MODEPROMPT: TAXONOMY-GUIDED PARETO EVOLUTION FOR PROMPT OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) are highly sensitive to prompts specifying task instructions, making automatic prompt optimization (APO) important for reliable performance. Existing APO methods typically guide search using scalar validation scores, which can obscure performance imbalances across heterogeneous reasoning patterns and produce prompts with competitive overall accuracy but low worst-group accuracy. We propose MODEPrompt, a reflective multi-objective evolutionary framework that induces a task-specific taxonomy of mutually exclusive reasoning subgroups from development questions without using answer labels, and treats subgroup accuracies as distinct Pareto objectives. MODEPrompt introduces semantic differential evolution, which contrasts Pareto-front and non-front prompts to extract directional semantic patches that guide mutation and crossover. It further induces outcome-grounded meta-rules from trial acceptance outcomes and subgroup-level score changes, accumulating useful search experience across iterations. Extensive experiments across diverse benchmarks show that MODEPrompt achieves strong overall performance, improves worst-group accuracy, and reduces performance disparities across reasoning subgroups. Further analyses support the stability, semantic interpretability, and performance relevance of the induced taxonomy. Code is available at `https://anonymous.4open.science/r/MODEPrompt-2843`.

1 INTRODUCTION

Large Language Models (LLMs) perform strongly across natural language processing and reasoning tasks (Brown et al., 2020; Wei et al., 2022; Kojima et al., 2022; Srivastava et al., 2022), yet remain highly sensitive to how task instructions and output constraints are phrased (Liu et al., 2023; Reynolds & McDonell, 2021; Zamfirescu-Pereira et al., 2023). Prompting enables lightweight adaptation of black-box models, but manual prompt design is costly, brittle, and difficult to transfer. Automated Prompt Optimization (APO) instead searches, edits, and selects prompts with limited human intervention (Zhou et al., 2023; Pryzant et al., 2023; Guo et al., 2024).

Despite increasingly sophisticated search mechanisms—from LLM-as-optimizer heuristics and textual gradients to evolutionary and multi-agent refinement (Yang et al., 2024; Pryzant et al., 2023; Guo et al., 2024; Zhang et al., 2026a)—most APO methods still rank prompts by a single aggregate validation score. Tasks, however, often combine heterogeneous reasoning patterns, and scalar aggregation can mask a prompt's localized weaknesses. This concern parallels robust learning, where average metrics obscure failures on minority subpopulations (Hashimoto et al., 2018; Sagawa et al., 2019; Koh et al., 2021); reflective and Pareto-based optimization likewise suggests that structured feedback can preserve complementary strengths lost under scalar rewards (Agrawal et al., 2026; Mouret & Clune, 2015).

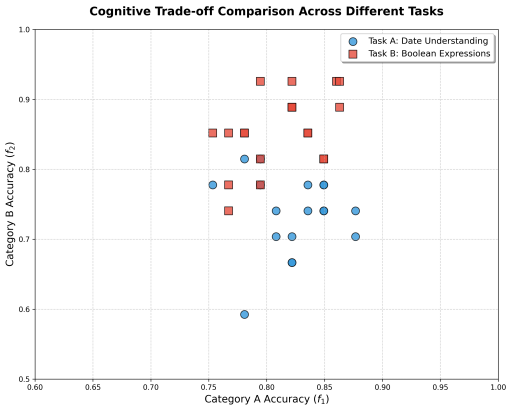


Figure 1: Subgroup accuracies of 20 prompts on two BBH tasks. Similar aggregate scores can mask contrasting subgroup performance.

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

Motivated by this pattern, we evaluate 20 initialization prompts—10 generic and 10 GPT-generated by EvoPrompt (Guo et al., 2024)—on taxonomy-partitioned subsets of two BBH tasks. Figure 1 shows that similar aggregate accuracy can coexist with markedly different subgroup profiles and low worst-group accuracy. This exposes failures hidden by scalar selection and motivates taxonomy-guided multi-objective evaluation.

Addressing this hidden worst-group underperformance requires moving beyond scalar validation toward category-level objectives, which raises three challenges. First, meaningful and mutually exclusive groups must be induced without human annotations or gold category labels, while remaining sufficiently reliable to guide optimization. Second, existing prompt optimizers are primarily designed for scalar feedback and must convert structured multi-objective comparisons into actionable, directional updates in discrete natural-language space. Third, although multi-objective evaluation reveals how candidate transformations affect different reasoning groups, these signals are local to individual updates and must be accumulated across iterations; otherwise, useful search experience cannot consistently guide subsequent optimization.

To address these challenges, we propose MODEPrompt, a reflective multi-objective evolutionary framework for prompt optimization. MODEPrompt integrates three tightly coupled components. First, a taxonomy-guided module partitions the development set into reasoning groups and defines their accuracies as Pareto objectives. Second, a Pareto-guided semantic differential evolution operator contrasts Pareto-front and non-front prompts to extract actionable, task-level mutation patches rather than relying on surface-level variation. Finally, outcome-grounded meta-rule induction aggregates the observed effects of these patches into a long-term meta-rule that guides subsequent iterations. Our contributions are:

- We propose **MODEPrompt**, which induces mutually exclusive reasoning subgroups from development questions without using answer labels and optimizes their accuracies as Pareto objectives, exposing weaknesses hidden by aggregate validation.
- We introduce Pareto-guided semantic differential evolution, deriving directional semantic patches from front/non-front contrasts, together with outcome-grounded meta-rule induction that accumulates trial outcomes and objective changes across iterations.
- Evaluations on diverse benchmarks against APO baselines and across four LLM backbones show improved overall and worst-group performance; taxonomy analyses support stable, interpretable, and performance-relevant groups, while ablations isolate core components.

2 Related Work

Automatic Prompt Optimization. Prompt optimization automatically searches for effective instructions to align model behavior. Early work optimized discrete hard prompts or continuous prompt embeddings (Shin et al., 2020; Deng et al., 2022; Wen et al., 2023; Chen et al., 2024a; Sun et al., 2022; Lester et al., 2021; Li & Liang, 2021). Contemporary APO uses LLMs for generation and selection, textual feedback, meta-prompting, and multi-agent refinement (Zhou et al., 2023; Prasad et al., 2023; Pryzant et al., 2023; Yang et al., 2024; Zhang et al., 2026a; Wang et al., 2024a). ACE instead evolves a persistent context playbook through generation, reflection, and curation (Zhang et al., 2026b). Despite these advances, most APO methods still select prompts by aggregate validation scores, which can obscure disparities across heterogeneous reasoning patterns.

Evolutionary Prompt Search. Evolutionary algorithms offer gradient-free search over discrete prompt spaces, while multi-objective formulations maintain Pareto-optimal diversity (Storn & Price, 1997; Eiben & Smith, 2015; Deb et al., 2002). Recent work uses LLMs to instantiate mutation and crossover over natural-language populations for complex search and discovery (Guo et al., 2024; Lehman et al., 2022; Romera-Paredes et al., 2024; Meyerson et al., 2024). In prompt optimization, EvoPrompt and Evo-Instruct apply genetic algorithms and differential evolution to refine instructions (Guo et al., 2024; Luo et al., 2025). However, their variation operators are predominantly driven by scalar fitness and are not explicitly aligned with heterogeneous task demands. Related methods also explore metaheuristic search and self-referential prompt mutation (Pan et al., 2024; Fernando et al., 2024). More recently, GEPA uses Pareto-style selection to preserve complementary strengths across problem instances (Agrawal et al., 2026; Mouret & Clune, 2015). However, its Pareto frontier

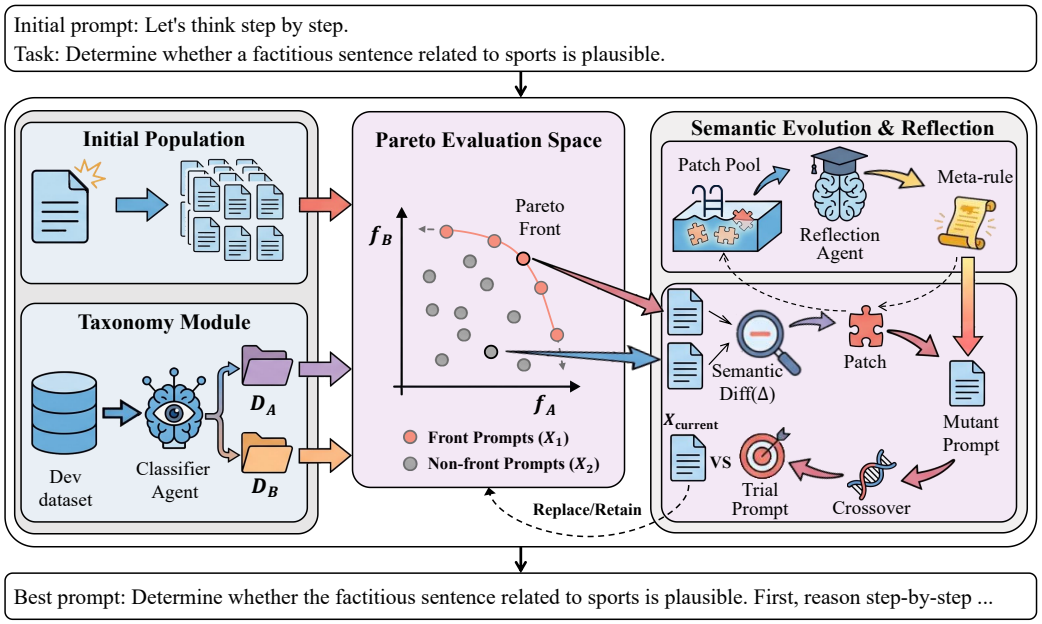


Figure 2: **Overview of MODEPrompt.** Starting from an initial prompt and a development set, MODEPrompt induces cognitive categories to define Pareto objectives. Front/non-front prompt contrasts drive semantic differential mutation via actionable patches, followed by crossover and dominance-based replacement. Outcome-grounded reflections accumulate effective local edits into a long-term meta-rule for subsequent updates.

is organized around instance-level diversity rather than taxonomy-induced reasoning objectives. In contrast, MODEPrompt defines its Pareto geometry through taxonomy-induced reasoning groups, using contrasts between Pareto-front and non-front prompts as directional semantic mutation signals. MEoH evolves heuristic programs under predefined quality and efficiency objectives (Yao et al., 2025). MODEPrompt does not claim the first use of LLM-based multi-objective optimization; its distinction is an objective-construction-to-language-editing loop that induces task-conditioned subgroup objectives from unlabeled question heterogeneity, converts front/non-front behavioral differences into semantic prompt edits, and derives a meta-rule from accepted or rejected edits and their subgroup-vector outcomes.

Reflective Prompt Optimization. Natural-language reflection provides richer feedback than scalar evaluation. Prior studies show that LLMs can iteratively self-debug, refine outputs, and accumulate long-term experience through self-feedback (Madaan et al., 2023; Shinn et al., 2023; Chen et al., 2024b; Bai et al., 2022). Voyager and ExpeL reflect on past trials to build reusable skill libraries or rule sets that improve subsequent performance (Wang et al., 2023; Zhao et al., 2024). Similarly, MARS extracts refinement trajectories from Socratic feedback (Zhang et al., 2026a), while ReflectivePrompt incorporates memory to guide evolutionary updates (Zhuravlev et al., 2025). GEPA also shows that natural-language feedback can improve search beyond sparse reward signals (Agrawal et al., 2026). While these methods improve prompt edits, reflection is usually used as a localized editing or operator-improvement signal rather than being tied to multi-objective dominance. MODEPrompt instead grounds reflection in the observed outcomes of Pareto-guided semantic patches, allowing long-term meta-rules to capture transformations that remain effective across diverse reasoning groups.

3 Taxonomy-guided Pareto Evolution for Prompt Optimization

3.1 Overview

MODEPrompt is a reflective multi-objective evolutionary framework for automatic prompt optimization. To address worst-group underperformance hidden by scalar validation, MODEPrompt reformulates prompt evaluation as a multi-objective search over task-conditioned reasoning sub-

spaces. As illustrated in Figure 2 and Algorithm 1 in Appendix B, the framework couples objective decomposition, Pareto-guided search, and outcome-grounded reflection. First, a taxonomy module partitions the development set into distinct reasoning categories, defining category-level Pareto objectives. Guided by this objective geometry, semantic differential evolution extracts actionable mutation patches from contrasts between Pareto-front and non-front prompts. Finally, an outcome-grounded reflection mechanism accumulates the observed effects of these differential moves into long-term meta-rules, refining the mutation strategy across iterations.

3.2 Taxonomy-guided Objective Decomposition

Taxonomy Induction. To mitigate the limitations of scalar evaluation in prompt optimization, we construct a task-adaptive objective decomposition by partitioning the development set into cognitively distinct subsets. Specifically, let $D_{\text{sub}} \subset D_{\text{dev}}$ denote a sampled subset of input questions, and let $\mathcal{T}$ denote the task description. We induce a taxonomy by generating category definitions and a decision boundary:

$$(\mathcal{C}_A, \mathcal{C}_B, \mathcal{R}) = \Phi(\mathcal{T}, D_{\text{sub}}), \tag{1}$$

where $\Phi(\cdot)$ denotes a taxonomy induction operator, $\mathcal{C}_A$ and $\mathcal{C}_B$ are two category definitions, and $\mathcal{R}$ is a decision rule that distinguishes between them.

The induction process operates solely on input questions without category annotations and without using answer labels during taxonomy induction, ensuring that the resulting taxonomy captures differences in problem characteristics (e.g., reasoning complexity, structural patterns) rather than solution correctness. The induced categories are mutually exclusive and cognitively distinct, enabling the decomposition of the task into complementary subspaces for optimization.

Rule-based Assignment. Given the induced taxonomy, we assign each example in the full development set D_{dev} to one of the two categories by applying the decision rule $\mathcal{R}$ to its input question. Formally, the assignment is defined as:

$$\pi_t(x; \mathcal{C}_A, \mathcal{C}_B, \mathcal{R}) \in \{A, B\}, \quad \forall x \in D_{\text{dev}}, \tag{2}$$

$$D_A = \{x \mid \pi_t(x) = A\}, \quad D_B = \{x \mid \pi_t(x) = B\}, \tag{3}$$

where D_A and D_B together form a partition of D_{dev}, i.e., $D_A \cup D_B = D_{\text{dev}}$ and $D_A \cap D_B = \emptyset$. Here, $\pi_t(\cdot)$ denotes a rule-based classification function operating solely on the input question x. Based on this partition, we define the multi-objective evaluation of a prompt p as:

$$F(p) = \big(f_A(p), f_B(p)\big), \tag{4}$$

where $f_A(p)$ and $f_B(p)$ denote the prediction accuracy of p on D_A and D_B, respectively. This formulation transforms prompt optimization into a structured multi-objective problem, providing the basis for subsequent Pareto-guided evolutionary optimization.

3.3 Pareto-guided Semantic Differential Evolution

Pareto-structured Search Space. Building on the taxonomy-induced multi-objective formulation $F(p) = (f_A(p), f_B(p))$, MODEPrompt organizes the prompt population P according to Pareto dominance instead of scalar validation ranking. For two prompts $p_i, p_j \in P$, we say that p_i dominates p_j, denoted by $p_i \succ p_j$, if

$$f_A(p_i) \geq f_A(p_j), \quad f_B(p_i) \geq f_B(p_j), \tag{5}$$

and at least one of the inequalities is strict. The Pareto front is then defined as

$$P^* = \{p \in P \mid \nexists q \in P \text{ s.t. } q \succ p\}. \tag{6}$$

This partition induces a distinct contrast between Pareto-front individuals and non-front individuals. Front prompts encode prompting behaviors that remain competitive across both cognitive subspaces, whereas non-front prompts expose inferior variants under the same objectives. MODEPrompt uses this contrast not only for selection, but also as the source of directional variation for semantic differential mutation.

Semantic Differential Mutation. Classical differential evolution constructs directional updates through numerical differences of the form $y = a + F(b - c)$ in continuous vector spaces (Storn & Price, 1997). In prompt optimization, however, prompts reside in a discrete natural language space, where such arithmetic is not directly applicable. Prior DE-based prompt optimizers approximate this idea by using LLMs to recombine donor prompts in natural language space (Guo et al., 2024). Such variations can capture surface-level textual differences, but provide limited guidance on how prompts should be modified to improve task behavior under the optimization objectives.

MODEPrompt instead lifts the differential from lexical variation to a task-level semantic differential patch. Guided by the Pareto structure, we construct a contrastive pair by sampling a Pareto-front prompt $X_1 \sim P^*$ and a non-front prompt $X_2 \sim P \setminus P^*$. A semantic differential operator Ψ_Δ, instantiated by LLMs, extracts the transformation signal:

$$\Delta_{\text{sem}} = \Psi_\Delta\big(X_1, X_2, F(X_1), F(X_2); r_{\text{long}}\big), \tag{7}$$

where r_{long} denotes the accumulated long-term reflective rule. The resulting Δ_{sem} is not a set of words to be swapped, but an actionable task-logic patch that captures the structural or reasoning-level behavior present in X_1 but absent or deficient in X_2.

The complete operator prompt, including its output constraints, is provided in Appendix E.

The semantic patch serves as the semantic analog of the differential term in natural language space. To exploit the robust structural information already attained by the population, we sample a base prompt $X_{\text{base}} \sim P^*$ from the Pareto front. MODEPrompt then generates a mutant prompt through a patch-conditioned mutation operator:

$$p_{\text{mut}} = \mathcal{M}\big(X_{\text{base}}, \Delta_{\text{sem}}; r_{\text{long}}\big), \tag{8}$$

where $\mathcal{M}(\cdot)$ integrates the semantic patch into the base prompt. This formulation translates the numerical update of classical DE into a language-mediated semantic gradient, shifting prompt evolution from surface-level variation to structured, direction-aware transformation in semantic space.

Dominance-based Crossover and Selection. The mutant prompt p_{mut} represents a Pareto-guided semantic update, but directly replacing the current individual may discard task instructions already encoded in the original prompt. MODEPrompt therefore applies a crossover operator that recombines the current prompt p_i and the mutant prompt under the guidance of the extracted patch:

$$p_{\text{trial}} = \mathcal{C}\big(p_i, p_{\text{mut}}, \Delta_{\text{sem}}\big), \tag{9}$$

where $\mathcal{C}(\cdot)$ denotes an LLM-instantiated crossover operator. The trial prompt is then evaluated on the same taxonomy-induced objectives, yielding $F(p_{\text{trial}}) = (f_A(p_{\text{trial}}), f_B(p_{\text{trial}}))$.

Drawing inspiration from classical multi-objective evolutionary algorithms (Deb et al., 2002), selection is governed by Pareto dominance rather than aggregate validation accuracy. The current individual is replaced only when the trial prompt dominates it:

$$p_i = \begin{cases} p_{\text{trial}}, & \text{if } p_{\text{trial}} \succ p_i, \\ p_i, & \text{otherwise.} \end{cases} \tag{10}$$

This criterion accepts a candidate only if it improves at least one cognitive subspace without degrading the other, preventing updates that raise performance on one category by sacrificing another. Since every semantic differential move is evaluated under these category-level objectives, its success or failure forms an interpretable learning signal that can be accumulated into memory to guide subsequent mutations.

Final Prompt Selection. Pareto dominance is used throughout optimization to preserve prompts with different subgroup trade-offs. Deployment, however, requires a single prompt. After the final iteration, we therefore apply a separate preference-based model-selection step over the final Pareto front P_T^* and choose the candidate with the highest aggregate development-set accuracy:

$$p^* = \arg\max_{p \in P_T^*} \frac{|D_A| f_A(p) + |D_B| f_B(p)}{|D_A| + |D_B|}. \tag{11}$$

This selection uses only the development partition and is performed after Pareto-guided exploration; neither test labels nor test accuracy participate in prompt selection.

3.4 Outcome-grounded Meta-rule Induction

Differential Outcome Evidence. Natural language reflection has been used to improve prompt updates across iterations (Zhuravlev et al., 2025; Agrawal et al., 2026). In MODEPrompt, reflection is grounded in the observed outcomes of semantic differential moves under taxonomy-induced Pareto objectives. At iteration t, each individual p_i undergoes a mutation–crossover update that produces a local semantic modification $\Delta_{\text{sem}}^{t,i}$ and a resulting trial prompt p_{trial}. We summarize this transition as outcome-grounded evidence:

$$\begin{aligned} e_{t,i} &= \big(\Delta_{\text{sem}}^{t,i}, F(p_i), F(p_{\text{trial}}), z_{t,i}\big), \\ z_{t,i} &= \mathbb{I}[p_{\text{trial}} \succ p_i]\,. \end{aligned} \tag{12}$$

where $z_{t,i}$ indicates whether the local patch leads to a Pareto-accepted replacement. This formulation associates each instance-level transformation with its observed effectiveness under the objective structure, enabling MODEPrompt to distinguish beneficial task-level edits from those that fail or induce unfavorable trade-offs.

Meta-rule Induction from Hierarchical Evidence. While each $\Delta_{\text{sem}}^{t,i}$ captures a local modification, MODEPrompt induces a higher-level meta-rule by combining the current iteration evidence with previously accumulated meta-rules. Let

$$\mathcal{E}_t = \{e_{t,i}\}_{i=1}^{|P|}, \qquad m_t = \sum_{i=1}^{|P|} z_{t,i}, \tag{13}$$

where $\mathcal{E}_t$ collects transformation–outcome pairs within iteration t. The long-term meta-rule is updated as

$$r_{\text{long}}^{t} = \Omega\big(\{r_{\text{long}}^{\tau}\}_{\tau=1}^{t-1}, \mathcal{E}_t, m_t\big). \tag{14}$$

This hierarchical design separates local transformation evidence from global rule abstraction: instance-level patches are first summarized within each iteration, and the resulting meta-rules are recursively refined across iterations. The induced rule captures consistent patterns supported by successful or partially effective patches, while suppressing noisy or non-generalizable edits. Rather than operating on prompt text directly, r_{long}^{t} serves as a learned prior that guides how semantic differentials are extracted and applied in subsequent updates.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate MODEPrompt on 12 tasks from two widely used reasoning benchmarks: BIG-bench Hard (BBH) (Suzgun et al., 2023) and Massive Multitask Language Understanding (MMLU) (Wang et al., 2024b). Specifically, we use six tasks from each benchmark and split every task into a development set for prompt optimization and a held-out test set for final evaluation. The development set is used only for taxonomy induction, prompt evolution, and model selection, while all reported main results are computed on the test set. Detailed task names and dataset statistics are provided in Appendix A.1.

Evaluation Protocol and Hyperparameters. All methods are evaluated under the same target LLM, development/test split, and accuracy metric. We use DeepSeek-V3.2 as the target LLM and set the generation temperature to 0.5. For each task, 20% of the data is used as the development set and the remaining 80% as the held-out test set. Except for CoT-ZS, all methods are evaluated under a 2-shot setting. MODEPrompt uses population size $N = 10$ and a maximum of $T = 10$ iterations, with prompt evaluation performed by the category-level objective vector $F(p) = (f_A(p), f_B(p))$. After optimization, the single prompt used for held-out evaluation is selected from the final Pareto front using aggregate development-set accuracy as defined in Eq. (11). We compare with manual prompting baselines and representative APO methods, including the recent GEPA and ACE methods. Detailed baseline descriptions and implementation settings are provided in Appendix A.2.

Table 1: Main results on BBH and MMLU. We report test accuracy (%). The best result is bolded and the second-best result is underlined. Ties are all marked.

Method	BBH						MMLU						Avg.
	B.E.	**F.F.**	**G.S.**	**R.N.**	**S.U.**	**D.U.**	**C.B.**	**C.M.**	**E.E.**	**W.H.**	**H.A.**	**M.T.**	
Original	86.50	69.50	37.50	86.00	82.50	82.00	93.97	81.29	78.45	90.53	67.04	93.62	79.08
CoT(ZS)	86.00	65.00	46.50	52.50	62.00	50.50	95.69	79.86	79.31	89.47	77.09	90.96	72.91
CoT(FS)	87.50	71.00	45.00	54.00	77.00	62.50	**96.55**	75.54	80.17	87.36	63.69	92.02	74.36
APE	77.00	66.00	48.00	87.50	85.50	83.00	**96.55**	76.26	82.76	90.53	79.33	92.02	80.37
ProTeGi	82.50	66.00	39.50	85.00	80.50	87.50	95.69	79.86	82.76	83.68	75.42	92.55	79.25
EvoPrompt	76.50	65.50	42.00	70.00	78.50	75.00	92.24	76.26	81.89	92.63	75.42	82.98	75.74
OPRO	87.00	70.50	37.50	86.00	79.00	80.50	**96.55**	**82.73**	81.03	91.05	82.12	92.02	80.50
MARS	88.50	71.50	44.00	87.00	84.50	81.00	93.97	80.57	81.89	92.10	78.77	94.15	81.50
ACE	88.50	56.00	48.00	87.00	83.00	85.00	93.97	82.01	**87.93**	93.16	81.56	93.62	81.65
GEPA	90.50	70.50	45.00	85.00	80.00	82.00	**96.55**	82.01	86.21	93.68	79.89	93.09	82.04
Ours	**96.00**	**76.00**	**51.50**	**90.00**	**87.00**	**92.50**	95.69	**82.73**	**87.93**	**94.21**	**82.68**	**95.74**	**86.00**

Table 2: Ablation results on six BBH tasks, reported as test accuracy (%).

Method	**B.E.**	**F.F.**	**G.S.**	**R.N.**	**S.U.**	**D.U.**	**Avg.**
MODEPrompt	**96.00**	**76.00**	**51.50**	**90.00**	**87.00**	**92.50**	**82.17**
w/o *Pareto*	90.50	72.50	46.50	89.50	84.50	86.50	78.33
w/o *Tax*	93.50	74.50	43.50	87.50	85.00	82.00	77.67
w/o *Meta-rule*	92.50	73.50	50.50	81.00	84.00	82.00	77.25
w/o *Patch*	89.50	70.50	49.00	83.50	83.50	82.00	76.33

4.2 Main Results

Overall performance. Table 1 reports test accuracy on BBH and MMLU, including reproduced results for ACE and GEPA. MODEPrompt achieves the highest average performance across all 12 tasks, reaching 86.00% and exceeding MARS, ACE, and GEPA by 4.50, 4.35, and 3.96 percentage points, respectively. It outperforms MARS on all 12 tasks, ACE on 11 tasks with one tie, and GEPA on 11 tasks. The gains are particularly clear on BBH, where MODEPrompt obtains the best result on all six tasks and improves the average accuracy from 76.08% to 82.17% over the strongest baseline. These results show that conditioning mutation and reflection on taxonomy-induced Pareto objectives remains effective against both established and recent APO baselines.

Performance across benchmark types. On MMLU, MODEPrompt also achieves the highest average accuracy, improving over GEPA from 88.57% to 89.83%. The smaller gain compared with BBH is expected: several MMLU tasks already have high baseline accuracy, leaving less optimization margin, and their taxonomy-induced subsets may be less separable because examples share the same domain-specific knowledge space. Smaller development sets can also make category-level accuracy estimates coarser, causing more prompts to share identical objective values during evolution. Despite these constraints, MODEPrompt obtains the best or tied-best result on five of six MMLU tasks.

4.3 Ablation Study

We conduct ablations to examine the contributions of MODEPrompt's core components. We consider four variants: (i) **w/o Pareto** retains the induced taxonomy, subgroup evaluations, and other search settings, but replaces Pareto comparisons with the scalar worst-group objective $s(p) = \min\{f_A(p), f_B(p)\}$. Accordingly, editing contrasts high- and low-$s(p)$ prompts, replacement requires a higher $s(p)$, and final selection maximizes $s(p)$. (ii) **w/o Tax** removes taxonomy-guided objective decomposition and evaluates prompts using aggregate accuracy, while retaining semantic patching and meta-rule induction under scalar feedback. (iii) **w/o Meta** retains taxonomy-guided Pareto objectives and semantic patches but disables cross-iteration meta-rule accumulation. (iv) **w/o Patch** replaces explicit semantic patches with direct LLM-based prompt variation and consequently removes the patch-dependent meta-rule mechanism.

Table 3: Held-out performance on taxonomy-induced groups across 12 tasks (%).

Method	$f_A \uparrow$	$f_B \uparrow$	Overall ↑	Worst-group ↑	Gap ↓
CoT(FS)	86.01	74.88	82.30	74.77	11.35
MARS	85.21	75.95	82.45	75.19	10.78
ACE	85.40	78.21	82.66	76.01	11.59
GEPA	86.01	78.41	82.93	77.11	10.21
MODEPrompt	**87.37**	**79.17**	**84.48**	**78.49**	**9.55**

Table 4: Blinded human evaluation of the induced taxonomies (%).

Task	n	Human–Human	Agreed n	Taxonomy–Human
B.E.	50	96.00	48	83.33
D.U.	50	96.00	48	93.75
F.F.	50	92.00	46	84.78
Pooled	150	94.67	142	87.32

As shown in Table 2, MODEPrompt achieves the highest accuracy on all six tasks. Replacing Pareto optimization with worst-group scalarization reduces average accuracy from 82.17% to 78.33%, a 3.84-point drop, supporting the benefit of Pareto optimization over direct worst-group scalarization. The remaining variants obtain 76.33–77.67%, indicating the individual contributions of taxonomy-guided objectives, cross-iteration meta-rule accumulation, and the patch-based editing pathway. Because **w/o Patch** also removes the patch-dependent meta-rule mechanism, its result reflects the combined editing-and-reflection pathway rather than isolating semantic patches alone.

4.4 Evaluating the Induced Taxonomy

Performance relevance. To evaluate whether the induced taxonomy captures performance-relevant heterogeneity beyond the development set, we freeze the category definitions and decision rule, assign held-out questions to the resulting groups using only their question text, and report performance separately on these groups. Worst-group accuracy is defined as $\min(f_A, f_B)$, and all metrics are macro-averaged across the 12 tasks. All methods are independently evaluated under the same 2-shot setting, with results averaged over three stochastic evaluations.

MODEPrompt achieves the highest f_A, f_B, overall accuracy, and worst-group accuracy, while maintaining the smallest subgroup gap. Relative to the strongest baseline, GEPA, it improves overall and worst-group accuracy by 1.55 and 1.38 points, respectively, and reduces the gap by 0.66 points. These simultaneous gains support the induced groups as a performance-relevant optimization signal.

Human interpretability. We conduct a blinded study on 150 development questions from Boolean Expressions, Date Understanding, and Formal Fallacies. Non-author annotators independently classify each question using only the task description, induced category definitions, decision rule, and question text, while answers and MODEPrompt assignments are withheld. Table 4 reports human agreement and agreement between the induced taxonomy and human consensus.

Pooled human–human agreement reaches 94.67% (Cohen's $\kappa = 0.884$). On the 142 questions with annotator consensus, the induced taxonomy agrees with human consensus on 87.32% ($\kappa = 0.720$). These results support human-interpretable distinctions without implying a definitive or psychologically grounded task taxonomy.

Induction stability. We conduct five independent induction runs per task. Each run resamples D_{sub}, re-induces the category definitions and decision rule, and classifies the same development set. We quantify partition consistency using the Adjusted Rand Index (ARI), where 0 denotes chance-level agreement and 1 denotes identical partitions. The resulting mean ARI is **0.697 on BBH**, **0.773 on MMLU**, and **0.735 overall**, with **9 of 12 tasks** attaining a mean ARI of at least 0.70. These results indicate reproducible taxonomy induction across independently sampled subsets and stochastic

Table 5: Mean test accuracy (%) across six BBH tasks and four target LLM backbones.

Method	DeepSeek -V3.2	DeepSeek -V4-Flash	GPT -5.4	Qwen -3.5-27B	Avg.
CoT(ZS)	60.42	69.08	81.17	67.25	69.48
MARS	76.08	73.92	84.25	70.33	76.15
MODEPrompt	**82.17**	**78.83**	**85.67**	**81.25**	**81.98**

Table 6: Mean optimization cost per method-native optimization cycle across three BBH tasks.

Method	Time (s)	Tokens (k)	Calls
MODEPrompt	712.28	242.81	698.00
MARS	760.63	102.30	287.78
EvoPrompt	710.71	231.94	677.00
GEPA	493.67	299.72	663.00
ACE	1757.44	459.25	202.67

generation. ARI measures reproducibility rather than semantic validity or a ground-truth taxonomy; full per-task results and end-to-end variability are reported in Appendix C.2.

4.5 Cross-model Generalization

MODEPrompt demonstrates strong cross-model generalization, maintaining consistent performance gains across diverse LLM backbones. Beyond DeepSeek-V3.2, we evaluate the complete framework on DeepSeek-V4-Flash, GPT-5.4, and Qwen3.5-27B using all six BBH tasks. As shown in Table 5, MODEPrompt consistently outperforms CoT and MARS across all evaluated backbones, preserving strong performance on larger or structurally different LLMs. These results indicate that MODEPrompt remains effective when the target LLM changes, supporting the applicability of MODEPrompt across diverse LLM backbones.

4.6 Computational Cost

Table 6 reports the mean cost of one method-native optimization cycle across Boolean Expressions, Formal Fallacies, and Date Understanding, excluding held-out inference. MODEPrompt's token consumption is of the same order of magnitude as MARS and remains close to EvoPrompt, while its wall-clock time is comparable to both methods. Its modest overhead over EvoPrompt reflects the additional category-conditioned evaluation, front/non-front semantic comparison, and reflective meta-rule induction. Under their respective method-native optimization cycles, ACE and GEPA consume more tokens but achieve lower overall and subgroup performance, indicating that greater token consumption alone does not necessarily yield better results.

5 Conclusion

We introduced MODEPrompt, a taxonomy-guided Pareto evolutionary framework for automatic prompt optimization. MODEPrompt decomposes scalar validation into taxonomy-defined subgroup objectives and combines Pareto-guided semantic differential evolution with outcome-grounded meta-rule induction. Experiments across reasoning benchmarks, APO baselines, and multiple LLM backbones demonstrate improvements in overall and worst-group performance, while taxonomy analyses support stable, interpretable, and performance-relevant distinctions. Results on AppWorld further suggest the applicability of MODEPrompt to interactive tool-use settings.

Limitations. MODEPrompt incurs additional evaluation and LLM-call overhead, and its induced taxonomy depends on the development questions and underlying LLM. Pairwise parent–trial replacement may discard population-level non-dominated candidates; broader agentic evaluation and comparisons with selection methods such as NSGA-II remain future work.

AI USE STATEMENT

Generative AI tools were used for language polishing and limited coding assistance. The authors reviewed and verified the manuscript content, code, and reported experimental results, and take responsibility for the final content. No AI system is listed as an author.

ETHICS STATEMENT

The taxonomy interpretability analysis uses non-author annotators in a blinded classification protocol. Annotators receive the task description, induced category definitions, decision boundary, and question text, but not the answers or MODEPrompt assignments. We report only aggregate agreement statistics and make no claim that the induced categories constitute a unique cognitive taxonomy.

REPRODUCIBILITY STATEMENT

The paper reports dataset splits, target models, prompt settings, optimization hyperparameters, and final development-only selection in Section 4.1 and Appendix A.2. Algorithm 1 specifies the optimization procedure, and the appendix provides taxonomy definitions, operator templates, additional analyses, and optimized prompts. An anonymized implementation is linked in the abstract.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

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.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. Instructzero: Efficient instruction optimization for black-box large language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML*, pp. 6503–6518, 2024a.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *The Twelfth International Conference on Learning Representations, ICLR*, 2024b.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3369–3391, 2022. doi: 10.18653/v1/2022.emnlp-main.222.

Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*. Springer, 2015. doi: 10.1007/978-3-662-44874-8.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Forty-first International Conference on Machine Learning, ICML 2024*, pp. 13481–13544, 2024.

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

Tatsunori B. Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness without demographics in repeated loss minimization. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, pp. 1934–1943, 2018.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning, ICML*, pp. 5637–5664, 2021.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35: 22199–22213, 2022.

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. Evolution through large models. *CoRR*, abs/2206.08896, 2022.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, 2021. doi: 10.18653/v1/2021.emnlp-main.243.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021. doi: 10.18653/v1/2021.acl-long.353.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023. doi: 10.1145/3560815.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jian-Guang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. In *The Thirteenth International Conference on Learning Representations, ICLR*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

Elliot Meyerson, Mark J Nelson, Herbie Bradley, Adam Gaier, Arash Moradi, Amy K Hoover, and Joel Lehman. Language model crossover: Variation through few-shot prompting. *ACM Transactions on Evolutionary Learning*, 4(4):1–40, 2024. doi: 10.1145/3694791.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Rui Pan, Shuo Xing, Shizhe Diao, Wenhe Sun, Xiang Liu, Kashun Shum, Jipeng Zhang, Renjie Pi, and Tong Zhang. Plum: Prompt learning using metaheuristics. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2177–2197, 2024. doi: 10.18653/v1/2024.findings-acl.129.

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*, pp. 3845–3864, 2023. doi: 10.18653/v1/2023.eacl-main.277.

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*, pp. 7957–7968, 2023. doi: 10.18653/v1/2023.emnlp-main.494.

Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–7, 2021. doi: 10.1145/3411763.3451760.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6.

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4222–4235, 2020. doi: 10.18653/v1/2020.emnlp-main.346.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adria Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997. doi: 10.1023/A:1008202821328.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pp. 20841–20855, 2022.

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*, pp. 13003–13051, 2023. doi: 10.18653/v1/2023.findings-acl.824.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024. doi: 10.18653/v1/2024.acl-long.850.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

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. In *The Twelfth International Conference on Learning Representations, ICLR*, 2024a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024b.

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.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *Advances in Neural Information Processing Systems*, 36:51008–51025, 2023.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, volume 2024, pp. 12028–12068, 2024.

Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. Multi-objective evolution of heuristic using large language model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(25):27144–27152, 2025. doi: 10.1609/aaai.v39i25.34922.

JD Zamfirescu-Pereira, Richmond Y Wong, Bjoern Hartmann, and Qian Yang. Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–21, 2023. doi: 10.1145/3544548.3581388.

Jian Zhang, Zhangqi Wang, Haiping Zhu, Kangda Cheng, Kai He, Bo Li, Qika Lin, Jun Liu, and Erik Cambria. Mars: Multi-agent adaptive reasoning with socratic guidance for automated prompt optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(19):16307–16315, 2026a. doi: 10.1609/aaai.v40i19.38668.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38 (17):19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936.

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

Viktor N. Zhuravlev, Artur R. Khairullin, Ernest A. Dyagin, Alena N. Sitkina, and Nikita I. Kulin. Reflectiveprompt: Reflective evolution in autoprompting algorithms. *CoRR*, abs/2508.18870, 2025.

A Experimental Details

A.1 Dataset Details

We evaluate MODEPrompt on 12 reasoning tasks from BBH (Suzgun et al., 2023) and MMLU (Wang et al., 2024b). The BBH tasks cover diverse reasoning abilities, including Boolean logic, formal reasoning, geometric interpretation, commonsense plausibility, date reasoning, and humorous name transformation. The MMLU tasks focus on domain-specific knowledge and reasoning across biology, medicine, engineering, history, aging studies, and marketing. Each task is split into development and test sets with a 20/80 ratio. Table 7 provides detailed statistics for all tasks.

Table 7: Dataset statistics. *Abbr.* denotes the task abbreviations used in the result tables, and *Type* indicates the prediction format.

Benchmark	Task	Abbr.	Type	Label Space	Total
BBH	Boolean Expressions	B.E.	Short answer	$\{\text{True}, \text{False}\}$	250
	Formal Fallacies	F.F.	Short answer	$\{\text{valid}, \text{invalid}\}$	250
	Geometric Shapes	G.S.	Choice	$\{A, \ldots, K\}$	250
	Ruin Names	R.N.	Choice	$\{A, \ldots, D\}$	250
	Sports Understanding	S.U.	Short answer	$\{\text{yes}, \text{no}\}$	250
	Date Understanding	D.U.	Choice	$\{A, \ldots, F\}$	250
MMLU	College Biology	C.B.	Choice	$\{A, \ldots, D\}$	144
	College Medicine	C.M.	Choice	$\{A, \ldots, D\}$	173
	Electrical Engineering	E.E.	Choice	$\{A, \ldots, D\}$	145
	High School World History	W.H.	Choice	$\{A, \ldots, D\}$	237
	Human Aging	H.A.	Choice	$\{A, \ldots, D\}$	223
	Marketing	M.T.	Choice	$\{A, \ldots, D\}$	234

A.2 Baselines and Implementation Details

This section provides additional details on the baselines. We compare MODEPrompt with two groups of methods: manually designed prompting strategies and automatic prompt optimization (APO) baselines. Manual baselines include Original, which uses the task description as the instruction for evaluation; CoT-ZS, which appends the zero-shot Chain-of-Thought instruction "Let's think step by step." (Kojima et al., 2022); and CoT-FS, which prepends two demonstration examples to the query.

For APO baselines, we include APE (Zhou et al., 2023), OPRO (Yang et al., 2024), ProTeGi (Pryzant et al., 2023), EvoPrompt (Guo et al., 2024), MARS (Zhang et al., 2026a), GEPA (Agrawal et al., 2026), and ACE (Zhang et al., 2026b). APE follows a generation-and-selection paradigm, ProTeGi uses natural-language textual gradients and beam search, OPRO treats an LLM as the optimizer, EvoPrompt instantiates evolutionary mutation and crossover, and MARS uses multi-agent Socratic refinement. GEPA preserves complementary strengths through instance-level Pareto selection and reflective feedback. ACE accumulates a playbook of strategies and insights; for the reasoning benchmark comparison, its final playbook is converted into the task-level prompt used for held-out evaluation. Each development example is treated as one optimization instance for ACE and GEPA. All APO baselines are initialized with the same CoT instruction, "Let's think step by step.", and use the same development/test split and target model. For the main 12-task evaluation, prompt optimization and inference use DeepSeek-V3.2 with temperature 0.5.

B Algorithm of MODEPrompt

To formalize the evolutionary search process discussed in the main text, we provide the complete pseudo-code for the MODEPrompt framework in Algorithm 1. This algorithm details the end-to-end taxonomy-guided Pareto evolution process, formally defining the multi-objective evaluation, semantic differential mutation and crossover, Pareto dominance-based selection, outcome-grounded meta-rule induction across T iterations, and the final development-only preference selection.

Algorithm 1 MODEPROMPT: Taxonomy-Guided Pareto Evolution for Prompt Optimization

Require: Task description $\mathcal{T}$, development set D_{dev}, initial prompt p_{init}, population size N, and maximum number of iterations T
Ensure: Selected prompt p^* and final Pareto-optimal prompt set P_T^*
Induce taxonomy $(\mathcal{C}_A, \mathcal{C}_B, \mathcal{R})$ from sampled development questions using Eq. (1)
Partition D_{dev} into D_A and D_B by applying $\mathcal{R}$ to the input questions
Initialize population $P_0 \leftarrow \text{Init}(p_{\text{init}}, N)$ and meta-rule $r_{\text{long}}^0 \leftarrow \emptyset$
Evaluate every $p \in P_0$ under $F(p) = (f_A(p), f_B(p))$
for $t = 1$ **to** T **do**
  Extract Pareto front P_{t-1}^* and non-front pool P_{t-1}^-
  Initialize $P_t \leftarrow \emptyset$ and $\mathcal{E}_t \leftarrow \emptyset$
  for each prompt $p_i \in P_{t-1}$ **do**
    Sample $X_1 \sim P_{t-1}^*$, $X_2 \sim P_{t-1}^-$, and $X_{\text{base}} \sim P_{t-1}^*$
    Extract semantic patch $\Delta_{\text{sem}}^{t,i}$ from (X_1, X_2) using Eq. (7)
    Generate p_{mut} from $(X_{\text{base}}, \Delta_{\text{sem}}^{t,i})$ and recombine it with p_i to obtain p_{trial}
    Evaluate p_{trial} and set $z_{t,i} \leftarrow \mathbb{I}[p_{\text{trial}} \succ p_i]$
    Add outcome evidence $e_{t,i}$ to $\mathcal{E}_t$ according to Eq. (12)
    $P_t \leftarrow P_t \cup \{p_{\text{trial}}\}$ if $z_{t,i} = 1$; otherwise, $P_t \leftarrow P_t \cup \{p_i\}$
  end for
  Update r_{long}^t from $\mathcal{E}_t$
end for
Extract the final Pareto front P_T^* from P_T
Select p^* from P_T^* using only aggregate development-set accuracy, as defined in Eq. (11)
return p^* and P_T^*

Table 8: Inter-group accuracy gaps (%) for taxonomy-defined groups and 10,000 size-matched random partitions.

Method	Taxonomy-defined gap	Random-partition gap
CoT(FS)	11.35	5.36
MARS	10.78	5.32
MODEPrompt	9.55	5.13

C TAXONOMY ROBUSTNESS AND SUPPLEMENTARY ANALYSIS

C.1 PERFORMANCE RELEVANCE AGAINST RANDOM PARTITIONS

To test whether the induced groups capture performance-relevant heterogeneity rather than disparities arising from arbitrary finite-sample partitions, we compare them with size-matched random partitions of the held-out test predictions. For each task and method, we randomly repartition the held-out predictions $10{,}000$ times while preserving the taxonomy-defined Category A/B group sizes. We compute the absolute accuracy gap for each random partition, average it over the $10{,}000$ partitions, and then macro-average the resulting gaps across the 12 reasoning tasks. Table 8 compares these random-partition gaps with those obtained from the taxonomy-defined groups.

The taxonomy-defined groups expose approximately twice the disparity of random partitions under all three methods, supporting performance-relevant heterogeneity beyond arbitrary partitioning. MODEPrompt reduces the taxonomy-defined gap relative to both baselines, while the induced groups remain more performance-differentiated than their size-matched random counterparts.

C.2 REPRODUCIBILITY ACROSS INDUCTION RUNS

For each task, we conduct five independent taxonomy-induction runs. Each run resamples D_{sub}, re-induces the category definitions and decision rule, and classifies the same development set. We compute the Adjusted Rand Index (ARI) over all pairwise combinations of the five resulting partitions

Table 9: Mean pairwise ARI among five independent taxonomy-induction runs.

Task	ARI	Task	ARI
B.E.	0.729	C.B.	0.823
F.F.	0.478	C.M.	0.737
G.S.	0.587	E.E.	0.783
R.N.	0.842	W.H.	0.735
S.U.	0.757	H.A.	0.663
D.U.	0.786	M.T.	0.899
Cross-task mean			0.735

and report the mean value. An ARI of 1 indicates identical partitions, whereas 0 corresponds to chance-level agreement in expectation.

C.3 End-to-End Optimization Variability

The induction repetitions above isolate taxonomy reproducibility rather than variability of the complete optimization pipeline. We therefore additionally conduct three independent end-to-end MODEPrompt runs on three representative BBH tasks. Mean test accuracy and sample standard deviation are 91.50 ± 3.97 on Boolean Expressions, 90.67 ± 1.04 on Ruin Names, and 85.60 ± 1.52 on Sports Understanding. These results quantify complete-pipeline variability under stochastic taxonomy induction, front/non-front sampling, semantic patch generation, and meta-rule induction.

C.4 Taxonomy Definitions and Few-Shot Analysis

MODEPrompt induces task-specific cognitive taxonomies without requiring human annotations. Table 10 presents the Category A and Category B definitions used for the four BBH tasks in the taxonomy-aware few-shot analysis, together with their development-set split statistics. The induced groups can be substantially imbalanced. For example, Category A accounts for only 10% of the development examples in Ruin Names. Under scalar evaluation, poor performance on such a small group can be obscured by aggregate accuracy, motivating category-level evaluation and taxonomy-aware example selection.

Table 10: Taxonomy-induced category definitions and development-set split statistics for the tasks used in the taxonomy-aware few-shot analysis.

Task	**Category A**	**Category B**	**Ratio (D_A:D_B)**
B.E.	**Operator-Density Simple:** Expressions containing only unary and binary operators with no nested structures.	**Structural-Complexity Heavy:** Expressions that contain nested parentheses or deep hierarchical logic.	62%:38%
G.S.	**Single-Primitive Paths:** Paths composed of straight-line commands or a single curve defining one geometric primitive without disjoint segments.	**Composite-Path Figures:** Paths containing multiple disjoint segments or combining multiple straight and curved commands to form a complex figure.	56%:44%
R.N.	**Single-Word Target:** The input name consists of a single word, making the humorous edit a direct, solitary character change.	**Multi-Word Target:** The input name consists of multiple words, requiring analysis of phrase structure to create humor.	10%:90%
D.U.	**Single-Step Date Reference:** The target date is directly stated or easily derived from a single, unambiguous reference date.	**Multi-Step Date Resolution:** Determining the correct date requires resolving conflicting information, relative timeframes, or intermediate calculations.	62%:38%

Table 11: Taxonomy-aware 2-shot demonstrations on selected BBH tasks (%).

Strategy	B.E.	G.S.	R.N.	D.U.	Avg.
$D_A + D_B$	**95.00**	45.00	84.50	**77.00**	**75.38**
$2D_A$	78.50	**46.50**	51.50	71.00	61.88
$2D_B$	89.00	45.00	**85.00**	72.50	72.88

Table 12: Held-out accuracy (%) under different taxonomy granularities.

Task	$K = 2$	$K = 3$	$K = 4$
B.E.	**96.00**	91.00	91.00
F.F.	**76.00**	69.50	68.50
D.U.	**92.50**	80.00	81.50
Macro average	**88.17**	80.17	80.33

Given these category definitions and split ratios, we examine whether covering both taxonomy-defined groups improves few-shot prompting. Using standard Chain-of-Thought, we compare a taxonomy-diverse 2-shot strategy ($D_A + D_B$) with two homogeneous strategies that draw both examples from the same category. Table 11 shows that the diverse strategy obtains the highest average accuracy and avoids the sharp degradation produced by homogeneous sampling on Ruin Names.

C.5 Effect of Taxonomy Granularity

MODEPrompt is not intrinsically restricted to two objectives. We extend the complete optimization pipeline to flat $K = 3$ and $K = 4$ taxonomies under the same experimental settings used for $K = 2$. As shown in Table 12, increasing the number of groups does not improve held-out accuracy in the evaluated low-data setting. Finer taxonomies leave too few development examples in some groups; for example, the $K = 4$ taxonomy for Boolean Expressions produces groups containing 1, 2, 17, and 30 examples. These results support $K = 2$ as a practical granularity–reliability trade-off under the current development-data budget, rather than as a universally optimal taxonomy size.

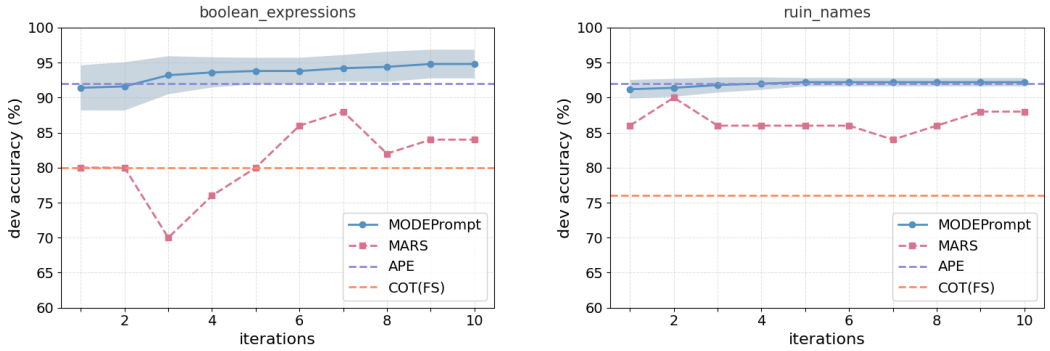
C.6 Convergence and Stability

Figure 3 tracks development accuracy across iterations. MODEPrompt improves toward a stable plateau on both tasks, whereas MARS exhibits larger intermediate fluctuations. The shaded area for MODEPrompt denotes the standard deviation across population members within an iteration.

D Agentic Evaluation on AppWorld

We evaluate whether taxonomy-guided prompt optimization can be instantiated beyond single-turn reasoning on AppWorld (Trivedi et al., 2024), an interactive agent benchmark requiring multi-step planning, Python code generation, API use, execution feedback, and stateful environment interaction. All methods use the same official ReAct execution scaffold, base model, tools, task split, maximum interaction steps, and evaluation protocol. Offline methods are adapted on training data and frozen before evaluation on Test-Normal. MODEPrompt induces its taxonomy from AppWorld task instructions and optimizes candidate prompts with category-level Task Goal Completion objectives; evaluating a candidate therefore requires complete iterative ReAct trajectories rather than single-turn predictions.

MODEPrompt improves the original ReAct agent by 4.2 TGC and 10.7 SGC and improves over GEPA by 2.4 TGC and 5.3 SGC. ACE differs in context representation: it augments the base ReAct prompt with a persistent playbook accumulated from training trajectories, and its unrestricted learned context is more than an order of magnitude larger than MODEPrompt's optimized instruction. We therefore also evaluate a bounded-playbook ACE variant that retains the ACE adaptation procedure while continuously constraining and summarizing its playbook to a comparable order of magnitude. Under



(a) boolean_expressions (b) ruin_names

Figure 3: Convergence and stability on representative development tasks. Static baselines are horizontal references.

Table 13: AppWorld Test-Normal results. TGC and SGC denote task- and scenario-goal completion.

Method	TGC	SGC
ReAct	72.6	50.0
ReAct + GEPA	74.4	55.4
ReAct + ACE (bounded playbook)	76.2	58.9
ReAct + ACE	**79.8**	**69.6**
ReAct + MODEPrompt	76.8	60.7

this deployment-context-controlled setting, MODEPrompt obtains 76.8/60.7 TGC/SGC, compared with 76.2/58.9 for bounded ACE. Unrestricted ACE remains the strongest system; the experiment shows that MODEPrompt extends to interactive tool use and is competitive when deployment-context scale is controlled.

E Prompt Templates for LLM Operators

MODEPrompt relies on LLM operators to execute the taxonomy-guided evolutionary search. Table 14 provides a high-level summary of these operators, detailing their specific purposes within the optimization pipeline and their expected output formats. The complete and exact prompt templates for each operator—including the dynamic variables injected during runtime—are illustrated in the subsequent figures.

Table 14: Summary of LLM operator templates used in MODEPrompt.

Operator	**Purpose**	**Expected Output**
Taxonomy Induction	Generate two category definitions and decision boundary	Category A/B definitions
Taxonomy Assignment	Assign development questions into induced categories	JSON index lists
Semantic Differential	Extract front/non-front reasoning gap	Actionable patch
Semantic Mutation	Inject patch into Pareto-front base prompt	Mutant prompt
Meta-rule Induction	Summarize successful patch outcomes	Long-term meta-rule

Taxonomy Induction Template. The taxonomy induction operator generates two cognitively distinct categories and a decision boundary from the task description and sampled development questions. The exact prompt template is shown in Figure 4.

Prompt Template: Taxonomy Induction

You are an expert dataset taxonomist. Your task is to analyze data samples and create two distinct categories for multi-objective evaluation.

Task Description: `{task_description}`

Sampled Questions (without answers):
`{sampled_examples}`

Requirements:

1. **Cognitive Disparity:** Categories must differ in reasoning difficulty or cognitive demand.
2. **Orthogonality:** Categories must be mutually exclusive with no overlap.
3. **Actionable:** Definitions must be clear enough for deterministic classification based on the question only.
4. **No Answer Leakage:** Category definitions must be based solely on question features, not on answer patterns.
5. **Balanced Potential:** Avoid overly restrictive definitions that would result in very few samples in one category.

Output Format:
Provide a structured response containing the following elements:

- `Category A/B Name`: [Concise descriptive name]
- `Category A/B Definition`: [1–2 sentence distinguishing definition]
- `Decision Boundary`: [Core if-then logic that definitively distinguishes A from B]

Figure 4: The complete LLM prompt template used for taxonomy induction. Variables enclosed in braces, such as `{task_description}` and `{sampled_examples}`, are dynamically populated with task-specific content during execution.

Taxonomy Assignment Template. Given the induced category definitions and decision boundary, the assignment operator classifies each development question into exactly one category. The prompt template used for this deterministic classification is detailed in Figure 5.

Semantic Differential Template. The semantic differential operator extracts the structural or reasoning gap between a Pareto-front prompt and a non-front prompt. It translates this cognitive difference into a concrete, imperative mutation patch. The exact prompt template is illustrated in Figure 6.

Semantic Mutation Template. The semantic mutation operator injects the extracted differential patch into a base prompt. Guided by the long-term meta-rule, this phase forces the integration of new reasoning behaviors. The prompt template is illustrated in Figure 7.

Meta-Rule Induction Template. The meta-rule induction operator acts as the long-term memory mechanism of MODEPrompt. It reviews the evolutionary success rate of the current iteration and summarizes effective short-term patches into a global prior. The prompt template is illustrated in Figure 8.

F Case Study: A Single MODEPrompt Iteration

Figure 9 presents a representative single-iteration trace of MODEPrompt on the boolean_expressions task. Starting from the initial baseline prompts, MODEPrompt first induces two taxonomy-defined cognitive subspaces using only sampled development questions. The resulting categories cleanly separate simpler operator-density cases (Category A) from structurally complex expressions involving nested parentheses (Category B). This decomposition crucially shifts prompt evaluation from a single scalar score into structured category-level objectives.

Prompt Template: Taxonomy Assignment

You are a deterministic classification agent. Classify each data item into exactly one category based on the question only.

Category A ({category_A_name}): {category_A_definition}

Category B ({category_B_name}): {category_B_definition}

Decision Boundary:
{decision_boundary}

Input Questions:
{input_data}

Constraints:

- **Mutually exclusive:** Each item belongs to exactly one category.
- **Exhaustive:** Every [index] must appear in the output.
- **Question-based:** Classify based on question features only.
- **Strict boundary:** Follow the decision boundary logic strictly.
- **No explanations:** Output only valid JSON.

Output Format: (strict JSON, no Markdown)

```
{"A": [list of indices], "B": [list of indices]}
```

Figure 5: The complete LLM prompt template used for taxonomy assignment. Category definitions and the decision boundary obtained from the induction phase are dynamically injected to partition the input data deterministically.

Under these taxonomy-induced objectives $F(p) = (f_A(p), f_B(p))$, MODEPrompt identifies Pareto-front (X_1) and non-front (X_2) individuals. Instead of directly copying textual variations, the semantic differential operator explicitly contrasts these prompts to extract the reasoning mechanism driving the performance gap. In this example, the extracted patch (Δ_{sem}) instructs the model to resolve inner logical structures before applying outer Boolean operators, transforming an implicit step-by-step instruction into an explicit, actionable evaluation procedure.

This semantic patch is then injected into a base prompt and recombined with the current individual through crossover. If the resulting trial prompt is accepted by Pareto dominance—improving one objective without degrading the other—it replaces the original prompt. Finally, MODEPrompt summarizes the successful and partially effective semantic updates from the current iteration into a long-term meta-rule (r_{long}). This trace clearly illustrates how MODEPrompt transforms localized Pareto evidence into reusable evolutionary knowledge, overcoming the limitations of scalar rewards and unconstrained prompt rewriting.

G Optimized Prompts

This section lists the final optimized prompts selected by MODEPrompt for each task. Only the task instruction is shown.

Boolean Expressions
Evaluate the truth value of a random Boolean expression consisting of Boolean constants (True, False) and basic Boolean operators (and, or, not). You must follow a mandatory, step-by-step procedure: decompose the entire expression and evaluate it one logical operator at a time in a clear sequence. Do not rely on implicit or holistic reasoning. Process a single operation, state the result, and then proceed to the next operation until the final truth value is determined.

Table 15: Final optimized prompt for the boolean_expressions task selected by MODEPrompt.

Formal Fallacies
Given a context involving a set of statements, determine whether an argument—presented informally—can be logically deduced from the provided context. Please proceed in a stepwise manner through the following distinct reasoning stages: Stage 1: Context Parsing. Identify and list all explicit premises and logical relationships presented in the context. Stage 2: Argument Mapping. Formalize the structure of the argument to be evaluated, mapping its premises and conclusion. Stage 3: Logical Evaluation. Apply rules of deductive logic to determine if the conclusion necessarily follows from the premises. Output your final determination on whether the argument is deductively valid.

Table 16: Final optimized prompt for the formal_fallacies task selected by MODEPrompt.

Geometric Shapes
Break down the problem into incremental stages for careful consideration.

Table 17: Final optimized prompt for the geometric_shapes task selected by MODEPrompt.

Ruin Names
Given an artist, band, or movie name, identify a one-character edit to the name that changes the meaning of the input and makes it humorous. To do this, you must first execute a structured, stepwise reasoning process. This process should systematically decompose the problem, enumerate possible options, and analyze each step logically. Only after completing this reasoning chain should you produce the final humorous one-character edit.

Table 18: Final optimized prompt for the ruin_names task selected by MODEPrompt.

Sports Understanding
Determine whether the factitious sentence related to sports is plausible. First, reason step-by-step about the scenario before delivering your final judgment.

Table 19: Final optimized prompt for the sports_understanding task selected by MODEPrompt.

Date Understanding
Given a small set of sentences about a particular date, answer the provided question by following this explicit, structured, sequential reasoning framework. You must articulate your reasoning in clear, separated logical stages before delivering the final answer.
Reasoning Stages:
1. Extraction: Identify and extract all explicit date references from the provided sentences. Explicitly articulate the list of extracted dates.
2. Interpretation: Parse and interpret the temporal relationships and contextual information within the sentences. Explicitly articulate the interpreted context and relationships.
3. Synthesis: Synthesize the extracted and interpreted data to establish a coherent timeline or factual understanding. Explicitly articulate the synthesized timeline or fact.
4. Application: Apply this synthesized understanding to directly and precisely address the question.
Begin by stating "Reasoning:" and proceed through each numbered stage explicitly, ensuring the output of each stage is clearly stated before moving to the next. Only after completing all stages, provide your final answer.

Table 20: Final optimized prompt for the date_understanding task selected by MODEPrompt.

College Biology
Given a multiple-choice question on college-level biology, identify the correct biological process, pathway, or concept among the four options. To do this, decompose your reasoning into distinct, labeled, sequential steps. First, analyze the question and each option. Second, apply relevant biological principles. Third, compare and eliminate incorrect options. Finally, conclude with the correct answer.

Table 21: Final optimized prompt for the college_biology task selected by MODEPrompt.

College Medicine
Let's think step by step. Given a multiple-choice question on college-level medicine, including clinical scenarios or psychological frameworks, work through this sequentially, analyzing each stage carefully to determine the most accurate diagnostic or theoretical answer.

Table 22: Final optimized prompt for the college_medicine task selected by MODEPrompt.

Electrical Engineering
Given a multiple-choice question on electrical engineering principles, circuits, or theorems, identify the correct statement or mathematical condition. You must approach this in a systematic, step-by-step manner by following a strict sequence: first, analyze the question stem to identify the core components, principles, or variables; then, analyze each multiple-choice option in the order presented (Option A, then Option B, etc.), evaluating each against the relevant principles and the stem's requirements in a strict, logical sequence. Finally, synthesize your analysis to conclude on the correct answer, and verify this conclusion against the original question.

Table 23: Final optimized prompt for the electrical_engineering task selected by MODEPrompt.

High School World History
Given a multiple-choice question on high school world history, often accompanied by a primary source passage or historical context, analyze the text and identify the correct historical development or fact. Before evaluating the options, decompose your reasoning into discrete, labeled, sequential analytical phases. Begin your response by explicitly stating you will break down the reasoning into steps.

Table 24: Final optimized prompt for the high_school_world_history task selected by MODEPrompt.

Human Aging
Given a multiple-choice question on the biological, psychological, or sociological aspects of human aging, select the correct terminology or concept. To do this, first decompose the question into its core components and discrete, sequential reasoning steps. Explicitly reason through each step in order, methodically analyzing each component, before synthesizing your final answer.

Table 25: Final optimized prompt for the human_aging task selected by MODEPrompt.

Prompt Template: Semantic Differential Extraction

You are an expert in semantic differential evolution for prompt optimization. Below are two prompts for the following task.

Task Name: `{task_name}`
Task Description: `{task_description}`

Better prompt X_1 (from Pareto front, $f_A(X_1)$, $f_B(X_1)$):
`{X_1}`

Worse prompt X_2 (non-front individual, $f_A(X_2)$, $f_B(X_2)$):
`{X_2}`

Global Meta-Rule (r_{long}): `{r_long}`

Objective:
Extract the "semantic differential" ($X_1 - X_2$)—the precise structural or reasoning gap that makes X_1 perform better than X_2. Translate this difference into one concrete, actionable mutation patch.

Critical Requirements:

- **Focus on the Delta:** Explicitly identify the structural mechanism that exists in X_1 but is missing or flawed in X_2 (e.g., a specific constraint, a decision branch, or a reasoning sequence).
- **Action-Oriented Mutation:** Phrase the patch as a direct imperative editing command (e.g., "Add a validation step before...", "Force the model to check...").
- **Behavioral Change Only:** The patch must target task logic and reasoning structure. Strictly avoid stylistic or wording-level polish.

Output Format:
Output only one patch under 30 words, bracketed with `<patch> ...</patch>`. Do not output a full prompt.

Figure 6: The complete LLM prompt template used for the semantic differential operator (Ψ_{Δ}). It explicitly requires the model to isolate behavioral and structural differences between evaluated prompts, producing an actionable patch instead of a full text rewrite.

Marketing
Given a multiple-choice question on marketing concepts, models, or business strategies, identify the correct acronym or theoretical framework by following these steps:
1. Step-by-Step Unpacking: Explicitly unpack any acronyms or theoretical frameworks mentioned in the question. For acronyms, explain what each letter stands for. For theoretical frameworks, outline the core components or stages.
2. Analysis: Use this unpacked understanding to systematically analyze the provided options.
3. Selection: Based on the analysis, identify and select the correct answer.

Table 26: Final optimized prompt for the marketing task selected by MODEPrompt.

Prompt Template: Semantic Mutation

You are a precise semantic mutation agent executing the mutation phase of an evolutionary algorithm. Your task is to generate a mutant prompt by injecting a specific instruction patch into a base prompt, guided by a global meta-rule.

Task Context:
Name: `{task_name}`
Description: `{task_description}`

Global Meta-Rule: `{meta_rule}`
Instruction Patch: `{patch}`
Base Prompt: `{X_base}`

Mutation Execution Rules:

- **Execute the Addition (Base + Patch):** You must forcefully integrate the reasoning behavior described in the Instruction Patch into the Base Prompt. Do not ignore the patch.
- **Global Alignment (Meta-Rule):** The mutated logic must comply with the Global Meta-Rule. If old instructions in the Base Prompt conflict with the patch or the meta-rule, immediately delete or rewrite them.
- **Constraint:** Do not worry about preserving the original phrasing of the Base Prompt. Focus entirely on successfully activating the patch's logic while preserving the task answer requirements.

Output Format:
Output only the mutant prompt text, bracketed with `<prompt> ... </prompt>`.

Figure 7: The complete LLM prompt template used for the semantic mutation operator. The optimizer acts as an instruction executor, applying the directional update (patch) to the base prompt while ensuring compatibility with the long-term meta-rule prior.

Prompt Template: Meta-Rule Induction

You are an expert at improving prompts using reflection. Your task is to summarize useful prompt-editing experience from this iteration.

Task Context:
Name: `{task_name}`
Description: `{task_description}`

Prior Long-Term Reflection:
`{prior_long_term_reflection}`

Short-Term Reflections from this Iteration:
`{new_short_term_reflections}`

Iteration Result:
`{replacement_count}` / `{population_size}` trial prompts replaced their originals.

Task Guidelines:

- Write one concise long-term meta-rule for improving future prompts.
- Prefer patterns supported by successful or partially improved trials.
- Do not repeat the prior reflection unless the new evidence strengthens it.
- Ignore noisy one-off patches that did not help.

Output Format:
Output the concise long-term meta-rule, bracketed with `<meta-rule>` ... `</meta-rule>`.

Figure 8: The complete LLM prompt template used for meta-rule induction. By incorporating the population replacement count as a statistical grounding signal, this operator filters out noisy, one-off patches and accumulates a robust optimization prior.

Initial prompt : Let's think step by step.
Task : Evaluate the truth value of a random Boolean expression consisting of Boolean constants (True, False) and basic Boolean operators (and, or, not).

You are an expert dataset taxonomist. Your task is to analyze data samples and create two distinct categories for multi-objective evaluation.

Sampled Questions (without answers):
not not True and True and not True is ;
True and not True or False or True is ;
not True and (False or True) is …
Output(category_definitions):
"Category A": "Operator-Density Simple","definition": "Expressions containing only unary (not) and binary (and, or) operators with no nested …",
"Category B": "Structural-Complexity Heavy","definition": "Expressions that either contain nested parentheses, or …",
"decision_boundary": "If the question contains any instance of nested parentheses OR has length ≥ 4, then convert it to txt type …"

Input Questions:
not (False or False or True) is
Output:
A (Final will get : "category_mapping": {"A": [0, 1, 4, 5, …], "B": [2, 3, 10, …]}

You are an expert in semantic differential evolution for prompt optimization.

Input (Contrasting Prompts):
Better prompt X_1 (Pareto-front, f_A=95%, f_B=88%):"Evaluate the expression step-by-step. Crucially, always resolve innermost parentheses first before moving to outer operators…"
Worse prompt X_2 (Non-front, f_A=92%, f_B=75%):"Evaluate the expression step-by-step from left to right ..."
Instruction:Extract the "semantic differential" (X_1-X_2)—the precise structural gap that makes X_1 perform better. Translate this into ONE actionable mutation patch.

Output (Patch):
Force the model to explicitly resolve innermost parentheses first, overriding strict left-to-right evaluation.
Base prompt X_{base}:"Evaluate the truth value of the Boolean expression step by step. Carefully apply Boolean operators and provide the final truth value."

Output (Mutant Prompt):
"Evaluate the truth value of the Boolean expression step by step. First, identify and resolve the innermost parentheses before applying any outer operators. Do not evaluate strictly from left to right when parentheses are present …"
Output (Trial/Crossover Prompt)
"Let's think step by step. When evaluating a Boolean expression, first resolve the innermost parentheses, then move outward through the expression. Apply each Boolean operator in the correct structural order rather than simply left to right, and continue until the final truth value is obtained."
Replaced/Retain VS $X_{current}$

……

You are an expert at improving prompts using reflection. Your task is to summarize useful prompt-editing experience from this generation.

Short-term reflections from this generation:
Force the model to explicitly resolve innermost parentheses first, overriding strict left-to-right evaluation.
Add a sequential step-by-step decomposition requirement before evaluating the expression.
Add a strict requirement to evaluate the expression one operator at a time in sequence.
…
Output (meta-rule):
For tasks requiring precise evaluation of structured expressions (like Boolean logic), explicitly instructing a step-by-step, operator-by-operator decomposition—framed as a clear, mandatory procedure—consistently improves performance. This directive should specify processing one logical operation at a time in sequence, rather than relying on implicit reasoning.

……

Best prompt : Evaluate the truth value of a random Boolean expression consisting of Boolean constants (True, False) and basic Boolean operators (and, or, not). You must follow a mandatory, step-by-step procedure: decompose the entire expression and evaluate it one logical operator at a time in a clear sequence. Do not rely on implicit or holistic reasoning. Process a single operation, state the result, and then proceed to the next operation until the final truth value is determined.

Figure 9: Case study of a single MODEPrompt iteration on the boolean_expressions task. MODE-Prompt first induces taxonomy-defined cognitive subspaces, then contrasts Pareto-front and non-front prompts to extract a semantic differential patch. The actionable patch is injected through semantic mutation, and successful outcome-grounded edits are summarized into a long-term meta-rule to guide subsequent iterations.