

# Prompt Recursive Search: A Living Framework with Adaptive Growth in LLM Auto-Prompting

Xiangyu Zhao<sup>a</sup> and Chengqian Ma<sup>a</sup>

<sup>a</sup>Xiamen University

ORCID (Xiangyu Zhao): <https://orcid.org/0009-0007-0903-0352>, ORCID (Chengqian Ma): <https://orcid.org/0009-0005-9269-3880>

**Abstract.** Large Language Models (LLMs) exhibit remarkable proficiency in addressing a diverse array of tasks within the Natural Language Processing (NLP) domain, with various prompt design strategies significantly augmenting their capabilities. However, these prompts, while beneficial, each possess inherent limitations. The primary prompt design methodologies are twofold: The first, exemplified by the Chain of Thought (CoT), involves manually crafting prompts specific to individual datasets, hence termed **Expert-Designed Prompts (EDPs)**. Once these prompts are established, they are unalterable, and their effectiveness is capped by the expertise of the human designers. When applied to LLMs, the static nature of EDPs results in a uniform approach to both simple and complex problems within the same dataset, leading to the inefficient use of tokens for straightforward issues. The second method involves prompts autonomously generated by the LLM, known as **LLM-Derived Prompts (LDPs)**, which provide tailored solutions to specific problems, mitigating the limitations of EDPs. However, LDPs may encounter a decline in performance when tackling complex problems due to the potential for error accumulation during the solution planning process. To address these challenges, we have conceived a novel **Prompt Recursive Search (PRS)** framework that leverages the LLM to generate solutions specific to the problem, thereby conserving tokens. The framework incorporates an assessment of problem complexity and an adjustable structure, ensuring a reduction in the likelihood of errors. We have substantiated the efficacy of PRS framework through extensive experiments using LLMs with different numbers of parameters across a spectrum of datasets in various domains. Compared to the CoT method, the PRS method has increased the accuracy on the BBH dataset by 8% using Llama3-7B model, achieving a 22% improvement.

**Table 1.** Analysis of prompt methodologies in relation to their capabilities, utilizing the following symbols: “★” for comprehensive support, “☆” for limited support, and “✗” for absence of support. CE: Computational Resource Utilization Efficiency, IH: Independence from Human Expertise, SE: Supervision of Errors in the Inference Process.

Scheme	CE	IH	SE
Expert-Designed Prompt [26, 24, 27, 3]	✗	✗	★
LLM-Derived Prompt [18, 10, 34]	★	★	✗
Prompt Recursive Search	★	★	★

## 1 Introduction

Stem cells differentiate into other types of cells with distinct functions upon induction by signaling molecules[2, 20]. A similar process, as depicted in 2, is proposed by us in the use of Large Language Models (LLMs) to solve problems. Previous work[26, 9, 31, 30, 13] has indicated that a single interaction often fails to adequately address problems. Instead, a more effective approach is to divide the problem-solving process into multiple steps: first proposing a solution to the problem, and then executing it step by step by the LLM. Each interaction with the LLM involves input to the model and its subsequent output. A single interaction can lead to the refinement, detailing, summarization, or execution of a solution. Each interaction corresponds to the realization of a thought within the thoughts that make up the solution. The functionality of a thought evolves from a more basic state to become increasingly specific, much like how a stem cell differentiates into a cell with a particular function. Similarly, an original thought can differentiate in response to various problems, giving rise to thoughts capable of solving different issues.

Control over the evolution of thoughts traditionally involves two methods. The first method entrusts the transformation process of thoughts entirely to experienced domain experts. These experts can, based on their understanding of problems within a particular field, plan the problem-solving process in a sequential chain, addressing the problem step by step. This is the contribution of the Chain of Thought (CoT) approach[26]. Prior to this, the practice of presenting a problem to LLMs and expecting a direct answer placed excessive demands on the LLMs, yielding suboptimal results and failing to fully leverage LLM’s potential. Given that there is often more than one method to solve a problem, and multiple approaches can yield answers for a specific issue, we can compare the answers derived from various methods to select the most optimal one, thus obtaining the best solution to the problem. Plan-and-Solve (PS)[24] Prompting is a concrete implementation of this line of thinking. Building upon this, the Tree of Thought (ToT)[27] further extends this work by considering a scenario where a thought represents a critical step in the solution. If this thought is flawed, continuing to reason based on it can lead to more severe errors. Therefore, the authors of ToT propose a prompt structure that allows for a retreat from an erroneous solution path to reconsider and pursue the correct approach. This structure is hierarchical, and the feature is known as the backtracking function during the tree traversal process. On the foundation of ToT, the Graph of Thought (GoT) [3] structure is even more flexible. By performing operations such as aggregation and refinement on

multiple thoughts, the expert-designed prompt structure represented by GoT is highly fault-tolerant. The outcome of one thought can be verified by multiple thoughts, and the knowledge derived from multiple thoughts can be integrated into a single thought.

We refer to the prompt design structures represented by CoT (Chain of Thought), ToT (Tree of Thought), and GoT (Graph of Thought) as **Expert-Designed Prompts (EDP)**: These prompt frameworks are manually crafted, thus necessitating a substantial amount of human expert experience. The approach demands a high level of expertise from the designer, who must draw on their professional experience and conduct multiple experiments to derive a prompt structure capable of addressing various scenarios within a particular domain. Given that this method accounts for a multitude of potential prompt structures, it is universally effective for all problems, which may result in a complex structure. While this approach often excels in tackling complex issues, it can be inefficient for simpler problems, as many steps within the prompt structure become redundant, leading to unnecessary computational resource wastage. Furthermore, if the human expert responsible for designing the prompt lacks sufficient proficiency or fails to consider all possible scenarios comprehensively, the static nature of the prompt, once designed, can lead to the persistent presence of any inherent flaws in the structure during each application.

To mitigate the dependency of Large Language Model (LLM) prompt design on human experience, a second approach to prompt design has been proposed. The primary issues with human-engineered prompt structures are twofold: Firstly, due to the limitations of human cognition, human experts cannot fully account for all scenarios within a domain. Secondly, once a prompt structure is designed by a human expert, it becomes immutable, which can lead to the waste of substantial computational resources when addressing simple problems, as the comprehensively complex structure, initially a strength, becomes redundant. However, entrusting the task of prompt structure design to an LLM addresses these concerns. Given the vast training corpora of large language models, their breadth of knowledge can surpass that of individual human experts. Additionally, the prompt framework design functionality provided by LLMs can commence upon the presentation of a specific problem, thereby being tailored to that particular issue. For problems that are straightforward and can be resolved in a single step, the LLM-generated prompt structure will not incur the unnecessary computational resource expenditure associated with the redundant steps found in human expert-designed prompt structures.

In the work of Automatic Prompt Engineering (APE)[34], the Large Language Model (LLM) initially proposes a set of prompts that encompass solutions to specific problems. The quality of these prompts varies, leading to a selection process where lower-quality prompts are discarded, and higher-quality ones are retained. Then LLM can use the higher-quality prompts as references to generate similar ones, enhancing the diversity of the prompt collection. This iterative process continues until satisfactory prompts are obtained. During the prompt generation by the LLM, the model can act not only as a problem solver providing solutions to domain-specific questions but also as a question generator, posing some extreme or marginal questions within a domain. These questions can be utilized to optimize the prompts, thereby improving their quality. Building on this approach, Intent-based Prompt Calibration (IPC)[10] optimizes the prompts using these edge-case questions after designing the prompts and related questions, ensuring the quality of the prompts by leveraging these domain-marginal issues. Following the proposal of prompts by the LLM, to ensure their quality, we can

employ methods mentioned in the first two approaches: evaluating and discarding prompts with lower quality, or constraining them with edge-case questions. Additionally, we can utilize the Automatic Prompt Optimization[18] method: composing all possible prompts into a prompt space, using a prompt generated by the LLM as the starting point within this space, and optimizing the prompt based on its evaluation from LLM as the textual gradient. Through repeated iterations, the optimal solution can be found within the prompt space, employing a textual gradient descent method to identify the most ideal solution. The solutions proposed by the LLM, in conjunction with the problem context or textual gradient, often achieve effects comparable to those of human-designed prompts, which we term as **LLM-Derived Prompts (LDP)**.

However, when dealing with excessively complex problems, such as large-scale planning issues, the complete delegation of the planning of solutions to the LLM can become problematic. With numerous steps involved, there is a high likelihood of error propagation and accumulation. A minor error at any stage can be rapidly magnified across multiple steps, leading to suboptimal performance by the LLM.

Expert-Designed Prompts (EDPs) often necessitate a more complex structure to address all scenarios within a dataset, which can result in redundant steps when EDPs are applied to solve simpler problems within that dataset. On the other hand, LLM-Derived Prompts (LDPs) delegate the entire task of prompt design to the Large Language Model (LLM). During the process of generating solutions to problems, the LLM is highly susceptible to the accumulation of errors, which significantly increases the difficulty of accurately generating solutions for complex tasks.

We have identified that the characteristics of Expert-Designed Prompts (EDPs) and LLM-Derived Prompts (LDPs) are, to a certain extent, complementary. Consequently, after thorough consideration of the advantages and disadvantages of both EDPs and LDPs, we propose an entirely new **Prompt Recursive Search(PRS)** framework for prompt design which is depicted in 2. This framework enables Large Language Models (LLMs) to avoid overly complex planning when tackling simplistic problems, while also allowing LLMs to undertake a portion of the prompt design work, thus reducing the high dependency on human experts. Drawing inspiration from the differentiation process of human stem cells: when the body, prompted by signaling molecules, requires cells to perform a specific task, stem cells differentiate into new cells with distinct characteristics to address issues that other cells cannot. Our prompt framework, when confronted with a new problem, first evaluates the problem and has the LLM assign a complexity score. If the problem's complexity is deemed too high, we break down the problem into multiple steps for resolution. The simplified problems, post-dissection, become new targets, for which the LLM provides solutions. This is akin to allowing the LLM to complete a differentiation design process for the prompt, resulting in prompts with specific functionalities. During the differentiation steps, we recursively apply this differentiation function until the problem has been sufficiently broken down into simple components.

By employing LLMs with varying numbers of parameters, we compared the PRS method with traditional prompt design approaches on the BBH dataset, which spans multiple domains, thereby validating the effectiveness of the PRS method. Through this paper, we have made the following contributions:

- We have categorized existing prompts based on their source of acquisition into two types: Expert-Designed Prompts (EDP) and

If you follow these instructions, do you return to the starting point? Always face forward. Take 7 steps backward. Take 10 steps right. Take 4 steps forward. Take 1 step left. Options: Yes. No.  
Assess the complexity of the following question, then only reply "The complexity score is [s].", where s is an integer from 1 to 10.

 The complexity score is 6.

 Let's first understand the following question, and then devise a plan consisting of a total of 3 steps to solve it then assess the complexity of these plans. Your output must be of the following format.



1. Define Initial Orientation: Assume a forward direction (e.g., north) as the starting point with coordinates (0, 0).

2. Simulate Movement: Translate each instruction into coordinate changes: "backward" as subtraction from the y-coordinate, "right" as addition to the x-coordinate, "forward" as addition to the y-coordinate, and "left" as subtraction from the x-coordinate.

3. Analyze Final Position: Calculate the final coordinates after all movements and determine if they match the starting point (0, 0) to conclude if the path leads back to the origin.



1. Initial Position: (0, 0).  
 2. Take 7 steps backward: Position becomes  $(0, 0 - 7) = (0, -7)$ .  
 3. Take 10 steps right: Position becomes  $(0 + 10, -7) = (10, -7)$ .  
 4. Take 4 steps forward: Position becomes  $(10, -7 + 4) = (10, -3)$ .  
 5. Take 1 step left: Position becomes  $(10 - 1, -3) = (9, -3)$ .  
 6. The final position is (9, -3), which is not the same as the starting point (0, 0). Therefore, the answer is: No.



2.1 Subtract 7 from the y-coordinate to simulate taking 7 steps backward.  
 2.2 Add 10 to the x-coordinate to simulate taking 10 steps to the right.  
 2.3 Add 4 to the y-coordinate to simulate taking 4 steps forward.  
 2.4 Subtract 1 from the x-coordinate to simulate taking 1 step to the left.

**Final answer:**  
 1-order solution;  
 2-order solution;  
 Fixed prompt;  
 Unfixed prompt.

**Figure 1.** Example of our proposed method Prompt Recursive Search with complexity score limitation (PRS).

LLM-Derived Prompts (LDP);

- By introducing biological principles, we have proposed an automated framework for addressing complex problems that integrates the advantages of both EDP and LDP.
- Through validation on the BBH[21] dataset across multiple domains using models with parameters spanning different orders of magnitude, we have confirmed the effectiveness of this method.

## 2 Related Work

### 2.1 LLM-Derived Prompts

Large Language Models (LLMs) are an advanced tool in the field of Natural Language Processing (NLP) technology[32, 33]. They are based on deep learning architectures, such as Transformers[22], and have learned complex patterns of language through extensive training on vast amounts of text data. These models are not only capable of understanding the nuances of language but also generating coherent and grammatically correct text, thereby performing well on a variety of language tasks. They can be used in a wide range of applications, including chat-bots[17], recommendation systems[12], content creation aids[5], and educational tools[16, 14]. However, LLMs may also replicate and amplify biases present in their training data, so ethical[4, 7, 1, 14] and bias[28, 15] considerations must be taken into account in their design and use. Notable examples of LLMs include the BERT[6] and GPT[19] models, which have achieved significant success in tasks involving natural language understanding[19, 23] and generation[11, 8].

### 2.2 Expert-Designed Prompts

#### 2.2.1 Chain of Thought(CoT)

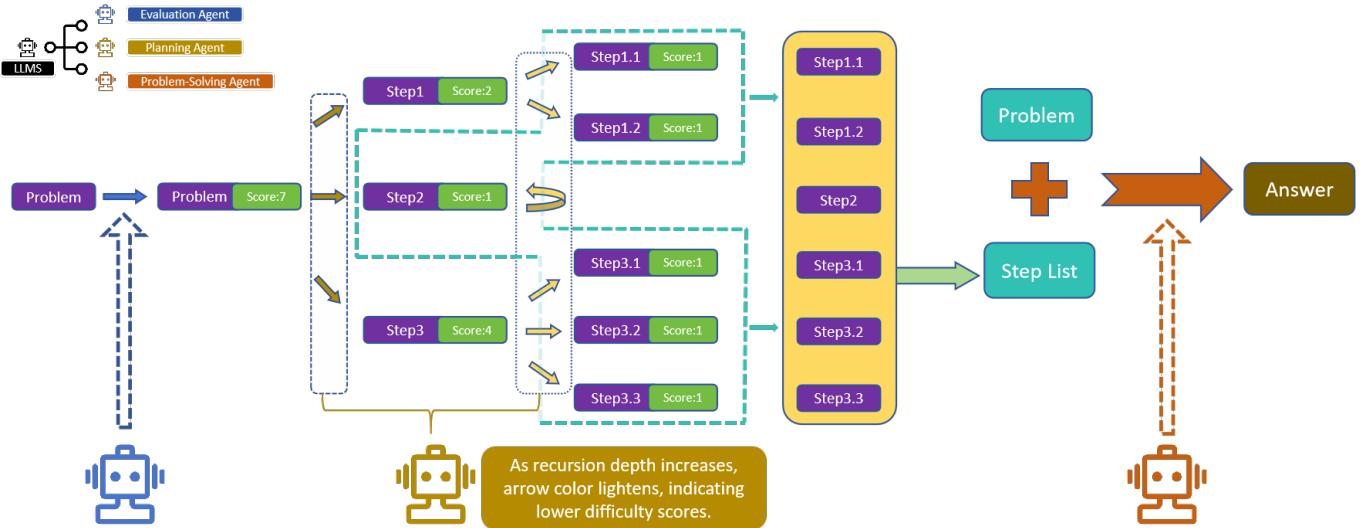
This technology is designed to enhance the ability of Large Language Models (LLMs) to handle complex issues. It simulates the continuous thought process of human problem-solving by embedding a series of logical reasoning steps within the input prompts. These steps progressively guide the model to the solution of the problem, aiding in the processing of tasks that require continuous logic or mathematical computation. By demonstrating examples of problem-solving, the accuracy and logicality of the model's output can be improved, even without specific task training.

#### 2.2.2 Plan-and-Solve (PS) Prompting

Compared with Chain of Thought (CoT), Plan-and-Solve Prompting significantly improves the performance of Large Language Models (LLMs) in multi-step reasoning tasks by guiding the models to formulate plans for solutions and then execute them. It has demonstrated superior performance over Zero-shot-CoT thinking across multiple datasets and is comparable to CoT methods that require manual examples, showcasing the potential to stimulate the reasoning capabilities of LLMs without the need for manual examples.

#### 2.2.3 Tree of Thoughts

The Tree of Thought (ToT) framework aims to enhance the capabilities of Large Language Models (LLMs) in problem-solving. It allows the model to explore various reasoning paths and to self-assess its decisions, leading to more deliberate choices. This approach views problem-solving as a search process within a "thought tree," where



**Figure 2.** Pipeline of our proposed method Prompt Recursive Search with complexity score limitation (PRS).

each "thought" is a coherent text block that represents a step towards the solution.

#### 2.2.4 Graph of Thoughts

Graph of Thoughts (GoT) significantly enhances the performance of Large Language Models (LLMs) in complex task processing by conceptualizing the reasoning process as a graph structure composed of nodes, which represent the model's "thoughts," and edges, which denote the connections between these thoughts. Not only does GoT improve the quality of task execution, with a 62% increase in efficiency over existing technologies in sorting tasks, but it also achieves a reduction in costs by more than 31%. The design of GoT greatly facilitates the approximation of LLMs' reasoning capabilities to human thought patterns.

#### 2.2.5 Automatic Prompt Engineer(APE)

The manual design of prompts necessitates a high level of expertise from the designer and is constrained by the relatively limited knowledge base and subjective factors of human experts compared to Large Language Models (LLMs). Handcrafted prompts are subject to numerous limitations stemming from human factors, which LLMs are not bound by. Therefore, entrusting the task of prompt design to an LLM can both resolve these constraints and eliminate the inconvenience of manual prompt design, thereby automating the use of LLMs. Initially, the LLM is tasked with generating a set of prompts, which are then not utilized in their entirety. Instead, these prompts are evaluated and the highest-quality ones are selected. Based on the highest-quality prompts, a variety of similar prompts are generated to enhance their diversity.

#### 2.2.6 Intent-based Prompt Calibration(IPC)

Large Language Models (LLMs) are highly sensitive to the input content, where even seemingly irrelevant random noise can significantly alter the LLM's output. The root cause of this sensitivity is a lack of an accurate and comprehensive understanding of the problem. To ensure that LLMs consider the problem thoroughly, based on the user's requirements and the initial prompt, new prompts and data samples are iteratively generated. The prompts are then optimized using edge data, allowing the LLM to fully contemplate the problem.

This operation of generating data samples opens up new avenues for prompt design: not only can we have LLMs generate answers for us, but we can also have LLMs generate questions that can, in turn, lead to better answers.

#### 2.2.7 Automatic Prompt Optimization

After the generation of prompts by a LLM, it is often necessary to refine them. This refinement can be accomplished through simple ranking and selection, known as Approach by APE, or by making modifications to the prompt based on the data provided by the LLM, referred to as IPC. However, when the objective is to identify the most appropriate prompt within a dense space of prompts, gradient descent emerges as a more rational method. The process begins with the LLM generating a response based on an initial prompt. Subsequently, the LLM analyzes the response in conjunction with the prompt to produce Textual Gradients. These Textual Gradients are then integrated to formulate a new prompt. This cycle of generation and refinement is repeated iteratively until the optimal prompt is achieved.

### 3 Methodology

Our approach is founded upon the principles of the EDP and the LDP, taking into account that while EDP is adept at addressing straightforward problems with a comprehensive framework designed to encompass all scenarios of such issues, it may necessitate superfluous token expenditure when applied to simpler problems. On the other hand, LDP can lead to the accumulation of errors; if an LLM is tasked with designing prompts without constraints or corrections, it must adhere to a process that involves planning a solution and then executing it. Errors introduced during the planning phase can propagate through to subsequent steps, potentially undermining the problem-solving process. To mitigate the costs associated with manually designed prompts, we delegate the responsibility of devising solutions to the LLM itself. The LLM will design tailored solutions for each problem, thereby avoiding unnecessary token usage. Furthermore, to prevent the accumulation of errors during the LLM's prompt design phase, we employ complexity detection methods and procedural planning techniques to constrain the LLM and validate its solution planning. Our method is showed in 1 and 2.

---

**Algorithm 1:** Design Solution

---

**Input:** Problem P, Steps Steps  
**Output:** Solution Solution

```
1 Solution ← []
2 Initial_Steps ← Get_Initial_Solution(P, Steps)
3 foreach Step in Initial_Steps do
4   Complexity ← Evaluate_Complexity(Step)
5   if Complexity > Threshold then
6     New_Steps ← Calculate_Steps(Complexity)
7     Solution ←
8       Solution + Design_Solution(Step, New_Steps)
9   else
10    | Solution ← Solution + [Step]
11 end
12 return Solution
```

---

**Algorithm 2:** PRS Pipeline

---

**Input:** Problem description  $P_{desc}$   
**Output:** Answer Ans

```
// Evaluate Complexity
1 C ← Evaluate_Complexity( $P_{desc}$ )
// Calculate Steps
2 Steps ← Calculate_Steps(C)
// Design Solution
3 Sol ← Design_Solution( $P_{desc}$ , Steps)
// Get Answer
4 Ans ← Get_Answer( $P_{desc}$ , Sol)
5 return Ans
```

---

### 3.1 Evaluating the Complexity of a Problem

Before devising solutions for a problem, we first assess its complexity to facilitate the design of more comprehensive solutions for more complex issues and more straightforward solutions for simpler ones. The complexity level can provide effective guidance for formulating our solutions. In the process of evaluating the complexity of a problem by a Large Language Model (LLM), we consider the characteristics of textual descriptions of problems: directly describing the complexity using natural language can lead to an inability to distinctly differentiate between problems based on this metric. Therefore, we opt to use numerical values instead of the original complexity descriptions. We categorize the complexity of problems into ten levels, represented by ten integers ranging from 1 to 10.

### 3.2 Planning Solution for a Problem

Upon determining the complexity of a problem, our focus shifts to designing solutions based on that complexity. In practice, the aspect of a solution that is directly related to complexity is the number of steps involved. Through extensive experimentation, we have correlated the complexity levels proposed by the Large Language Model (LLM) for problems with the steps required in their solutions. Our findings indicate a linear relationship: the number of steps to resolve a problem typically ranges between one and five, and the complexity level of a problem, when divided by two, often yields the number of steps needed for its resolution. This macroscopic statistical pattern allows us to seamlessly incorporate the initial complexity assessment into the design of solutions, thereby specifying the number of steps required for resolution.

### 3.3 Recursively Proposing Solutions to Problems

Following the first two steps, we have established the necessary steps for a solution, which are determined based on the complexity of the problem. This approach, underpinned by our discovered relationship between complexity levels and solution steps, mitigates the accumulation of errors during the problem formulation by the LLM. It addresses the shortcomings of the LLM-Derived Prompt (LDP) while fully leveraging its strengths in devising tailored solutions for specific problems, thereby also resolving the issues associated with the Expert-Designed Prompt (EDP).

### 3.4 Obtaining a complete solution

Subsequently, we can employ the identified solution steps to propose resolutions. Recognizing that providing detailed steps for complex problems in a single response demands a high level of proficiency from the LLM, we adopt a recursive strategy for the more intricate steps of the solution. Each complex step is treated as a new problem to be broken down, and a more refined solution is proposed, prompting the LLM to elaborate on the specifics of the solution. This recursive process continues until all solutions are simplified to the point where they can be resolved in a single step. By transforming complex problems into simpler ones, we prevent the LLM from being overwhelmed by complexity and ensure the provision of accurate solutions.

### 3.5 Obtaining an Answer according to the solution

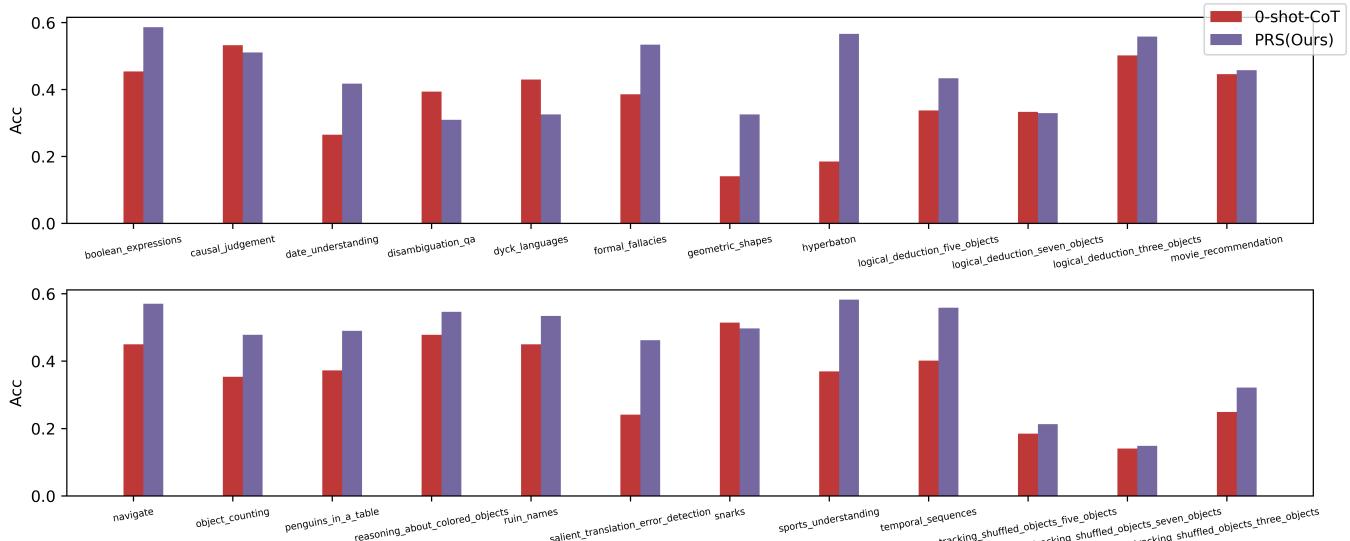
After obtaining a solution through the preceding plan, we can then apply this solution to address the problem. Additionally, we can impose constraints on the format of the answers outputted by the Large Language Model (LLM) with respect to specific datasets, ensuring the production of coherent and appropriate responses.

## 4 Experiments

### 4.1 Model Selection

In the process of model selection, considering that Large Language Models (LLMs) with a substantial number of parameters inherently possess higher performance, the advent of prompt design methodologies has been predominantly aimed at enhancing the capabilities of LLMs with moderate to smaller parameter counts. Moreover, the efficacy of prompt design methods is more pronounced in LLMs with smaller parameter volumes. We can enhance the performance of LLMs by designing prompts, fully tapping into the potential of LLMs; however, LLMs with a larger number of parameters already possess relatively good capabilities, and even with the use of prompt techniques, the improvement in LLM performance may not be very significant. We will demonstrate this point through the results of ablation experiments. Consequently, we have elected to utilize the Yi-34B model, which has a moderately sized parameter volume, alongside the Meta-Llama-3-8B model, which features a smaller parameter count.

**Yi-34B** The Yi-34B model[29] is a middle-scale language model trained on a diverse corpus of 3 terabytes, designed with a bilingual focus. It has demonstrated significant potential across various aspects such as language comprehension, common sense reasoning, and reading comprehension, indicative of the emergent capabilities associated with large models[25]. This emergent capability implies that LLMs can exhibit human-like logical thinking, reasoning, and



**Figure 3.** Comparison between PRS (ours) and 0-shot-cot on the BBH dataset.

analytical abilities, which are precisely the skills that LLMs need to engage in reasoning with the aid of prompts. Suitable for a multitude of applications, the Yi-34B model is fully accessible for academic research and concurrently offers free commercial licensing upon application.

**Meta-Llama-3-8B** The Llama-3 model represents an auto-regressive language model engineered with an enhanced Transformer [22] architecture. It features refined iterations that employ supervised fine-tuning (SFT) coupled with reinforcement learning augmented by human feedback (RLHF), ensuring alignment with human-centric values of utility and safety. Llama 3 has been pre-trained on an extensive corpus exceeding 15 trillion tokens, sourced from a variety of public repositories. For fine-tuning, it leverages a compilation of publicly accessible instructional datasets alongside a substantial collection of over 10 million examples annotated by humans.

## 4.2 Dataset Introduction

The BIG-Bench Hard (BBH) is a subset culled from the original BIG-Bench assessment suite, focusing on tasks that pose a challenge to existing language models. BBH comprises 23 tasks and 27 sub-datasets, and when creating the BBH dataset, researchers adhered to specific filtering criteria, including the number of examples in the task, the presence of human rater performance data, the type of task, and the performance of previous models, among others. This dataset is designed to drive improvements in language model performance on complex reasoning tasks and provides a valuable benchmark for future research. This dataset is relatively difficult and covers a wide range of topics, with many of its sub-datasets focusing on assessing the reasoning capabilities of LLMs, making it an ideal tool to test the abilities of PRS.

## 4.3 Set Up

We have confirmed the linear relationship between the complexity of a problem and the number of steps required to solve it through experiments conducted across multiple datasets. On the Llama3 model and the BBH dataset, this linear relationship is specifically manifested as the number of steps required to solve a problem being half of its complexity. Therefore, once we have ascertained the complexity level of

a problem via a Large Language Model (LLM), we utilize this linear relationship to translate the complexity into the corresponding steps necessary for problem resolution.

## 4.4 Evaluation Metrics.

After obtaining the answer from the Large Language Model (LLM), it is necessary to compare it with the correct answer to assess its accuracy and calculate the rate of correctness. There are two primary methods for comparison: The first method involves the LLM itself comparing the provided answer to the correct one and then making a judgment. This approach is contingent on the performance of the LLM, which carries the potential for error. The second method is applicable only when the correct answers in the dataset adhere to a more uniform and standardized format. In this case, we simply instruct the LLM to pay attention to the format when presenting the answer. Subsequently, we can employ regular expressions to extract the answer and use string methods for comparison. This method demands less from the LLM's performance but has a more limited range of applicability. The answer format for the problems in the BBH dataset is relatively standardized, therefore we can employ the second method, which is more accurate for making judgments.

Since the correct answers in the dataset we used have specific formatting requirements, we provided the LLM with the answer to the first sample in the dataset as a template, instructing the LLM to mimic the format of the first answer when responding. The evaluation of the LLM's responses begins with the second sample, ensuring that the format of the LLM's answers is maintained while preventing data leakage.

Before determining the correctness of the answers, we categorized the types of answers in the dataset, dividing them into single-choice, multiple-choice, numerical, etc., and formulated appropriate regular expressions to match the answers accordingly.

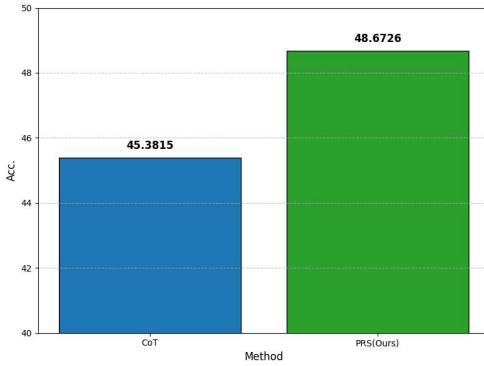
## 4.5 Result

We conducted extensive experiments on the BBH dataset, where out of its 27 sub-datasets, only 24 were found to be valuable for exploration. The "dyck\_languages" and "multistep\_arithmetic\_two" tasks, which were discarded, primarily assess the LLM's ability to discern

symbols, which is unrelated to the performance of the prompt. PRS outperformed CoT in 19 out of 25 sub-datasets, the accuracy rate improved from 36% with CoT to 44% with PRS, which is an increase of 22%.

#### 4.6 Ablation Experiment

By applying this framework to the Yi-34B model, we also compared the CoT and PRS methods and found that the average accuracy of PRS increased from 45% with the CoT method to 49%, achieving a 9% improvement. This demonstrates that our method is also applicable to LLMs with a larger number of parameters and also validates our viewpoint that the enhancement effect of prompt design methods on LLMs with large parameter volumes is relatively less pronounced.



**Figure 4.** The ablation experiments based on the **Yi-34B** model on the BBH dataset. The results represent metrics on the formal\_fallacies subset.

## 5 Conclusion

Traditional prompt design methods can be divided into LDP and EDP, where EDP heavily relies on the subjective experience of human experts. Its immutable characteristics once involved make it waste redundant computational resources when dealing with simple problems. On the other hand, LDP is generated and optimized by LLMs, but due to the lack of effective supervision during the prompt generation process, it is prone to error accumulation. We have integrated the advantages of both by designing a brand-new prompt framework called PRS. It eliminates the high dependence on human expert knowledge through automatic prompt design and saves computational resources. By effectively supervising the complexity of the problem, it prevents error accumulation during the reasoning process. We have demonstrated the effectiveness of this framework by comparing its performance with the CoT method using the Llama3-7B model across multiple domain datasets, as well as with the ablation experiments using the Yi-34 model.

## 6 Limitations

By summarizing traditional prompt design techniques, we have categorized traditional prompt design methods into EDP (Expert-Designed Prompts) and LDP (LLM-Derived Prompts). Upon discovering the complementary nature of the advantages between the two, inspired by the differentiation phenomenon of stem cells in biological principles, we have innovatively proposed a new prompt structure: Prompt Recursive Search (PRS). This structure can automati-

cally design prompts for problems, which has the advantage of saving computational resources. At the same time, by judging the complexity of the problem, it supervises the errors in the prompt design process, thereby ensuring accuracy. Through experiments in multiple domains and with various models, we have demonstrated the effectiveness of PRS. However, it is undeniable that our framework still has certain shortcomings, which are as follows:

1. We mandate that the Large Language Model (LLM) outputs answers in a specific format through our prompt, as deviation from this format would render the true-false judgment infeasible. However, the LLM is not always compliant with the prescribed output format, and answers that do not conform to the format could be either erroneous or correct. Our condition for deeming an LLM-provided answer as correct stipulates that not only must the content be accurate, but the format must also be correct. This essentially elevates the criteria for acceptable responses. Situations where the content is correct but the format is not are not accounted for in our assessment, leading to an underestimation of our true accuracy rate in the statistics we compile.
2. Our framework is predicated on a crucial assumption: the complexity of a problem is directly proportional to the number of steps required to solve it. We have substantiated this intuitive assumption through numerous experiments, yet it is important to recognize that this assumption represents a macroscopic rule and there are a few exceptional cases that do not conform to it. Consequently, the number of steps necessary to resolve a problem remains a topic worthy of exploration. We thus leave this question for future research endeavors.
3. Even for the same input, the responses from the Large Language Model (LLM) can vary with each invocation, thus reflecting that the performance of the Prompt Recursive Search (PRS) framework has an inherent degree of randomness. The experimental results presented in this paper are the average of three trials, which may still differ from the true capabilities of the method.
4. Due to the inherently strong capabilities of Large Language Models (LLMs) with a larger number of parameters, applying the Prompt Recursive Search (PRS) method proposed in this paper does not yield a significant enhancement in this kind LLM's abilities. In fact, the effectiveness of this framework is contingent upon the LLM's analytical capacity regarding the problem at hand. Consequently, LLMs with a larger parameter count are better suited to leverage the full potential of the PRS framework. Our ablation study proved this perspective but did not delve into a detailed investigation. We reserve the exploration of this aspect for future work.

In summary, we have proposed a brand-new method for prompt design and have validated it through extensive experiments, providing a starting point for future research.

## Acknowledgements

By using the `ack` environment to insert your (optional) acknowledgements, you can ensure that the text is suppressed whenever you use the `doubleblind` option. In the final version, acknowledgements may be included on the extra page intended for references.

## References

- [1] J. Bang, B.-T. Lee, and P. Park. Examination of ethical principles for llm-based recommendations in conversational ai. *2023 International Conference on Platform Technology and Service (PlatCon)*, pages 109–113, 2023. URL <https://api.semanticscholar.org/CorpusID:262977578>.
- [2] A. Becker, E. A. McCulloch, and J. E. Till. Cytological demonstration of the clonal nature of spleen colonies derived from transplanted mouse marrow cells. *Nature*, 197:452–454, 1963. URL <https://api.semanticscholar.org/CorpusID:11106827>.
- [3] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawska, L. Gianiuzzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczek, and T. Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In M. J. Wooldridge, J. G. Dy, and S. Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20–27, 2024, Vancouver, Canada*, pages 17682–17690. AAAI Press, 2024. doi: 10.1609/AAAI.V38I16.29720. URL <https://doi.org/10.1609/aaai.v38i16.29720>.
- [4] J. Cabrera, M. S. Loyola, I. Magaña, and R. Rojas. Ethical dilemmas, mental health, artificial intelligence, and llm-based chatbots. In *International Work-Conference on Bioinformatics and Biomedical Engineering*, 2023. URL <https://api.semanticscholar.org/CorpusID:259335839>.
- [5] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun. A comprehensive survey of ai-generated content (aige): A history of generative ai from gan to chatgpt. *ArXiv*, abs/2303.04226, 2023. URL <https://api.semanticscholar.org/CorpusID:257405349>.
- [6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL <https://api.semanticscholar.org/CorpusID:52967399>.
- [7] L. Goetz, M. Trengove, A. A. Trotsuk, and C. A. Federico. Unreliable llm bioethics assistants: Ethical and pedagogical risks. *The American Journal of Bioethics*, 23:89–91, 2023. URL <https://api.semanticscholar.org/CorpusID:263774936>.
- [8] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. Bang, D. Chen, W. Dai, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55:1 – 38, 2022. URL <https://api.semanticscholar.org/CorpusID:246652372>.
- [9] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *ArXiv*, abs/2205.11916, 2022. URL <https://api.semanticscholar.org/CorpusID:249017743>.
- [10] E. Levi, E. Brosh, and M. Friedmann. Intent-based prompt calibration: Enhancing prompt optimization with synthetic boundary cases. *ArXiv*, abs/2402.03099, 2024. URL <https://api.semanticscholar.org/CorpusID:267412105>.
- [11] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. rahman Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Annual Meeting of the Association for Computational Linguistics*, 2019. URL <https://api.semanticscholar.org/CorpusID:204960716>.
- [12] Q. Liu, N. Chen, T. Sakai, and X.-M. Wu. A first look at llm-powered generative news recommendation. *ArXiv*, abs/2305.06566, 2023. URL <https://api.semanticscholar.org/CorpusID:263891105>.
- [13] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Welleck, B. P. Majumder, S. Gupta, A. Yazdanbakhsh, and P. Clark. Self-refine: Iterative refinement with self-feedback. *ArXiv*, abs/2303.17651, 2023. URL <https://api.semanticscholar.org/CorpusID:257900871>.
- [14] G. F. N. Mvondo, B. Niu, and S. Eivazinezhad. Generative conversational ai and academic integrity: A mixed method investigation to understand the ethical use of llm chatbots in higher education. *SSRN Electronic Journal*, 2023. URL <https://api.semanticscholar.org/CorpusID:261311676>.
- [15] A. F. Oketunji, M. Anas, and D. Saina. Large language model (llm) bias index - llmbi. *ArXiv*, abs/2312.14769, 2023. URL <https://api.semanticscholar.org/CorpusID:266521434>.
- [16] M. S. Orenstrakh, O. Karnalim, C. A. Suárez, and M. Liut. Detecting llm-generated text in computing education: A comparative study for chatgpt cases. *ArXiv*, abs/2307.07411, 2023. URL <https://api.semanticscholar.org/CorpusID:259924631>.
- [17] S. Peng, W. Swiatek, A. Gao, P. Cullinan, and H. Chang. Ai revolution on chat bot: Evidence from a randomized controlled experiment. *ArXiv*, abs/2401.10956, 2024. URL <https://api.semanticscholar.org/CorpusID:267068546>.
- [18] R. Pryzant, D. Iter, J. Li, Y. T. Lee, C. Zhu, and M. Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Conference on Empirical Methods in Natural Language Processing*, 2023. URL <https://api.semanticscholar.org/CorpusID:258546785>.
- [19] A. Radford and K. Narasimhan. Improving language understanding by generative pre-training. 2018. URL <https://api.semanticscholar.org/CorpusID:49313245>.
- [20] L. Siminovitch, E. A. McCulloch, and J. E. Till. The distribution of colony-forming cells among spleen colonies. *Journal of cellular and comparative physiology*, 62:327–36, 1963. URL <https://api.semanticscholar.org/CorpusID:43875977>.
- [21] M. Suzgun, N. Scales, N. Scharli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. hsin Chi, D. Zhou, and J. Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Annual Meeting of the Association for Computational Linguistics*, 2022. URL <https://api.semanticscholar.org/CorpusID:252917648>.
- [22] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. URL <https://api.semanticscholar.org/CorpusID:13756489>.
- [23] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP@EMNLP*, 2018. URL <https://api.semanticscholar.org/CorpusID:5034059>.
- [24] L. Wang, W. Xu, Y. Lan, Z. Hu, Y. Lan, R. K.-W. Lee, and E.-P. Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Annual Meeting of the Association for Computational Linguistics*, 2023. URL <https://api.semanticscholar.org/CorpusID:258558102>.
- [25] Z. Wang, S. Mao, W. Wu, T. Ge, F. Wei, and H. Ji. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. 2023. URL <https://api.semanticscholar.org/CorpusID:259765919>.
- [26] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. hsin Chi, F. Xia, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903, 2022. URL <https://api.semanticscholar.org/CorpusID:246411621>.
- [27] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *ArXiv*, abs/2305.10601, 2023. URL <https://api.semanticscholar.org/CorpusID:258762525>.
- [28] K.-C. Yeh, J.-A. Chi, D.-C. Lian, and S.-K. Hsieh. Evaluating interfaced llm bias. In *Taiwan Conference on Computational Linguistics and Speech Processing*, 2023. URL <https://api.semanticscholar.org/CorpusID:264555752>.
- [29] A. A. Young, B. Chen, C. Li, C. Huang, G. Zhang, G. Zhang, H. Li, J. Zhu, J. Chen, J. Chang, K. Yu, P. Liu, Q. Liu, S. Yue, S. Yang, S. Yang, T. Yu, W. Xie, W. Huang, X. Hu, X. Ren, X. Niu, P. Nie, Y. Xu, Y. Liu, Y. Wang, Y. Cai, Z. Gu, Z. Liu, and Z. Dai. Yi: Open foundation models by 01.ai. *ArXiv*, abs/2403.04652, 2024. URL <https://api.semanticscholar.org/CorpusID:268264158>.
- [30] J. Yu, R. He, and R. Ying. Thought propagation: An analogical approach to complex reasoning with large language models. *ArXiv*, abs/2310.03965, 2023. URL <https://api.semanticscholar.org/CorpusID:263831197>.
- [31] Z. Zhang, A. Zhang, M. Li, and A. J. Smola. Automatic chain of thought prompting in large language models. *ArXiv*, abs/2210.03493, 2022. URL <https://api.semanticscholar.org/CorpusID:252762275>.
- [32] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, and M. Du. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 2023. URL <https://api.semanticscholar.org/CorpusID:261530292>.
- [33] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J. Nie, and J. rong Wen. A survey of large language models. *ArXiv*, abs/2303.18223, 2023. URL <https://api.semanticscholar.org/CorpusID:257900969>.
- [34] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba. Large language models are human-level prompt engineers. *ArXiv*, abs/2211.01910, 2022. URL <https://api.semanticscholar.org/CorpusID:253265328>.