Exploiting Edited Large Language Models as General Scientific Optimizers

Qitan Lv[1], **Tianyu Liu**[1*], **Hong Wang**[1†],
[1] University of Science and Technology of China
{qitanlv, tianyu_liu, wanghong1700}@mail.ustc.edu.cn

Abstract

Large language models (LLMs) have been widely adopted in mathematical optimization in scientific scenarios for their extensive knowledge and advanced reasoning capabilities. Existing methods mainly focus on utilizing LLMs to solve optimization problems in a prompt-based manner, which takes observational feedback as additional textual descriptions. However, due to LLM's **high sensitivity to the prompts** and **tendency to get lost in lengthy prompts**, these methods struggle to effectively utilize the observational feedback from each optimization step, which severely hinders the applications for real-world scenarios. To address these challenges, we propose a conceptually simple and general bi-level optimization method, namely **G**eneral **S**cientific **O**ptimizers (GSO). Specifically, GSO first utilizes inner-level simulators as experimental platforms to evaluate the current solution and provide observational feedback. Then, LLMs serve as knowledgeable and versatile scientists, generating new solutions by refining potential errors from the feedback as the outer-level optimization. Finally, simulations together with the expert knowledge in LLMs are jointly updated with bi-level interactions via model editing. Extensive experiments show that GSO consistently outperforms existing state-of-the-art methods using *six* different LLM backbones on *seven* different tasks, demonstrating the effectiveness and a wide range of applications.

1 Introduction

Optimization is ubiquitous across a wide range of scientific domains, spanning mathematics, physics, chemistry, pharmacology, etc (Amari, 1993; Intriligator, 2002; Cova and Pais, 2019). Various research endeavors innovate within its field, creating methods tailored to its specific challenges and nuances to automate and accelerate the process of mathematical optimization in scientific scenarios (Wang et al., 2023b). However, the need for customization of optimization algorithms to address specific challenges highlights the absence of a universally applicable philosophy (Popper, 2005; Fortunato et al., 2018), which is crucial for establishing a standardized optimization framework and enhancing the efficiency of scientific research. We aim to transcend specific domains and provide a generalized approach to boost mathematical optimization in scientific scenarios.

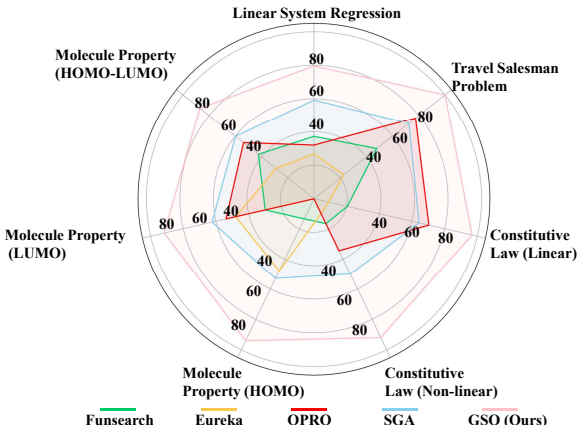

Figure 1: GSO achieves state-of-the-art performance on a broad range of scientific optimization tasks compared with existing methods, using LLama 3 8B (Team, 2024b) as the backbone. Results of other five LLMs are in Figures 10 and 11.

Standing out as versatile tools with vast knowledge repositories, large language models (LLMs) have recently risen to prominence in optimization across scientific domains for their expansive knowledge bases, advanced reasoning capabilities, and human-friendly natural language interface (AI4Science and Quantum, 2023). Extensive research efforts have been devoted to boosting general mathematical optimization in scientific scenarios. Canonical methods mainly focus on fine-tuning LLMs using domain-specific data

[*]Equal Contributions.
[†]The Corresponding author.

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 5212–5237

to align natural language with scientific information, such as chemical structures (Li et al., 2024b; Chithrananda et al., 2020) or drug structures (Liu et al., 2021). However, these approaches are constrained to specific domains and require substantial amounts of data and extensive computation resources for broader applicability. Recently, prompt-based iterative optimization methods—which enhance the inherent capabilities of pre-trained LLMs by incorporating the optimization feedback to LLMs—have emerged as a promising approach for advancing scientific optimization (Yang et al., 2023). Extensive researches have explored leveraging LLMs as optimizers or agents (Zhang et al.) for tasks such as mathematical problem-solving (Romera-Paredes et al., 2024a; Yang et al., 2023), conducting chemical experiments (Boiko et al., 2023), advancing physical scientific discovery (Ma et al.), molecular discovery (Li et al., 2024a), and drug discovery (Sharma and Thakur, 2023).

Albeit with multiple benefits of the prompt-based methods, they confront one significant challenge that severely hinders their general applications—struggling to effectively utilize observational feedback. This challenge primarily stems from two limitations inherent in existing prompt-based methods: **(i) LLMs are shown to be sensitive to the prompt format** (Lu et al., 2022; Wei et al., 2023; Madaan and Yazdanbakhsh, 2022). In particular, semantically similar prompts can yield drastically different performance (Kojima et al., 2022; Zhou et al.; Yang et al., 2023), and the optimal prompt formats may be model-specific and task-specific, which severely limits the generalizability across different scientific tasks. **(ii) LLMs may get lost in lengthy prompt** (Lv et al., 2024). In multi-round iterative optimization, the input prompt can become increasingly lengthy to trace the optimization trajectory, which may distract the LLMs' reasoning and result in the *lost in the middle issue* (Shi et al., 2023). Despite LLMs' ability to process long contexts, performances significantly decrease as the input grows longer, even for models explicitly designed for long contexts (Lv et al., 2024).

To address this challenge, we propose a conceptually simple, flexible, and general method, namely **G**eneral **S**cientific **O**ptimizers (GSO). Specifically, GSO is a bi-level optimization method involving **inner-level** optimization and **outer-level** optimization, together with **bi-level** interaction between them. For a given optimization task, GSO will iteratively conduct the following process: (i) the inner-level optimization first employs simulators serving as an experimental platform to provide observational feedback for reasoning and refining; (ii) the **outer-level** optimization then utilizes knowledgeable LLMs as reasoning agents to generate hypotheses, reason with observational feedback, and refine the previous hypothetical solutions. It also devises a dynamic exploit-and-explore strategy to adaptively adjust the LLM reasoning trajectory based on observational feedback; (iii) finally, the **bi-level** optimization jointly updates the simulations together with the knowledge embedded within LLMs via model editing.

GSO is a novel framework to boost general mathematical optimization in scientific scenarios. As shown in Figure 1, extensive experiments demonstrate the effectiveness and generalization of our GSO, leading significant and consistent superiority using *six* different LLM backbone on *seven* different tasks than existing state-of-the-art methods.

2 Preliminaries

Model Editing Model editing mainly focuses on altering the internal knowledge within LLMs. It aims to update a range of intricate learned concepts, such as logical reasoning, spatial awareness, and numerical understanding, to make tailored adjustments to the model's behavior. In this paper, we follow (Zhong et al., 2023; Zhang et al., 2024b) to update factual knowledge represented as (subject s, relation r, object o). An LLM is expected to retrieve a memory corresponding to o when given a natural language prompt. Editing a fact involves replacing the existing knowledge triple (s, r, o) with a new one (s, r, o^*). For simplicity, an edit is denoted as $e = (s, r, o, o^*)$. Given a set of edits $\mathcal{E} = \{e_1, e_2, \ldots\}$ and an original model f_{θ_0}, sequential model editing applies each edit consecutively, i.e., $\mathcal{F}(f_{\theta_{n-1}}, e_n) = f_{\theta_n}$, where f_{θ_n} refers to the model after n edits.

3 Related Work

Model Editing Model editing involves modifying the memorized knowledge contained in LLMs. Various kinds of complex learned beliefs such as logical, spatial, or numerical knowledge are expected to be edited. Many studies explore the role of MLP layers in Transformers, revealing that these layers store knowledge, which can be localized to specific neurons and edited (Da et al.,

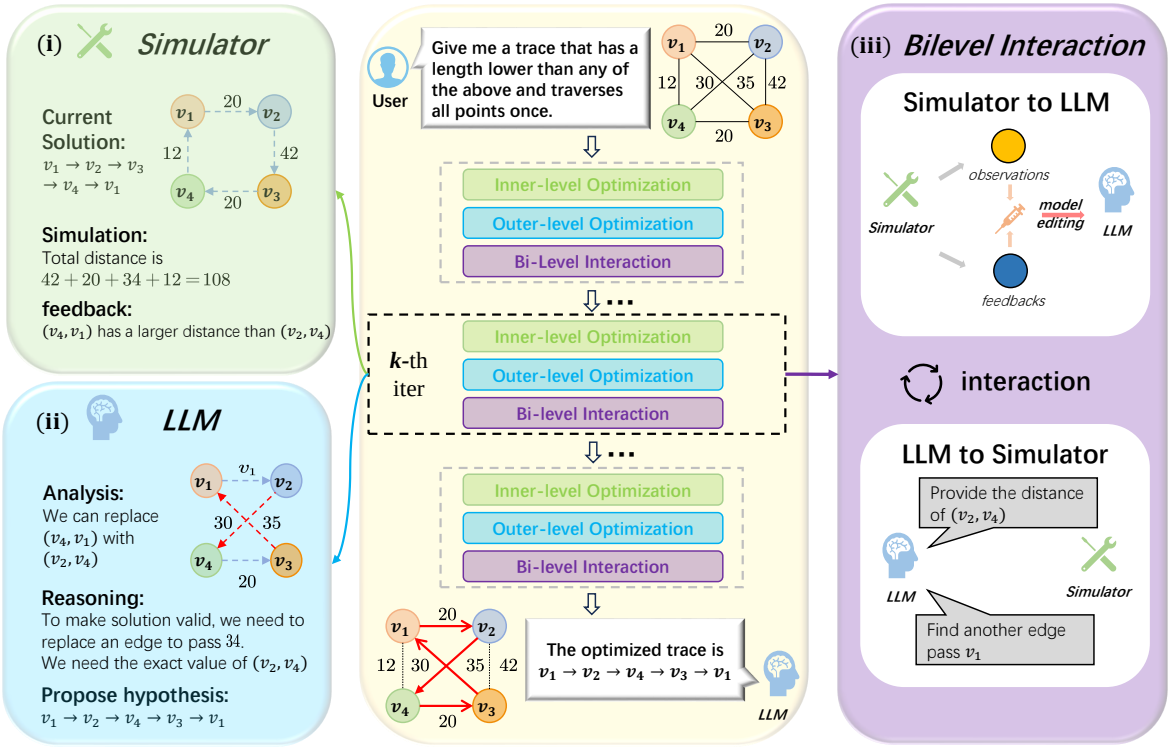

Figure 2: The overview of GSO. For a given optimization task, GSO iteratively conducts the inner-level optimization, outer-level optimization, and bi-level interaction sequentially. The workflow is as follows: (i) the inner-level simulator Φ conducts numerical simulations based on the current step's hypothetical solution s_k ($v_1 \to v_2 \to v_3 \to v_4 \to v_1$) and returns observational feedback $f_k, \mathcal{L}_k$ (the edge (v_4, v_1) has a larger distance than the edge (v_2, v_4), current total distance: 108); (ii) the outer-level LLM $\mathcal{M}_{\theta_k}$ generates new hypothetical solutions s_{k+1} ($v_1 \to v_2 \to v_4 \to v_3 \to v_1$) based on the observational feedback $f_k, \mathcal{L}_k$; (iii) the bi-level interaction jointly updates simulations in conjunction with the expert knowledge within the LLMs through model editing.

2021; Geva et al., 2020, 2022). KE (De Cao et al., 2021) and MEND (Mitchell et al.) train a hypernetwork to compute gradient adjustments for updating model parameters. ROME (Meng et al., 2022) and MEMIT (Meng et al.) apply the Locate-Then-Edit strategy, which first locates MLP storing factual knowledge, and then edits such knowledge by injecting a new key-value pair in the MLP module.

LLMs for Scientific Optimization A recent line of research explores methods to enhance LLM performance by incorporating natural language feedback as prompts to revise model outputs in improving reasoning (Shinn et al., 2023; Madaan et al., 2024), code generation (Chen et al., 2023; Olausson et al., 2023), dialogue applications (Nair et al., 2023; Yuan et al., 2024; Liu et al., 2024b), and so on (Wang et al., 2023a; Kim et al., 2024). LLMs have also been demonstrated to optimize complex problems by utilizing tools (Sumers et al.). AlphaGeometry's (Trinh et al., 2024) success in solving complex geometry problems without human demonstrations underscores the potential of LLMs in automating complex tasks. OPRO (Yang et al., 2023) employs LLMs as black-box optimizers for complex reasoning tasks. Eureka (Ma et al., 2023b) generates multiple solutions in each step to increase the success rate of produced codes. Funsearch (Romera-Paredes et al., 2024b) utilizes an evolutionary strategy to avoid local optima by using LLMs along with a systematic evaluator. SGA (Ma et al.) introduces a bilevel optimization framework to enhance the knowledge-driven capabilities of LLMs by integrating simulations.

4 Method

4.1 Bi-level Optimization Pipeline

We first briefly outline the bi-level pipeline of our GSO, describing how the inner-level and the outer-level optimization are jointly conducted. Specifically, for a scientific optimization task y (e.g., the traveling salesman problem), GSO aims to predict the optimal solution $\hat{s}$, by which GSO iteratively optimizes the solution s_k from an initial solution s_0, where k denotes the iteration step. Each itera-

tion consists of three parts: the inner-level simulation platform, outer-level LLM optimization, and bi-level interactions. An overview of GSO is in Figure 2.

The inner-level simulator **first** serves as an experimental platform, where a simulator Φ is employed to take a intermediate solution s_k and its scientific expression $\mathcal{E}_k$ (e.g., coordinates of the nodes in the TSP task) as inputs, and outputs the corresponding observational feedback f_k and the optimization objective $\mathcal{L}_k$ (e.g., the total distance of the current route in the TSP task):

$$f_k, \mathcal{L}_k = \Phi\left(s_k; \mathcal{E}_k\right).$$

The outer-level optimization **then** utilizes an LLM with its parameter θ_k (denote by $\mathcal{M}_{\theta_k}$) as a reasoning agent, propose a new hypothesis to optimize the current solution s_k:

$$\mathcal{E}_{k+1}, s_{k+1} = \mathcal{M}_{\theta_k}\left(\{\mathcal{L}_i, f_i, \mathcal{E}_i, s_i\}_{i=0}^{k}; \mathcal{P}\right).$$

Here $\mathcal{P}$ denotes the input prompt to LLM, and $\{\mathcal{L}_i, f_i, \mathcal{E}_i, s_i\}_{i=0}^{k}$ represents the historical optimization trajectory.

The bi-level interaction finally utilizes the feedback f_k from inner-level simulation and the solutions s_k provided by the outer-level LLM as a new key-value pair to update the LLM parameters θ through model editing:

$$\mathcal{M}_{\theta_{k+1}} = \mathcal{F}\left(\mathcal{M}_{\theta_k}, s_k, f_k\right),$$

where θ_{k+1} represents the new parameter matrix obtained by applying a model editing step to the previous parameters θ_k, $\mathcal{M}_{\theta_{k+1}}$ represents the according updated LLM, and $\mathcal{F}$ denotes the model editing process. We can then define the overall optimization task:

$$\begin{aligned} &\min_{s} \mathcal{L}\left(y\left(\mathcal{E}, s; \Phi\right)\right) \\ &\text{s.t. } G\left(\mathcal{E}, s; \Phi\right) \leq 0, \end{aligned}$$

where $G\left(\cdot\right) \leq 0$ indicates that the current solution satisfies the constraints of the given optimization problem (for example, in the TSP task, the solution must visit all points and return to the starting point).

4.2 Inner-level Simulation Platforms

The inner-level optimization conducts numerical simulations for a hypothetical solution. Simulations can provide domain-specific knowledge, transferring information from the simulation to optimization outputs (e.g., feedback in the form of total distance for the TSP task). These outputs, paired with the predicted solution s, are then fed back into the LLMs to iteratively refine the hypothesis. The feedback can include the simulation loss relative to the target metric $\mathcal{L}$, as well as auxiliary information during the optimization process, which can provide more insights for improving solutions. For instance, if $\mathcal{L}$ represents the total distance of a solution for a TSP task, the auxiliary information may include the specific distance between two given points or the relative magnitude of distances between pairs of points (e.g. simulator can provide auxiliary information that (v_4, v_1) has a larger distance than (v_2, v_4) in Figure 2).

4.3 Outer-level LLM Optimization

Many studies have demonstrated that LLMs are capable of discovering high-quality solutions and can match or even outperform hand-designed heuristic algorithms (Yang et al., 2023; Romera-Paredes et al., 2024c). Therefore, we design outer-level optimization to: **(i)** analyze the current optimization task and make reasonable hypotheses; **(ii)** utilize the feedback from the simulations (including experimental phenomena and loss values, etc.), and update the LLM's knowledge through model editing; **(iii)** design and propose new potential solutions and input into the simulation platforms.

Inspired by the scientific optimization process by human scientists, we observe that when confronted with a new problem, scientists tend to be *daring adventurers*, utilizing expert knowledge and problem parameters to thoroughly explore the search space, eliminating improbable parameter regions. This may significantly reduce the subsequent solution space, yielding solutions that satisfy optimization constraints, though they may not be optimal (Wuestman et al., 2020). Then, after gaining a deeper understanding of the problem through initial attempts, they change to be *cautious followers* to keep the trajectory and trail previous solutions, adhering more closely to previous optimizations. This allows them to continually improve the metrics while satisfying optimization constraints, ultimately approaching the optimal solution. We thus design a **dynamic exploitation and exploration strategy** to mimic this process.

Specifically, we calculate the loss change as $\Delta L = \frac{L_{\text{prev}} - L_{\text{cur}}}{L_{\text{prev}}}$. If $\Delta L > 0$, this indicates that the current hypothetical solutions are better than

the previous ones. We then apply a lower decoding temperature to allow the LLM to follow the trace, i.e., $T_{\text{cur}} = T_{\text{pre}} \times \left(\frac{1}{1+\Delta L}\right)$; if $\Delta L < 0$, this indicates that current solutions are worse than the previous ones. We thus apply a higher decoding temperature to be more exploratory and adventurous, i.e., $T_{\text{cur}} = T_{\text{pre}} \times (1 + |\Delta L|)$. We also clip the temperature to $[0, 1]$ in case that $T_{\text{cur}} < 0$ or $T_{\text{cur}} > 1$. Empirically, we divide solutions into $\mathcal{S}_{\text{exploit}}$ and $\mathcal{S}_{\text{explore}}$. We observe that (i) $\mathcal{S}_{\text{exploit}}$ often contains repetitive solutions from previous iterations, and (ii) $\mathcal{S}_{\text{explore}}$ tends to yield solutions to be informative for guiding optimization, or otherwise infeasible.

4.4 Bi-level Interactions

The key challenge in integrating the two levels of optimization is developing a protocol that facilitates efficient, structured, and flexible communication between them. Therefore, we propose an interaction strategy considering the two aspects. **On the one hand**, **from simulation platforms to LLMs**, drawing upon (Meng et al., 2022), we incorporate the feedback from simulation platforms as new knowledge edited into the model. We **first** locate how facts are stored within the parameters of LLMs. We begin by analyzing and identifying which specific layers and their parameters W have the strongest causal effect on predictions of individual facts through **causal tracing** (Meng et al., 2022) (Detailed steps and results of the causal tracing for different LLMs are in Appendix B). **Next**, we integrate the experimental results from the simulation platforms as new knowledge, formatted as triples, into the model. For instance, in the context of the TSP problem, suppose the simulation platform's feedback template includes: (i) the path length for the solution $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4$ is 108, (ii) the distance between v_1 and v_4 is greater than that between v_2 and v_4, and etc. These statements are then transformed into triples, such as $(v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4,\ \text{has distance of},\ 108)$ and $(v_1, v_4),\ \text{greater than},\ (v_2, v_4))$, which are subsequently edited into the model. **Finally**, by collecting such triples (s, r, o), we compile a series of key-value pairs for a set of vector keys $S = [s_1, s_2, \ldots, s_n]$ and corresponding vector values $O = [o_1, o_2, \ldots, o_n]$ in each iteration step. Therefore, for the u new (s, r, o) pairs obtained from the next iteration step, the objective of editing can be given by the following formula:

$$W_1 \triangleq \arg\min_{\hat{W}} \left(\sum_{i=1}^{n} \left\| \hat{W} s_i - o_i \right\|^2 + \sum_{i=n+1}^{n+u} \left\| \hat{W} s_i - o_i \right\|^2 \right)$$

Here, W denotes the original matrix and W_1 denotes the edited weight matrix. Due to this straightforward algebraic structure, any fact can be inserted directly once (s, o) is determined. We can update the parameters of the LLM by editing the newly obtained u parameter-feedback pairs all at once, which adaptively utilizes the feedback from each simulation platform interaction into the LLM. It directly updates the LLM's weight parameters and prevents the input prompt length from increasing with iterations, alleviating the loss-in-the-middle issue during long-text and multi-round optimization processes. **On the other hand, from LLMs to simulation platforms**, we leverage the LLM as a domain expert to guide the setup of each subsequent simulation based on the results of the previous results. This process is akin to an experienced scientist providing tailored guidance for the experimental configuration of the next step simulation based on the current results. The simulation platforms can then continue running experiments based on the guidance of the LLM to provide observational feedback. Specifically, each time the outer-level LLM receives feedback from the previous iteration of simulation platforms, it can simultaneously request intermediate results from the inner-level platforms to aid in its reasoning. Taking the TSP problem as an example, when the LLM provides the solution $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_1$, it can also request the platforms to provide the specific distance between two points. This additional feedback can then assist in refining the reasoning for the next iteration.

5 Experiments

5.1 Problem Descriptions

We provide a brief definition of each task. More detailed definitions are in Appendix F. For the **linear system regression** task, the objective is to estimate the linear coefficients to model relationships between input and corresponding output (Fisher, 1922). We use **"#steps"** (optimization steps for successfully finding the optimal solution) **as the metric**. We follow the same dataset setting as OPRO (Yang et al., 2023). For the **TSP** task, the objective is that given a set of n nodes with known coordinates, it seeks to find the shortest possible route

Table 1: Results of our GSO against **6** baselines using GPT-J 6B, Llama3 8B, and Mistral 7B as three representative backbone models (for more results of different backbone models, please see Appendix E). Our experiments encompass **7** different tasks, which are divided into linear system regression (LSR) **(a)**, travel salesman problem (TSP) **(b)**, constitutive law prediction **(c-d)**, and molecule property prediction **(e-g)**. We report the mean ± standard error of each optimization result. The symbol N/A indicates that the model cannot provide a feasible solution for the current task. **A lower value is preferable across all tasks.** The best results are highlighted in **bold** text.

Backbone	Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
		(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
GPT-J 6B	**Vanilla**	N/A	6.0 ± 2.0	N/A	N/A	N/A	N/A	N/A
	CoT	N/A	4.4 ± 1.9	N/A	N/A	N/A	N/A	N/A
	Funsearch	29.7 ± 10.2	0.9 ± 0.1	77.4 ± 19.5	58.7 ± 17.2	230.8 ± 30.0	301.6 ± 51.9	34.9 ± 6.8
	Eureka	45.1 ± 19.8	1.9 ± 0.8	155.3 ± 31.2	91.7 ± 18.5	390.0 ± 33.8	388.0 ± 70.3	55.3 ± 8.4
	OPRO	41.0 ± 18.0	0.4 ± 0.2	60.9 ± 22.6	81.3 ± 10.2	455.9 ± 83.3	155.8 ± 18.9	23.7 ± 4.0
	SGA	23.9 ± 5.0	0.2 ± 0.1	84.7 ± 17.9	73.2 ± 21.0	238.1 ± 17.5	89.5 ± 13.0	15.5 ± 3.1
	GSO (ours)	**12.3 ± 4.8**	**0.0 ± 0.0**	**15.7 ± 5.0**	**55.4 ± 8.0**	**70.1 ± 17.7**	**83.1 ± 13.9**	**8.5 ± 2.0**

Backbone	Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
		(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
Llama3 8B	**Vanilla**	N/A	6.0 ± 2.0	N/A	N/A	N/A	N/A	N/A
	CoT	N/A	4.4 ± 1.9	3397.0 ± 298.9	N/A	N/A	N/A	N/A
	Funsearch	15.7 ± 8.3	1.3 ± 0.5	198.9 ± 30.8	335.9 ± 112.1	433.7 ± 25.5	175.5 ± 28.8	91.4 ± 19.8
	Eureka	18.3 ± 5.1	2.1 ± 0.2	255.6 ± 80.9	401.3 ± 98.0	260.7 ± 38.9	130.1 ± 15.9	155.1 ± 28.3
	OPRO	17.0 ± 4.7	0.3 ± 0.1	74.1 ± 10.3	227.8 ± 33.2	537.1 ± 69.7	115.5 ± 35.3	51.9 ± 10.4
	SGA	10.2 ± 3.7	0.5 ± 0.1	89.1 ± 5.9	150.0 ± 40.3	235.1 ± 55.7	94.1 ± 22.0	30.5 ± 15.3
	GSO (ours)	**5.1 ± 1.0**	**0.0 ± 0.0**	**8.1 ± 2.1**	**20.1 ± 3.9**	**30.1 ± 14.9**	**20.9 ± 9.3**	**9.7 ± 3.6**

Backbone	Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
		(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
Mistral 7B	**Vanilla**	N/A	6.0 ± 2.0	N/A	N/A	N/A	N/A	N/A
	CoT	N/A	4.4 ± 1.9	N/A	N/A	N/A	N/A	N/A
	Funsearch	67.3 ± 14.7	1.3 ± 0.3	317.9 ± 43.0	199.4 ± 31.0	335.9 ± 61.3	310.4 ± 33.1	86.8 ± 17.4
	Eureka	37.9 ± 13.1	2.5 ± 0.8	337.3 ± 51.1	379.0 ± 139.1	201.3 ± 33.7	331.7 ± 25.4	110.4 ± 19.5
	OPRO	N/A	0.3 ± 0.1	88.6 ± 12.1	275.7 ± 54.1	55.7 ± 13.9	252.4 ± 31.7	12.5 ± 3.9
	SGA	27.9 ± 5.8	0.3 ± 0.1	145.1 ± 11.7	227.3 ± 19.9	130.5 ± 33.1	55.7 ± 12.1	55.4 ± 18.4
	GSO (ours)	**5.6 ± 3.0**	**0.1 ± 0.0**	**3.2 ± 1.7**	**13.0 ± 2.1**	**23.9 ± 5.1**	**2.8 ± 1.5**	**1.7 ± 0.4**

that visits each node once and returns to the starting point. We follow the same dataset setting as OPRO (Yang et al., 2023) and use the Gurobi solver (Gurobi Optimization, LLC, 2024) to construct the oracle solutions and compute the **optimality gap as the metric** (the difference between the distance in the solution by the evaluated approach and by the oracle solution, divided by the distance of the oracle solution). For the **constitutive law prediction** task, we consider both fitting linear and non-linear materials and follow the same dataset setting as SGA (Ma et al.). We use **mean square error (MSE) as the metric**. For the **molecular property prediction** task, we consider three tasks: predict a molecule's highest occupied molecular orbital (HOMO), lowest unoccupied molecular orbital (LUMO), and the HOMO-LUMO gap based on their conformations and quantum mechanical properties. We follow SGA to use the QM9 dataset (Ramakrishnan et al., 2014) for experiments. We also use **MSE as the metric**.

5.2 Experiment Setups

Implementation Details. We apply LLMs including GPT-J 6B (Radford et al., 2019), Llama3 8B (Team, 2024b), Mistral 7B (Jiang et al., 2023), Llama2 13B (Team, 2023c), Yi9b (Young et al., 2024), and Internlm 7B (Team, 2023b). We follow (Ma et al., 2023b) to conduct all experiments five times using different random seeds to guarantee stable and reproducible results. More details of implementation details are in Appendix F.

We consider **six** strong baselines for evaluation: (i) vanilla LLMs without additional modules. Vanilla LLMs represent the original capabilities of LLMs. (ii) **CoT** prompting (Wei et al., 2022) solves the problem by looking at step-by-step solutions from examples. We provide 5 examples with explanations as the initial solutions. (iii) **Funsearch** (Romera-Paredes et al., 2024b) utilizes evolutionary strategy to avoid local optimum. (iv) **Eureka** (Ma et al., 2023b) generates multiple solutions in each iteration to improve the success rate.

Table 2: The results of the ablation study of our GSO on the seven scientific optimization tasks, using Llama3 8B as the backbone model (We provide more results for the other *five* backbone models, in Appendix G).

Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
	(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
$\text{GSO}_{w/o\ edit}$	33.7 ± 10.1	0.7 ± 0.3	116.1 ± 30.1	141.5 ± 36.9	307.7 ± 31.4	331.2 ± 89.2	164.3 ± 38.9
$\text{GSO}_{w/o\ dynamic}$	6.5 ± 3.5	0.0 ± 0.0	23.7 ± 9.9	76.9 ± 11.4	35.0 ± 17.1	45.0 ± 19.3	21.9 ± 7.3
GSO	**5.1 ± 1.0**	**0.0 ± 0.0**	**8.1 ± 2.1**	**20.1 ± 3.9**	**30.1 ± 14.9**	**20.9 ± 9.3**	**9.7 ± 3.6**

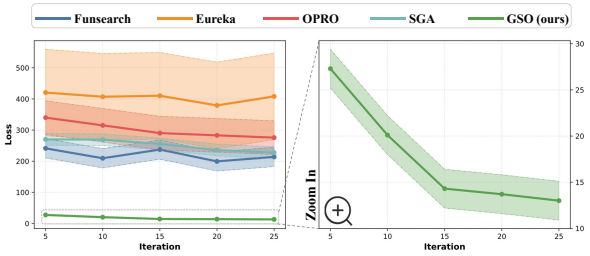

Figure 3: We visualize the average MSE loss values of each method for the non-linear Constitutive Law task (d) across five random seeds at the same optimization steps using Mistral 7B as the backbone model, with shading representing the standard deviation.

(v) **OPRO** (Yang et al., 2023) highlights the advantages of involving a sorted optimization trajectory. (vi) **SGA** (Ma et al.) utilizes a top-k heap to generate more diverse solutions.

5.3 Main Results

We conduct our experiments on the 7 designed tasks using GPT-J 6B, Llama3 8B, and Mistral 7B as three representative backbone models in Table 1. We also provide more results for other *three* different backbone models in Table 5 in Appendix E to demonstrate the versatility of our GSO across different backbones. GSO enables tasks, which are challenging to effectively optimize with traditional Vanilla and CoT methods, to become feasible. We also observe that GSO significantly and consistently outperforms existing methods on the scientific optimization tasks, which demonstrates the effectiveness of our GSO. Notably, for the molecule property prediction task on predicting the HOMO-LUMO gap, GSO achieves a maximum precision improvement of over 32.6× when utilizing Mistral 7B as the backbone model. These results underscore the importance of effectively utilizing the observational feedback to adaptively adjust its optimization directions. The universality of *six* popular open-source backbone models in Appendix E also suggests the generalizability of our GSO.

5.4 Ablation Study

To further investigate the contribution of each component within GSO, we conduct a series of ablation experiments on the entire framework. Specifically, we denote GSO without *edit* as $\text{GSO}_{w/o\ edit}$, GSO without the dynamic strategy as $\text{GSO}_{w/o\ dynamic}$, respectively. We use Llama3 8B as the backbone model, the results of other backbone models are in Appendix G. We present the ablation results of GSO using Llama 3 8B as the backbone model in Table 2. More Results using the other five different backbone models are in Appendix G. As shown in Table 2, the absence of any component within GSO results in a performance degradation of the entire framework. Notably, $\text{GSO}_{w/o\ edit}$ exhibits more significant impacts on the performance of GSO, which demonstrates the importance of effectively utilizing the observational feedback to adaptively adjust the optimization direction.

5.5 Case Study

Robustness of the Prompt As mentioned in Section 1, one appealing feature of our GSO compared to other methods is its robustness to prompts. Prompts with similar semantics do not require meticulous crafting to yield consistently promising results. To provide more insights into our GSO, we manually generated two prompts and used OpenAI o1 to rewrite an additional three based on the original task prompts for each task (details of the generated prompts are in Appendix D). As shown in Table 3, we observe that prompt-based methods tend to be sensitive to the prompts, with semantically similar prompts often leading to fluctuating results across many tasks. This necessitates significant effort from users to perform prompt 'tuning' during application. In contrast, **our GSO consistently produces robust results across different prompts, with minimal variation between semantically similar prompts.** This reduces the need for extensive prompt "tuning," highlighting its potential for broader real-world applications.

Table 3: We evaluate our GSO against other baseline methods on five prompts that vary in format but are semantically similar, and report their average results of each method, using Mistral 7B as the representative backbone model. The symbol N/A indicates that the model cannot provide a feasible solution for the current task.

Backbone	Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
		(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
Mistral 7B	**Funsearch**	52.2 ± 29.4	2.4 ± 1.5	411.0 ± 193.4	254.9 ± 109.3	473.3 ± 130.2	403.1 ± 135.0	173.0 ± 66.3
	Eureka	60.25 ± 29.2	3.5 ± 2.6	303.7 ± 180.3	179.0 ± 65.8	219.0 ± 40.6	287.9 ± 99.0	148.9 ± 55.0
	OPRO	N/A	0.4 ± 0.3	107.6 ± 61.4	224.8 ± 34.2	94.3 ± 50.6	593.7 ± 266.8	40.5 ± 28.0
	SGA	38.7 ± 20.3	0.3 ± 0.2	55.4 ± 43.9	203.2 ± 22.3	211.7 ± 98.0	73.0 ± 18.8	62.9 ± 33.2
	GSO (ours)	**8.3 ± 3.3**	**0.1 ± 0.0**	**5.5 ± 3.1**	**20.9 ± 8.3**	**18.8 ± 7.0**	**3.9 ± 2.1**	**2.2 ± 0.4**

Table 4: Results of our GSO against popular closed-source LLMs, including GPT-4o and Claude-3.5. We present the results of our GSO Mistral 7B as the backbone model for comparison.

Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
	(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
GPT-4o	6.4 ± 1.5	0.2 ± 0.0	59.7 ± 7.4	45.5 ± 9.1	181.4 ± 33.7	313.5 ± 45.4	67.8 ± 10.9
Claude-3	12.1 ± 2.0	0.2 ± 0.1	128.1 ± 13.5	103.0 ± 5.2	162.3 ± 21.2	519.3 ± 37.4	37.8 ± 8.3
GSO	**5.6 ± 3.0**	**0.1 ± 0.0**	**3.2 ± 1.7**	**13.0 ± 2.1**	**23.9 ± 5.1**	**2.8 ± 1.5**	**1.7 ± 0.4**

Loss Curve and Decoding Temperature We also investigate the impact of the number of lengthy input prompts as optimization iterations increase on each method. We present the feedback loss trend for the non-linear constitutive law task (d) in Figure 3 as an example. We observe that GSO achieves significantly lower loss and exhibits a clear convergence trend compared to existing baselines. This demonstrates that GSO can effectively leverage feedback from each iteration to achieve stable and consistent improvements in scenarios involving lengthy prompts. Notably, some methods show improvement at the beginning (i.e., loss reduction) when the initial prompt is short and the optimization space is large, but as the number of optimization steps increases, the feedback loss fluctuates, making further optimization difficult. In contrast, GSO effectively leverages each step feedback to adaptively adjust its optimization direction, leading to a stable and consistent loss decrease. These results demonstrate that **GSO can effectively observational feedback from lengthy prompts to facilitate further optimization, thereby alleviating the loss in the middle issue.**

Comparison with Advanced Closed-source LLMs We also compared our GSO framework on open-source models with the current state-of-the-art closed-source model, GPT-4o[1] and Claude-3-Sonnet[2]. As shown in Table 4, our GSO utilizing Mistral 7B can consistently outperform GPT-4o and Claude-3-Sonnet, demonstrating the effectiveness of our approach. Note that our method is orthogonal to the choice of backbone model, making it a versatile plug-and-play module that can be directly applied to more advanced LLMs to further achieve enhanced results.

6 Conclusion

In this paper, we propose a novel **G**eneral **S**cientific **O**ptimizers method, effectively enabling LLMs to utilize observational feedback from each optimization step. Specifically, GSO consists of a bi-level optimization framework: outer-level LLMs function as knowledgeable and versatile scientists, generating new hypotheses to optimize experimental hyperparameters; inner-level simulations function as experimental platforms to perform numerical simulations to these hypotheses and provide observational feedback; a bi-level interaction then update the simulators together with the expert knowledge within LLMs via model editing. GSO effectively guides LLMs to derive a more precise and stable optimization direction, yielding superior optimization results. Extensive experiments on *six* different open-source and *seven* different scientific tasks demonstrate the superiority of our GSO, delivering consistent, robust, generalizable, and nearly monotonic improvement. We view our GSO as a trailblazer, establishing a new paradigm for utilizing LLMs and simulations as bi-level optimization to further advancements in scientific optimizations.[3]

[1] https://openai.com/o1/

[2] https://www.anthropic.com/news/claude-3-5-sonnet

[3] More discussions on GSO are in Appendix K.

Acknowledgements

The authors would like to thank all the anonymous reviewers for their insightful comments.

7 Limitations

We consider a few limitations and future directions. (i) The content generated by LLMs exhibits a certain degree of randomness, and the optimization process cannot guarantee interpretability or transparency. (ii) LLM inference requires large computational resources and thus increases expense. It paves the way for research on LLM inference acceleration to expedite our GSO (Leviathan et al., 2023; Liu et al., 2024a). (iii) GSO requires access to model weights, which limits its applicability to closed-source models like GPT-4 and Claude 3.5. We believe that as the community progresses and the performance gap between open-source and closed-source models narrows, our GSO will be able to demonstrate its capabilities more effectively.

References

Microsoft Research AI4Science and Microsoft Azure Quantum. 2023. The impact of large language models on scientific discovery: a preliminary study using gpt-4. *arXiv preprint arXiv:2311.07361*.

Shun-ichi Amari. 1993. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196.

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. 2023. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578.

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

Angelica Chen, David Dohan, and David So. 2024a. Evoprompting: language models for code-level neural architecture search. *Advances in Neural Information Processing Systems*, 36.

Angelica Chen, Jérémy Scheurer, Tomasz Korbak, Jon Ander Campos, Jun Shern Chan, Samuel R Bowman, Kyunghyun Cho, and Ethan Perez. 2023. Improving code generation by training with natural language feedback. *arXiv preprint arXiv:2303.16749*.

Hanzhu Chen, Xu Shen, Qitan Lv, Jie Wang, Xiaoqi Ni, and Jieping Ye. 2024b. Sac-kg: Exploiting large language models as skilled automatic constructors for domain knowledge graph. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4345–4360.

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

Xinyun Chen and Yuandong Tian. 2019. Learning to perform local rewriting for combinatorial optimization. *Advances in neural information processing systems*, 32.

Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Richard Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'aurelio Ranzato, et al. 2022. Towards learning universal hyperparameter optimizers with transformers. *Advances in Neural Information Processing Systems*, 35:32053–32068.

Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. 2020. Chemberta: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*.

Tânia FGG Cova and Alberto ACC Pais. 2019. Deep learning for deep chemistry: optimizing the prediction of chemical patterns. *Frontiers in chemistry*, 7:809.

Jeff Da, Ronan Le Bras, Ximing Lu, Yejin Choi, and Antoine Bosselut. 2021. Analyzing commonsense emergence in few-shot knowledge models. *arXiv preprint arXiv:2101.00297*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*.

Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. 2018. Learning heuristics for the tsp by policy gradient. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26–29, 2018, Proceedings 15*, pages 170–181. Springer.

Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W Cohen. Differentiable reasoning over a virtual knowledge base. In *International Conference on Learning Representations*.

Vanessa Didelez and Iris Pigeot. 2001. Causality: models, reasoning, and inference.

R. A. Fisher. 1922. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A*, 222:309–368.

Santo Fortunato, Carl T Bergstrom, Katy Börner, James A Evans, Dirk Helbing, Staša Milojević, Alexander M Petersen, Filippo Radicchi, Roberta Sinatra, Brian Uzzi, et al. 2018. Science of science. *Science*, 359(6379):eaao0185.

Elias Frantar and Dan Alistarh. 2023. Qmoe: Practical sub-1-bit compression of trillion-parameter models. *arXiv preprint arXiv:2310.16795*.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Bruce Golden, Lawrence Bodin, T Doyle, and W Stewart Jr. 1980. Approximate traveling salesman algorithms. *Operations research*, 28(3-part-ii):694–711.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*.

Gurobi Optimization, LLC. 2024. Gurobi Optimizer Reference Manual.

Gregory Gutin and Abraham P Punnen. 2006. *The traveling salesman problem and its variations*, volume 12. Springer Science & Business Media.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2024. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36.

Shibo Hao, Bowen Tan, Kaiwen Tang, Bin Ni, Xiyan Shao, Hengzhe Zhang, Eric Xing, and Zhiting Hu. 2023. Bertnet: Harvesting knowledge graphs with arbitrary relations from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5000–5015.

Keld Helsgaun. 2017. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12:966–980.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Michael D Intriligator. 2002. *Mathematical optimization and economic theory*. SIAM.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. 2016. The material point method for simulating continuum materials. In *Acm siggraph 2016 courses*, pages 1–52.

Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. 1995. The traveling salesman problem. *Handbooks in operations research and management science*, 7:225–330.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2024. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Wouter Kool, Herke Van Hoof, and Max Welling. 2018. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*.

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. 2022. Evolution through large models. *Preprint*, arXiv:2206.08896.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Jiatong Li, Yunqing Liu, Wenqi Fan, Xiao-Yong Wei, Hui Liu, Jiliang Tang, and Qing Li. 2024a. Empowering molecule discovery for molecule-caption translation with large language models: A chatgpt perspective. *IEEE Transactions on Knowledge and Data Engineering*.

Junxian Li, Di Zhang, Xunzhi Wang, Zeying Hao, Jingdi Lei, Qian Tan, Cai Zhou, Wei Liu, Yaotian Yang, Xinrui Xiong, Weiyun Wang, Zhe Chen,

Wenhai Wang, Wei Li, Shufei Zhang, Mao Su, Wanli Ouyang, Yuqiang Li, and Dongzhan Zhou. 2024b. Chemvlm: Exploring the power of multimodal large language models in chemistry area. *Preprint*, arXiv:2408.07246.

Junxian Li, Di Zhang, Xunzhi Wang, Zeying Hao, Jingdi Lei, Qian Tan, Cai Zhou, Wei Liu, Yaotian Yang, Xinrui Xiong, Weiyun Wang, Zhe Chen, Wenhai Wang, Wei Li, Shufei Zhang, Mao Su, Wanli Ouyang, Yuqiang Li, and Dongzhan Zhou. 2024c. Chemvlm: Exploring the power of multimodal large language models in chemistry area. *Preprint*, arXiv:2408.07246.

Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, and Winston Hu. 2024a. Parallel speculative decoding with adaptive draft length. *Preprint*, arXiv:2408.11850.

Tianyu Liu, Qitan Lv, Jie Wang, Shuling Yang, and Hanzhu Chen. 2024b. Learning rule-induced subgraph representations for inductive relation prediction. *Advances in Neural Information Processing Systems*, 36.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Zhichao Liu, Ruth A Roberts, Madhu Lal-Nag, Xi Chen, Ruili Huang, and Weida Tong. 2021. Ai-based language models powering drug discovery and development. *Drug Discovery Today*, 26(11):2593–2607.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

Qitan Lv, Jie Wang, Hanzhu Chen, Bin Li, Yongdong Zhang, and Feng Wu. 2024. Coarse-to-fine highlighting: Reducing knowledge hallucination in large language models. In *Forty-first International Conference on Machine Learning*.

Pingchuan Ma, Peter Yichen Chen, Bolei Deng, Joshua B Tenenbaum, Tao Du, Chuang Gan, and Wojciech Matusik. 2023a. Learning neural constitutive laws from motion observations for generalizable pde dynamics. In *International Conference on Machine Learning*, pages 23279–23300. PMLR.

Pingchuan Ma, Tsun-Hsuan Wang, Minghao Guo, Zhiqing Sun, Joshua B Tenenbaum, Daniela Rus, Chuang Gan, and Wojciech Matusik. Llm and simulation as bilevel optimizers: A new paradigm to advance physical scientific discovery. In *Forty-first International Conference on Machine Learning*.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023b. Eureka: Human-level reward design via coding large language models. In *The Twelfth International Conference on Learning Representations*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Aman Madaan and Amir Yazdanbakhsh. 2022. Text and patterns: For effective chain of thought, it takes two to tango. *arXiv preprint arXiv:2209.07686*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Elliot Meyerson, Mark J Nelson, Herbie Bradley, Adam Gaier, Arash Moradi, Amy K Hoover, and Joel Lehman. 2023a. Language model crossover: Variation through few-shot prompting. *arXiv preprint arXiv:2302.12170*.

Elliot Meyerson, Mark J Nelson, Herbie Bradley, Adam Gaier, Arash Moradi, Amy K Hoover, and Joel Lehman. 2023b. Language model crossover: Variation through few-shot prompting. *arXiv preprint arXiv:2302.12170*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. In *International Conference on Learning Representations*.

Douglas C. Montgomery, Elizabeth A. Peck, and Geoffrey G. Vining. 2021. *Introduction to Linear Regression Analysis*, 6th edition edition. Wiley.

Varun Nair, Elliot Schumacher, Geoffrey Tso, and Anitha Kannan. 2023. Dera: enhancing large language model completions with dialog-enabled resolving agents. *arXiv preprint arXiv:2303.17071*.

Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. 2018. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31.

Leland Gerson Neuberg. 2003. Causality: models, reasoning, and inference, by judea pearl, cambridge university press, 2000. *Econometric Theory*, 19(4):675–685.

Theo X Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. 2023. Demystifying gpt self-repair for code generation. *arXiv preprint arXiv:2306.09896*.

OpenAI. 2020. Chatgpt: A large-scale generative model for conversation.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Karl Popper. 2005. *The logic of scientific discovery*. Routledge.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. 2014. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7.

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

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

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

Daniel J Rosenkrantz, Richard E Stearns, and Philip M Lewis, II. 1977. An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, 6(3):563–581.

George A. F. Seber and Alan J. Lee. 2012. *Linear Regression Analysis*, 2nd edition edition. Wiley.

Gaurav Sharma and Abhishek Thakur. 2023. Chatgpt in drug discovery.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2(5):9.

Mohammad Shoeybi, Md.MostofaAli Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *Cornell University - arXiv,Cornell University - arXiv*.

Deborah Sulsky, Shi-Jian Zhou, and Howard L Schreyer. 1995. Application of a particle-in-cell method to solid mechanics. *Computer physics communications*, 87(1-2):236–252.

Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*.

Anthropic Team. 2024a. The claude 3 model family: Opus, sonnet, haiku.

Gemini Team. 2023a. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

InternLM Team. 2023b. Internlm: A multilingual language model with progressively enhanced capabilities.

Llama2 Team. 2023c. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Llama3 Team. 2024b. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

PaLM Team. 2022. Palm: Scaling language modeling with pathways. *Preprint*, arXiv:2204.02311.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482.

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

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. 2023b. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Huaijie Wang, Lingxiao Ma, Fan Yang, Ruiping Wang, Yi Wu, and Furu Wei. 2023c. Bitnet: Scaling 1-bit transformers for large language models. *arXiv preprint arXiv:2310.11453*.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. In *The Eleventh International Conference on Learning Representations*.

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

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. 2023. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*.

Mignon Wuestman, Jarno Hoekman, and Koen Frenken. 2020. A typology of scientific breakthroughs. *Quantitative Science Studies*, 1(3):1203–1222.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. 2024. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*.

Weizhe Yuan, Kyunghyun Cho, and Jason Weston. 2024. System-level natural language feedback. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2773–2789.

Di Zhang, Wei Liu, Qian Tan, Jingdan Chen, Hang Yan, Yuliang Yan, Jiatong Li, Weiran Huang, Xiangyu Yue, Dongzhan Zhou, et al. 2024a. Chemllm: A chemical large language model. *arXiv preprint arXiv:2402.06852*.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. 2024b. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*.

Starkson Zhang, Alfredo Pearson, and Zhenting Wang. Autonomous generalist scientist: Towards and beyond human-level automatic research using foundation model-based ai agents and robots (a position).

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. 2024. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209.

Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*.

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

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*.

A More Related Works

A.1 Classical Language Model Methods for Scientific Optimization

Besides the methods using LLMs for scientific optimization in Section 3, these are several classical studies that have fine-tuned language models (LMs) to function as mutation and crossover operators within evolutionary algorithms (Meyerson et al., 2023a; Zhang et al., 2024a; Li et al., 2024c). LMX (Meyerson et al., 2023b) employs language models guided by few-shot exemplars to generate evolutionary crossovers in tasks like image and code generation, enhancing the model's ability to adapt and innovate across diverse domains. ELM (Lehman et al., 2022) trains an LLM to generate code diffs, which serve as the mutation operator, and introduces a fine-tuning method to enhance performance in the Sodarace domain for robotic simulation. Evo-Prompting (Chen et al., 2024a) leverages large language models to evolve neural network architectures by integrating evolutionary search with soft prompt tuning. OptFormer (Chen et al., 2022) incorporates trajectories as input for optimization by training a transformer model on extensive hyperparameter optimization datasets. These methods primarily focus on fine-tuning LLMs with domain-specific data to align natural language with scientific information. However, these approaches are domain-bound and demand substantial data which also limits their broader applicability.

A.2 Large Language Models

Language models such as GPT (Radford et al., 2018), BERT (Kenton and Toutanova, 2019), RoBERTa (Liu et al., 2019), and Megatron-LM (Shoeybi et al., 2019) have led to a learning paradigm shift in natural language processing (NLP). Models are first pre-trained on extensive volumes of unlabeled text corpora with language modeling objectives and then fine-tuned on downstream tasks. Recently, large language models (LLMs) including LLama (Team, 2024b, 2023c) ChatGPT (OpenAI, 2020) GPT4 (OpenAI, 2023), PaLM (Team, 2022), Gemini (Team, 2023a), and Claude3 (Team, 2024a) have shown great performance in both few-shot and even zero-shot scenarios (Brown et al., 2020). To further enhance the interpretability of these LLMs, some research endeavors explain LLMs through attribution analysis (Wang et al.; Hanna et al., 2024; Gurnee et al.; Chen et al., 2024b). Another line of work aims to retrieve the knowledge explicitly from LLMs as the basis for interpreting them, including the reasoning task (Shi et al., 2023) and the QA task (Hao et al., 2023; Dhingra et al.; Guu et al., 2020).

B Results of the Causal Tracing for Different LLMs

Causal tracing has emerged as a pivotal methodology for dissecting and understanding the internal mechanisms of model (Didelez and Pigeot, 2001). This technique facilitates the identification and modification of specific factual associations within a model without necessitating comprehensive retraining (Neuberg, 2003). In the context of model editing, causal tracing enables precise interventions by isolating the neural correlates responsible for particular behaviors or outputs. We follow (Meng et al., 2022) to build a **causal graph** (Neuberg, 2003) to describe dependencies between the hidden variables. This graph illustrates numerous pathways from the input on the left to the output (next-word prediction) at the lower right. Our aim is to **determine whether specific hidden state variables are more important than others in the process of recalling a fact**.

To quantify each state's contribution to a correct factual prediction, we analyze all of LLM's internal activations across three runs: **a clean run** that accurately predicts the fact, **a corrupted run** where the prediction is impaired, and **a corrupted-with-restoration run** that evaluates the ability of a single state to restore the correct prediction.

Let $\mathbb{P}[o]$, $\mathbb{P}_*[o]$, and $\mathbb{P}_{*,\text{ clean } h_i^l}[o]$ denote the probability of emitting the given entity o under the clean, corrupted, and corrupted-with-restoration runs, respectively; dependence on the input x is omitted for notational simplicity. The **total effect** (TE) is defined as the difference between two probabilities:

$$\text{TE} = \mathbb{P}[o] - \mathbb{P}_*[o].$$

The **indirect effect** (IE) of a specific mediating state $\hat{h_i}^l$ is defined as the difference between the probability of o under the corrupted condition and the probability of o when that state is restored to its clean version, while the subject remains in a corrupted state:

$$\text{IE} = \mathbb{P}_{*,\text{clean } h_i{}^l}[o] - \mathbb{P}_*[o].$$

By averaging over a sample of statements, we can derive the **average indirect effect (AIE)** for

Table 5: Results of our GSO against **6** baselines using Llama2 13B, Yi 9B, and Internlm 7B as the backbone models. Our experiments encompass **7** different tasks, which are divided into linear system regression (LSR) **(a)**, travel salesman problem (TSP) **(b)**, constitutive law prediction **(c-d)**, and molecule property prediction **(e-g)**. For the LSR task, we use the number of steps of successfully finding the optimal solution as the metric. For the TSP task, we use the optimality gap as the metric. For the rest five tasks, we use MSE loss as the metric. We calculate the mean ± standard error of each optimization result. The symbol N/A indicates that the model is unable to provide a feasible solution for the current task. A lower value is preferable across all tasks. The best results are highlighted in **bold** text.

Backbone	Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
		(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
Llama2 13B	**Vanilla**	N/A	6.0 ± 2.0	N/A	N/A	N/A	N/A	N/A
	CoT	N/A	4.4 ± 1.9	N/A	N/A	N/A	N/A	N/A
	Funsearch	N/A	2.5 ± 1.0	170.9 ± 20.1	255.1 ± 31.1	108.5 ± 13.0	115.5 ± 20.7	59.7 ± 8.3
	Eureka	N/A	2.7 ± 1.5	211.7 ± 50.4	131.7 ± 21.9	98.2 ± 11.3	220.1 ± 30.1	39.3 ± 13.9
	OPRO	28.7 ± 10.8	0.9 ± 0.3	55.8 ± 10.4	165.8 ± 52.9	168.5 ± 31.7	355.4 ± 43.7	35.9 ± 10.7
	SGA	30.0 ± 10.2	0.2 ± 0.1	31.7 ± 10.1	55.4 ± 15.8	87.1 ± 17.0	89.5 ± 17.7	27.6 ± 5.4
	GSO (ours)	**5.0 ± 1.6**	**0.1 ± 0.1**	**2.9 ± 1.3**	**7.4 ± 3.1**	**48.1 ± 12.0**	**79.1 ± 13.3**	**8.3 ± 3.2**

Backbone	Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
		(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
Yi 9B	**Vanilla**	N/A	6.0 ± 2.0	N/A	N/A	N/A	N/A	N/A
	CoT	N/A	4.4 ± 1.9	N/A	N/A	N/A	N/A	N/A
	Funsearch	19.6 ± 6.0	2.2 ± 0.2	201.4 ± 10.3	229.5 ± 31.3	201.0 ± 55.3	198.9 ± 40.1	135.0 ± 20.7
	Eureka	8.6 ± 4.0	1.8 ± 0.6	130.1 ± 22.1	381.1 ± 98.8	1559.1 ± 100.7	301.9 ± 38.7	98.3 ± 10.1
	OPRO	9.5 ± 4.4	0.3 ± 0.2	94.0 ± 23.3	163.2 ± 28.9	850.7 ± 94.0	746.9 ± 31.0	129.1 ± 23.5
	SGA	22.5 ± 5.1	0.7 ± 0.4	50.3 ± 7.9	104.4 ± 19.3	133.9 ± 23.1	**168.4 ± 59.0**	39.6 ± 5.3
	GSO (ours)	**3.0 ± 0.8**	**0.0 ± 0.0**	**5.9 ± 2.1**	**89.1 ± 33.9**	**67.9 ± 23.1**	172.9 ± 43.1	**5.5 ± 2.1**

Backbone	Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
		(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
Internlm 7B	**Vanilla**	N/A	6.0 ± 2.0	N/A	N/A	N/A	N/A	N/A
	CoT	N/A	4.4 ± 1.9	N/A	N/A	N/A	N/A	N/A
	Funsearch	19.1 ± 1.3	0.8 ± 0.2	193.8 ± 30.9	318.3 ± 87.1	110.4 ± 19.3	150.0 ± 23.7	70.3 ± 10.9
	Eureka	29.5 ± 4.2	1.3 ± 0.4	211.0 ± 30.3	230.5 ± 39.9	211.1 ± 40.1	139.6 ± 38.2	50.8 ± 9.4
	OPRO	27.6 ± 3.8	0.2 ± 0.1	59.6 ± 16.2	130.8 ± 19.1	257.9 ± 93.0	195.1 ± 33.9	45.3 ± 9.7
	SGA	14.6 ± 5.5	**0.0 ± 0.0**	133.5 ± 29.3	191.4 ± 15.0	93.7 ± 25.4	94.8 ± 19.3	17.9 ± 4.5
	GSO (ours)	**10.0 ± 2.1**	**0.0 ± 0.0**	**10.4 ± 3.1**	**37.0 ± 12.9**	**47.2 ± 10.8**	**73.5 ± 19.6**	**4.2 ± 2.3**

Table 6: We consider an imaginary constitutive law setting to prevent the LLM from cheating by memorization and report the results using Mistral 7B as the representative backbone model.

Funsearch	Eureka	OPRO	SGA	GSO
201.1 ± 39.0	291.0 ± 20.8	190.0 ± 10.3	59.7 ± 5.9	**17.1 ± 4.0**

each hidden state variable. Specifically, when restoring hidden states from the original run, we substitute the values computed originally for the corresponding layer and token, allowing subsequent computations to proceed without further modification. Taking Llama3 8B as an example, as shown in Figure 4, we observe that the 18-22th layers for the last subject token demonstrate the most causality in the AIE metric. Therefore, we use the 18-22th MLP layers as the layers where model edits are applied to update knowledge by modifying the parameters. We also visualize other mentioned LLMs in Figures 5, 6, 7, 8, and 9 to provide a more comprehensive results.

C Prompt Templates for Each Task

We list the prompt templates for different tasks to offer more visually intuitive results for each task in Table 7, respectively. More detailed prompt information for the best performance of each task and dataset can be seen within the code.

D Details of Augmented Prompts

As mentioned in Section 5.5, we discussed the robustness of the GSO method compared to traditional prompt-based approaches when handling semantically similar prompts. Specifically, we manually generated five prompts and used OpenAI o1 to rewrite an additional five prompts based on the original task prompts. We now list these augmented prompts in Tables 8 and 9 to provide more insights

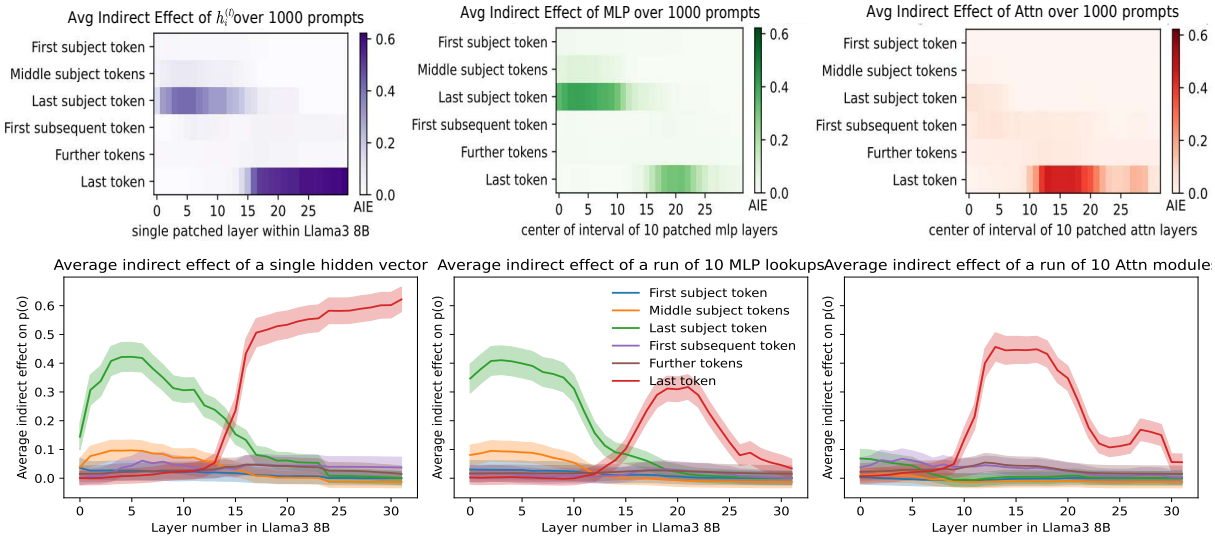

Figure 4: Causal tracing visualization results for Llama3 8B. The causal impact on output probability is mapped for (a) the effect of each hidden state on the prediction, (b) the effect of MLP activations alone, and (c) the effect of attention activations alone. We also give according to mean causal traces of over a sample of 1000 factual statements, shown as a line plot with 95% confidence intervals, which is below the first three figures. The confidence intervals confirm that the distinctions between peak and non-peak causal effects at both early and late sites are significant.

into our GSO.

E More Results of Different Edited Models

As mentioned in Section 5.3, we select GPT-J 6B, Llama3 8B, and Mistral 7B as representative models in Table 1. In this section, to further demonstrate the generalization and versatility of GSO, we also conducted experiments on several popular open-source LLMs at different scales, including Llama2 13B, Yi-9B, and Internlm 7B. As shown in Table 5, we can still observe that our GSO method significantly outperforms existing baselines. This further demonstrates the effectiveness of our GSO approach. We also present radar charts for each model to provide a more intuitive performance comparison in Figures 10 and 11. We apply a linear mapping to assign a score of 100 to values with a zero loss. Higher loss values correspond to progressively lower scores, with an inability to answer the question resulting in a score of zero. For formatting reasons, we also provide the radar chart of Llama3 8B. The effectiveness of our GSO across various popular open-source models further demonstrates its strong versatility and generalization capabilities.

F More Details of Experiment Setups and Task Definitions

We present more details of experiment setups and task definitions in this section.

Experiment Setups For the experimental setup, we apply several representative open-source LLMs: Llama3 8B[4], GPT-J-6B[5], Llama2 13B[6], Yi 9B[7], InternLM 7B[8], and Mistral 7B[9]. Other open-source models can also be replaced based on specific requirements. All experiments were conducted on a single Nvidia A100 GPU (80GB). Our approach is implemented using PyTorch 2.4.1[10] and Huggingface's Transformers 4.44.2[11]. For each task, we set a maximum of 100 iterations, with a limit of 2048 tokens generated per iteration. The optimization process terminates if no performance improvement or accurate result is observed after 10 consecutive iterations, or if an accurate result is reached earlier. More detailed configurations for the best perfor-

[4] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
[5] https://huggingface.co/EleutherAI/gpt-j-6b
[6] https://huggingface.co/meta-llama/Llama-2-13b-chat-hf
[7] https://huggingface.co/01-ai/Yi-1.5-9B-Chat
[8] https://huggingface.co/internlm/internlm2_5-7b-chat
[9] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3
[10] https://pytorch.org/
[11] https://github.com/huggingface/transformers

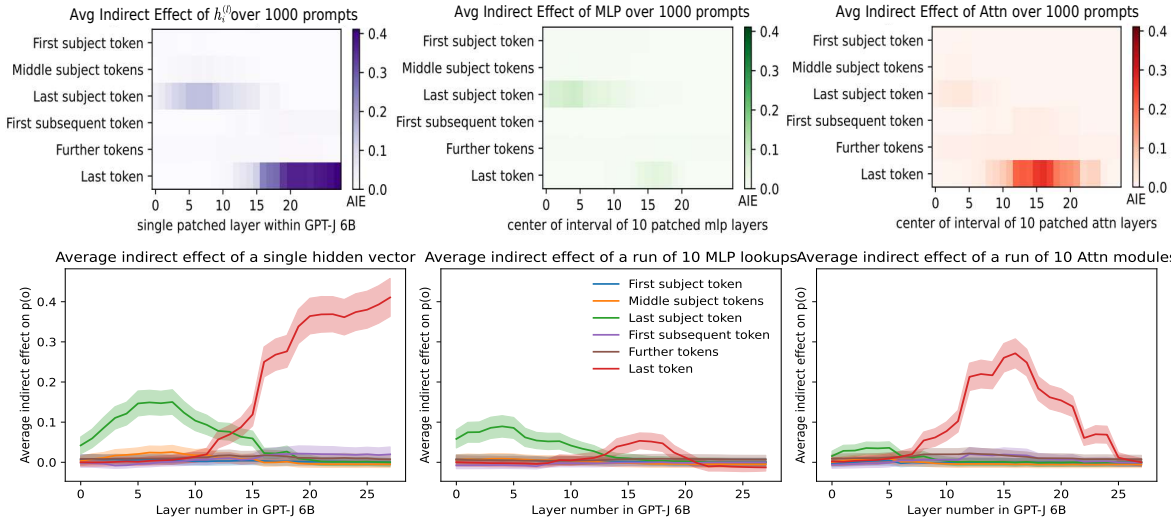

Figure 5: Causal tracing visualization results for GPT-J 6B. The causal impact on output probability is mapped for (a) the effect of each hidden state on the prediction, (b) the effect of MLP activations alone, and (c) the effect of attention activations alone. We also give according to mean causal traces of over a sample of 1000 factual statements, shown as a line plot with 95% confidence intervals, which is below the first three figures. The confidence intervals confirm that the distinctions between peak and non-peak causal effects at both early and late sites are significant.

mance of each task and dataset can be seen within our code.

For task definitions, we provide a more detailed of each task definition and setting.

Linear System Regression In linear system regression, the objective is to estimate the linear coefficients that most effectively model the relationship between input variables and their corresponding responses, within a probabilistic framework (Fisher, 1922). We follow the dataset setting as OPRO (Yang et al., 2023) which we include 437 samples. Linear system regression is a fundamental technique in statistics and machine learning, widely applied in both theoretical and practical settings (Montgomery et al., 2021; Seber and Lee, 2012).

Traveling Salesman Problem (TSP) The Traveling Salesman Problem (TSP) (Jünger et al., 1995; Gutin and Punnen, 2006) is a classical combinatorial optimization problem that has garnered significant attention in the literature, with numerous algorithms proposed, ranging from heuristic methods to exact solvers (Rosenkrantz et al., 1977; Golden et al., 1980; Gurobi Optimization, LLC, 2024; Helsgaun, 2017). Recently, the problem has also been approached through the use of deep neural networks (Kool et al., 2018; Deudon et al., 2018; Chen and Tian, 2019; Nazari et al., 2018), underscoring its adaptability to modern machine learning techniques. Formally, given a set of n nodes with known coordinates, the TSP seeks to determine the shortest possible route that visits each node exactly once and returns to the starting point.We follow the dataset setting as OPRO (Yang et al., 2023), which we include 177 TSP problems. At each optimization step, a maximum of 8 new solutions are generated. To evaluate the performance of various methods, we utilize the Gurobi solver (Gurobi Optimization, LLC, 2024) to generate oracle solutions and compute the optimality gap. The optimality gap is defined as the relative difference between the distance of the solution obtained by the tested approach and that of the oracle solution, normalized by the oracle solution's distance.

Constitutive Law Prediction Identifying constitutive laws from elastic material remains one of the most challenging tasks in fields such as physics, materials science, and mechanical engineering. In this paper, we follow the dataset settings as (Ma et al., 2023a; Ma et al.), including 357 different linear and non-linear materials and utilizing differentiable Material Point Method (MPM) simulators (Sulsky et al., 1995; Jiang et al., 2016). The primary objective of this task is to uncover both the discrete structure and continuous parameters of a constitutive law, specifically identifying the material models $\varphi(\cdot)$ along with their associated mate-

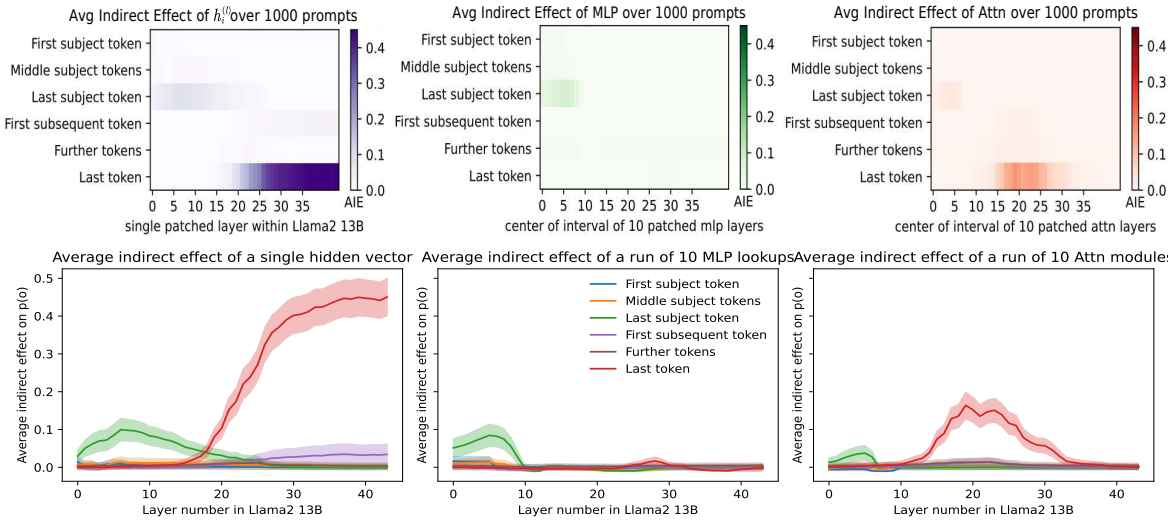

Figure 6: Causal tracing visualization results for Llama2 13B. The causal impact on output probability is mapped for (a) the effect of each hidden state on the prediction, (b) the effect of MLP activations alone, and (c) the effect of attention activations alone. We also give according to mean causal traces of over a sample of 1000 factual statements, shown as a line plot with 95% confidence intervals, which is below the first three figures. The confidence intervals confirm that the distinctions between peak and non-peak causal effects at both early and late sites are significant.

rial parameters θ, from a ground-truth trajectory of particle positions $\hat{X}_{t\in[1,\ldots,T]}$, where T represents the number of time steps. In this context, we consider two classes of constitutive laws: $\varphi_E(\cdot;\theta_E)$ for modeling elastic materials and $\varphi_P(\cdot;\theta_P)$ for modeling plastic materials, both of which contain both linear and non-linear materials and are formally defined as the following:

$$\varphi_E\left(\mathbf{F};\theta_E\right) \mapsto \boldsymbol{\tau}$$
$$\varphi_P\left(\mathbf{F};\theta_P\right) \mapsto \mathbf{F}^{\text{corrected}},$$

where $\mathbf{F} \in \mathbb{R}^{3\times3}$ is the deformation gradient, $\tau \in \mathbb{R}^{3\times3}$ is the Kirchhoff stress tensor, $\mathbf{F}^{\text{corrected}} \in \mathbb{R}^{3\times3}$ is the deformation gradient after elastic return-mapping correction, and θ_E and θ_P are the continuous material parameters for elastic and elastic constitutive laws respectively. Given a specific constitutive law, we input it to the differentiable simulation and yields a particle position trajectory:

$$X_{t\in[1,\ldots,T]} = \text{sim}\left(\varphi\left(\cdot;\theta\right)\right),$$

and we optimize the constitutive law by fitting the output trajectory to the ground truth $\hat{X}_{t\in[1,\ldots,T]}$.

Molecule Property Prediction Predicting the Highest Occupied Molecular Orbital (HOMO), Lowest Unoccupied Molecular Orbital (LUMO), and HOMO-LUMO gap value of molecules is a key challenge in computational chemistry, particularly in the fields of quantum chemistry, material science, and drug design. In this task, we aim to predict the HOMO value based on molecular descriptors, such as molecular weight, Topological Polar Surface Area (TPSA), Octanol-water partition coefficient (logP), etc (maximum to 2k different properties). Although traditional quantum mechanical methods such as Density Functional Theory (DFT) are widely used for HOMO calculations, their computational cost can be prohibitive for large datasets. In this work, we adopt a machine learning approach to predict the HOMO value using these physical-chemical properties as input features. We follow the dataset settings as (Ma et al., 2023a; Ma et al.), including 238 different molecules.

The primary goal of this task is to construct a model that accurately predicts the HOMO energy level ϵ_{HOMO} by learning a mapping $\psi(\cdot)$ between the input molecular properties $x \in \mathbb{R}^n$, such as molecular weight, TPSA, and logP, and the HOMO energy $\epsilon_{\text{HOMO}} \in \mathbb{R}$. The machine learning model is trained on a dataset containing known molecular properties and their corresponding HOMO values, and optimized to minimize the prediction error relative to the true HOMO values $\hat{\epsilon}_{\text{HOMO}}$.

Given a set of molecular descriptors, the model outputs the predicted HOMO value:

$$\epsilon_{\text{HOMO}} = \psi(x),$$

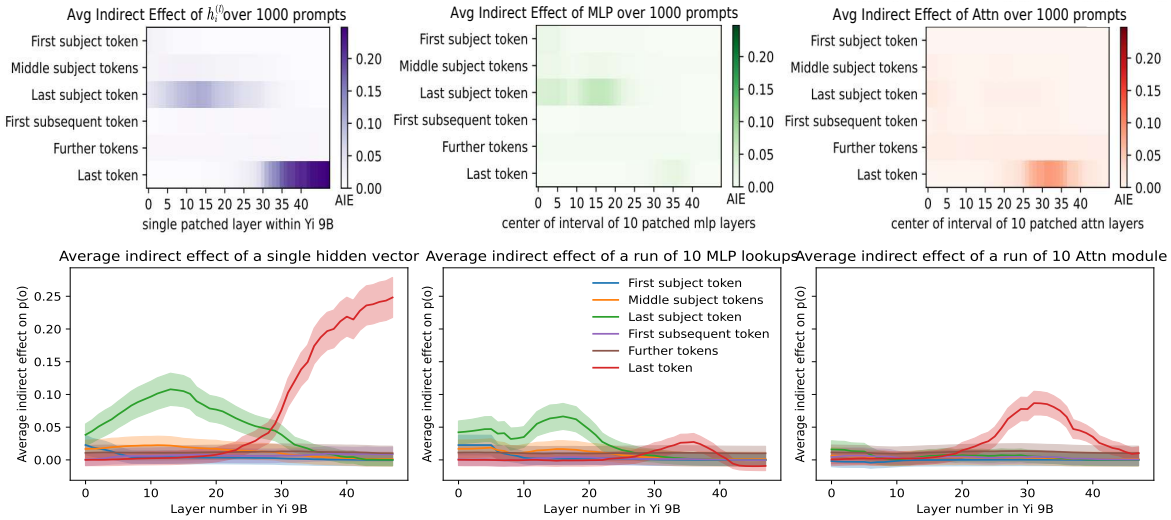

Figure 7: Causal tracing visualization results for Yi 9B. The causal impact on output probability is mapped for (a) the effect of each hidden state on the prediction, (b) the effect of MLP activations alone, and (c) the effect of attention activations alone. We also give according to mean causal traces of over a sample of 1000 factual statements, shown as a line plot with 95% confidence intervals, which is below the first three figures. The confidence intervals confirm that the distinctions between peak and non-peak causal effects at both early and late sites are significant.

and the objective is to minimize the loss function, defined as the difference between the predicted and true HOMO values:

$$L = \|\psi(x) - \hat{\epsilon}_{\text{HOMO}}\|^2 .$$

This approach provides a computationally efficient alternative to traditional quantum mechanical methods, offering the potential for high-throughput screening of molecular libraries for their electronic properties, including HOMO values.

G More Results of Ablation Study

In Section 5.4, we report the results of the ablation study using Llama3 8B as the backbone model. In this section, we will further present the results using GPT-J-6B, Llama2 13B, Yi 9B, InternLM 7B, and Mistral 7B as backbone models to obtain more insights into the individual components constituting GSO across various backbone models. As illustrated in Tables 11, 12, 13, 14, and 15, we still observe that the absence of each component within GSO leads to a decline in performance across diverse domains for almost all applied backbone models in the seven tested scientific optimization tasks, which further demonstrates that GSO organically integrates the two-level optimization into a unified framework as well. We also observe that the absence of model editing leads to a more significant decline in accuracy, which further demonstrates the importance of effectively utilizing observational feedback to adaptively adjust the optimization direction.

H More Case Study on Generalization or Memorization

To investigate whether the improvements brought by our method are solely due to the LLM having seen solutions during the training phase, we designed an experiment aimed at eliminating this factor by having the model invent an imaginary constitutive law that does not exist on Earth. We combined the von Mises plasticity constitutive law, granular material, and weakly compressible fluid in proportions of 50%, 30%, and 20%, respectively, creating a new constitutive law that represents an exceedingly complex hypothetical material. As shown in Table 6, our method was still able to discover the constitutive law with minimal quantitative loss compared to other existing baselines. These results also indicate that GSO does not simply rely on memorization to achieve results but instead indeed effectively performs optimization.

I Inference Time Comparisons

We note that GSO requires an additional process to utilize the optimization feedback by applying model editing techniques to update LLM's parameters. Compared to the vanilla model, this process

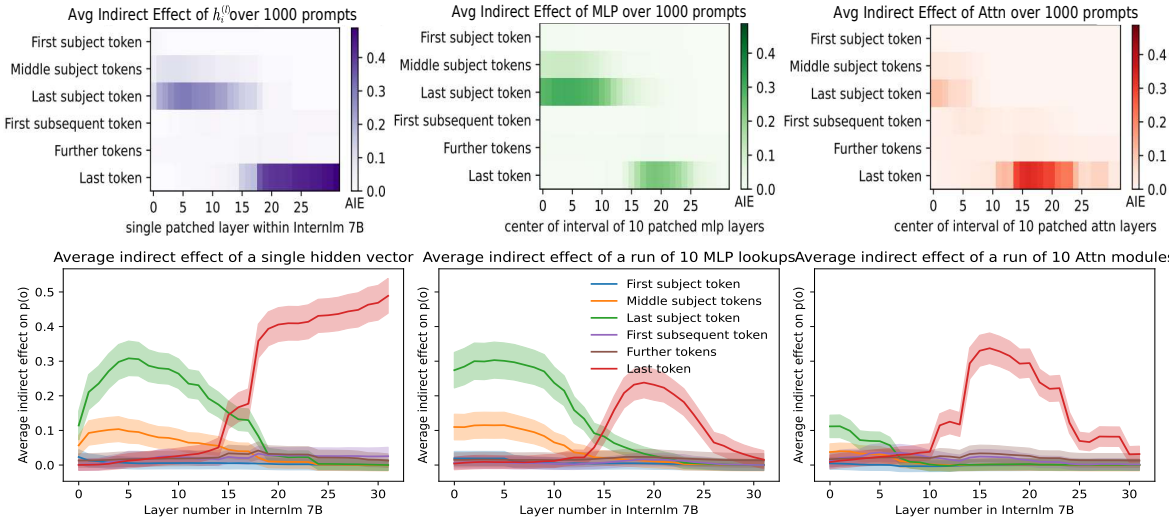

Figure 8: Causal tracing visualization results for Internlm 7B. The causal impact on output probability is mapped for (a) the effect of each hidden state on the prediction, (b) the effect of MLP activations alone, and (c) the effect of attention activations alone. We also give according to mean causal traces of over a sample of 1000 factual statements, shown as a line plot with 95% confidence intervals, which is below the first three figures. The confidence intervals confirm that the distinctions between peak and non-peak causal effects at both early and late sites are significant.

could potentially introduce extra inference time. However, we also observe that GSO effectively reduces the input prompt length through this approach, thereby decreasing the LLM's inference time. Hence, to explore the interplay between these two effects, we record and compare the average inference time per sample of different methods using a single Nvidia A100 GPU (80GB), including vanilla, CoT, OPRO, Eureka, Funsearch, SGA, and GSO on each mentioned scientific optimization task to explore the influence of additional inference time and provide more insight of our GSO.

We report the average inference time per sample as a metric, as shown in Table 10. We observe from the table that although the incorporation of model editing within the GSO framework introduces just marginal additional inference time costs. On average, the increase in inference time cost per sample due to the introduction of GSO, compared to the vanilla LLM, is 7.75s. Notably, when utilizing Mistral 7B as the backbone model, this additional inference time is only 5.8s, yet it offers a maximum precision improvement of 32.6× in the HOMO-LUMO gap prediction task compared to all other methods. Furthermore, GSO exhibits higher inference efficiency compared to existing methods. We may focus on exploring ways to further reduce the time cost of the GSO, including lightweighting LLMs to get a faster calculation (Zhu et al., 2023; Hsieh et al., 2023) or adopting more efficient (Wang et al., 2023c; Zhao et al., 2024; Frantar and Alistarh, 2023) and rational large model inference strategies such as speculative decoding (Leviathan et al., 2023; Liu et al., 2024a) as for future works.

J Scientific Artifacts

The data we collect in specialized domains is publicly available and viewable online. The data owners have indicated that the data can be used for scientific research or have not indicated that the data cannot be used for scientific research, and our collection process is also in compliance with regulations. Moreover, there is no unique identification of individuals or offensive content in these data.

K More Discussions On GSO

K.1 What is the key advantage of using LLMs to optimize over some traditional optimization algorithms, especially on classical optimization problems?

The key advantage is that one can use natural language to describe the optimization problem. Instead of formally defining the optimization problem and deriving the update step with a programmed solver. This makes optimization more accessible to general users who may not have extensive domain knowledge of the specific types of optimization tasks in question, and it may also enhance

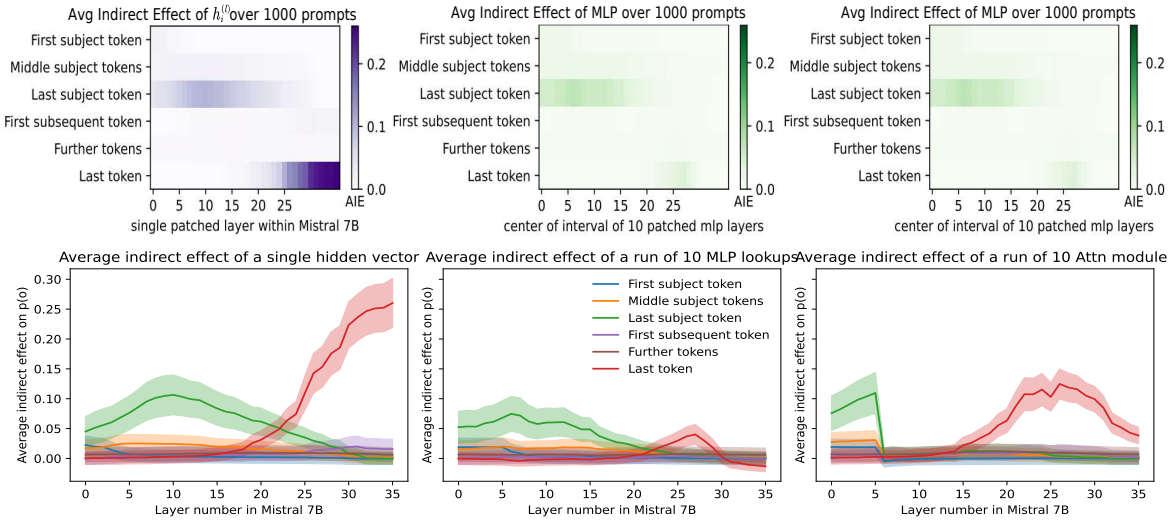

Figure 9: Causal tracing visualization results for mistral 7B. The causal impact on output probability is mapped for (a) the effect of each hidden state on the prediction, (b) the effect of MLP activations alone, and (c) the effect of attention activations alone. We also give according to mean causal traces of over a sample of 1000 factual statements, shown as a line plot with 95% confidence intervals, which is below the first three figures. The confidence intervals confirm that the distinctions between peak and non-peak causal effects at both early and late sites are significant.

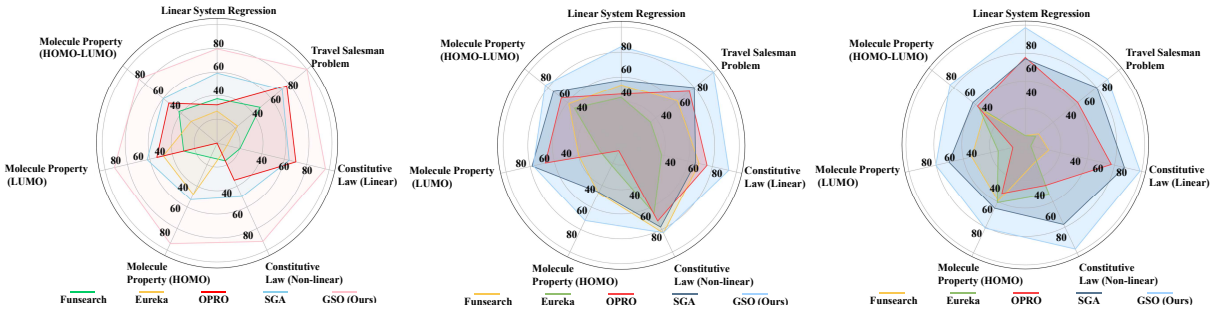

Figure 10: GSO achieves state-of-the-art performance on a broad range of scientific optimization tasks compared with existing methods, using LLama 3 7B, GPT-J 6B, and Llama2 13B as backbone models, respectively. We linearly map the evaluation metrics to [0, 100] for presentation.

the productivity of optimization experts who work on these tasks daily. Moreover, employing LLMs for optimization is both domain-agnostic and task-agnostic, offering excellent universality and generalization. It enables rapid understanding and optimization of relatively unknown problems and can assist in finding more refined solutions. With the rapid advancement of LLMs, their abilities are improving swiftly, indicating the great potential of LLM-based optimization methods to address more complex and even unknown optimization tasks.

K.2 What specific impact does the adoption of the exploitation/exploration phase actually have, given that sometimes satisfactory results can be achieved without introducing it?

To investigate how the dynamic exploitation/exploration strategy works, we designed an ablation experiment in Section 5.4 and Appendix G. Statistically, the strategy has a relatively positive effect, especially for tasks like predicting the HOMO-LUMO gap of molecules. We find that: **Without exploitation**, GSO sometimes finds it difficult to effectively provide consistent high-quality parameter hypotheses. **Without exploration**, GSO sometimes becomes trapped in local optima, unable to further optimize the task to gain additional insights. Consequently, using a balanced exploitation/explo-

Table 7: Prompt templates for each task. In the prompts, "str(Examples)" represents initial solutions for each task. "str(Nodes)" and "str(Properties)" represent some basic properties of the materials/molecules.

Optimization Tasks	**Prompt Templates**
Linear System Regression	You will help me minimize a function with two input variables w, b. I have some (w, b) pairs and the function values at those points. The pairs are arranged in descending order based on their function values, where lower values are better. Below are some examples: str(Examples) Give me a new (w, b) pair that is different from all pairs above, and has a function value lower than any of the examples.
Travel Salesman Problem	You are given a list of points with coordinates below: str(Nodes) Below are some previous traces and their lengths. The traces are arranged in descending order based on their lengths, where lower values are better: str(Traces) Give me a new trace that is different from all traces above, and has a length lower than any of the above. The trace should traverse all points exactly once. The trace should start with <trace> and end with </trace>.
Constitutive Law Prediction (Linear and Non-Linear)	You are given a list of materials with corresponding properties below: str(Properties) Below are some previous stress-strain responses for each material, sorted in descending order based on compliance, where lower compliance values indicate better structural integrity: str(Examples) Provide a new constitutive law that differs from all the laws above, ensuring that it produces a compliance value lower than any of the existing responses. The constitutive law should describe the relationship between stress and strain accurately for the material and should start with <law> and end with </law>.
Molecule Property Prediction (HOMO value, LUMO value, and HOMO-LUMO gap)	You are given a list of molecules with their chemical properties below: str(Properties). Your goal is to predict the HOMO (can be adjusted to LUMO or HOMO-LUMO gap depending on different tasks) values for the listed molecules. Below are some examples: str(Examples) The HOMO value for each molecule should be distinct from any previously reported values, and the predictions should ensure an accurate representation of their electronic properties. Each predicted value should be formatted to start with <value> and end with </value>

ration strategy enables one to escape local optima and obtain consistent loss decay. As shown in Figure 3, GSO with this strategy is the only one that continues to progress toward better results, whereas others exhibit plateaued stagnation curves.

K.3 Is GSO still effective when dealing with high-dimensional data or extremely intricate optimization tasks?

Our experimental design includes both simple, low-dimensional ideal, and high-dimensional complex scientific optimization problems aimed at minimizing the sim-to-real gap to the greatest extent possible. This ensures that GSO generates meaningful and practical optimizations rather than questionable toys. We believe this is a strength rather than a weakness for GSO. Moreover, the success of GSO stems from the effective utilization of optimization feedback at each step and a dynamic exploration/-exploitation strategy, which is domain-agnostic and can be applied to other domains. We also acknowledge that when the underlying physics of the opti-

Table 8: Augmented prompt templates for linear system regression and travel salesman problem. In the prompts, "str(Examples)" represents initial solutions for each task. "str(Nodes)" represents some basic properties of the materials.

Optimization Tasks	**Augmented Prompts**
Linear System Regression	(i) You will help me minimize a function with two input variables w, b. I have some (w, b) pairs and the function values at those points. The pairs are arranged in descending order based on their function values, where lower values are better. Please provide a new (w, b) pair distinct from those above, with a lower function value than any previous pair. (ii) You are assisting in minimizing a function with two variables w and b. Provided are some (w, b) pairs along with their function values, sorted in descending order where lower values are better. Generate a (w, b) pair not seen in the above list, ensuring it yields a function value lower than any listed. (iii) Help me find the minimum of a function dependent on variables w and b. Below are (w, b) pairs and their corresponding function values, arranged from highest to lowest (lower is better). Give a (w, b) pair not seen in the above list, ensuring it yields a function value lower than any listed. (iv) I need assistance in minimizing a function with inputs w and b. Here are some (w, b) pairs and their function values, listed in descending order of their function values (lower values indicate better results). Give me a better (w, b) pair that is not included above. (v) Your task is to help minimize a function of two variables w and b. The following are (w, b) pairs and their function values, sorted from highest to lowest (lower values are preferable). Provide a new and better (w, b) pair from those above.
Travel Salesman Problem	(i) Given the coordinates of points: str(Nodes). Here are some previous routes and their lengths, sorted from longest to shortest (shorter is better). Give me a new trace that is different from all traces above, and has a length lower than any of the above. The trace should traverse all points exactly once. The trace should start with <trace> and end with </trace>. (ii) Consider the following points with coordinates: str(Nodes). Generate a new trace that differs from all previous traces and is shorter than any of them. Ensure that this trace covers each point exactly once, begins with <trace>, and ends with </trace>. (iii) You have a set of points at these coordinates below: str(Nodes). Please provide a unique trace that is distinct from all preceding traces and has a length shorter than any of the prior traces. This trace should start with <trace> and conclude with </trace>, while visiting each point one time. (iv) Here are points with their coordinates: str(Nodes). Please craft a new trace that is unique from all previous traces and shorter in length than any listed above. This trace should traverse all points a single time, beginning with <trace> and ending with </trace>. (v) You are provided with points with their coordinates: str(Nodes). Produce a trace that does not resemble any of the existing traces and has a length less than the shortest one listed. It should begin with <trace>, finish with </trace>, and cover each point exactly once.

Table 9: Augmented prompt templates for constitutive law prediction and molecule property prediction. In the prompts, "str(Examples)" represents initial solutions for each task. "str(Properties)" represent some basic properties of the molecules.

Optimization Tasks	**Augmented Prompts**
Constitutive Law Prediction (Linear and Non-Linear)	(i) You have been provided with a list of materials and their associated properties: str(Properties). You will find previous stress-strain responses for each material, where lower compliance indicates stronger structural integrity: str(Examples) Create a new constitutive law that is distinct from all those listed. The law should begin with <law> and end with </law>. (ii) Provided below are some materials with their properties: str(Properties). It also includes prior stress-strain responses for each material, where lower values reflect better structural integrity: str(Examples). Develop a new constitutive law that is different from all previous laws. The law should start with <law> and end with </law>. (iii) You have a list of materials and their properties outlined below: str(Properties). Alongside, there are previous stress-strain responses for each material with lower compliance suggesting stronger structural integrity: str(Examples). Formulate a new constitutive law. This law should start with <law> and end with </law>. (iv) Below is a list of materials and their corresponding properties: str(Properties) The data is in descending order, where a lower compliance value indicates greater structural integrity: str(Examples). Construct a new constitutive law that is distinct from all the above, ensuring that it yields a compliance value below all existing responses. The law should start with <law> and end with </law>. (v) A list of materials and their properties is given below: str(Properties). Below are previous responses, where lower values suggest better structural integrity: str(Examples). Design a new constitutive law. The law should begin with <law> and end with </law>.
Molecule Property Prediction (HOMO value, LUMO value, and HOMO-LUMO gap)	(i) Provided below is a list of molecules and their properties: str(Properties). Predict the HOMO values for these molecules (or adjust to LUMO/HOMO-LUMO gap as needed). Ensure each HOMO value accurately represents electronic properties. Format predictions with <value> at the start and </value> at the end. (ii) Below are molecules with their chemical properties: str(Properties). Your task is to predict HOMO values (or LUMO/HOMO-LUMO gap) for each. Make sure predictions reflect electronic properties accurately, formatted as <value>...</value>. (iii) You have a list of molecules with properties: str(Properties). Predict the HOMO (or LUMO/HOMO-LUMO gap) values. Use <value>...</value> to format predictions. (iv) Here are molecules and properties: str(Properties). Predict HOMO values (or LUMO/HOMO-LUMO gap) that capture properties accurately. Prediction should begin with <value> and end with </value>. (v) Provided is a list of molecules: str(Properties). Predict HOMO values (or LUMO/HOMO-LUMO gap), ensuring accurate electronic property representation. Format each as <value>...</value>.

Table 10: The inference time (per sample on average) on the TSP task for the baseline methods, including Vanilla, CoT, OPRO, Eureka, Funsearch, SGA, GSO, respectively. We report the results using Llama3 8B, GPT-J 6B, Llama2 13B, Yi9B, Internlm 7B, and Mistral 7B as backbone models, respectively. (Unit: seconds)

Backbone Models	Vanilla	CoT	OPRO	Eureka	Funsearch	SGA	GSO (ours)
Llama3 8B	21.9	21.9	34.0	32.1	40.9	37.3	28.9
GPT-J 6B	20.1	21.3	30.8	27.5	44.5	35.8	29.0
Llama2 13B	40.2	41.4	53.3	47.6	61.0	55.3	51.0
Yi 9B	28.8	28.9	38.8	41.0	53.3	50.5	35.9
Internlm 7B	16.9	17.4	25.4	21.0	36.9	37.0	23.8
Mistral 7B	18.2	18.5	27.2	20.4	33.9	37.3	24.0

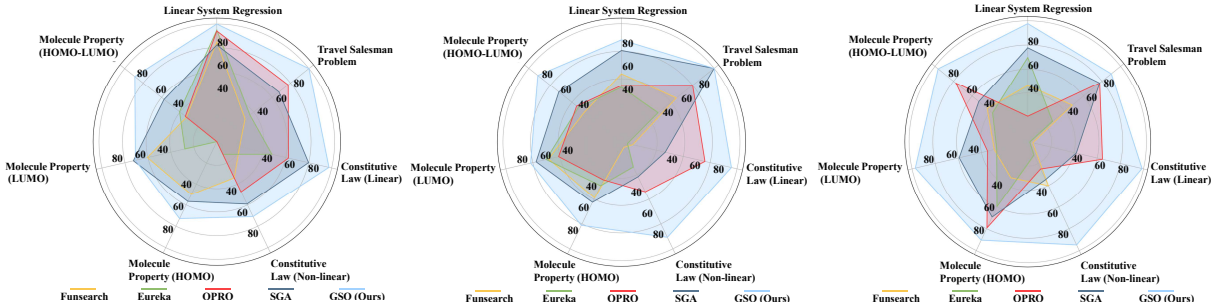

Figure 11: GSO achieves state-of-the-art performance on a broad range of scientific optimization tasks compared with existing methods, using Yi 9B, Internlm 7B, and Mistral 7B as backbone models, respectively. We linearly map the evaluation metrics to [0, 100] for presentation.

mization problem is too complex, this represents a limitation of our approach. A potential extension could focus on data compression by pre-training an auto-encoder to project high-dimensional data into a latent space.

K.4 What is the difference between updating LLM parameters using model editing and employing methods like fine-tuning?

The differences are two-folds:**(i)** the key difference lies in their **scope and flexibility**. Fine-tuning typically involves adjusting a large portion of the model's parameters over a dataset, often requiring extensive computational resources and time (Hu et al., 2021; Zhu et al., 2020), and it can result in overfitting or catastrophic forgetting of prior knowledge (Luo et al., 2023). Model editing is a more targeted approach, allowing for precise modifications to specific parts of the model in response to new information or tasks without altering the broader knowledge encoded within the model. (ii) Another difference is its **efficiency**; it enables rapid updates to the model without the need for extensive retraining. This makes it particularly suitable for dynamic optimization tasks where quick adjustments are necessary. Moreover, model editing preserves the generalization abilities of the LLM while fine-tuning may risk degrading its performance on unrelated tasks. In this sense, model editing offers a more controlled and adaptive method for refining LLM behavior, especially in domain-agnostic and task-agnostic scenarios, making it an ideal tool for iterative optimization processes.

Table 11: The results of the ablation study of our GSO on the seven scientific optimization tasks, using GPT-J 6B as the backbone model. The symbol N/A indicates that the model is unable to provide a feasible solution for the current task.

Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
	(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
$\text{GSO}_{w/o\ edit}$	N/A	0.5 ± 0.1	37.1 ± 10.0	185.3 ± 36.0	180.2 ± 19.1	497.1 ± 98.3	25.1 ± 4.1
$\text{GSO}_{w/o\ dynamic}$	15.1 ± 3.2	0.0 ± 0.1	19.9 ± 6.1	131.7 ± 26.0	120.9 ± 64.4	135.9 ± 27.1	15.4 ± 4.1
GSO (ours)	**12.3 ± 4.8**	**0.0 ± 0.0**	**15.7 ± 5.0**	59.9 ± 10.3	**70.1 ± 17.7**	**95.5 ± 10.0**	**8.5 ± 2.0**

Table 12: The results of the ablation study of our GSO on the seven scientific optimization tasks, using Llama2 13B as the backbone model.

Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
	(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
$\text{GSO}_{w/o\ edit}$	22.3 ± 3.9	0.2 ± 0.0	73.3 ± 16.0	113.3 ± 20.1	540.2 ± 38.3	613.1 ± 49.9	332.1 ± 61.2
$\text{GSO}_{w/o\ dynamic}$	13.7 ± 2.9	**0.1 ± 0.1**	6.3 ± 2.0	20.5 ± 4.0	**35.3 ± 5.0**	87.6 ± 14.1	19.3 ± 6.0
GSO (ours)	**5.0 ± 1.6**	**0.1 ± 0.1**	**2.9 ± 1.3**	**7.4 ± 3.1**	48.1 ± 12.0	**79.1 ± 13.3**	**8.3 ± 3.2**

Table 13: The results of the ablation study of our GSO on the seven scientific optimization tasks, using Yi 9B as the backbone model.

Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
	(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
$\text{GSO}_{w/o\ edit}$	9.0 ± 1.5	0.4 ± 0.1	55.4 ± 10.1	408.1 ± 39.0	821.0 ± 112.0	982.1 ± 101.5	155.0 ± 30.1
$\text{GSO}_{w/o\ dynamic}$	5.5 ± 1.7	**0.0 ± 0.0**	19.1 ± 3.0	**80.9 ± 5.0**	73.0 ± 5.3	194.0 ± 20.1	9.0 ± 2.3
GSO (ours)	**3.0 ± 0.8**	**0.0 ± 0.0**	**5.9 ± 2.1**	89.1 ± 33.9	**67.9 ± 23.1**	**172.9 ± 43.1**	**5.5 ± 2.1**

Table 14: The results of the ablation study of our GSO on the seven scientific optimization tasks, using Internlm 7B as the backbone model.

Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
	(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
$\text{GSO}_{w/o\ edit}$	12.0 ± 3.1	0.3 ± 0.1	65.1 ± 10.5	420 ± 82.1	490.0 ± 55.0	782.5 ± 133.2	150.2 ± 10.5
$\text{GSO}_{w/o\ dynamic}$	**9.0 ± 1.1**	0.1 ± 0.0	17.5 ± 4.3	79.0 ± 13.1	78.5 ± 20.0	159.3 ± 23.0	19.3 ± 2.5
GSO (ours)	10.0 ± 2.1	**0.0 ± 0.0**	**10.4 ± 3.1**	**37.0 ± 12.9**	**47.2 ± 10.8**	**73.5 ± 19.6**	**4.2 ± 2.3**

Table 15: The results of the ablation study of our GSO on the seven scientific optimization tasks, using mistral 7B as the backbone model.

Method	Linear System	Travel Salesman	Constitutive Law		Molecule Property		
	(a) ↓	(b) ↓	(c) ↓	(d) ↓	(e) ↓	(f) ↓	(g) ↓
$\text{GSO}_{w/o\ edit}$	13.0 ± 2.5	0.3 ± 0.1	55.4 ± 13.5	31.7 ± 10.4	151.1 ± 20.0	45.4 ± 10.1	115.3 ± 20.1
$\text{GSO}_{w/o\ dynamic}$	9.4 ± 2.0	**0.0 ± 0.0**	14.1 ± 3.0	16.0 ± 3.0	47.3 ± 6.8	10.5 ± 1.9	16.8 ± 4.0
GSO (ours)	**5.6 ± 3.0**	0.1 ± 0.0	**3.2 ± 1.7**	**13.0 ± 2.1**	**23.9 ± 5.1**	**2.8 ± 1.5**	**1.7 ± 0.4**