Towards Optimizing with Large Language Models

Pei-Fu Guo b08303101@ntu.edu.tw
Ying-Hsuan Chen r12922044@ntu.edu.tw
Yun-Da Tsai f08946007@csie.ntu.edu.tw
Shou-De Lin sdlin@csie.ntu.edu.tw

Abstract

In this work, we conduct an assessment of the optimization capabilities of LLMs across various tasks and data sizes. Each of these tasks corresponds to unique optimization domains, and LLMs are required to execute these tasks with interactive prompting. That is, in each optimization step, the LLM generates new solutions from the past generated solutions with their values, and then the new solutions are evaluated and considered in the next optimization step. Additionally, we introduce three distinct metrics for a comprehensive assessment of task performance from various perspectives. These metrics offer the advantage of being applicable for evaluating LLM performance across a broad spectrum of optimization tasks and are less sensitive to variations in test samples. By applying these metrics, we observe that LLMs exhibit strong optimization capabilities when dealing with small-sized samples. However, their performance is significantly influenced by factors like data size and values, underscoring the importance of further research in the domain of optimization tasks for LLMs.

I Introduction

In recent years, Large Language Models have demonstrated exceptional capabilities in reasoning across a variety of natural language-based tasks [5]. However, their potential extends beyond multiple-choice questions or single-question answering. This work explores LLMs' effectiveness in optimization across diverse tasks and data sizes. Optimization, in this context, refers to the process of iteratively generating and evaluating solutions to improve upon a given objective function.

Our research assesses how well LLMs perform in interactive optimization settings, where each optimization step involves the generation of new solutions based on past solutions and their associated values. We conduct our study with four different types of optimization algorithms: Gradient Descent, Hill Climbing, Grid Search, and Black Box Optimization. In order to provide a comprehensive evaluation of LLM performance, we introduce three distinct metrics. These metrics provide a multifaceted view of task performance and are applicable across a broad spectrum of optimization tasks, reducing sensitivity to sample variations.

Our preliminary findings indicate that LLMs exhibit impressive optimization capabilities, particularly when confronted with tasks involving small-sized data samples. However, the performance of these models is notably influenced by factors, including the sample size and the range of values involved in the process. These observations underscore the need for further research and exploration within the domain of optimization tasks tailored for Large Language Models.

To summarize, we aim to make the following contributions in this preliminary study: (1) Exploring LLMs in interactive optimization scenarios with iterative prompting. (2) Introduce three novel evaluation metrics designed to provide a comprehensive assessment of LLM performance in optimization tasks. (3) Delve into factors that influence LLM performance in optimization, with a particular emphasis on the impact of data size and task type.

II Related Work

In various optimization scenarios, the utilization of Large Language Models (LLMs) has become indispensable for the development of optimization algorithms or agent systems capable of handling complex and informative text-based feedback [1, 3, 6]. In this section, we summarize three significant related works that leverage LLMs to tackle optimization and reinforcement learning challenges:

1. Optimization by PROmpting (OPRO): OPRO harnesses LLMs as versatile optimizers by describing optimization tasks in natural language prompts. It iteratively generates and evaluates solutions from these prompts, demonstrating superior performance on tasks like linear regression and traveling salesman problems. OPRO outperforms human-designed prompts by up to 50% on challenging tasks.

2. Reflexion: Reflexion introduces a novel framework for training language agents that rely on linguistic feedback rather than traditional reinforcement learning. It achieves remarkable results, such as a 91% pass@1 accuracy on coding tasks, surpassing previous state-of-the-art models by 11%.

3. EvoPrompt: EvoPrompt automates prompt optimization by connecting LLMs with evolutionary algorithms. This approach outperforms human-designed prompts by up to 25% and existing automatic prompt generation methods by up to 14%. It highlights the synergy between LLMs and conventional algorithms.

These works collectively showcase the adaptability and effectiveness of LLMs in addressing optimization and learning challenges across various domains.

III Methodology

We demonstrate that LLMs with iterative prompting can serve as optimizers and could be formulated similarly to different types of optimization algorithms. In particular, we present a case study on Gradient-Descent, Hill-Climbing (Heuristic), Grid-Search, and Black-Box Optimization (LLMs as a black box optimizer) and conduct an assessment of the optimization capabilities of Large Language Models across a range of distinct parameter search tasks.

3-1 Tasks

We design four optimization tasks that require the model to algorithmically search for the optimal value of parameters. These tasks encompass Gradient-Descent, Hill-Climbing, Grid-Search, and Black-Box Optimization, each representing unique optimization domains: gradient-based, meta-heuristics, decision-theoretic, and Bayesian. In terms of parameter types, Grid-Search and Hill-Climbing involve discrete search spaces, while Gradient-Descent and Black-Box Optimization tackle continuous search spaces.

In the **Gradient-Descent** task, we instruct Large Language Models to undertake a conventional gradient descent optimization process based on the loss function they have defined. Language Models need to compute the gradient and update the parameters using the gradient information and the learning rate given. This task assesses the model's proficiency in advanced calculations and its grasp of the principles of gradient descent.

In the **Hill-Climbing** task, we introduce custom instructions in the search algorithm to evaluate the LLM's capability to adhere to predefined rules they have not seen before. LLMs start with an initial solution and iteratively explore nearby solutions by making small incremental changes. In our task, neighboring solutions are generated by selecting a specific element within the solution and either increasing or decreasing it by one each time. Subsequently, the neighbor solution with the minimum loss is chosen as the new solution and passed to the next iteration.

In the **Grid-Search** task, Large Language Models are tasked with generating all grid points and systematically searching for the point that results in the lowest loss according to the given loss function. This task assesses the LLM's ability to conduct exhaustive searches and locate optimal solutions within a predefined search space.

In the **Black-Box Optimization** task, we treat the LLMs as black boxes that try to fit an unknown loss function. We provide the LLM with a limited set of solutions, each paired with its respective true loss value. The LLM's objective is to discover a new solution that has a lower loss than the existing solutions in each iteration by themselves. This task evaluates the LLM's ability to make informed decisions and optimize in a less transparent or more abstract problem-solving context.

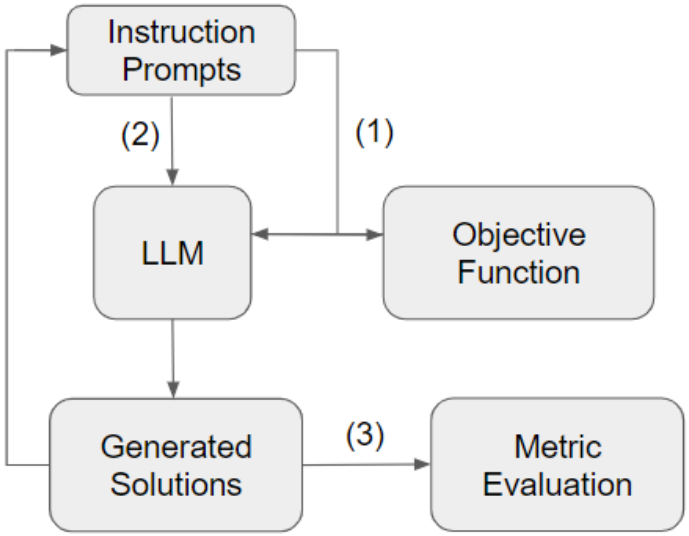

Figure 1: Overview of our prompting strategy. (1) Define the loss function. (2) Given instruction prompt, LLM generates solution and pass result to next iteration. (3) Evaluate LLM final output.

3-2 Prompting Strategy

We applied Chain of Thoughts as our prompting method. The Language model will accomplish each step with reasoning thoughts as intermediate outputs. Each optimization iteration is composed of three steps: (1) Define the loss function. (2) Given an instruction prompt, LLM generates a solution and passes the result to the next iteration. (3) Evaluate LLM final output. In each of these tasks (optimization algorithm), LLMs are initially required to formulate a loss function based on the provided sample. Following the formulation of this loss function, they embark on a process of iterative exploration to determine the optimal values for the variables that will minimize the loss associated with the function. Figure 1 shows an overview of how LLM performs optimization in interactive settings. The following illustrates a sample meta-prompt for the prompting strategy. Detailed meta prompt instructions can be found in Appendix A. Complete prompt examples for each optimization algorithm can be found in Appendix B.

Gradient-Descent

***Define Loss function**

User :

Given the data points (y1, y2, y3) = (2,6,0), what is the MSE loss function with respect to the $\hat{y}$s for a hypothetical set of predicted $\hat{y}$s values?

LLM :

The MSE loss function for the given data points (y1, y2, y3) = (2,6,0) with respect to ŷs is: $\frac{1}{3}[(\hat{y}_1 - 2)^2 + (\hat{y}_2 - 6)^2 + (\hat{y}_3 - 0)^2]$

***First Iteration**
User :
Please minimize the loss function using gradient descent with learning rate of 0.1 at point (ŷ1, ŷ2, ŷ3, ...) = (2,3,4). What is the point we eventually end up after one update? Your answer includes two parts an explanation with calculation and a short answer of result.
LLM :
Explanation : To minimize the Mean Squared Error (MSE) loss function using gradient descent ...
Short Answer: After calculation, the next update point is ($\hat{y}1_{new}$, $\hat{y}2_{new}$, $\hat{y}3_{new}$) = (1.92, 3.05, 3.94)

***Second Iteration**
User :
Please minimize the loss function using gradient descent with learning rate of 0.1 at point (ŷ1, ŷ2, ŷ3, ...) = (1.92, 3.05, 3.94). What is the point we eventually end up after one update? Your answer includes two parts an explanation with calculation and a short answer of result.
LLM :
Explanation : To minimize the Mean Squared Error (MSE) loss function using gradient descent ...
Short Answer: After calculation, the next update point is ($\hat{y}1_{new}$, $\hat{y}2_{new}$, $\hat{y}3_{new}$) = (1.81, 3.12, 3.41)
⋮

IV Experimental

4-1 Dataset

In the experiment, we generate synthetic datasets where each instance belongs to $[0, 10]^d$. In the experiments, we create five datasets with d values chosen from the set $\{3, 6, 12, 24, 48\}$ in order to investigate sensitivity to the size of input data points.
We then apply these data to a loss function (Mean Squared Error) and find the true solution for each parameter search task. These true solutions and their losses serve as the ground truth for the tasks and are used as a benchmark to compare with the solutions found by Large Language Models afterward.

4-2 Detailed Settings

In our experiment, we performed five repetitions of the test for each sample in the dataset, with the LLM conducting ten iterations of parameter search in each repetition. We excluded excessively biased results to prevent our metrics from being skewed by a minority of poorly performing test outcomes. All the samples have been confirmed to exhibit a significant reduction in their respective loss within ten iterations when optimized by the ground truth algorithm.

We assess the performance of Large Language Models: GPT-turbo-3.5, GPT-4. To create an interactive environment, we utilize the chat-mode of GPTs, where the entire conversation history serves as the prompt. This approach enables the LLMs to retain the memory of prior search results and reasoning paths from past interactions. With each iteration, new instructions are added to the ongoing conversation records and the entire dialogue is presented as the prompt. If the dialogue exceeds the token limit, we will remove the earliest portions of the conversation to ensure compliance with the limit. See Appendix B for detailed examples.

We set the model temperature to 0.8, which adds randomness to the generated responses. This intentional randomness helps us evaluate the stability of LLM solutions when provided with identical initial

settings in the search. Due to resource and token length limitations, we only test the Gradient Descent task and BlackBox Optimization task on all datasets and LLMs. For Grid Search and Hill Climbing tasks, experiments are conducted only on datasets with sample sizes 3, 6, and 12 and exclusively run on GPT-turbo-3.5.

V Metric Design

To comprehensively evaluate task performance from different aspects, we designed three distinct metrics. The first is the *goal metric*, gauging the LLM's advancement in accomplishing the optimization task. The second is the *policy metric*, assessing the proximity of the final model output to the ground truth. Lastly, the *uncertainty metric*, which quantifies the level of variability in the LLM's solutions when exposed to identical conditions.

These metrics have the advantage of being versatile in evaluating LLM performance across various optimization tasks, streamlining the process for practitioners to assess multiple task performances concurrently. Additionally, these metrics are less sensitive to variations in test samples because of the use of ratio measures rather than difference measures.

5-1 Goal Metric

One of our objectives in this study is to evaluate the optimization capabilities of Large Language Models. To achieve this, we employ the goal metric, which serves as a quantitative measure of the extent to which the optimization process leads to improvements in objective function values. We define the **goal metric** of a test sample j as :

$$G_j = \frac{1}{N}\sum_{i=1}^{N}\frac{loss_{LLM,init} - loss_{LLM,i}}{loss_{LLM,init}}$$

where $loss_{LLM,init}$ is the initial solution loss of sample j, $loss_{LLM,i}$ is the LLM output loss of trial i, and N is the number of trials per sample. The higher the metric value, the greater the progress in optimization. The goal metric plays a crucial role in our evaluation framework, particularly in scenarios where ground truth is absent, such as the Black-Box optimization scenarios.

5-2 Policy Metric

Beyond self-improvement, it's essential to evaluate the Large Language Models' ability to function akin to a truth model algorithm. The metric designed for this evaluation is the policy metric, which serves as an indicator of the LLM's proficiency in adhering to task-specific instructions. We define the **policy metric** of a test sample j as :

$$P_j = \frac{1}{N}\sum_{i=1}^{N}\frac{loss_{LLM,i} - loss_{truth}}{loss_{truth}}$$

where $loss_{LLM,i}$ is the LLM output loss of trial i, $loss_{truth}$ is the ground truth of sample j and N is the number of trials. The policy metric measures the disparity between the ground truth and the LLM's output. A lower policy metric value indicates a more effective alignment of the LLM's actions with the prescribed guidelines. When the value is negative, it means that LLM's performance surpasses the ground truth.

5-3 Uncertainty Metric

Stability is a crucial characteristic in optimization tasks. We hope that the LLMs produce identical results in every trial involving the same sample, even under conditions with temperatures greater than

zero. We define the **uncertainty metric** of a test sample j as :

$$U_j = \frac{1}{N}\sum_{i=1}^{N}(loss_{LLM,i} - \overline{loss_{LLM}})^2$$

where $loss_{LLM,i}$ is the LLM output of the i-th trial, $\overline{loss_{LLM}}$ is the mean of the trial outputs and N is the number of trials. A stable LLM can be more trusted for tasks that demand consistent and reproducible results. In our case, if the language model truly understands the context of problems, the final optimal output should be identical in every trial of the same sample.

VI Results and Analysis

We summarize the outcomes of our experiment and subsequently examine the common trends observed across all experiments. In every plot, the x-axis displays the number of data points within each sample. In the case of the goal metric and policy metric plots, the y-axis illustrates the average metric value for the respective tasks, while the shaded area in a lighter color delineates the confidence interval of the metric, denoted as $[value - std, value + std]$. As for the uncertainty metric plot, the y-axis showcases the uncertainty metric value, which corresponds to the standard deviation of the LLM final solution loss.

6-1 Results

Figure 2 presents the performance of GPT-turbo-3.5 across all tasks. The Grid-Search task doesn't appear in the goal metric graph due to its non-iterative nature. Similarly, Black-Box Optimization isn't represented in the policy metric graph because ground truth is unattainable.

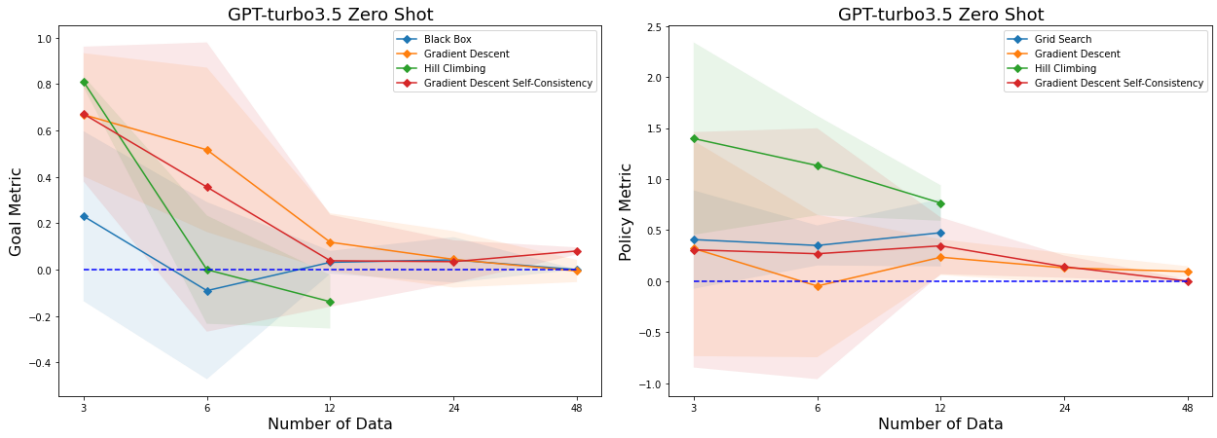

Figure 2: GPT-turbo-3.5 average zero-shot performance across tasks

We can see that GPT-turbo-3.5 exhibits optimization capabilities in most cases. Impressively, in the Gradient-Descent task, GPT-turbo-3.5 even surpasses the ground truth, particularly in the case of the 6 data points sample. It's also surprising that the model achieves respectable results in the Grid-Search task, considering it must compute a vast number of grid points, which increase exponentially as the data per sample expands. The model faces challenges in the Hill-Climbing task, evident from a policy metric significantly exceeding zero. This suggests that meta-heuristics may pose greater difficulty for LLMs compared to other tasks.

Remarkably, GPT-turbo-3.5 and GPT-4 excel at performing Black-Box tasks in small-size samples. From Figure 3, we can see that GPT-turbo-3.5 performs notably with three data points, whereas GPT-4 excels

at three and six data points. Interestingly, as the number of data points increases, the performance of both models gradually diminishes. Eventually, GPT-4 establishes itself as the superior performer.

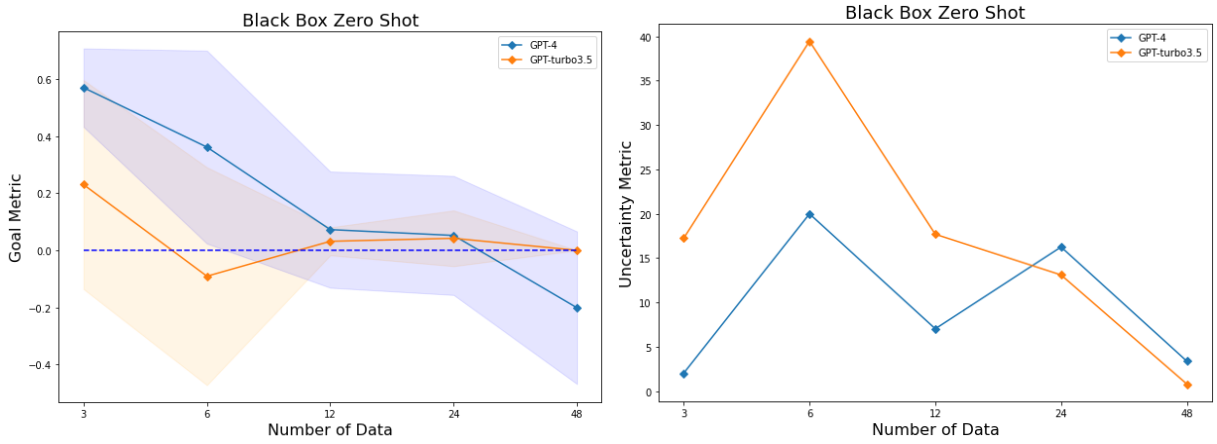

Figure 3: GPTs average performance of Black-Box Task

In the context of the Gradient-Descent Task, both models exhibit strong performance, with GPT-4 edging out GPT-turbo-3.5 by a slight margin. Figure 4 underscores this by revealing a policy metric that consistently hovers near zero, signifying a remarkable alignment between the LLM's output and the ground truth. Despite a decline in the goal metric as the sample size increases, the consistently low and stable value of the policy metric underscores the fact that GPT's performance in the gradient-descent task is nearly on par with the truth model or algorithm.

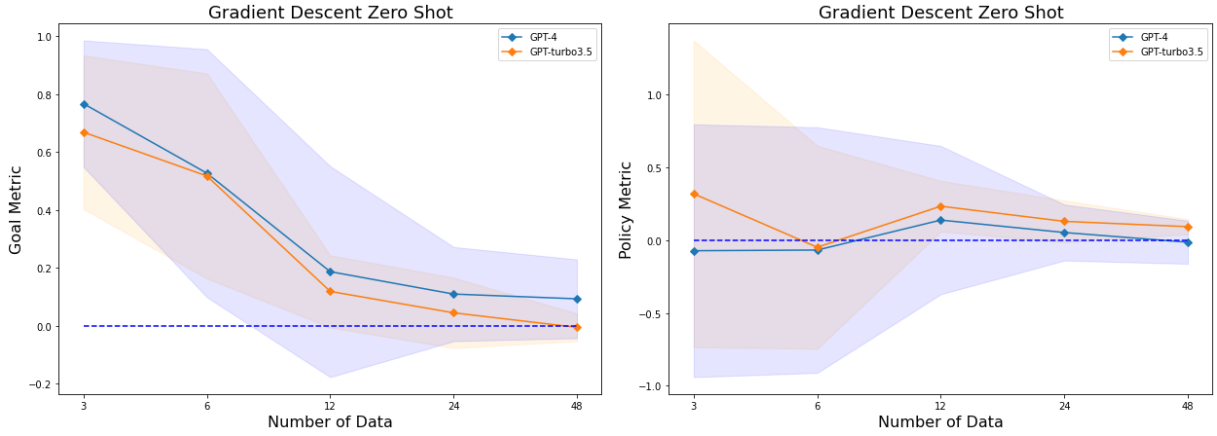

Figure 4: GPTs average performance of Gradient-Descent Task

6-2 Analysis

LLMs showcase optimization prowess across diverse domains. While not all solutions consistently achieve substantial improvements or align perfectly with ground truth, the predominantly positive goal metric values across most tasks and datasets indicate their capability for optimization. Among these tasks, Gradient Descent emerges as the leading performer, while the Hill-Climbing task poses greater challenges. Remarkably, GPT-4 even surpasses ground truth in a dataset comprising six data points, underscoring its exceptional aptitude for calculus. This underscores the notion that LLMs possess a versatile capacity to optimize within various problem spaces, which allows for the possibility of switching

between different optimization methods within a single task.

LLM can be real optimizers. In the Black-Box Optimization task, LLMs are treated as black-box optimizers, which try to fit an unknown loss function. In each iteration, they are asked to offer their preferred solutions based on the discovered solutions and their true scores. Notably, when there are three data points, a more pronounced optimization capability is observed. Despite the performance decline with an increase in data points (variables), this observation intriguingly suggests that LLMs may possess inherent optimization capabilities of their own.

Data quantity influence stability. In cases where samples have fewer data points, it's common to observe high uncertainty metric values accompanied by significant variations in policy and goal metrics. This suggests that while Large Language Models generally perform well in most trials, we may still encounter instances of failure, such as getting stuck in neither global nor local optima, optimizing in the wrong direction, and regenerating solutions that already appeared when attempting the same task again with the same sample. Figure 2 and 5 both highlight the conspicuous pattern of uncertainty, where the uncertainty initially rises and then gradually decreases. This behavior is consistent across different tasks and models, warranting further investigation to fully understand its underlying causes.

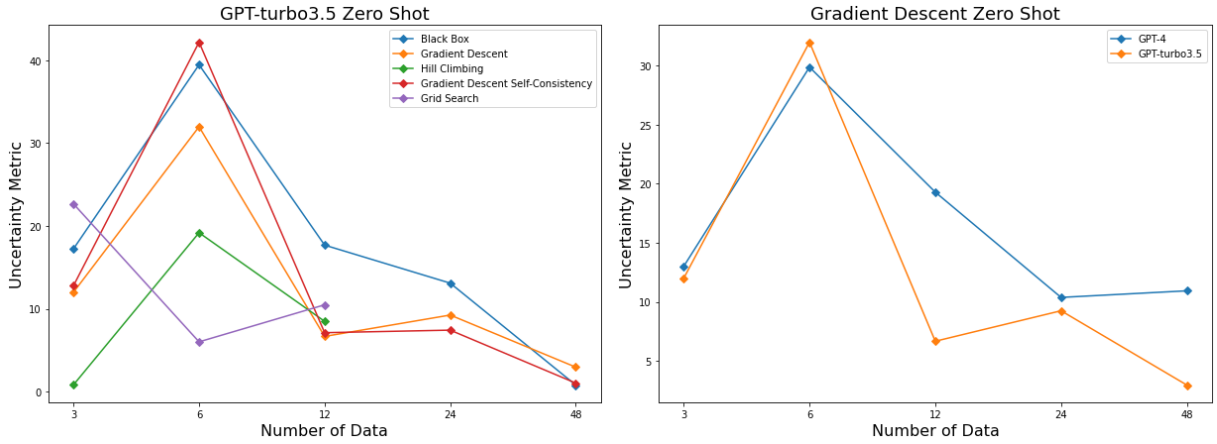

Figure 5: Uncertainty Metric of different tasks and models

Prompting methods may contribute to stability. In the Gradient-Descent task, we employ the self-consistency technique [4], where we conduct five repetitions for each iteration and select the solution that emerges most frequently. From Figure 6, we can see that GPT-4 performances increase largely, and the confidence interval for both the policy-metric and goal-metric narrows, indicating improved stability and reliability. Nonetheless, this approach does not yield favorable outcomes when applied to GPT-turbo-3.5. This suggests the need for further investigation within the realm of variance reduction to better understand the contributing factors to these results.

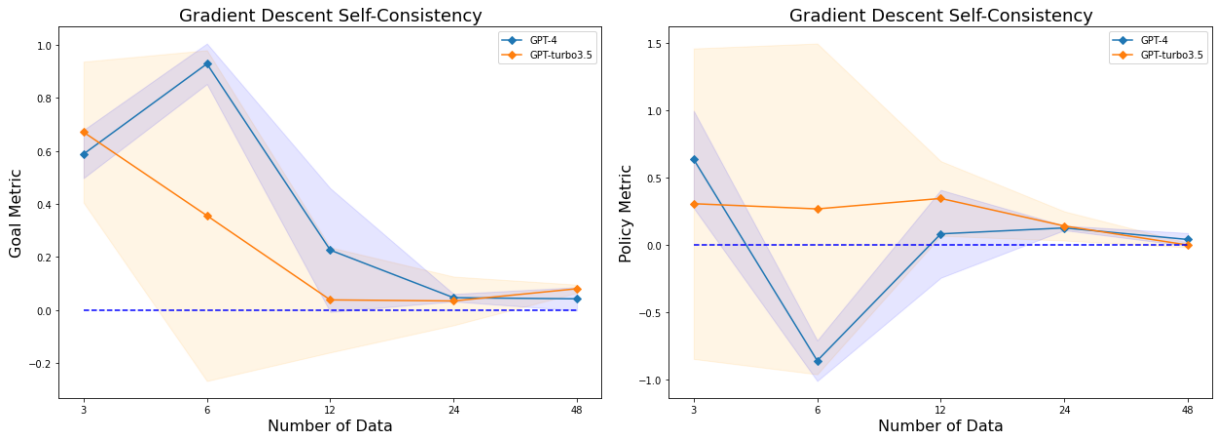

Figure 6: Average Performance of Gradient-Descent Task with Self-Consistency

Sensitivity to data samples. It's worth considering that the aforementioned results may be influenced by the inherent randomness in the generation of test samples. Previous research has indicated that Large Language Models (LLMs) may demonstrate preferences for particular numbers, words, and symbols [2], which can introduce a level of bias in their responses. Given the high sensitivity of LLMs to the input prompt, the initial starting points and data provided can exert a significant influence on their outputs. In essence, the impact of instruction description and data initialization should be carefully considered when interpreting the results of LLM-based experiments to ensure a more accurate assessment of their performance.

VII Conclusion

In this paper, we present our in-depth examination of assessing Large Language Models (LLMs) within the realm of optimization. We investigate LLMs' performance across four optimization tasks that necessitate their comprehension of algorithmic instructions and their ability to generate new solutions based on previous solutions and their corresponding values.

Our evaluation shows that LLMs showcase optimization prowess across diverse domains. Among the four tasks we examined, LLMs exhibit their greatest strengths in the Gradient-Descent task, displaying remarkable proficiency in this area. However, they encounter more pronounced difficulties in the meta-heuristics task, where they must adhere to predefined rules that they have not encountered previously. Furthermore, LLMs demonstrate impressive skills in the grid search task, showcasing their ability to conduct exhaustive searches effectively. In the Black-Box task, LLMs excel, particularly when dealing with limited sample sizes, suggesting inherent optimization abilities within them.

Interestingly, in cases where samples have fewer data points(variables), it's common to observe a higher variance in solution quality. The notable pattern of uncertainty is characterized by an initial surge followed by a gradual decrease as data size increases and persistence across tasks and models.

In conclusion, our exploration of Large Language Models (LLMs) for optimization has unveiled a host of unresolved questions that beckon future research. Chief among these challenges is the quest to mitigate the sensitivity of LLMs to both data value and size, a paramount concern for enhancing their practical utility.

References

[1] Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *arXiv:2309.08532*, 2023.

[2] Alex Renda, Aspen Hopkins, and Michael Carbin. Can llms generate random numbers? evaluatingllm sampling in controlled domains. 2023.

[3] Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *arXiv:2303.11366*, 2023.

[4] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, and D Zhou. Self-consistency improves chain of thought reasoning in language models. In *arXiv preprint arXiv:2203.11171*, 2022.

[5] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D Zhou. Chain of thought prompting elicits reasoning in large language models. In *arXiv preprint arXiv:2201.11903*, 2022.

[6] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *arXiv preprint arXiv:2309.03409*, 2023.

APPENDIX

A Instruction Prompts

Objective Function Prompt :

Q :
Given the data points (y1, y2, ...) = {data}, what is the MSE loss function with respect to the ŷs for a hypothetical set of predicted ŷs values?
A :
The MSE loss function for the given data points (y1, y2, ...) = {data} with respect to ŷs is:

Gradient-Descent Prompt :

Q :
Please minimize the loss function using gradient descent with learning rate of 0.1 at point (ŷ1, ŷ2, ŷ3, ...) = {point}. What is the point we eventually end up after one update? Your answer includes two parts an explanation with calculation and a short answer of result.
A :
Explanation : Lets think step by step ...
Short Answer: After calculation, the next update point is ($\hat{y}1_{new}$, $\hat{y}2_{new}$, $\hat{y}3_{new}$, ...) =

Grid-Search Prompt (Create Grid Points) :

Q :
I want to do grid search on the ŷs and the range of them are the integers of {low_bound} to {high_bound}. Generate all possible combinations of ŷs values from the specify range. What are the combinations? Your answer includes two parts an explanation with calculation and a list containing all the combinations.
A :
Explanation : Lets think step by step ...
List : [write all the combinations here]

Grid-Search Prompt (Select) :

Q :
For every combinations of ŷs, calculate its MSE loss. Which combination has the smallest MSE loss? Your answer includes two parts an explanation with calculation and a list containing the combination with the smallest MSE loss.
A :
Explanation : Lets think step by step
List : [write the combination with smallest MSE loss]

Hill-Climbing Prompt (Generate Neighbors) :

Q :
I want to minimize the loss function using hill climbing. Generate neighboring solutions by either adding 1 or minus 1 to a specific element in the current solution. The current solution is solution. Your answer includes two parts an explanation with calculation and a list containing all neighbor solutions(eg. [(ŷ1, ŷ2,....), (ŷ1, ŷ2,....), ...]).
A :
Explanation : Let's think step by step ...
List : [write neighbor solutions here]

Hill-Climbing Prompt (Select) :

Q :
For every neighbor solution, calculate its MSE loss. Which neighbor has the smallest MSE loss? Your answer includes two parts an explanation with calculation and a list containing the smallest MSE loss neighbor solution(eg. [(ŷ1, ŷ2,....)]).
A :
Explanation : Lets think step by step ...
List : [write best neighbor solution here]

Black-Box Prompt :

Q :
You want to minimize an unknown MSE loss function by guessing the values of the ŷs. When you guess, you should take consider of the past guessing result so that your new guess will have smaller loss than the past results. Pass guessing result are {pass_result}. Base on the previous guesses, what is your next guess?
A :
(ŷ1, ŷ2,....) = [your answer]

B Examples

For tasks except Black Box Optimization, we use the whole dialogue as a prompt. Take the gradient-descent task as an example, the first prompt will be the Define Loss Function section, the second prompt will be the Define Loss Function section plus the First Iteration section and so on.

Gradient-Descent :

***Define Loss function**
User :
Given the data points (y1, y2, y3) = (2,6,0), what is the MSE loss function with respect to the ŷs for a hypothetical set of predicted ŷs values?
LLM :
The MSE loss function for the given data points (y1, y2, y3) = (2,6,0) with respect to ŷs is: $\frac{1}{3}[(\hat{y}_1 - 2)^2 + (\hat{y}_2 - 6)^2 + (\hat{y}_3 - 0)^2]$

***First Iteration**
User :
Please minimize the loss function using gradient descent with learning rate of 0.1 at point (ŷ1, ŷ2, ŷ3) = (2,3,4). What is the point we eventually end up after one update? Your answer includes two parts an explanation with calculation and a short answer of result.
LLM :
Explanation : To minimize the Mean Squared Error (MSE) loss function using gradient descent ...
Short Answer: After calculation, the next update point is ($\hat{y}1_{new}$, $\hat{y}2_{new}$, $\hat{y}3_{new}$) = (1.92, 3.05, 3.94)

***Second Iteration**
User :
Please minimize the loss function using gradient descent with learning rate of 0.1 at point (ŷ1, ŷ2, ŷ3) = ((1.92, 3.05, 3.94). What is the point we eventually end up after one update? Your answer includes two parts an explanation with calculation and a short answer of result.
LLM :
Explanation : To minimize the Mean Squared Error (MSE) loss function using gradient descent ...
Short Answer: After calculation, the next update point is ($\hat{y}1_{new}$, $\hat{y}2_{new}$, $\hat{y}3_{new}$) = (1.81, 3.12, 3.41)
⋮

Grid-Search :

***Define Loss function**
User :
Given the data points (y1, y2, y3) = (2,6,0), what is the MSE loss function with respect to the ŷs for a hypothetical set of predicted ŷs values?
LLM :
The MSE loss function for the given data points (y1, y2, y3) = (2,6,0) with respect to ŷs is: $\frac{1}{3}[(\hat{y}_1 - 2)^2 + (\hat{y}_2 - 6)^2 + (\hat{y}_3 - 0)^2]$

***Create Grid Points**
User :
I want to do grid search on the ŷs and the range of them are the integers of 2 to 3. Generate all possible combinations of ŷs values from the specify range. What are the combinations? Your answer includes two parts an explanation with calculation and a list containing all the combinations.
LLM :
Explanation : To generate all the grid points, you should ...
List : [(2, 2, 2), (2, 2, 3), (2, 3, 2), (2, 3, 3), (3, 2, 2), (3, 2, 3), (3, 3, 2), (3, 3, 3)]

***Select Best**
User :
For every combinations of ŷs, calculate its MSE loss. Which combination has the smallest MSE loss? Your answer includes two parts an explanation with calculation and a list containing the combination with the smallest MSE loss.

LLM :
Explanation : To select the grid points with smallest MSE value ...
List : [(2, 3, 3)]
⋮

Hill-Climbing :

***Define Loss function**
User :
Given the data points (y1, y2, y3) = (2,6,0), what is the MSE loss function with respect to the ŷs for a hypothetical set of predicted ŷs values?
LLM :
The MSE loss function for the given data points (y1, y2, y3) = (2,6,0) with respect to ŷs is: $\frac{1}{3}[(\hat{y}_1 - 2)^2 + (\hat{y}_2 - 6)^2 + (\hat{y}_3 - 0)^2]$

***Generate Neighbors**
User :
I want to minimize the loss function using hill climbing. Generate neighboring solutions by either adding 1 or minus 1 to a specific element in the current solution. The current solution is (10,10,10). Your answer includes two parts an explanation with calculation and a list containing all neighbor solutions(eg. [(ŷ1, ŷ2,....), (ŷ1, ŷ2,....), ...]).
LLM :
Explanation : To generate all the neighbors ...
List : [(11, 10, 10), (10, 11, 10), (10, 10, 11), (9, 10, 10), (10, 9, 10), (10, 10, 9)].

***Select Best**
User :
For every neighbor solution, calculate its MSE loss. Which neighbor has the smallest MSE loss? Your answer includes two parts an explanation with calculation and a list containing the smallest MSE loss neighbor solution(eg. [(ŷ1, ŷ2,....)]).
LLM :
Explanation : To select the neighbor solution with smallest MSE value ...
List : [(9, 10, 10)]
⋮

As for the Black-Box task, the first prompt will be the Loss Function section and for the subsequent iterations, the prompt will be the Loss Function section and the prompt with new past results.

Black-Box Optimization :

***Define Loss function**
User :
Given the data points (y1, y2, y3) = (2,6,0), what is the MSE loss function with respect to the ŷs for a hypothetical set of predicted ŷs values?
LLM :
The MSE loss function for the given data points (y1, y2, y3) = (2,6,0) with respect to ŷs is: $\frac{1}{3}[(\hat{y}_1 - 2)^2 + (\hat{y}_2 - 6)^2 + (\hat{y}_3 - 0)^2]$

First Iteration
User :

You want to minimize an unknown MSE loss function by guessing the values of the ŷs. When you guess, you should take consider of the past guessing result so that your new guess will have smaller loss than the past results. Pass guessing result are $f(2,4,6) = 10$, $f(1,5,0) = 2$. Base on the previous guesses, what is your next guess?
LLM :
$(\hat{y}1, \hat{y}2, \hat{y}3) = [1,3,5]$

Second Iteration
User :
You want to minimize an unknown MSE loss function by guessing the values of the ŷs. When you guess, you should take consider of the past guessing result so that your new guess will have smaller loss than the past results. Pass guessing result are $f(2,4,6) = 10$, $f(1,5,0) = 2$, $f(1,3,5) = 5$. Base on the previous guesses, what is your next guess?
LLM :
$(\hat{y}1, \hat{y}2, \hat{y}3) = [1,1,1]$
$\vdots$