Bandit-Based Prompt Design Strategy Selection Improves Prompt Optimizers

Anonymous ACL submission

Abstract

Prompt optimization aims to search for effective prompts that enhance the performance of large language models (LLMs). Although existing prompt optimization methods have discovered effective prompts, they often differ from sophisticated prompts carefully designed by human experts. Prompt design strategies, representing best practices for improving prompt performance, can be key to improving prompt optimization. Recently, a method termed the Autonomous Prompt Engineering Toolbox (APET) has incorporated various prompt design strategies into the prompt optimization process. In APET, the LLM is needed to implicitly select and apply the appropriate strategies because prompt design strategies can have negative effects. This implicit selection may be suboptimal due to the limited optimization capabilities of LLMs. This paper introduces Optimizing Prompts with sTrategy Selection (OPTS), which implements explicit selection mechanisms for prompt design. We propose three mechanisms, including a Thompson sampling-based approach, and integrate them into EvoPrompt, a well-known prompt optimizer. Experiments optimizing prompts for two LLMs, Llama-3-8B-Instruct and GPT-4o mini, were conducted using BIG-Bench Hard. Our results show that the selection of prompt design strategies improves the performance of EvoPrompt, and the Thompson sampling-based mechanism achieves the best overall results.

1 Introduction

Large language models (LLMs) such as GPT-4 (OpenAI et al., 2024), Gemini (Team et al., 2024), and Llama 3 (Grattafiori et al., 2024) have demonstrated superior abilities in a variety of domains, including medicine (Nori et al., 2023), law (Katz et al., 2024), and code generation (Rozière et al., 2024). Since well-crafted prompts improve the performance of LLMs, prompt engineering (i.e., designing better prompts) plays a key role in the area (Bsharat et al., 2024; Schulhoff et al., 2024). Despite its importance, prompt engineering is laborious as it requires a lot of time for refinement and sufficient knowledge of the tasks. To design effective prompts with less effort, research on prompt optimization has been actively conducted. In particular, discrete prompt optimization, which optimizes prompts within the natural language space, has attracted attention. This approach is valuable as it typically allows for the optimization of prompts for black-box LLMs, such as GPT-4, while also providing interpretable results (Chang et al., 2024). To explore effective prompts within the large natural language space, several methods have been proposed, which include emulating evolutionary algorithms using LLMs (Guo et al., 2024; Fernando et al., 2023; Cui et al., 2024). These methods have discovered effective prompts, but they often differ from sophisticated prompts carefully designed by human experts.

Prompt design strategies, which provide guidelines for creating effective prompts, can be key to boosting the performance of prompt optimizers. In fact, Chain-of-Thought (CoT; Wei et al., 2022) and Role Prompting (Wang et al., 2024a) have been employed in prompt optimization, leading to better prompts (Agarwal et al., 2024). Recently, Kepel and Valogianni (2024) proposed a method termed the Autonomous Prompt Engineering Toolbox (APET), which incorporated various prompt design strategies into the optimization process. APET fed all prepared strategies into an LLM to generate a new prompt that incorporates the strategies. However, not all strategies should be incorporated because prompt design strategies can have negative effects depending on both the LLM and the task (Zheng et al., 2024; Deng et al., 2024). In APET, an LLM that generates prompts is required to implicitly select appropriate strategies, which may lead to suboptimal performance

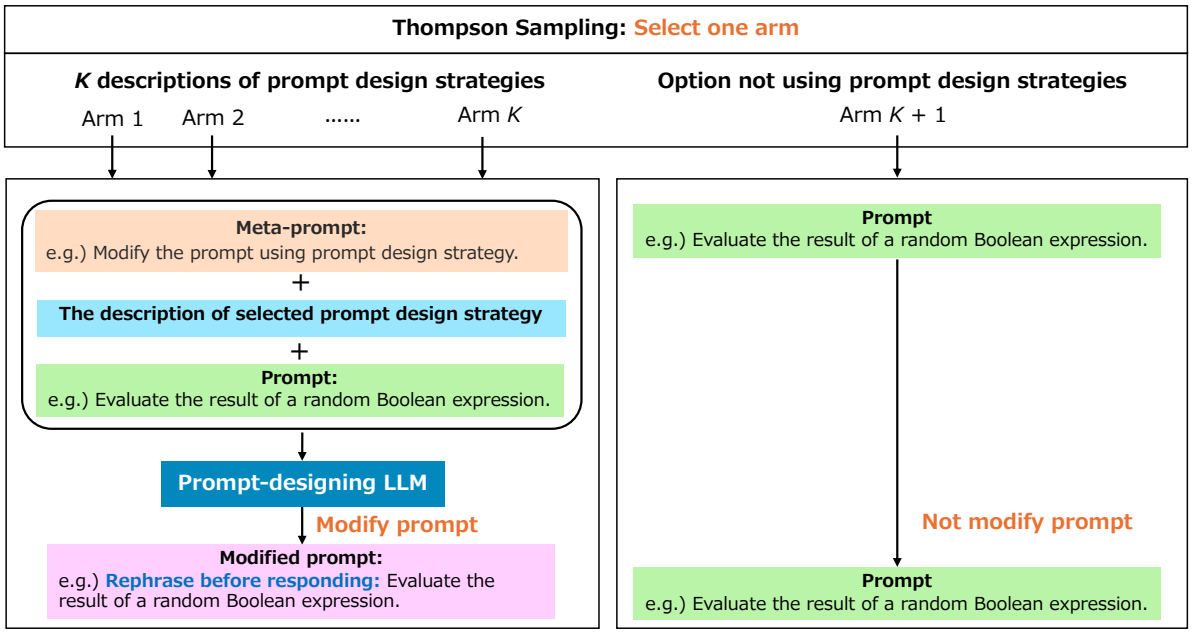

Figure 1: Overview of OPTS(TS), which shows how the prompt generated from the prompt optimizer is modified. If one of the first K arms is selected, the description of the prompt design strategy corresponding to the selected arm is passed to the prompt-designing LLM. If the $(K+1)$-th arm is selected, the prompt is not modified in any way.

because LLMs cannot perform optimization effectively (Huang et al., 2024).

In this paper, we introduce explicit selection mechanisms for prompt design strategies for the first time. We also propose three selection methods, including one based on Thompson sampling (TS; Thompson, 1933; Russo et al., 2018). By integrating them with the existing prompt optimizer, EvoPrompt (Guo et al., 2024), we show that explicit strategy selection effectively leverages existing knowledge of prompt design and enhances the performance of prompt optimizers. Moreover, the optimizer with the TS-based selection mechanism outperforms other existing methods.

In summary, our contributions are as follows:

- We propose explicit prompt design strategy selection mechanisms, including a method based on Thompson sampling, for prompt optimizers.
- We experimentally show that the proposed selection mechanism enhances EvoPrompt. TS-based selection improves EvoPrompt's performance by up to 50% when using GPT-4o mini for both generating prompts and solving downstream tasks.
- We also compare the TS-based selection with APET-based selection and uniform sampling-based selection. The results demonstrate that TS-based selection is overall superior.

2 Related Work

Prompt design strategy. The term prompt design strategy refers to a well-established guideline for designing prompts that have been empirically known to be effective. Chain-of-Thought (CoT; Wei et al., 2022) and Role Prompting (Wang et al., 2024a) are notable examples. CoT asks the LLMs to generate not only the answer, but also the reasoning process that leads to the answer. Role Prompting is a strategy that includes phrases in the prompt that give the LLM a role. Various prompt design strategies have been proposed so far (Schulhoff et al., 2024; Xu et al., 2023; Li et al., 2023; Xu et al., 2024; Deng et al., 2024; Lu et al., 2023; Bsharat et al., 2024), yet they are not always useful. Indeed, CoT and Role Prompting can lead to worse results (Deng et al., 2024; Zheng et al., 2024). As their efficacy depends on the LLM and task, users need to make a non-obvious decision on whether to use them.

Discrete prompt optimization. Discrete prompt optimization optimizes prompts in natural language space. To effectively deal with natural language space, several prompt optimizers emulate the process of black-box optimization algorithms by using LLMs. These methods are useful because optimized prompts have high interpretability, while they can be applied to LLMs that can be accessed

through black-box APIs such as GPT-4 (OpenAI et al., 2024). GRIPS (Prasad et al., 2023) repeatedly edits the phrases in the prompt, and APE (Zhou et al., 2023) repeatedly generates prompts using LLMs based on Monte Carlo search. Unlike APE and GRIPS, ProTeGi (Pryzant et al., 2023) uses a mechanism in which incorrect answers made by an LLM and the corresponding prompt are fed into another LLM to generate a proposal to improve the prompt, and another LLM responsible for designing prompts then modifies the prompt according to the proposal. In addition to this mechanism, PromptAgent (Wang et al., 2024b) also uses Monte Carlo Tree Search to efficiently optimize prompts. Besides these, adv-ICL (Long et al., 2024), which applies adversarial learning, has been proposed. In OPRO (Yang et al., 2024), instead of using an LLM to suggest a proposal to improve the prompts, an LLM directly generates new prompts using three items: previously generated prompts, their scores, and a description of the downstream task.

Recently, several methods combining LLMs with evolutionary algorithms have been proposed (Guo et al., 2024; Jin et al., 2024; Fernando et al., 2023; Cui et al., 2024). EvoPrompt (Guo et al., 2024), a representative method among them, emulates the optimization process of Genetic Algorithm (GA) or Differential Evolution (DE). In contrast to EvoPrompt (Guo et al., 2024), PromptBreeder (Fernando et al., 2023) also optimizes the prompt that is used for generating new prompts. PhaseEvo (Cui et al., 2024) optimizes both task instruction and examples and achieves highly effective optimization by dividing optimization into multiple stages.

In addition to dividing optimization into multiple stages, PromptWizard (Agarwal et al., 2024) utilizes prompt design strategies such as CoT and Role Prompting, but it lacks a strategy selection mechanism and applies them in all cases. EoT prompting (Jin et al., 2024) optimizes zero-shot CoT (Kojima et al., 2022) using evolutionary algorithms. APET (Kepel and Valogianni, 2024) is the most relevant to our study. In APET, a prompt and descriptions of prompt design strategies are input to an LLM. The LLM then implicitly selects strategies and generates a new prompt. In contrast, we propose explicit strategy selection mechanisms that assist prompt optimizers in exploiting appropriate strategies.

3 Proposed Methods

In this section, we describe our proposed methods for selecting prompt design strategies. We then introduce prompt optimization algorithms that combine EvoPrompt (Guo et al., 2024) with the strategy selection methods. We term our methods Optimizing Prompts with sTrategy Selection (OPTS).

Terminology *Task-solving LLM* is an LLM that is applied to and solves downstream tasks, while *Prompt-designing LLM* is another LLM that produces helpful prompts for task-solving LLMs. In contrast to *prompt*, which represents an instruction for a task-solving LLM, *meta-prompt* refers to an instruction for a prompt-designing LLM.

3.1 Selection of Prompt Design Strategies

In the following, we discuss three different methods: the TS-based selection called OPTS(TS), the uniform sampling-based selection called OPTS(US), and the APET-based selection called OPTS(APET).

OPTS(TS) OPTS(TS) selects prompt design strategies using Thompson sampling (TS; Thompson, 1933; Russo et al., 2018), which is a multi-armed bandit algorithm and empirically shows superior performance (Chapelle and Li, 2011).

The overview of OPTS(TS) is shown in Figure 1. There are K arms, each corresponding to one of the K descriptions of the prompt design strategies. Also, we append a special arm called the *inaction arm*, which corresponds to the option of not using the prompt design strategy, making a total of $K+1$ arms. The inaction arm is needed because none of the predefined strategies may improve the prompts at all. To instantiate TS, we use the beta distributions as priors for the expected reward. Once one of the first K arms is sampled by TS, we feed the description of the corresponding prompt design strategy into the prompt-designing LLM along with the meta-prompt and the prompt to be modified. The LLM then modifies the prompt based on the input. The meta-prompt is the same as that used in APET, whose details are explained in Appendix A. After evaluating the generated prompt with the task-solving LLM and calculating its score, a reward is calculated using the score, and the distributions in TS are updated based on the reward. Throughout this paper, we compute the reward r for a prompt with the score s as

$$r = \mathbf{1}\left[s > \max \tilde{\mathcal{S}}\right] \in \{0, 1\} \ , \qquad (1)$$

Algorithm 1 EvoPrompt(DE)-OPTS

Require: Initial prompts $\mathcal{P}_0 = \{p_1, p_2, \ldots, p_N\}$, population size N, number of iterations T, development set D_{dev} consisting of input and correct output pairs (x, y), scoring function g, task-solving LLM f_T

Evaluation of initial prompts: $\mathcal{S}_0 \leftarrow \left\{ s_i = \frac{1}{|D_{\text{dev}}|} \sum_{(x,y)\in D_{\text{dev}}} g\left(y, f_T\left(p_i, x\right)\right) : p_i \in \mathcal{P}_0 \right\}$
for $t = 1$ to T **do**
  for p_i in $\mathcal{P}_{t-1}$ **do** ▷ p_i: the i-th parent prompt
    Sample donors: $p_{r_1}, p_{r_2} \in \mathcal{P}_{t-1}$, where p_{r_1}, p_{r_2}, and p_i differ from each other.
    Crossover and Mutation: $p_i' \leftarrow f_D(m_{\text{de}}, (p_i, p_{r_1}, p_{r_2}, p_{\text{best}}))$
    where p_{best} is the current best prompt. ▷ f_D: Prompt-Designing LLM
    ▷ m_{de}: Meta-prompt for DE-based crossover and mutation
    OPTS: Generate p_i'' from p_i' by incorporating prompt design strategies (Refer to Section 3.1)
    Selection: $p_i^* = \underset{p\in\{p_i, p_i''\}}{\text{argmax}} \frac{1}{|D_{\text{dev}}|} \sum_{(x,y)\in D_{\text{dev}}} g\left(y, f_T\left(p, x\right)\right)$
    ▷ Keep the better one in the population
    Update probability distribution if the TS-based selection is used (Refer to Section 3.1)
  end for
  Update: $\mathcal{P}_t \leftarrow \{p_i^* : 1 \leq i \leq N\}$
end for
Return the best prompt $p^* = \text{argmax}_{p\in\mathcal{P}_T} \frac{1}{|D_{\text{dev}}|} \sum_{(x,y)\in D_{\text{dev}}} g\left(y, f_T\left(p, x\right)\right)$

where $\tilde{\mathcal{S}}$ is the set of scores of the parent prompts, which come from EvoPrompt (Guo et al., 2024) described in Section 3.2, and $\mathbf{1}[\cdot]$ is the indicator function.

OPTS(US) In OPTS(US), each arm is selected according to a uniform distribution. OPTS(US) is similar to OPTS(TS), except that the probability of selecting each arm is equal and is not updated.

OPTS(APET) OPTS(APET) is the selection method based on APET (Kepel and Valogianni, 2024). It is slightly different from APET in that it has an additional option equivalent to the inaction arm. OPTS(APET) first randomly decides with a probability of 0.5 whether to modify a prompt based on the prompt design strategies. If it decides to modify the prompt, the prompt-designing LLM is applied to the prompt to incorporate the prompt design strategies. The prompt-designing LLM receives the meta-prompt, the description of all prompt design strategies, and the prompt to be modified. Then, it implicitly selects the prompt design strategies and modifies the prompt according to them.

3.2 EvoPrompt with OPTS

We combine the proposed selection methods with EvoPrompt (Guo et al., 2024). We adopt EvoPrompt because it is effective yet sufficiently simple, allowing us to focus solely on evaluating the strategy selection methods. Also, it has variants depending on whether GA or DE is employed. This feature allows us to assess the impact of prompt design strategy selection on different optimization algorithms. The algorithm integrated OPTS into EvoPrompt(DE) is shown in Algorithm 1, while that based on EvoPrompt(GA) is shown in Appendix B. Note that a response generation by LLM f is denoted by $f\left(p, x\right)$. We insert OPTS after the crossover and mutation process of EvoPrompt. After evaluating the newly generated prompts with the task-solving LLM, the scores are used to determine the next generation's population and, if necessary, to update the distribution of the arms in TS. See Appendix B for the details of the algorithm.

4 Experiments

We experimentally evaluate three strategy selection methods we introduce: OPTS(TS), OPTS(US), and OPTS(APET). Combined with EvoPrompt, these methods are applied to various tasks and compared with the existing baseline methods.

4.1 Dataset

We evaluate the proposed method using BIG-Bench Hard (BBH; Suzgun et al., 2022). BBH is a collection of the tasks that are challenging for LLMs. Details of each task can be found in the original BBH paper. For each task, we randomly sample 50 examples from the test set and use them as the development set, as done in (Guo et al., 2024). The

Prompt Design Strategy	Remarks
ExpertPrompting	Assign expert roles to task-solving LLMs (Xu et al., 2023).
Chain-of-Thought	Let task-solving LLMs also generate a reasoning process (Wei et al., 2022).
Tree-of-Thought	Let task-solving LLMs iteratively choose the best of multiple reasoning paths, backtracking as necessary (Yao et al., 2023).
Emotion Prompting	Incorporate phrases that appeal to human emotions (Li et al., 2023).
Re-Reading	Instruct task-solving LLMs to reread the question (Xu et al., 2024).
Style Prompting	Specifies the desired output style (Lu et al., 2023).
Rephrase and Respond	Let task-solving LLMs rephrase the question before responding (Deng et al., 2024).
Avoiding bias	A more generalized version of the 13th principle of the 26 principles of prompting (Bsharat et al., 2024).
Making prompt specific	Based on Best practices for prompt engineering published by OpenAI.[1]
Shortening the prompt	Based on the experimental result that accuracy can decrease as prompts become longer (Levy et al., 2024).
Adding necessary information	One of the strategies used in APET (Kepel and Valogianni, 2024).

Table 1: Prompt design strategies used in the experiment. The concrete descriptions of each strategy are provided in Appendix C.

development set is used for evaluating prompts in the optimization process. At the end of the optimization, the prompt with the highest score on the development set is tested on the test set excluding the development set.

4.2 Metrics

We use accuracy as the scoring function. When evaluating a prompt using task-solving LLMs, answer parts are first extracted from the responses generated by the LLMs. The regular expression used in lm-evalation-harness (Gao et al., 2024) is used to extract the answer parts. In the regular expression, the parts following "the answer is " are extracted. Then, the scoring function gives a value of 1 if they exactly match the desired responses and 0 otherwise.

4.3 Implementation Details

We evaluate the strategy selection methods with DE-based EvoPrompt. We also evaluate GA-based algorithm, which is presented in Appendix D. We use the 3-shot prompts from the original BBH paper (Suzgun et al., 2022), but we optimize only the task description and leave the examples unchanged. We conduct two experiments in which Llama-3-8B-Instruct (Grattafiori et al., 2024) and GPT-4o mini are used as the task-solving LLMs. In the experiments using Llama-3-8B-Instruct, we use all 27 tasks of the BBH. In the experiments using GPT-4o mini, due to the high API cost, we sample only 3 out of the 27 tasks. We run three trials with different random seeds in both our experiments with Llama-3-8B-Instruct and GPT-4o mini. We use GPT-4o mini for the prompt-designing LLMs, set the population size to 10, and perform optimization for 50 iterations. The prompt design strategies we use are listed in Table 1.

4.4 Initial Task Descriptions

The initial task descriptions given at the beginning of the optimization are prepared by the following procedure. First, we select five task descriptions from the 20 prepared descriptions based on their evaluation scores on the development set. The 20 task descriptions consist of those used in the original BBH paper (Suzgun et al., 2022) and their 19 paraphrases generated by GPT-4o mini. For paraphrasing, we use the instruction "Generate 19 variations of the following instruction while keeping the semantic meaning," which is a slightly modified version of the meta-prompt created by Zhou et al. (2023). Then, we use the 10 task descriptions, consisting of the five selected task descriptions and their respective paraphrases by GPT-4o mini, as the initial task descriptions. When paraphrasing the selected descriptions, in the same way as Guo et al. (2024), we use the meta-prompt for resampling with the instruction "Generate a variation of the following instruction while keeping the semantic meaning," which is created by Zhou et al. (2023).

4.5 Baseline methods

We use the manual prompts that use those introduced in the BBH paper, APET, and EvoPrompt as

[1] https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api. Accessed: Jan. 31, 2025.

Task ID	Task Name	Manual Prompt	APET	EvoPrompt(DE)	EvoPrompt(DE)-OPTS(APET)	EvoPrompt(DE)-OPTS(US)	EvoPrompt(DE)-OPTS(TS)
0	boolean expressions	54.00	67.50	74.50 (1.08)	79.83 (2.05)	**84.00** (0.41)	82.50 (4.95)
1	causal judgement	2.19	0.00	40.39 (3.00)	42.34 (2.98)	40.15 (4.88)	**45.50** (3.00)
2	date understanding	0.00	0.00	12.33 (8.01)	18.00 (0.00)	**19.17** (1.65)	18.00 (0.00)
3	disambiguation qa	19.50	22.50	30.00 (1.87)	38.33 (4.11)	**47.33** (6.54)	42.50 (5.31)
4	dyck languages	6.50	0.00	6.67 (0.85)	6.50 (0.71)	6.17 (0.85)	**7.50** (0.00)
5	formal fallacies	29.50	0.00	40.67 (1.31)	**44.83** (2.66)	42.50 (1.22)	43.50 (3.56)
6	geometric shapes	16.50	24.50	**36.00** (0.41)	33.00 (0.41)	33.00 (4.32)	35.83 (2.62)
7	hyperbaton	53.00	3.50	54.67 (0.85)	**70.00** (0.82)	59.50 (5.02)	60.50 (4.42)
8	logical deduction five objects	12.00	3.50	14.67 (1.03)	29.00 (6.48)	24.67 (7.15)	**37.17** (13.21)
9	logical deduction seven objects	5.50	3.00	5.83 (0.24)	10.17 (1.25)	**13.17** (0.85)	13.00 (1.87)
10	logical deduction three objects	44.00	20.50	45.83 (3.42)	70.17 (5.10)	71.83 (2.09)	**78.83** (5.98)
11	movie recommendation	40.50	0.00	49.50 (5.12)	49.50 (2.83)	48.00 (2.48)	**54.33** (1.89)
12	multistep arithmetic two	**53.50**	52.00	**53.50** (1.78)	52.17 (2.46)	50.33 (0.85)	52.50 (1.47)
13	navigate	84.50	38.50	83.17 (0.47)	80.17 (2.66)	81.67 (3.40)	**84.67** (1.03)
14	object counting	**87.50**	27.50	85.83 (0.62)	85.67 (0.62)	85.00 (0.82)	85.83 (0.24)
15	penguins in a table	26.04	8.33	22.57 (3.93)	28.47 (7.23)	**41.67** (5.17)	40.97 (7.90)
16	reasoning about colored objects	21.00	6.50	**53.00** (3.54)	50.00 (4.64)	45.67 (5.27)	49.50 (3.27)
17	ruin names	18.50	65.50	27.17 (2.90)	**68.67** (7.26)	64.83 (2.39)	67.67 (3.66)
18	salient translation error detection	12.50	6.50	46.17 (3.52)	46.50 (4.71)	**51.17** (1.55)	**51.17** (3.66)
19	snarks	24.22	0.00	55.47 (6.72)	58.59 (3.31)	**65.10** (4.25)	64.06 (1.10)
20	sports understanding	51.52	0.00	61.45 (1.67)	61.62 (1.43)	76.94 (7.76)	**78.45** (4.54)
21	temporal sequences	57.00	6.00	65.83 (3.06)	65.83 (1.18)	60.33 (0.62)	**66.00** (1.47)
22	tracking shuffled objects five objects	67.50	18.00	66.50 (2.16)	66.17 (0.94)	68.17 (1.25)	**69.33** (1.25)
23	tracking shuffled objects seven objects	47.50	19.50	52.00 (1.22)	52.50 (1.87)	**54.33** (1.65)	49.50 (0.71)
24	tracking shuffled objects three objects	80.50	80.50	78.67 (1.55)	77.50 (2.83)	79.17 (1.65)	**81.67** (2.01)
25	web of lies	93.50	42.00	**95.00** (0.00)	**95.00** (0.00)	**95.00** (0.00)	**95.00** (0.00)
26	word sorting	44.50	41.00	45.50 (0.71)	45.83 (4.11)	**46.17** (1.70)	45.33 (2.25)
	AVG	39.00	20.62	48.26	52.83	53.89	**55.59**

Table 2: Accuracy on the test set for 27 tasks from BBH, evaluated with Llama-3-8B-Instruct as the task-solving LLM. The scores are averaged over three trials with different seeds. The values in parentheses represent the standard deviation. The bold scores indicate that the prompt optimized by the method achieved the highest average score. The prompt used in "date understanding" (task ID 2) did not contain the problem to solve due to the prompt template error. Although it looks meaningless (all problems in the task have the same prompt resulting in the same response), the comparison of the candidate methods is completely fair. We will prepare the corrected experimental results at the earliest point.

Task ID	EvoPrompt(DE)	EvoPrompt(DE)-OPTS(TS)
8	2.67 (1.43)	**52.33** (11.45)
19	79.17 (0.37)	**79.43** (1.33)
23	80.67 (4.37)	**81.83** (3.47)

Table 3: Accuracy on the test set for three tasks from BBH, evaluated with GPT-4o mini. The task IDs are the same as in Table 2. The scores in this table are the average scores over three trials. The values in parentheses represent the standard deviation.

the baseline methods. In APET, we use the APET procedure to incorporate prompt design strategies into task description in manual prompts.

4.6 Main Result

Table 2 shows the test scores using Llama-3-8B-Instruct as the task-solving LLM.

Effects of OPTS. First, we focus on the effect of the explicit selection mechanism in OPTS. Table 2 shows that all three variants of OPTS increase the average scores of EvoPrompt(DE) by approximately 4.5% to 7.5% compared with the naive EvoPrompt(DE). In particular, for the task “ruin names” (task ID 17), all three variants of EvoPrompt(DE)-OPTS outperform EvoPrompt(DE) by about 40%. The results indicate that OPTS can improve the prompt optimizer.

Comparing OPTS Variants. We compare three selection methods: OPTS(TS), OPTS(US), and OPTS(APET). Table 2 shows that, with EvoPrompt(DE), OPTS(TS) outperforms OPTS(US) by about 1.7% and OPTS(APET) by about 2.7% on average. This result indicates that OPTS(TS) can select more suitable strategies for task-solving LLMs and tasks. In addition, OPTS(APET) is inferior to OPTS(US), implying that LLMs have an

Method	Description	Score
Manual Prompt	Select the humorous edit that 'ruins' the input movie or musical artist name.	18.50
EvoPrompt(DE)	Decide on the eccentric twist that 'spoils' the name of the movie or music artist.	31.00
APET	(1) **Imagine you are a creative expert in humor and wordplay, skilled at crafting amusing edits that playfully distort movie or musical artist names. Your task is to select the humorous edit that 'ruins' the given input name in a funny and clever way.** (2) **Let's think step-by-step: First**, (3) **identify the original name provided**. (2) **Next**, (3) **brainstorm potential humorous edits that could transform the name into something amusing while maintaining a connection to the original**. (2) **Finally**, choose the edit that best exemplifies the concept of 'ruining' the name in a (4) **lighthearted manner**. (5) **Read the question again to ensure clarity before proceeding**. (6) **Remember, your goal is to evoke laughter and joy through your selection!**	65.50
EvoPrompt(DE)-OPTS(TS)	(2) **Let's think step-by-step! First,** carefully (5) **read the question again** and (3) **identify a movie title or musician whose name lends itself to a humorous spoof.** (2) **Next,** (3) **creatively reimagine that title or name with an absurdly funny twist that maintains the essence of the original while injecting a comedic element**. (2) **Finally,** present your funniest version clearly, ensuring it is both memorable and entertaining. (6) **Let your creativity shine through in this process!**	70.50

Table 4: Examples of the discovered descriptions (optimized parts within the prompts). These achieved the highest score in each method with Llama-3-8B-Instruct as the task-solving LLM and "ruin names" as the task. Underlined texts represent (1) ExpertPrompting, (2) Chain-of-Thought, (3) Making prompt specific, (4) Style Prompting, (5) Re-Reading, and (6) Emotion Prompting.

incorrect bias in selecting prompt design strategies.

Differences between Tasks. Table 2 also shows that the improvement achieved by strategy selection methods varies depending on the task. For example, in the tasks of "ruin names" (task ID 17) and "logical deduction three objects" (task ID 10), EvoPrompt(DE)-OPTS(TS) outperforms EvoPrompt(DE) by approximately 40% and 30%, respectively. On the other hand, OPTS degrades the performance of EvoPrompt(DE) in several tasks. A possible reason is that effective prompt design strategies differ for each task. In tasks with significant improvement, the eleven candidate strategies used in the experiment likely include effective options. In contrast, it may be difficult for tasks with performance degradation to achieve better performance using only the strategies available among the eleven candidates. Owing to the inaction arm, we note that the performance degradation is insignificant compared with the degree of performance improvement.

4.7 Results Using Another Task-Solving LLM

We also evaluated EvoPrompt(DE)-OPTS(TS) using another task-solving LLM, GPT-4o mini, which is one of the most widely used LLMs. We conducted experiments on three randomly chosen tasks from BBH. The experimental setup was the same as the experiment using Llama-3-8B-Instruct. Table 3 shows the accuracy of the test set. We observe that EvoPrompt(DE)-OPTS(TS) outperforms EvoPrompt(DE) in all three tasks. In particular, for "logical deduction five objects" (task ID 8), OPTS(TS) increases the accuracy by approximately 50%. This result suggests that OPTS is likely to be effective regardless of the task-solving LLM.

Before or After OPTS	Description
Before	Let's think step-by-step. Identify a humorous variation that spoofs the title of a film or music artist, inventing a funny alteration while preserving its original essence. Before providing your answer, ensure full clarity and understanding.
After	Let's think step-by-step. **First**, identify a film or music artist whose title can be humorously spoofed. **Next**, brainstorm a funny alteration that captures the essence of the original title while adding a comedic twist. **Finally**, clearly articulate your humorous variation, ensuring that it maintains the original's core meaning.

Table 5: Example of a prompt where Chain-of-Thought prompting is already used, and Chain-of-Thought prompting is selected again and modified accordingly. The task, task-solving LLM, and seed settings are the same as in Table 4.

5 Analysis

In this section, we analyze OPTS(TS) from two perspectives: the analyses of the discovered task descriptions and the case where a strategy was applied more than once.

Analysis of Discovered Descriptions. Table 4 shows the discovered task descriptions when using Llama-3-8B-Instruct as the task-solving LLMs and "ruin names" as the task to be solved. We can see that the task description discovered by EvoPrompt(DE) does not use any prompt design strategies and has a structure similar to that of a manual prompt, which is used to obtain the initial prompts for EvoPrompt. We consider that, although the crossover and mutation performed by the prompt-designing LLM in EvoPrompt can change the phrases in the prompt, it is difficult to change the structure significantly. In addition, unlike EvoPrompt(DE), EvoPrompt(DE)-OPTS(TS) discovered the task description with various prompt design strategies, including CoT, Re-Reading, Making prompt specific, and Emotion Prompting. This result means that OPTS(TS) significantly improves the task description using prompt design strategies. When comparing EvoPrompt(DE)-OPTS(TS) with APET, APET incorporated more strategies while its score was lower than EvoPrompt(DE)-OPTS(TS). Implicit selection by APET has the advantage of selecting and incorporating multiple prompt design strategies at once, but it may also include unnecessary strategies. Furthermore, we observe that the task description of EvoPrompt(DE)-OPTS(TS) uses several prompt design strategies in combination, although OPTS(TS) selects only one prompt design strategy at a time. Indeed, Re-Reading and Making prompt specific are used within the CoT. This shows that EvoPrompt(DE)-OPTS(TS) possesses the ability to combine multiple prompt design strategies as well as APET.

Analysis of Repeated Selection of a Strategy. During the optimization process of EvoPrompt-OPTS(TS), we observed that OPTS(TS) sampled a prompt design strategy that had already been incorporated into the prompt, thereby further modifying it. Table 5 illustrates a case in which CoT was applied again to a prompt within the same trial as Table 4. The example shows that reapplying CoT further aligned the prompt with the strategy. Also, the feature introduced in the step can be observed in the best prompt in Table 4. This example suggests that repeatedly applying the same strategy helps incorporate it more effectively.

6 Conclusion and Discussion

We introduced explicit selection mechanisms into prompt optimization to effectively leverage existing knowledge of prompt design. Experiments have demonstrated that the three methods we introduced improve the performance of the prompt optimizers. In particular, the method based on Thompson sampling is the best among those we compared. The prompts discovered by our methods effectively incorporate several prompt design strategies, which EvoPrompt alone was unable to discover. Our results highlight the importance of leveraging existing knowledge and selecting it explicitly.

Limitations

There are four limitations that remain for future research: (1) We formulated OPTS(TS) as the Bernoulli bandit, but alternative reward formulation may improve optimization performance. (2)

Although OPTS can be easily integrated into various prompt optimizers, we have not attempted to introduce it to other optimizers than EvoPrompt. (3) We used Thompson sampling to select the prompt design strategy but did not evaluate other sophisticated methods, such as contextual bandit algorithms. (4) The performance of OPTS mechanism can vary depending on the prompt design strategy prepared, but this is not clarified in this paper. These points remain topics for future research.

Ethical Considerations

Our method may involve the risk of being used to optimize prompts that generate malicious content, such as malware or fake news, even though this is not the intention of our method. At present, it is extremely difficult to reduce this risk. Although our method may be beneficial to some malicious users, we expect that our method can be more beneficial to many other benevolent users.

References

Eshaan Agarwal, Joykirat Singh, Vivek Dani, Raghav Magazine, Tanuja Ganu, and Akshay Nambi. 2024. PromptWizard: Task-aware prompt optimization framework. *Preprint*, arXiv:2405.18369.

Sondos Mahmoud Bsharat, Aidar Myrzakhan, and Zhiqiang Shen. 2024. Principled instructions are all you need for questioning LLaMA-1/2, GPT-3.5/4. *Preprint*, arXiv:2312.16171.

Kaiyan Chang, Songcheng Xu, Chenglong Wang, Yingfeng Luo, Xiaoqian Liu, Tong Xiao, and Jingbo Zhu. 2024. Efficient prompting methods for large language models: A survey. *Preprint*, arXiv:2404.01077.

Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of Thompson sampling. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

Wendi Cui, Jiaxin Zhang, Zhuohang Li, Hao Sun, Damien Lopez, Kamalika Das, Bradley A. Malin, and Sricharan Kumar. 2024. PhaseEvo: Towards unified long-context prompt optimization for large language models. In *First Workshop on Long-Context Foundation Models @ ICML 2024*.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2024. Rephrase and respond: Let large language models ask better questions for themselves. *Preprint*, arXiv:2311.04205.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. A framework for few-shot language model evaluation.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

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

Beichen Huang, Xingyu Wu, Yu Zhou, Jibin Wu, Liang Feng, Ran Cheng, and Kay Chen Tan. 2024. Exploring the true potential: Evaluating the black-box optimization capability of large language models. *Preprint*, arXiv:2404.06290.

Feihu Jin, Yifan Liu, and Ying Tan. 2024. Zero-shot chain-of-thought reasoning guided by evolutionary algorithms in large language models. *Preprint*, arXiv:2402.05376.

Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo Arredondo. 2024. GPT-4 passes the bar exam. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 382(2270):20230254.

Daan Kepel and Konstantina Valogianni. 2024. Autonomous prompt engineering in large language models. *Preprint*, arXiv:2407.11000.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15339–15353, Bangkok, Thailand. Association for Computational Linguistics.

Cheng Li, Jindong Wang, Yixuan Zhang, Kaijie Zhu, Wenxin Hou, Jianxun Lian, Fang Luo, Qiang Yang,

and Xing Xie. 2023. Large language models understand and can be enhanced by emotional stimuli. *Preprint*, arXiv:2307.11760.

Do Long, Yiran Zhao, Hannah Brown, Yuxi Xie, James Zhao, Nancy Chen, Kenji Kawaguchi, Michael Shieh, and Junxian He. 2024. Prompt optimization via adversarial in-context learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7308–7327, Bangkok, Thailand. Association for Computational Linguistics.

Albert Lu, Hongxin Zhang, Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2023. Bounding the capabilities of large language models in open text generation with prompt constraints. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1982–2008, Dubrovnik, Croatia. Association for Computational Linguistics.

Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. 2023. Capabilities of GPT-4 on medical challenge problems. *Preprint*, arXiv:2303.13375.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. GPT-4 technical report. *Preprint*, arXiv:2303.08774.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2023. 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, Dubrovnik, Croatia. Association for Computational Linguistics.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, Singapore. Association for Computational Linguistics.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, and 7 others. 2024. Code Llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, and 1 others. 2018. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, Pranav Sandeep Dulepet, Saurav Vidyadhara, Dayeon Ki, Sweta Agrawal, Chau Pham, Gerson Kroiz, Feileen Li, Hudson Tao, Ashay Srivastava, and 12 others. 2024. The prompt report: A systematic survey of prompting techniques. *Preprint*, arXiv:2406.06608.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *Preprint*, arXiv:2210.09261.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 1331 others. 2024. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

William R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294.

Noah Wang, Z.y. Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Jian Yang, Man Zhang, Zhaoxiang Zhang, Wanli Ouyang, Ke Xu, Wenhao Huang, Jie Fu, and Junran Peng. 2024a. RoleLLM: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14743–14777, Bangkok, Thailand. Association for Computational Linguistics.

Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric Xing, and Zhiting Hu. 2024b. PromptAgent: Strategic planning with language models enables expert-level prompt optimization. In *The Twelfth International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. 2023. ExpertPrompting: Instructing large language models to be distinguished experts. *Preprint*, arXiv:2305.14688.

Xiaohan Xu, Chongyang Tao, Tao Shen, Can Xu, Hongbo Xu, Guodong Long, Jian-Guang Lou, and

Shuai Ma. 2024. Re-reading improves reasoning in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15549–15575, Miami, Florida, USA. Association for Computational Linguistics.

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.

Mingqian Zheng, Jiaxin Pei, Lajanugen Logeswaran, Moontae Lee, and David Jurgens. 2024. When "a helpful assistant" is not really helpful: Personas in system prompts do not improve performances of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15126–15154, Miami, Florida, USA. Association for Computational Linguistics.

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

A Meta-Prompt for OPTS

The meta-prompt we used was based on APET (Kepel and Valogianni, 2024) as shown in Table 6. We fed this meta-prompt into the prompt-designing LLMs after replacing the `<strategy>` tag with descriptions of the prompt design strategies and the `<input>` tag with the prompt to be modified. For APET and OPTS(APET), the `<strategy>` tag was replaced with the list of K tags from `<strategy 1>` to `<strategy K>`, where the description of each prompt design strategy is inserted.

B Details of EvoPrompt-OPTS

The algorithm that integrates OPTS into EvoPrompt(GA) is shown in Algorithm 2. In the following, we provide supplementary explanations regarding EvoPrompt processes.

Crossover and Mutation. The prompt-designing LLM performs crossover and mutation based on the meta-prompts and prompts fed into them. The meta-prompts for EvoPrompt(GA) and EvoPrompt(DE) are described in Tables 7 and 8, respectively.

In EvoPrompt(GA), two prompts are selected as parents using roulette wheel selection. They are fed into the prompt-designing LLM along with the meta-prompt by replacing `<prompt1>` and `<prompt2>` in the meta-prompt provided in Table 7 with them. The LLM then generates an offspring prompt according to the meta-prompt.

In EvoPrompt(DE), four prompts in the current population are used to generate an offspring prompt: two randomly selected prompts p_{r_1} and p_{r_2}, the current best prompt p_{best}, and a parent prompt. Each prompt in the current population is selected once as the parent prompt. Those four prompts are fed into the prompt-designing LLM by replacing `<prompt0>`, `<prompt1>`, `<prompt2>`, and `<prompt3>` in Table 8 with the parent prompt, p_{r_1}, p_{r_2}, and p_{best}, respectively. The LLM then performs crossover and mutation to generate an offspring prompt according to the meta-prompt.

C Descriptions of Prompt Design Strategies

Table 9 provides the descriptions for each of the 11 prompt design strategies used in our experiment.

D GA-Based Experimental Results

In this section, we present the GA-based experimental results. Table 10 shows the result of the experiment using Llama-3-8B-Instruct as a task-solving LLM, and Table 11 shows the result of the experiment using GPT-4o mini as a task-solving LLM. Tables 10 and 11 show that, as in the DE-based experiments, the performance of EvoPrompt (GA) is enhanced by selecting suitable prompt design strategy using OPTS. In addition, when comparing the variations of OPTS, we observe a tendency that OPTS(TS) is overall the best, followed by OPTS(US) and OPTS(APET), as in the case of the EvoPrompt(DE). In terms of the extent of improvement, combining the mechanism for selecting a prompt design strategy with EvoPrompt(DE) has a greater impact than that with EvoPrompt(GA). These results suggest that, although the extent of improvement varies slightly depending on the evolutionary algorithm used in the prompt optimizer, the mechanism to select the prompt design strategy can enhance the prompt optimizer regardless of the evolutionary algorithm used.

System prompt	Imagine yourself as an expert in the realm of prompting techniques for LLMs. Your expertise is not just broad, encompassing the entire spectrum of current knowledge on the subject, but also deep, delving into the nuances and intricacies that many overlook. Your job is to reformulate prompts with surgical precision, optimizing them for the most accurate response possible. The reformulated prompt should enable the LLM to always give the correct answer to the question.
User prompt	Your available prompting techniques include, but are not limited to the following: - <strategy> Your approach is methodical and analytical, yet creative. You use a mixture of the prompting techniques, making sure you pick the right combination for each instruction. You see beyond the surface of a prompt, identifying the core objectives and the best ways to articulate them to achieve the desired outcomes. Output instructions:""" You should ONLY return the reformulated prompt. Make sure to include ALL information from the given prompt to reformulate. """ Given above information and instructions, reformulate below prompt using the techniques provided: """ <input> """

Table 6: Meta-prompt for OPTS and APET (Kepel and Valogianni, 2024). When APET or OPTS(APET) is used, K `<strategy>` tags (i.e., `<strategy 1>`, `<strategy 2>`, ..., `<strategy` K`>`) are provided, and each of them is replaced with one prompt design strategy description.

E License for Artifacts

BIG-Bench Hard and lm-evaluation-harness are licensed under the MIT License. Llama-3-8B-Instruct is licensed under META LLAMA 3 COMMUNITY LICENSE AGREEMENT. Our code will also be released under the MIT license.

F Artifact Use Consistent with Intended Use

We declare that we have used the BIG-Bench Hard dataset and Llama-3-8B-Instruct in accordance with their original intended use. Additionally, we prohibit the use of the code we release for optimizing prompts to generate malicious content, except for research purposes.

G Experimental Environment

Our experiments were conducted on a computer running Ubuntu 22.04 with an AMD EPYC 7502P CPU and an NVIDIA A100 GPU, and on another computer running Ubuntu 22.04 with an AMD EPYC 7702P CPU and an NVIDIA A100 GPU. We used openai 1.40.8 as the python library to access GPT-4o mini, and vllm 0.6.3.post1 as the python library to access llama-3-8B-Instruct.

User prompt	Please follow the instruction step-by-step to generate a better prompt. 1. Crossover the following prompts to generate a new prompt: Prompt 1: Your task is to classify the comment as one of the following categories: terrible, bad, okay, good, great. Prompt 2: In this task, you are given sentences from movie reviews. The task is to classify a sentence as one of the following categories: terrible, bad, okay, good, great. 2. Mutate the prompt generated in Step 1 and generate a final prompt bracketed with <prompt> and </prompt>. 1. Crossover Prompt: In this task, you are given comments from movie reviews. Your task is to classify each comment as one of the following categories: terrible, bad, okay, good, great. 2. <prompt>Given a sentence from a movie review, classify it into one of the following categories: terrible, bad, okay, good, or great.</prompt> Please follow the instruction step-by-step to generate a better prompt. 1. Crossover the following prompts and generate a new prompt: Prompt 1: <prompt1> Prompt 2: <prompt2> 2. Mutate the prompt generated in Step 1 and generate a final prompt bracketed with <prompt> and </prompt>. 1.

Table 7: Meta-prompt for crossover and mutation in EvoPrompt(GA) (Guo et al., 2024).

Algorithm 2 EvoPrompt(GA)-OPTS

Require: Initial prompts $\mathcal{P}_0 = \{p_1, p_2, \ldots, p_N\}$, population size N, number of iterations T, development set D_{dev} consisting of input and correct output pairs (x, y), scoring function g, task-solving LLM f_T

Evaluation of initial prompts: $\mathcal{S}_0 \leftarrow \left\{ s_i = \frac{1}{|D_{\text{dev}}|} \sum_{(x,y)\in D_{\text{dev}}} g\left(y, f_T\left(p_i, x\right)\right) : p_i \in \mathcal{P}_0 \right\}$
for $t = 1$ to T **do**
  for $i = 1$ to N **do**
    Sampling parents by roulette wheel: $p_{r_1}, p_{r_2} \in \mathcal{P}_{t-1}$
    Crossover and Mutation: $p_i' \leftarrow f_D(m_{\text{ga}}, (p_{r_1}, p_{r_2}))$
      ▷ f_D: prompt-designing LLM
      ▷ m_{ga}: Meta-prompt for GA-based crossover and mutation
    OPTS: Generate p_i'' from p_i' by incorporating prompt design strategies (Refer to Section 3.1)
    Evaluation: $s_i'' \leftarrow \frac{1}{|D_{\text{dev}}|} \sum_{(x,y)\in D_{\text{dev}}} g\left(y, f_T\left(p_i'', x\right)\right)$
    Update probability distribution if the TS-based selection is used (Refer to Section 3.1)
  end for
  $\hat{\mathcal{S}}_t \leftarrow \{s_i'' : 1 \le i \le N\}$, $\hat{\mathcal{P}}_t \leftarrow \{p_i'' : 1 \le i \le N\}$
  Update score: $\mathcal{S}_t \leftarrow$ select the best N scores in $\mathcal{S}_{t-1} \cup \hat{\mathcal{S}}_t$
  Update: $\mathcal{P}_t \leftarrow$ select the best N prompts in $\mathcal{P}_{t-1} \cup \hat{\mathcal{P}}_t$ according to $\mathcal{S}_{t-1} \cup \hat{\mathcal{S}}_t$,
end for
Return the best prompt $p^* = \text{argmax}_{p\in\mathcal{P}_T} \frac{1}{|D_{\text{dev}}|} \sum_{(x,y)\in D_{\text{dev}}} g\left(y, f_T\left(p, x\right)\right)$

User prompt	Please follow the instruction step-by-step to generate a better prompt. 1. Identify the different parts between the Prompt 1 and Prompt 2: Prompt 1: Rewrite the input text into simpler text. Prompt 2: Rewrite my complex sentence in simpler terms, but keep the meaning. 2. Randomly mutate the different parts 3. Combine the different parts with Prompt 3, selectively replace it with the different parts in step 2 and generate a new prompt. Prompt 3: Rewrite the given input text into simpler English sentences while preserving the same meaning, so it can be understood by non-native English speakers. 4. Crossover the prompt in the step3 with the following basic prompt and generate a final prompt bracketed with <prompt> and </prompt>: Basic Prompt: Make the sentence easier for people who do not speak English fluently to comprehend. 1. Identifying the different parts between Prompt 1 and Prompt 2: Prompt 1: Rewrite the input text into simpler text. Prompt 2: Rewrite my complex sentence in simpler terms, but keep the meaning. Different parts: "input text" vs "my complex sentence" "simpler text" vs "simpler terms, but keep the meaning" 2. Randomly mutate the different parts: "input text" -> "provided text" "my complex sentence" -> "the difficult sentence" "simpler text" -> "easier language" "simpler terms, but keep the meaning" -> "simpler words while maintaining the meaning" 3. Combine the different parts with Prompt 3, selectively replace it with the different parts in step 2 and generate a new prompt: Prompt 3: Rewrite the given input text into simpler English sentences while preserving the same meaning, so it can be understood by non-native English speakers. New Prompt: Transform the provided text into easier language while maintaining the meaning, making it accessible for non-native English speakers. 4. Crossover the prompt in step 3 with the following basic prompt and generate a final prompt bracketed with <prompt> and </prompt>: Basic Prompt: Make the sentence easier for people who do not speak English fluently to comprehend. Final Prompt: <prompt>Convert the difficult sentence into simpler words while preserving the meaning, so it's easier for non-native English speakers to understand.</prompt> Please follow the instruction step-by-step to generate a better prompt. 1. Identify the different parts between the Prompt 1 and Prompt 2: Prompt 1: <prompt1> Prompt 2: <prompt2> 2. Randomly mutate the different parts 3. Combine the different parts with Prompt 3, selectively replace it with the different parts in step2 and generate a new prompt. Prompt 3: <prompt3> 4. Crossover the prompt in the step3 with the following basic prompt and generate a final prompt bracketed with <prompt> and </prompt>: Basic Prompt: <prompt0> 1. """

Table 8: Meta-prompt for crossover and mutation in EvoPrompt(DE) (Guo et al., 2024).

ExpertPrompting	Crafting an expert who is an expert at the given task, by writing a high-quality description about the most capable and suitable agent to answer the instruction in second person perspective.
Chain-of-Thought	Explaining step-by-step how the problem should be tackled, and making sure the model explains step-by-step how it came to the answer. You can do this by adding "Let's think step-by-step".
Tree-of-Thought	Imagining three different experts who are discussing the problem at hand. All experts will write down 1 step of their thinking, then share it with the group. Then all experts will go on to the next step, etc. If any expert realises they're wrong at any point then they leave.
Adding necessary information	Making sure all information needed is in the prompt, adding where necessary but making sure the question remains having the same objective.
Re-Reading	For a given prompt, add a phrase such as "Read the question again" that instructs the Large Language Models to reread the question before generating an answer. This strategy is particularly effective for complex tasks and helps enhance the quality and reliability of the model's outputs.
Style Prompting	Clearly define the desired style in the given prompt. For example, you might say, "Write a formal letter about..." or "Create a casual conversation discussing...". This guidance helps the model produce text that matches the requested stylistic elements, whether it's formal, informal, technical, or poetic.
Rephrase and Respond	For a given prompt, add a phrase that instructs the Large Language Models to rephrase the question before responding, such as "Rephrase and expand the question, and respond.
Making prompt specific	Make the description of the given prompt more specific. This makes it easier for Large Language Models to correctly execute prompt instructions.
Avoiding bias	To allow Large Language Models to make logical and unbiased inferences, add phrases to a given prompt that instruct it to remove opinionated content. This helps the model concentrate on providing responses based on careful analysis and logical reasoning, minimizing biases.
Shortening the prompt	If a given prompt has long instructions, make it shorter by condensing it to only the essential parts.
Emotion Prompting	At the end of the prompt, add a phrase that evokes a strong emotion. When doing so, keep the following four points in mind: 1. Define emotional goals: Identify the emotional response you want to evoke, such as encouragement, motivation, or reassurance. 2. Use positive language: Incorporate words and phrases that are positive and supportive. Examples include "believe in your abilities," "excellent," "success," and "outstanding achievements". 3. Emphasize key words: Use techniques like exclamation marks and capitalized words to highlight important aspects and to enhance the emotional impact. 4. Incorporate social and self-esteem cues: Design stimuli that leverage social influence (e.g., group membership, others' opinions) and boost self-esteem and motivation. This can help regulate the emotional response of the Large Language Models and tap into intrinsic motivation.

Table 9: Descriptions of each prompt design strategy. Descriptions of ExpertPrompting, Chain-of-Thought, Tree-of-Thought, and Adding necessary information are quoted from APET (Kepel and Valogianni, 2024). The descriptions Re-Reading, Style Prompting, Rephrase and Respond, and Emotion Prompting are modified versions of descriptions created using GPT-4o from Xu et al. (2024), Lu et al. (2023), Deng et al. (2024), and Li et al. (2023), respectively. Description of Making prompt specific is created by us and is inspired by OpenAI's prompt engineering best practice. Description of Avoiding bias is created by us by referring to and generalizing the 13th principle of the 26 prompting principles proposed in Bsharat et al. (2024). Description of Shortening the prompt was created by us.

Task ID	Task name	Manual prompt	APET	EvoPrompt(GA)	EvoPrompt(GA)-OPTS(APET)	EvoPrompt(GA)-OPTS(US)	EvoPrompt(GA)-OPTS(TS)
0	boolean expressions	54.00	67.50	72.83 (3.30)	84.33 (3.66)	83.00 (2.94)	**85.67** (1.65)
1	causal judgement	2.19	0.00	37.71 (1.82)	40.88 (2.60)	**44.53** (3.10)	**44.53** (0.60)
2	date understanding	0.00	0.00	0.00 (0.00)	12.00 (8.49)	**18.00** (0.00)	7.17 (10.14)
3	disambiguation qa	19.50	22.50	32.17 (4.33)	37.00 (2.86)	42.67 (3.27)	**49.67** (6.91)
4	dyck languages	6.50	0.00	**6.67** (0.94)	6.50 (0.71)	5.83 (0.24)	6.50 (0.00)
5	formal fallacies	29.50	0.00	42.50 (0.71)	**45.17** (2.05)	43.83 (0.85)	44.50 (0.82)
6	geometric shapes	16.50	24.50	29.83 (7.85)	**37.33** (6.14)	33.00 (4.26)	32.67 (2.95)
7	hyperbaton	53.00	3.50	54.83 (0.94)	61.67 (1.55)	57.67 (3.57)	**63.67** (6.51)
8	logical deduction five objects	12.00	3.50	12.33 (1.31)	20.00 (1.22)	24.67 (4.03)	**25.83** (7.96)
9	logical deduction seven objects	5.50	3.00	6.83 (3.01)	7.83 (3.09)	**12.17** (1.55)	11.33 (1.70)
10	logical deduction three objects	44.00	20.50	58.17 (5.72)	64.00 (3.89)	59.67 (6.02)	**66.67** (3.30)
11	movie recommendation	40.50	0.00	48.67 (1.18)	49.67 (4.17)	**53.50** (4.90)	52.50 (1.63)
12	multistep arithmetic two	**53.50**	52.00	50.33 (2.25)	49.83 (1.25)	46.83 (2.72)	50.17 (1.03)
13	navigate	**84.50**	38.50	79.67 (4.25)	80.17 (2.32)	83.83 (3.01)	82.33 (2.25)
14	object counting	**87.50**	27.50	85.50 (0.71)	82.67 (3.70)	86.67 (0.62)	84.33 (0.62)
15	penguins in a table	26.04	8.33	25.00 (1.47)	29.86 (8.26)	30.90 (2.99)	**32.29** (1.47)
16	reasoning about colored objects	21.00	6.50	50.17 (9.33)	47.50 (1.08)	47.00 (6.38)	**52.33** (2.72)
17	ruin names	18.50	65.50	51.00 (13.83)	63.17 (3.52)	**67.33** (0.24)	66.83 (0.47)
18	salient translation error detection	12.50	6.50	31.67 (9.74)	49.00 (4.42)	48.17 (4.48)	**52.67** (1.03)
19	snarks	24.22	0.00	45.83 (6.03)	52.60 (15.94)	**64.84** (2.92)	60.16 (4.42)
20	sports understanding	51.52	0.00	62.63 (0.00)	62.63 (0.00)	**65.15** (2.58)	61.95 (6.20)
21	temporal sequences	57.00	6.00	**63.67** (3.70)	59.33 (2.49)	61.83 (1.43)	61.33 (2.78)
22	tracking shuffled objects five objects	67.50	18.00	67.67 (0.47)	67.83 (0.24)	**68.67** (2.66)	67.33 (1.03)
23	tracking shuffled objects seven objects	47.50	19.50	**53.33** (2.49)	51.50 (0.82)	52.00 (1.08)	50.33 (1.84)
24	tracking shuffled objects three objects	80.50	80.50	81.00 (0.71)	81.67 (0.62)	80.67 (0.24)	**82.33** (0.62)
25	web of lies	93.50	42.00	**96.33** (1.03)	95.83 (0.62)	**96.33** (1.43)	96.00 (1.47)
26	word sorting	44.50	41.00	**46.83** (1.65)	43.67 (4.40)	46.00 (7.08)	44.17 (1.03)
	AVG	39.00	20.62	47.90	51.25	52.77	**53.16**

Table 10: Accuracy on the test set for 27 tasks from BBH, evaluated with Llama-3-8B-Instruct as the task-solving LLM. The scores are averaged over three trials with different seeds. The values in parentheses represent the standard deviation. The bold scores indicate that the prompt optimized by the method achieved the highest average score. The prompt used in "date understanding" (task ID 2) did not contain the problem to solve due to the prompt template error. Although it looks meaningless (all problems in the task have the same prompt resulting in the same response), the comparison of the candidate methods is completely fair. We will prepare the corrected experimental results at the earliest point.

Task ID	EvoPrompt(GA)	EvoPrompt(GA)-OPTS(TS)
8	1.67 (0.85)	**48.17** (5.14)
19	**79.43** (1.33)	78.65 (1.84)
23	75.67 (1.25)	**88.33** (1.55)

Table 11: Accuracy on the test set for three tasks from BBH, evaluated with GPT-4o mini. The task IDs are the same as in Table 2. The scores in the table are the average scores over three trials. The values in parentheses represent the standard deviation.