of difficulty—hard enough that weak prompts fail, but tractable enough that well-crafted prompts succeed. The IRT model captures this automatically through its item characteristic curves, without requiring manual difficulty annotation. As the prompt population improves over optimization rounds, the difficulty boundary shifts upward, and POES's active scheduling tracks this shift by selecting progressively harder (but still discriminative) examples.

H.4 POES Failure Modes: When Discrimination Utility Collapses

Despite its aggregate gains, POES can underperform on individual tasks. Here we examine two failure cases under OPRO $\times$ Llama-3.1-8B (mean over 2 seeds): Temporal Sequences (`bbh_temporal_sequences`) and Tracking Shuf. 7-obj (`bbh_tracking_shuffled_objects_seven_objects`). They illustrate train$\rightarrow$test mismatch and training saturation, respectively. Table 14 reports seed-mean test scores for all schedulers. POES falls below both the per-task best baseline and the average of the five baselines' seed means in these cases: 0.810 vs. 0.898 on Temporal Sequences, and 0.920 vs. 0.944 on Tracking Shuf. 7-obj. These

examples qualify the family-level findings in Result 2 and help identify conditions in which POES is less effective.

Table 14. Two representative failure cases for POES. **Bold** marks the per-task best baseline. POES's *training* sub-score is competitive in both cases (col. **Sub.**), but the held-out test score lags. The two rows correspond to two distinct failure modes diagnosed below: (i) train→test distribution mismatch, and (ii) training saturation that flattens the IRT discrimination signal.

Task	N_{tr}	N_{te}	**POES Sub.**	**POES Test**	Random	SESS	IPOMP	Anchor	Pred.
Temporal Sequences	200	50	0.95	0.81	0.89	0.91	0.86	0.91	**0.92**
Tracking Shuf. 7-obj	200	50	1.00	0.92	**0.99**	0.97	0.92	0.93	0.91

Failure mode (i): train→test distribution mismatch (Temporal Sequences). On Temporal Sequences POES achieves a training sub-score of 0.95 – nominally on par with the best baselines – yet its held-out test score drops to 0.81, ranking last among six schedulers. Inspecting the scheduler trace, POES's warm-start swap engine remains highly active throughout: Across the $T{=}10$ optimization rounds the swap-loop acceptance ratio exceeds 97% (consistent with the suite-level 96.8% in Table 10), and the selected subset shifts on *every* round (avg. subset shift ≈ 0.30). The facility-location coverage objective is therefore being optimized aggressively on the *training* pool; what fails is the implicit assumption that the chosen training subset's TF-IDF coverage transfers to the test split. For Temporal Sequences the train/test partitions cover slightly different scheduling templates, so a subset that maximizes training coverage may under-represent the test distribution. **Symptom check:** (i) high training sub-score, (ii) low test score, (iii) high swap-acceptance ratio. **Implication:** POES's submodular guarantees (Proposition 2, Proposition 3) hold *on the training pool*; transfer to test requires the train/test similarity assumption that random sampling implicitly enforces.

Failure mode (ii): Training saturation flattens IRT discrimination (Tracking Shuf. 7-obj). On Tracking Shuf. 7-obj *every* scheduler—including POES—reaches a training sub-score of 1.00 within the first few rounds. Under such saturation, the estimated abilities of the two leading prompts move close together, so $\Delta\theta_t$ becomes small and, by Proposition 1, every raw utility $u_t(i) \approx (\Delta\theta_t)^2 \cdot \mathcal{I}(\bar{\theta}_t; b_i)$ shrinks by the same factor. Min-max normalization removes this common factor, so the normalized term $\tilde{u}_t$ does *not* vanish: It ranks items by $\mathcal{I}(\bar{\theta}_t; b_i)$, i.e., by how close their difficulty is to the common ability level, and keeps its full weight in G_t. Only if all $u_t(i)$ are exactly equal (zero range) does our convention set $\tilde{u}_t \equiv 0$, reducing the selection to coverage alone. In the saturated regime, the discrimination term therefore still drives selection, toward items near the common ability level, although with $\Delta\theta_t \approx 0$ no item is expected to separate the two prompts. POES's test score (0.92) lags Random (0.99) by 7 pp despite identical training performance; one possible explanation is that random draws spread evaluation across the pool, whereas POES keeps concentrating it on items that no longer separate the leaders. **Symptom check:** (i) saturated training sub-scores (≈ 1.0 across all schedulers), and (ii) Random matches or exceeds the structured baselines on test. **Implication:** POES needs the leading prompts to remain separable on the training pool (non-negligible $\Delta\theta_t$) to extract signal from the IRT model; once they have converged, the normalized discrimination term no longer reflects a difference between them.

I Limitations and Societal Impacts

Scope of the theory. Our four propositions characterize the scheduling layer alone—the per-round selection of S_t given the current prompt population and IRT estimates—and do not bound end-to-end APO convergence. The quality of the returned prompt depends jointly on the optimizer's proposals (OPRO/EvoPrompt), the worker model, the scheduler, and the match between training pool and test distribution. Proposition 4 additionally assumes a uniform drift bound δ, which our diagnostics do not certify (Section F).

Statistical scope. The main comparisons use two seeds per optimizer–model configuration and held-out splits of about 50 examples per task, so one test example corresponds to roughly 2 pp. Per-task differences of a few points are therefore sensitive to seed and split variation. The shared defaults were informed by offline parameter sensitivity evaluated on held-out splits of representative

tasks (Sections C and E.4). We emphasize aggregate results and their consistency across six optimizer–model configurations rather than individual per-task gaps. Our comparison with full-set evaluation is limited to the discussion in Section G.

Distributional assumptions and dependence on the optimizer. POES shapes feedback on the training pool and implicitly assumes that this pool is representative of the test distribution; under train/test mismatch (Section H.4, failure mode (i)), a more discriminative training subset need not yield higher test accuracy. POES also determines how candidates are compared, not which candidates exist: When the optimizer proposes weak or near-identical prompts, or when all candidates saturate on the training pool, the discrimination signal can weaken, reducing the potential gains.

Outcome type, scale, and design choices. The main comparisons use pools of $N{\approx}200$ and binary $y_{p,i} \in \{0, 1\}$ outcomes; a supplementary GSM8K study evaluates $N \in \{200, 500, 1000, 2000\}$ with a fixed $k{=}20$ (Section E.3). This demonstrates performance on one task up to $N{=}2000$ but does not establish scaling across tasks or beyond that range. Facility-location coverage requires $O(N^2)$ pairwise similarities; for substantially larger pools, lazy-greedy evaluation [35] and sparse or approximate nearest-neighbor similarities are natural extensions. For continuous-metric generation tasks, the binary discrimination utility could be replaced by the predicted score difference between the two leading candidates, or by Beta/Gaussian IRT variants, while coverage and bounded swaps carry over unchanged; our guarantees are specific to the binary formulation. Finally, top-two discrimination, the 1PL model, TF-IDF coverage, and the controller constants were validated on subsets of tasks (Sections D and E.4); richer alternatives may help when candidate prompts are closely matched, item discrimination varies strongly, or lexical similarity is a poor proxy for task similarity.

Societal impacts. *Positive impacts.* At a fixed evaluation budget, POES selects more informative examples for prompt comparison, and our experiments show improved accuracy–cost tradeoffs on representative tasks. This may help practitioners obtain better prompts without increasing the number of examples evaluated per candidate; any energy savings depend on the chosen budget and deployment setting. *Potential negative impacts and mitigations.* Like any APO technique, more efficient prompt optimization could in principle accelerate the iteration of unsafe prompts (e.g., jailbreaks) targeting deployed LLMs; POES does not introduce capabilities beyond standard APO but inherits this dual-use concern. We see two natural mitigations: (i) deploying POES on top of safety-aligned worker models with input/output filters, since POES optimizes against whatever evaluation signal the worker provides and so safety filters propagate into the optimization loop; (ii) responsible release—our reference implementation will be released with example configurations on benign reasoning benchmarks rather than red-teaming workflows. *Fairness considerations.* The IRT-based discrimination utility ranks examples by how well they separate top prompts, which could in principle under-sample rare or out-of-distribution items if those items have low posterior discrimination; the explicit facility-location coverage term in G_t is designed precisely to counteract this collapse, but practitioners deploying POES on safety-critical decisions should audit the resulting subset for representation of protected slices. As a scheduling-layer technique POES does not modify the LLM itself and inherits the standard societal considerations of APO research without introducing new high-risk capabilities.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the contributions (formulation of evaluation scheduling for APO as an online adaptive testing problem, the POES algorithm with formal guarantees, and experimental validation across multiple optimizers and benchmarks), and the experimental claims are supported by Section 3. The design-variant results are reported as task-level observations without attributing differences between configurations to one isolated change (Section 3.4), and the theoretical claims are scoped to the scheduling layer (Section 2.7).

   Guidelines:

   - The answer [NA] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [NA] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The main text discusses the scheduling-level scope of the theory and binary outcomes in the conclusion (Section 5). The supplementary GSM8K scaling study up to $N{=}2000$ is reported in Section E.3. Detailed limitations in Section I cover the scheduler-level scope of the theory, limited seed counts and held-out split sizes, offline parameter sensitivity, dependence on train/test similarity and candidate quality, and the need to validate larger pools across further tasks and continuous metrics; two failure cases are analyzed in Section H.4.

   Guidelines:

   - The answer [NA] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All four propositions are formally stated in Section 2.7 with complete proofs in Section B. Assumptions (1PL IRT model, online MAP estimation with an isotropic Gaussian L2 prior, binary outcomes, bounded inter-round drift for Prop. 4) are explicitly stated alongside each proposition, and Section 2.7 states that the guarantees concern the scheduling objective for a given candidate pool rather than end-to-end APO convergence.

   Guidelines:

   - The answer [NA] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section C provides complete hyperparameter configurations (Table 4), Section C.3 describes all datasets, and implementation details including model IDs (Llama-3.1-8B-Instruct and Qwen-2.5-7B-Instruct workers, GPT-OSS-120B meta-optimizer), API endpoints (vLLM serving), and random seeds are fully specified. The full POES procedure is given as pseudocode in Algorithm 1 (Section A).

   Guidelines:

   - The answer [NA] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The supplementary material contains code, scripts, and configuration files for reproducing the main experiments. All datasets used in our experiments (BBH, BigBench-IPOMP, MMLU, GSM8K, GSM-Hard, MATH, MultiArith) are publicly available benchmarks, and the worker models (Llama-3.1-8B-Instruct and Qwen-2.5-7B-Instruct) are openly accessible. We will publicly release code for POES and all baselines, complete configurations and prompts, task splits and manifests, scheduler traces, per-example outputs, and the statistical-analysis scripts (Section C, "Release plan").

   Guidelines:

   - The answer [NA] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 3 describes the experimental setup (optimizers, base models, evaluation protocol), Section C provides all hyperparameters in Table 4 (subset size, IRT update

schedule, prior parameters), and Section C.3 details all benchmark configurations including train/test splits.

Guidelines:

- The answer [NA] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Table 1 and Table 2 report two-seed means, while Table 3 reports single-seed results, without per-cell error bars or significance tests for the main comparisons. Consistency across six optimizer–model configurations provides complementary evidence but does not replace within-configuration uncertainty estimates. The scheduler diagnostics in Section F report means with standard deviations across main-suite runs; the limited seed count and small held-out splits are acknowledged in Section I.

   Guidelines:

   - The answer [NA] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section C specifies GPU types (2 $\times$ NVIDIA A100 80GB), model serving details (vLLM), and total compute ($\sim$1,000 GPU-hours for the full experiment suite of 35 main + 57 MMLU tasks $\times$ 7 methods $\times$ multiple seeds), and Table 3 reports both token consumption and wall-clock time for POES and all baselines, making the per-run compute footprint explicit.

   Guidelines:

   - The answer [NA] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work focuses on evaluation-scheduling for automatic prompt optimization and does not involve human subjects, deception, or harmful applications. We have reviewed and conform to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [NA] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No] , they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Section I ("Societal impacts") discusses both directions: positive impacts (improved accuracy–cost tradeoffs at a fixed evaluation budget, with potential benefits for resource-constrained researchers); potential negative impacts (the standard APO dual-use concern that more efficient prompt optimization could accelerate unsafe-prompt iteration); proposed mitigations (deployment on top of safety-aligned worker models with input/output filters, and responsible release with benign-benchmark configurations); and fairness considerations (the explicit facility-location coverage term in G_t counteracts subset collapse onto highly discriminative items, with an explicit recommendation to audit subsets for protected-slice representation in safety-critical deployments).

    Guidelines:

    - The answer [NA] means that there is no societal impact of the work performed.
    - If the authors answer [NA] or [No] , they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This work releases a scheduling algorithm for automatic prompt optimization. It does not release any new pre-trained model or scraped dataset, and therefore poses no high-risk-for-misuse concerns that would require additional safeguards.

    Guidelines:

    - The answer [NA] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All benchmarks (BBH, BigBench-IPOMP, MMLU, GSM8K, GSM-Hard, MATH, MultiArith) and models (Llama-3.1-8B-Instruct, Qwen-2.5-7B-Instruct, GPT-OSS-120B) are properly cited at first use, and all baseline optimizers (OPRO, EvoPrompt-GA/DE) and scheduling baselines (SESS, IPOMP) are credited to their original authors. All assets are publicly available under their respective licenses and terms of use.

    Guidelines:

    - The answer [NA] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new asset introduced is the POES scheduler implementation. Upon release, the code will include documentation, configuration files, and instructions for reproducing all experiments reported in the paper.

Guidelines:

- The answer [NA] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [NA] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work does not involve human-subjects research, so IRB review is not applicable.

    Guidelines:

    - The answer [NA] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

Justification: LLMs are a core component of the methodology: Section 2 and Section 3 fully describe the use of Llama-3.1-8B-Instruct and Qwen-2.5-7B-Instruct as the worker LLMs whose responses define the IRT outcomes, and GPT-OSS-120B as the meta-optimizer LLM that proposes prompt edits, including model identifiers and vLLM serving configuration.

Guidelines:

- The answer [NA] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.