OptLLM: Optimal Assignment of Queries to Large Language Models

Yueyue Liu[†], Hongyu Zhang[‡*], Yuantian Miao[†], Van-Hoang Le[†], Zhiqiang Li[§]

[†]School of Information and Physical Sciences, The University of Newcastle, Newcastle, Australia
[‡]School of Big Data and Software Engineering, Chongqing University, Chongqing, China
[§]School of Computer Science, Shaanxi Normal University, Shaanxi, China

yueyue.liu@uon.edu.au, hyzhang@cqu.edu.cn, sky.miao@newcastle.edu.au, vanhoang.le@uon.edu.au, lizq@snnu.edu.cn

Abstract—Large Language Models (LLMs) have garnered considerable attention owing to their remarkable capabilities, leading to an increasing number of companies offering LLMs as services. Different LLMs achieve different performance at different costs. A challenge for users lies in choosing the LLMs that best fit their needs, balancing cost and performance. In this paper, we propose a framework for addressing the cost-effective query allocation problem for LLMs. Given a set of input queries and candidate LLMs, our framework, named OptLLM, provides users with a range of optimal solutions to choose from, aligning with their budget constraints and performance preferences, including options for maximizing accuracy and minimizing cost. OptLLM predicts the performance of candidate LLMs on each query using a multi-label classification model with uncertainty estimation and then iteratively generates a set of non-dominated solutions by destructing and reconstructing the current solution. To evaluate the effectiveness of OptLLM, we conduct extensive experiments on various types of tasks, including text classification, question answering, sentiment analysis, reasoning, and log parsing. Our experimental results demonstrate that OptLLM substantially reduces costs by 2.40% to 49.18% while achieving the same accuracy as the best LLM. Compared to other multi-objective optimization algorithms, OptLLM improves accuracy by 2.94% to 69.05% at the same cost or saves costs by 8.79% and 95.87% while maintaining the highest attainable accuracy.

Index Terms—Large Language Models, Query Assignment, Multi-objective Optimization, Performance Prediction, Cost-performance Tradeoff

I. INTRODUCTION

Large Language Models (LLMs) are increasingly influential due to their ability to analyze and generate natural and programming languages. These models are trained on extensive text data [1] and have demonstrated impressive abilities to simulate human linguistic capabilities, posing a significant impact across multiple domains such as natural language processing (NLP) [2]–[4] and programming tasks [5]–[7].

Many companies (such as OpenAI[1], AI21[2], TogetherAI[3],etc.) now offer LLMs as services through public Application Programming Interfaces (APIs). These LLM services have different performance and pricing structures. Despite the remarkable achievements and rapid progress of LLMs across various fields, their adoption in practice is challenging.

---

*Corresponding author.

[1]https://openai.com
[2]https://www.ai21.com/
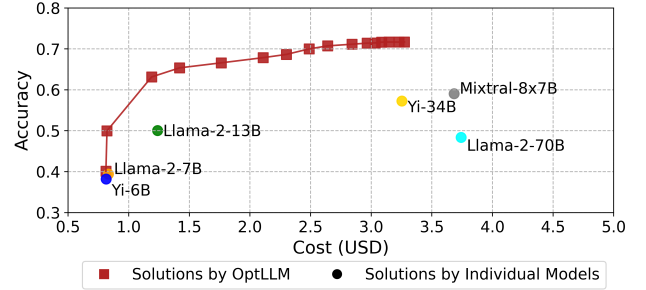[3]https://www.together.ai/

Fig. 1. Comparison of using each individual LLM with solutions by OptLLM. OptLLM can provide solutions with higher accuracy and lower cost than the best individual LLM (Mixtral 8x7B) on LLM-based log parsing task

First, utilizing LLMs can lead to considerable costs due to the large number of LLM queries required by a task [8]. For instance, using GPT-3 for a simplified customer service system could potentially cost a small business over $14,400 per month [9]. Second, the performance of an LLM, measured by the percentage of correctly processed queries (referred to as accuracy in this paper), may not be consistent across all tasks. Although expansive models such as GPT-4 are expected to wield greater computational strength and deliver superior outcomes, they incur higher costs and do not universally assure satisfactory results for all types of tasks. Small LLMs offer economical alternatives for executing less complex tasks, especially in scenarios where the extensive capabilities of larger models are superfluous [8]. Moreover, a task that may not be successfully completed by one LLM could potentially be effectively handled by another [10].

Given these challenges, it is imperative to make informed decisions about selecting an appropriate model for each query when leveraging LLMs in order to achieve a tradeoff between cost and performance expectations. Our OptLLM addresses this issue by predicting the performance of LLMs and optimizing the cost and accuracy effectively. Figure 1 demonstrates the superiority of OptLLM in achieving multi-objective performance, compared with the results obtained by using six LLMs individually on a log parsing task (the task of parsing raw log messages into templates). Specifically, an LLM named Mixtral 8x7B [11] achieves the highest accuracy of 59.05% at a cost of $3.68, whereas OptLLM provides

solutions that can reach a higher accuracy of 71.69% at a lower cost of $3.27. This highlights the potential of OptLLM to optimize the selection of LLMs for each query, resulting in improved performance and reduced cost.

This paper considers the assignment of queries to an appropriate LLM, as a multi-objective optimization problem with the objectives of minimizing cost and maximizing performance. Evolutionary algorithms (EAs), commonly employed in optimization problems like scheduling, planning, design, and management [12], [13], offer a potential approach to address the query assignment for LLMs. However, EAs struggle with efficiency and may converge to suboptimal solutions, especially when tackling large-scale, intricate problems with limited available information, leading to prolonged computation times and difficulties in scaling [14], [15].

To address the above challenges, this paper introduces OptLLM, an efficient and effective framework that selects a suitable LLM for a given set of queries in a predictable manner. OptLLM consists of two components: prediction and optimization. The prediction component employs multi-label classification to predict the possibility of candidate LLMs processing each query successfully. To handle prediction uncertainty, OptLLM uses a weighted mean to aggregate bootstrap sample predictions and calculates the standard deviation across samples to quantify the uncertainty. The optimization component starts with two extreme solutions: one with the highest predicted accuracy and the other with the lowest cost. OptLLM then iteratively generates non-dominated solutions through a destruction and reconstruction process. The destruction phase temporarily prioritizes one objective, while the reconstruction phase optimizes the solution based on heuristic rules. This optimization procedure enables OptLLM to address large-scale, intricate challenges with increased computational speed.

We demonstrate the generality by experimenting with various general NLP tasks and domain-specific tasks. The experimental results show that the solutions provided by OptLLM can reduce cost and improve accuracy simultaneously in all tasks. OptLLM achieves the same accuracy level as the best individual LLM while reducing costs by 2.40% to 49.18%. Compared to other multi-objective optimization algorithms, OptLLM offers solutions with accuracy improvements ranging from 2.94% to 69.05%, or cost savings between 8.79% and 95.87% while maintaining the highest accuracy attainable by each baseline algorithm.

Our major contributions are summarized as follows:
1) We treat the problem of assigning queries to LLMs as a multi-objective optimization problem, aiming to achieve a tradeoff between cost and accuracy.
2) We propose OptLLM, an effective and efficient framework that automatically assigns queries to suitable LLMs.
3) We demonstrate the generality and effectiveness of OptLLM through extensive experiments on various tasks. The results confirm that the solutions provided by OptLLM can reduce cost and improve accuracy across a wide range of tasks.
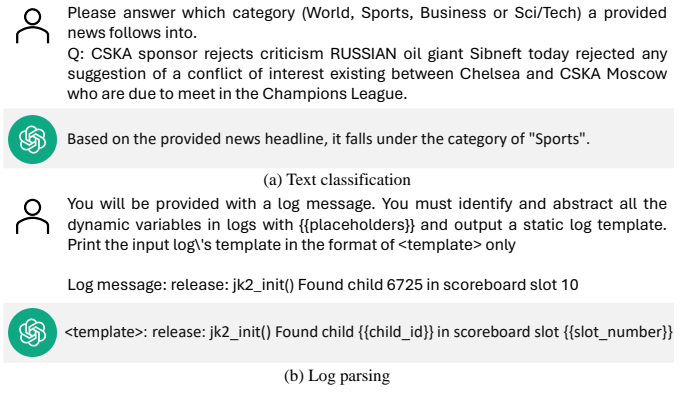

(a) Text classification

(b) Log parsing

Fig. 2. Examples of log parsing and text classification queries

II. BACKGROUND

A. Large Language Model and Its Application

In recent years, Large Language Models (LLMs) have emerged as powerful tools with remarkable capabilities in understanding and generating natural language text. Many LLMs, such as ChatGPT4 and Llama [16], [17], have been developed to tackle various NLP tasks, such as text classification, question answering, sentiment analysis, and reasoning. For example, text classification involves assigning a predefined category to a given piece of text, as exemplified in Figure 2-(a), where the task is to classify a news headline into one of four categories: World, Sports, Business, or Sci/Tech.

In addition to NLP tasks, LLMs are also applied to domain-specific tasks. In this paper, we focus on LLM-based log parsing [18] as a case study. Log parsing aims to extract structured information from unstructured log data generated by software systems. Log messages are system-generated lines of text containing event information. They often include dynamic variables that vary across instances of the same event type. Log parsing aims to normalize these variables by replacing them with placeholders, creating a static template that represents the common structure and content of log messages for a given event type. Figure 2-(b) illustrates the LLM-based log parsing process. By providing an instruction such as "abstract variables with placeholders to extract the correct template", we can guide the LLM to extract a template from a log message. In the example, the log message "release: jk2_init() Found child 6725 in scoreboard slot 10" is parsed into the template "release: jk2_init() Found child {{child_id}} in scoreboard slot {{slot_number}}", where the dynamic variables "6725" and "10" are replaced with the placeholders "child_id" and "slot_number", respectively.

B. Multi-objective Optimization

The multi-objective optimization problem (MOP) is a field of optimization concerned with problems involving multiple conflicting objectives that must be optimized simultaneously [19]. MOP has found wide-ranging applications across various domains, including scheduling, resource allocation,

and planning [20]. The goal is to identify a set of Pareto-optimal solutions [21] that represent the best possible trade-offs among the competing objectives. Formally, a multi-objective optimization problem can be formulated as:

$$min/max \ f(x) = min/max \ (f_1(x), f_2(x), \ldots, f_M(x))$$

where x is a μ-dimensional decision vector in the decision space, $f_M(x)$ is the Mth objective function, and M is the number of objectives.

This study employs the concept of Pareto dominance, which is the most common evaluation criterion in multi-objective optimization problems [22]. A solution s_a is said to dominate another solution s_b, denoted as $s_a \prec s_b$, if all objective values of s_a are at least as good as those of s_b, and strictly better in at least one objective. Specifically, in our research context, where one objective (total cost, donated as f_{cost}) is to be minimized and the other (accuracy, donated as f_{acc}) is to be maximized. Then solution s_a with objective values $(f_{cost}(a), f_{acc}(a))$ dominates solution s_b with objective values $(f_{cost}(b), f_{acc}(b))$ if and only if:

- $f_{cost}(a) \leqslant f_{cost}(b)$ and $f_{acc}(a) > f_{acc}(b)$, or
- $f_{cost}(a) < f_{cost}(b)$ and $f_{acc}(a) \geqslant f_{acc}(b)$

A solution s_a is called a **non-dominated solution** within a set of solutions S if no other solution in S dominates s_a. A solution s_a is a **Pareto-optimal solution** if no other feasible solution in the entire search space dominates s_a. The set of all Pareto-optimal solutions is known as the **Pareto front** or **reference point set**, representing the optimal trade-offs between the conflicting objectives. In an evolutionary algorithm, the set of all non-dominated solutions within the current population is called the **non-dominated set**. The goal of the multi-objective optimization process is to provide decision-makers with a diverse and well-balanced set of Pareto-optimal solutions, representing the best compromises among the multiple objectives.

III. PROBLEM DESCRIPTION

A. Problem Scope

Numerous LLMs with different capabilities and prices are now accessible via APIs. OptLLM aims to identify a set of cost-effective and high-performing solutions for leveraging LLMs in query-answering tasks, encompassing both general NLP tasks (such as text classification and summarization) and domain-specific tasks (such as log parsing). As discussed in Section II, multi-objective optimization problems can yield multiple Pareto-optimal solutions, unlike single-objective optimization problems typically have one solution deemed optimal. In this work, one objective is to maximize the percentage of queries processed accurately, while the other is to minimize the total cost of invoking LLM APIs.

B. Problem Formulation

Assume the user has a set of queries waiting to be processed by LLMs, represented as $J = \{j_1, j_2, \ldots, j_n\}$. Each query j_i

is characterized by the token number tn_i. The user has access to a set of LLMs $L = \{l_1, l_2, \ldots, l_m\}$, the price per token $price_k$ of invoking model l_k is available through the LLM provider. The cost $cost_i^k$ of submitting a query j_i on an LLM l_k is decided by the token number of the query content and the unit price of the LLM, which is calculated as follows:

$$cost_{i,k} = tn_i \times price_k \qquad (1)$$

Assume $acc_{i,k}$ is the result of leveraging the LLM l_k to query j_i, where $acc_{i,k} = 1$ represents the LLM processes the query correctly and 0 otherwise. We denote the decision variable $x_{i,k}$, which is a binary variable that is set as 1 when query i is assigned to the LLM k and 0 otherwise. This problem is then formalized as follows:

$$\text{Minimise } f_{cost} = \sum_{i=1}^{n} cost_i \qquad (2)$$

$$\text{Maximise } f_{acc} = \frac{1}{n} \sum_{i=1}^{n} acc_i \qquad (3)$$

s.t.

$$cost_i = \sum_{k=1}^{m} x_{i,k} \times cost_{i,k}, \forall i \in \{1, \ldots, n\} \qquad (4)$$

$$acc_i = \sum_{k=1}^{m} x_{i,k} \times acc_{i,k}, \forall i \in \{1, \ldots, n\} \qquad (5)$$

$$\sum_{k=1}^{m} x_{i,k} = 1, \forall i \in \{1, \ldots, n\} \qquad (6)$$

$$x_{i,k} \in \{0, 1\}, \forall i \in \{1, \ldots, n\}, \forall k \in \{1, \ldots, m\} \qquad (7)$$

Equations (2) and (3) are the objectives to minimize total cost and improve the percentage of queries that can be processed successfully. Constraints (4) and (5) define the cost and accuracy for each query i. Constraint (6) is the constraint that each query should be assigned to one LLM. Constraint (7) imposes restrictions on the decision variable.

IV. PROPOSED APPROACH

This section introduces OptLLM, a framework designed to efficiently allocate queries to the most suitable LLM while optimizing both cost and performance. OptLLM consists of two main components: prediction and optimization. The prediction component trains a model to estimate the probability of each LLM successfully processing a given query, while the optimization component uses the predicted probabilities to determine the optimal allocation of queries to LLMs. Figure 3 illustrates the architecture of OptLLM.

A. The Prediction Component

Determining the accuracy of LLMs on a specific input is challenging, as it remains uncertain until that input is actually processed by the LLM. To address this issue, we propose a multi-label classification model that incorporates uncertainty estimation techniques to enhance the reliability and stability of the performance predictions.
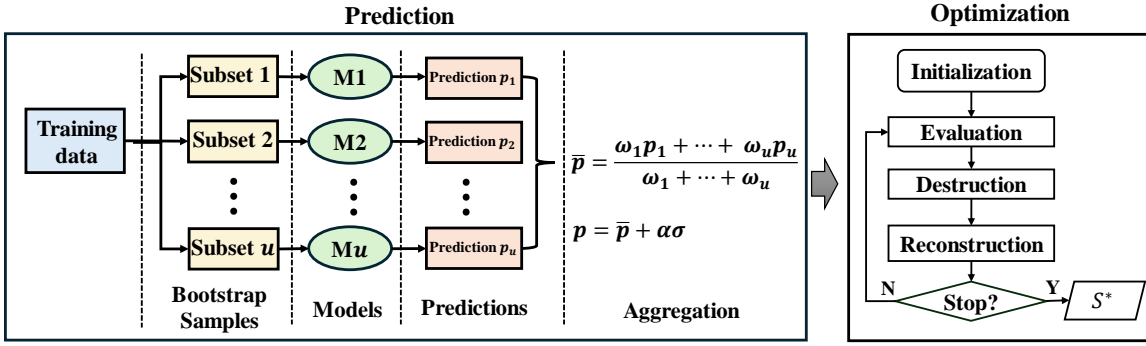
Fig. 3. The Framework of OptLLM

1) Bootstrap-based training and aggregation: The multi-label classifier takes the query content as input and predicts the accuracy of each candidate LLM for that specific query. The training set is collected by querying a small set of queries with each candidate LLM and recording their responses. Herein, a pre-trained word embedding model extracts features from the input query content, whilst the response results of the queried LLM are considered as labels. The word embeddings, which are trained on a large corpus of text data, enable the resulting feature vectors to capture more nuanced and contextual information about the input compared to traditional bag-of-words or count-based representations. Subsequently, we construct a set of prediction models trained with an ensemble of Random Forest classifiers on multiple bootstrap samples of the training and validation data. To obtain the final predictions, we aggregate the predictions from each bootstrap sample using a weighted mean approach. Assume there are u bootstrap samples, and for each sample, we evaluate the model's performance on the validation data by calculating the percentage of correctly predicted data, denoted as ω. The weighted mean prediction $\bar{p}$ is calculated as follows:

$$\bar{p} = \frac{\omega_1 p_1 + \ldots \omega_u p_u}{\omega_1 + \ldots \omega_u} \qquad (8)$$

2) Robust-aware predicted accuracy in aggregation: Predictions play a crucial role in the selection process during optimization. It is important to note that when the prediction model is not highly accurate, there is a risk that OptLLM may choose a solution with higher predicted accuracy but worse real-world performance. To mitigate this risk, we employ a robust optimization in aggregation phrase, a common approach for handling uncertainty by leveraging interval or discrete data from statistics or historical observations [23]. Specifically, we calculate the standard deviation σ of the predictions across the bootstrap samples to quantify the uncertainty. The robust-aware predicted accuracy is defined as follows. Formally, let $p_{i,k}$ represent the predicted accuracy of query j_i on LLM l_k. We formulate $p_{i,k}$ as:

$$p_{i,k} = \bar{p}_{i,k} + \alpha_{i,k} \times \sigma_{i,k} \qquad (9)$$

where $\bar{p}_{i,k}$ is the weighted mean prediction, and $\alpha_{i,k}$ is a parameter that controls the robustness of the predicted accuracy by adjusting the uncertainty term $\sigma_{i,k}$.

By incorporating uncertainty information into the predictions, our model can adapt its outputs based on the level of uncertainty, providing more robust and reliable results. The α parameter determines the robustness of the predictions by governing the impact of the uncertainty term on the final predictions. When α is set to 0, the uncertainty term is disregarded, and the robust predictions are equivalent to the mean predictions, indicating no additional consideration of uncertainty. Positive values of α increase the robustness by adding the uncertainty term to the mean predictions, with larger values resulting in more conservative predictions that account for a wider range of possible accuracies. Conversely, negative values of α decrease the robustness by subtracting the uncertainty term from the mean predictions, with larger negative values yielding more aggressive predictions that consider a narrower range of possible accuracies. During the search process, the accuracy objective of a candidate allocation solution is determined as follows:

$$f'_{acc} = \frac{1}{n} \sum_{i=1}^{n} p_i \qquad (10)$$

$$p_i = \sum_{k=1}^{m} x_{i,k} \times p_{i,k}, \quad \forall i \in \{1, \ldots, n\} \qquad (11)$$

where f'_{acc} represents the predicted overall accuracy objective of the solution, and p_i is the predicted accuracy for query j_i based on the LLM assignments in the current solution.

B. The Optimization Component

As shown in Algorithm 1, the optimization component consists of initialization and heuristic search phases. In the initialization phase, OptLLM generates two extreme optimal solutions s_h and s_c (Line 2 in Algorithm 1), where s_h has the highest predicted accuracy and s_c has the lowest cost. Then, the search phase of OptLLM performs destruction and reconstruction iteratively (Lines 7-13 in Algorithm 1) to generate a set of non-dominated solutions.

Algorithm 1 Function *Optimization*

Require: P: predicted accuracy table; C: cost table; GN: grid parameter
Ensure: S^*: non-dominated solutions output by the OptLLM;
  1: $S^* \leftarrow \emptyset$
  2: $s_h, s_c \leftarrow$ Initialization(P, C)
  3: $\Delta cost \leftarrow \frac{f_{cost}(s_h) - f_{cost}(s_c)}{GN}$
  4: $\Delta acc \leftarrow \frac{f_{acc}(s_h) - f_{acc}(s_c)}{GN}$
  5: $s_1 \leftarrow s_h$
  6: $s_2 \leftarrow s_c$
  7: **while** no termination criterion is met **do**
  8:     $\bar{s}_1, \bar{s}_2 \leftarrow$ DESTRUCTION_STRATEGY$((s_1, \Delta cost), (s_2, \Delta ccc))$
  9:     $\hat{s}_1, \hat{s}_2 \leftarrow$ RECONSTRUCTION_STRATEGY$(\bar{s}_1, \bar{s}_2)$
 10:     $S^* \leftarrow \hat{s}_1, \hat{s}_2$
 11:     $s_1 \leftarrow \hat{s}_1$
 12:     $s_2 \leftarrow \hat{s}_2$
 13: **end while**
 14: **return** S^*

Fig. 4. Solutions by destruction and reconstruction in optimization component. B is a feasible solution derived from the non-dominated solution A by releasing sufficient cost, and C is a new non-dominated solution obtained by reconstructing B.

1) Initialization: In the initialization phase, we generate two optimal solutions according to the concept of Pareto dominance. Let s_c denote the solution obtained by assigning each query to the cheapest LLM, resulting in the lowest cost. On the other hand, s_h represents the solution achieved by selecting the LLM with the highest predicted accuracy for each query. To prove that both s_c and s_h are Pareto solutions, we introduce the following lemma:

Lemma IV.1. *If s_c represents the solution with the lowest cost and s_h represents the solution with the highest accuracy, then both s_c and s_h are Pareto solutions.*

Proof. If s_c is not a Pareto solution, then there must exist a solution s such that s dominates s_c in at least one objective while being no worse in the other objective. Formally, $s \prec s_c$, where "$\prec$" indicates that solution s dominates solution s_c.

Let f_{cost} denote the cost objective and f_{acc} denote the accuracy objective. If $s \prec s_c$, then $f_{cost}(s) < f_{cost}(s_c)$ and $f_{acc}(s) \geq f_{acc}(s_c)$, or $f_{cost}(s) \leq f_{cost}(s_c)$ and $f_{acc}(s) > f_{acc}(s_c)$. However, s_c is the solution with the lowest cost, where $f_{cost}(s_c) < f_{cost}(s)$. This contradicts the definition of dominance, as s_c should be at least as good as s in all objectives and strictly better in at least one objective.

Similarly, s_h represents the solution with the highest accuracy, where $f_{acc}(s_h) > f_{acc}(s)$ for any solution s. According to the definition of Pareto dominance, there is no solution dominating s_c or s_h. Therefore, both s_c and s_h are Pareto solutions. This completes the proof. □

2) The heuristic search: During the heuristic search phase, OptLLM employs a process of destruction and reconstruction to iteratively generate a set of non-dominated solutions. During the destruction stage, OptLLM removes elements (queries have been assigned) from the current solution and reassigns them to maximize one objective while temporarily disregarding the other. This approach allows for a focused optimization of the prioritized objective. There are two search directions. Assume s_h is the solution with the highest expected accuracy with a relati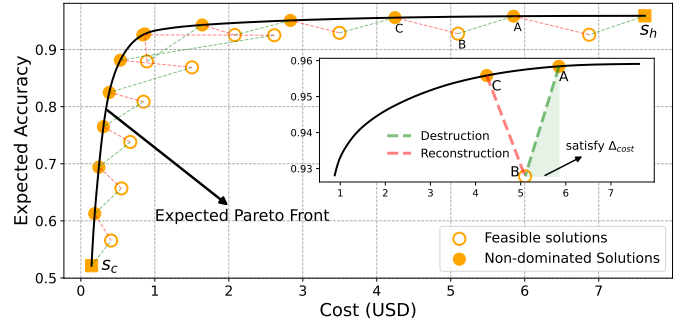vely high associated cost. In this case, OptLLM first destructs s_h to release as much cost as possible. Conversely, s_c represents the cheapest solution with a lower accuracy. During the destruction, OptLLM prioritizes increasing accuracy to the greatest extent possible by temporarily disregarding the cost objective. Subsequently, the reconstruction stage refines the solution obtained from destruction, transforming it into a non-dominated solution using a predefined scoring function that balances both objectives. For instance, as illustrated in Figure 4, solution B is a feasible solution derived from the non-dominated solution A by releasing sufficient cost. Subsequently, solution C is obtained by reconstructing solution B, resulting in a new non-dominated solution.

Destruction. Let $\hat{s}$ denote the solution obtained after the destruction phase. Taking optimizing cost objective as an example, $\hat{s}$ is expected to achieve cost savings of gap_{cost} compared to the original solution s, expressed as $f_{cost}(\hat{s}) - f_{cost}(s) \geq gap_{cost}$ (Lines 2-6 in Algorithm 2). The algorithm iteratively reassigns queries, beginning with those offering the largest cost reductions, until the cost of the solution $\hat{s}$ meets the specified condition. For a query j_i, if its current assignment is LLM k' and will be reassigned to k, the cost difference $cs_{i,k}$ is calculated as follows:

$$cs_{i,k} = cost_{i,k} - cost_{i,k'}, k \neq k'$$

For each query j_i, cs_i represents the largest cost savings achievable by selecting another LLM, calculated as:

$$cs_i = \min\{cs_{i,1}, cs_{i,2}, \ldots, cs_{i,m}\}$$

Similarly, if the goal is to optimize accuracy through destruction, $\hat{s}$ is expected to achieve an accuracy improvement of gap_{acc} compared to the original solution s (Lines 8-12 in Algorithm 2). This is expressed as $f_{acc}(\hat{s}) - f_{acc}(s) \geq gap_{acc}$. The accuracy improvement ($ai_{i,k}$) associated with reallocating query j_i from LLM k' to LLM k is defined as follows:

$$ai_{i,k} = acc_{i,k} - acc_{i,k'}, k \neq k'$$

Let ai_i denote the maximum accuracy improvement achievable by selecting an alternative LLM for query j_i, calculated as:

$$ai_i = \max\{ai_{i,1}, ai_{i,2}, \ldots, ai_{i,m}\}$$

Algorithm 2 Function *Destruction*
___
Require: gap_{cost} (or gap_{acc}): cost (or accuracy) gap
Ensure: $\hat{s}$: solution output by the function *Destruction*
1: **if** gap_{cost} **then**
2:      **while** $f_{cost} - \hat{f}_{cost} < gap_{cost}$ **do**
3:          $i = Min(cs_i)$ ▷ query with the largest cost reduction
4:          $\hat{s} \leftarrow$ reallocation(s, i)
5:          $s \leftarrow \hat{s}$
6:      **end while**
7: **else if** gap_{acc} **then**
8:      **while** $\hat{f}_{acc} - f_{acc} < gap_{acc}$ **do**
9:          $i = Max(ai_i)$ ▷ query with the largest accuracy improvement
10:          $\hat{s} \leftarrow$ reallocation(s, i)
11:          $s \leftarrow \hat{s}$
12:      **end while**
13: **end if**
14: **return** $\hat{s}$

Algorithm 3 Function *Reconstruction*
___
Require: $\hat{s}$: solution output by the function *Destruction*
Ensure: $\bar{s}$: solution output by the function *Reconstruction*
1: $i_1 \leftarrow$ query with the largest positive score $score_{i_1}$ in s
2: $i_2 \leftarrow$ query with the largest negative score $score_{i_2}$ in s
3: **while** $|score_{i_1}| > |score_{i_2}|$ and $score_{i_1} \cdot score_{i_2} < 0$ **do**
4:      $\hat{s} \leftarrow$ reallocation(s, i_1, i_2)
5:      **if** $(s \nprec \hat{s}) \wedge (\hat{s} \nprec s)$ **then** ▷ *Neither $\hat{s}$ nor s dominates each other*
6:          $\hat{S} \leftarrow \hat{S} \cup \{\hat{s}\}$
7:      **else if** $\hat{s} \prec s$ **then**
8:          $\hat{S} \leftarrow \{\hat{s}\} \cup (\hat{S} \setminus \{s\})$ ▷ *replace s with $\hat{s}$*
9:      **end if**
10:      $s \leftarrow \hat{s}$
11:      update $i_1, i_2, score_{i_1}, score_{i_2}$
12: **end while**
13: **return** $\hat{S}$

Reconstruction. Then, we perform reconstruction to optimize solution $\hat{s}$. During this process, the objective is to maximize the accuracy improvement per unit cost. In other words, for a given solution, a higher ratio of accuracy improvement to cost increase is considered more desirable. Based on this thought, we introduce the following lemma.

Lemma IV.2. *Assume there are two solutions s_1 and s_2. If $\frac{f_{acc}(s_1)}{f_{cost}(s_1)} \leq \frac{f_{acc}(s_2)}{f_{cost}(s_2)}$, then $s_1 \nprec s_2$.*

Proof. If $s_1 \prec s_2$, based on the definition, there are two possible situations: 1) $f_{cost}(s_1) \leq f_{cost}(s_2)$ and $f_{acc}(s_1) > f_{acc}(s_2)$. 2) $f_{cost}(s_1) < f_{cost}(s_2)$ and $f_{acc}(s_1) \geq f_{acc}(s_2)$.

In the first situation, if $f_{acc}(s_1) > f_{acc}(s_2)$, then $\frac{f_{acc}(s_1)}{f_{cost}(s_2)} > \frac{f_{acc}(s_2)}{f_{cost}(s_2)}$. Because $f_{cost}(s_1) \leq f_{cost}(s_2)$, we can easily have $\frac{f_{acc}(s_1)}{f_{cost}(s_1)} \geq \frac{f_{acc}(s_1)}{f_{cost}(s_2)} > \frac{f_{acc}(s_2)}{f_{cost}(s_2)}$. Similarly, in the second situation, we can conclude that $\frac{f_{acc}(s_1)}{f_{cost}(s_1)} > \frac{f_{acc}(s_1)}{f_{cost}(s_2)} \geq \frac{f_{acc}(s_2)}{f_{cost}(s_2)}$.

Therefore, we can have if $s_1 \prec s_2$, then $\frac{f_{acc}(s_1)}{f_{cost}(s_1)} > \frac{f_{acc}(s_2)}{f_{cost}(s_2)}$. This is logically equivalent to: if $\frac{f_{acc}(s_1)}{f_{cost}(s_1)} \leq \frac{f_{acc}(s_2)}{f_{cost}(s_2)}$, then $s_2 \nprec s_1$. This completes the proof. □

Based on Lemma IV.2, we reassign the queries in $\hat{s}$ until the quotient of accuracy and cost (i.e., $\frac{f_{acc}}{f_{cost}}$) cannot be increased

further. Let s be the original solution and s' be the solution after the reassignment. The difference between s and s' is caused by the reassignment of a query or a set of queries. By iteratively reassigning queries and applying the criterion from Lemma IV.2, we arrive at a non-dominated solution $\bar{s}$ that cannot be further improved, indicating that there is no new solution possible that can dominate the current solution.

V. EXPERIMENTAL DESIGN

A. Research Questions

We aim to answer the following research questions (RQs) through experiments.
RQ1: Can OptLLM allocate queries to LLMs effectively and efficiently?
RQ2: Does each core component of OptLLM contribute to the overall performance?
RQ3: How do different hyper-parameter settings affect the performance of OptLLM?

TABLE I
BENCHMARK SPECIFICATION

Dataset	Task Type	Train size (1%)	Val size (1%)	Test size (98%)
LogPai	Log parsing	320	320	31360
AGNEWS	Text classification	76	76	7448
COQA	Question answering	80	80	7822
HEADLINES	Sentiment analysis	100	100	9800
SCIQ	Reasoning	117	117	11443

B. Benchmark Setting

1) Natural Language Processing (NLP) tasks: To show the generality of OptLLM on different types of tasks, we have chosen four NLP tasks, including text classification(AGNEWS [24]), question answering (COQA [25]), sentiment analysis (HEADLINES [26]), and reasoning(SCIQ [27]). 12 candidate LLMs are selected from 4 mainstream providers: OpenAI[1] (GPT-Curie, ChatGPT, GPT-3, and GPT-4), AI21[2] (Jurassic-1 Large, Jurassic-1 Grande, and Jurassic-1 Jumbo), Cohere[4] (Xlarge and Medium), and Textsynth [5] (GPT-J, FAIRSEQ, and GPT-Neox). The raw data is provided by Chen et al. [10], which contains the inputs (prompts) sent to the LLMs, ground truth references, LLM outputs, and cost.

2) Domain-specific tasks: Furthermore, we have chosen an intelligent software engineering (SE) task, specifically focusing on LLM-based log parsing. We utilize log data sourced from the LogPai benchmark [28], [29] to interface with 8 LLMs, including TogertherAI[3] (Mixtral-8x7B, Llama-2-7B, Llama-2-13B, Llama-2-70B, Yi-34B, Yi-6B), AI21[2](Jurassic-2 Mid and Jurassic-2 Ultra). The LogPai benchmark consists of log data from 16 systems, including distributed systems, supercomputers, operating systems, mobile systems, server applications, and standalone software. The raw data includes inputs (queries and full prompts) sent to the LLMs, ground

___

[4]https://cohere.com/
[5]https://textsynth.com/

truth references, LLM outputs, and the corresponding execution cost. The details of the datasets are listed in Table I.

C. Baselines

1) Individual LLM: Because it is common practice to assign a task to a specific LLM, we select individual LLMs as baselines to compare against OptLLM's performance. We submit the entire set of queries to each individual LLM and assess the resulting cost and the proportion of queries that have been successfully completed.

2) Classic multi-objective optimization algorithms: As shown in Algorithm 1, OptLLM utilizes a heuristic search-based algorithm in optimization. We compare the effectiveness of this algorithm with well-known multi-objective optimization algorithms, including the Non-dominated Sorting Genetic Algorithm (NSGA-II) [30], Multi-objective Particle Swarm Optimisation (MOPSO) [31], and Multi-objective Evolutionary Algorithm with Decomposition (MOEA/D) [32]. These three algorithms have been extensively studied and have proven to be effective in solving a wide range of multi-objective optimization problems [20], [33]. In addition, three variants of classic algorithms are also compared, including R-NSGA-II [34], SMS-EMOA [35], and MOEA/D-GEN [36]. It is important to note that all the evaluated multi-objective optimization algorithms are integrated with the same prediction component as OptLLM, to enable a fair comparison of the optimization strategies.

D. Evaluation Metrics

1) Evaluating single solution performance: When evaluating the performance of a single solution, such as assigning all queries to a single LLM, directly comparing the optimization objectives is feasible. This approach allows for a straightforward assessment of the solution's effectiveness in terms of cost and accuracy. Two objectives considered in this evaluation are:

- f_{cost}: the total cost of invoking LLM APIs (see Equation 2)
- f_{acc}: the percentage of jobs that have been processed correctly (see Equation 3)

2) Multi-objective optimization evaluation metrics: To evaluate the performance of multi-objective optimization algorithms, we introduce widely used metrics: IGD and Δ [37]. The metrics require knowledge of the Pareto front, which represents the set of optimal trade-off solutions. We obtain the Pareto front through an exhaustive enumeration search, which serves as a reference for calculating the evaluation metrics.

True Pareto Front Generation: Based on the definition of Pareto dominance, an extremely optimal solution $s_{cheapest}$ can be easily obtained, where all jobs are allocated to the cheapest LLM to achieve the lowest cost. Then, we generate a new solution that is not dominated by $s_{cheapest}$. Specifically, we calculate the cost required by a new LLM to process the job correctly for all the jobs that are not solved in the current solution. The job with the smallest cost is reallocated to get a new optimal solution. Repeat this process until the accuracy cannot be increased.

Inverted Generational Distance (IGD): The IGD measures the distance between the obtained solution set and the true Pareto front (or the reference set), evaluating the quality of the obtained solution set [37]. Let the solution set $\Lambda = \{y_1, y_2, \ldots, y_{|\Lambda|}\}$ be the Pareto front for the problem and $\hat{\Lambda} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|\hat{\Lambda}|}\}$ be the solutions (approximation) obtained by an algorithm. We can define IGD by:

$$IGD(\hat{\Lambda}, \Lambda) = \frac{1}{|\Lambda|} \sqrt{\sum_{y \in \Lambda} \left(\min_{\hat{y} \in \hat{\Lambda}} d(\hat{y}, y) \right)^2}$$

where $|\Lambda|$ denotes the number of solutions in the true Pareto front (or the reference set). $d(\hat{y}, y)$ is the Euclidean distance (or any other appropriate distance metric) between a solution $\hat{y}$ in the obtained solution set $\hat{\Lambda}$ and the nearest solution y in the true Pareto front Λ. A lower value of IGD means a better performance, indicating the obtained solution set has a better distribution and better approximation to the reference set.

Δ **metric**: The Δ metric assesses the diversity and uniformity of the solutions distribution along the Pareto front by measuring Euclidean distances between consecutive solutions and comparing them to the average distance [37].

$$\Delta(\hat{\Lambda}) = \frac{d_f + d_l + \sum_{z=1}^{|\hat{\Lambda}|-1} |d_z - \bar{d}|}{d_f + d_l + (|\hat{\Lambda}| - 1)\bar{d}}$$

where d_f and d_l are the Euclidean distances between the extreme solutions and their nearest neighbors in the obtained solution set $\hat{\Lambda}$. d_z denotes the Euclidean distance between the zth solution $\hat{y}_z$ and its next solution $\hat{y}_{z+1}$ in the obtained solution set $\hat{\Lambda}$, sorted in ascending order of a chosen objective. $\bar{d}$ represents the mean value of the distances d_z. A smaller Δ metric indicates a higher diversity and distribution of solutions.

To ensure a fair comparison, all multi-objective algorithms have the same termination condition of 200 iterations. We also record the execution time (in minutes) required by each algorithm to generate the solution sets within this fixed number of iterations to evaluate the computational efficiency.

E. Implementation

To mitigate the impact of randomness and ensure the robustness of our findings, each experiment is conducted 10 times and the average value is reported. The implementations of all algorithms are based on Python 3.11. Specifically, we utilize the standard versions of NSGA-II, R-NSGA-II, and SMS-EMOA from the Pymoo library [38], while MOPSO and MOEA/D are obtained from the Pygmo library.

Accuracy table: The training and validation sets each comprise 1% of the data, while the remaining data is used for testing. All multi-objective optimization algorithms use the same predicted accuracy generated by the prediction component for evaluation during the search process.

Parameter setting: Optuna, a widely used hyperparameter optimization package [39], is employed to ensure the effectiveness and efficiency of all algorithms. We conduct parameter tuning using Optuna to choose optimal parameter settings for all algorithms. For OptLLM, bootstrap sample number μ is

set as 100, GN is set as 50, and α is set as 0.5. Due to space constraints, the results and optimal parameters for all baselines are provided on our project webpage[6].

VI. EXPERIMENTAL RESULTS

A. RQ1: Comparison with the Baselines

1) Comparison with individual LLMs: OptLLM generates a set of solutions rather than a single solution. For comparison purposes, we select the solution from OptLLM that achieves the same accuracy as the best-performing individual LLM for each dataset. Table II presents the cost savings achieved by OptLLM compared to the best individual LLM. For instance, on the AGNEWS dataset, GPT-4 outperforms other individual LLMs, attaining an accuracy of 0.90 at a cost of 126.58. OptLLM provides a solution with the same accuracy but at a lower cost of 75.77, resulting in a 40.14% cost saving. Across all datasets, OptLLM maintains the same level of accuracy as the best individual LLM while significantly reducing costs. The cost savings range from 2.40% for the SCIQ dataset to 49.18% for the LogPai dataset, demonstrating OptLLM's ability to achieve comparable performance to the best individual LLM while offering cost savings.

TABLE II
COST SAVINGS BY OPTLLM COMPARED WITH THE INDIVIDUAL LLM
(f_{cost} AND f_{acc} ARE THE COST AND ACCURACY OF THE SOLUTION)

Dataset	Best Individual LLM			Cost to reach the same accuracy	
	Model	f_{acc}	f_{cost}	f_{cost}	Cost Savings
AGNEWS	GPT-4	0.90	126.58	75.77	40.14%
COQA	GPT-4	0.27	216.01	152.63	29.34%
HEADLINES	GPT-4	0.86	65.28	40.91	37.33%
SCIQ	Xlarge	0.71	144.86	141.39	2.40%
LogPai	Mixtral-8x7B	0.59	3.68	1.87	49.18%

2) Comparison with classic multi-objective optimization algorithms: Table III presents the best solution with the highest accuracy generated by all algorithms. The results demonstrate that OptLLM consistently provides solutions with the highest accuracy among all the compared methods. For instance, OptLLM can generate a solution reaching an accuracy of 0.71, while the baseline algorithms can only provide solutions with an accuracy of around 0.45. Furthermore, Table IV illustrates the cost reduction achieved by OptLLM while maintaining the same accuracy as the highest accuracy attained by the baseline methods. Take the AGNEWS dataset as an example, MOPSO, the second-best performer among all the algorithms, achieves an accuracy of 82.18% at a cost of 51.15 dollars. While OptLLM attains the same accuracy for only 5.1 dollars, resulting in a substantial cost saving of 90.03%. This significant cost reduction demonstrates OptLLM's ability to provide high-performing solutions under strict cost constraints, making it a more cost-effective option compared to the baseline methods.

Table V presents the IGD and Δ of the solutions generated by OptLLM and other multi-objective optimization methods on the benchmarks. OptLLM achieves the best performance on all benchmark instances. The IGD value of the solutions found by OptLLM is much smaller than that achieved by its

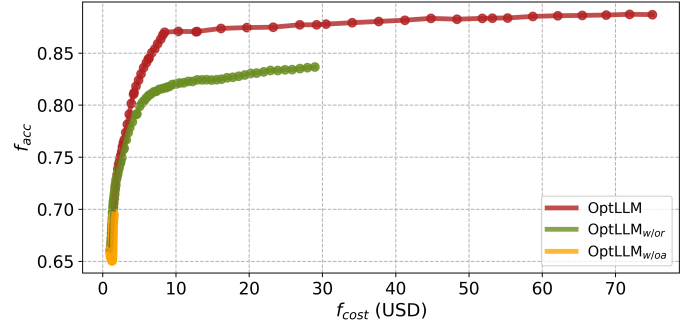[6]https://github.com/superyue72/OptLLM

Fig. 5. Ablation study on AGNEWS. The optimization component and robust-aware prediction help find the solution with higher accuracy. The solution with the highest accuracy by OptLLM$_{\text{w/o o}}$ is 69.54%, OptLLM$_{\text{w/o r}}$ is 83.67%, OptLLM is 88.74%

competitors. For example, on the AGNEWS dataset, OptLLM can achieve an IGD of 0.13, while the second best value is 11.61 by MOPSO. This indicates that the solutions generated by OptLLM are much closer to the Pareto Front, with a lower cost and higher accuracy. In terms of Δ, OptLLM can obtain the smallest value in all scenarios, showing the best diversity of the obtained solution set. In terms of computation time, to generate the same number of solutions, OptLLM is around 5 times faster than its competitors.

There are two possible reasons why OptLLM performs better. First, the solutions generated by the optimization component can be regarded as the ones with the best trade-off between cost and predicted accuracy (See Section IV). Second, multi-objective algorithms may be more prone to getting trapped in local optima, especially when the predicted accuracies for different queries are similar. In such cases, the algorithms may struggle to identify the globally optimal solutions without additional problem-specific information to guide the search process. OptLLM, on the other hand, incorporates heuristic rules through its prediction component, which helps to guide the optimization process and avoid local optima.

Overall, the experimental results demonstrate the superior performance of OptLLM in multiple aspects.

To verify the comparison, we conduct a statistical test, and the results are provided on our project webpage due to the space limitation[6].

> *In response to RQ1, our results show that OptLLM outperforms both single LLM allocation solutions and other multi-objective optimization strategies.*

B. RQ2: Ablation Study

To assess the performance improvement contributed by the optimization component of OptLLM, we conduct an ablation study. We design two variant algorithms: OptLLM$_{\text{w/o o}}$, which represents OptLLM without optimization, and OptLLM$_{\text{w/o r}}$, which represents OptLLM without robust-aware prediction. The results of OptLLM and its alternative versions are presented in Table VI.

TABLE III
THE SOLUTION WITH THE HIGHEST ACCURACY BY ALL ALGORITHMS

Dataset	AGNEWS		COQA		HEADLINES		SCIQ		LogPai	
Alg	f_{acc}	IMPROV	f_{acc}	IMPROV	f_{acc}	IMPROV	f_{acc}	IMPROV	f_{acc}	IMPROV
OptLLM	**0.90**	\	**0.27**	\	**0.86**	\	**0.70**	\	**0.71**	\
NSGA-II	0.74	21.62%	0.21	28.57%	0.82	4.88%	0.68	2.94%	0.42	69.05%
MOPSO	0.82	9.76%	0.23	17.39%	0.80	7.50%	0.68	2.94%	0.48	47.92%
MOEA/D	0.70	28.57%	0.22	22.73%	0.78	10.26%	0.67	4.48%	0.46	54.35%
RNSGA-II	0.77	16.88%	0.22	22.73%	0.79	8.86%	0.67	4.48%	0.46	54.35%
SMS-EMOA	0.76	18.42%	0.22	22.73%	0.79	8.86%	0.67	4.48%	0.45	57.78%
MOEA/D-GEN	0.82	9.76%	0.22	22.73%	0.80	7.50%	0.68	2.94%	0.47	51.06%

TABLE IV
COST (f_{cost}) SAVINGS BY OPTLLM TO MATCH THE BASELINE'S PERFORMANCE

Dataset	AGNEWS			COQA			HEADLINES			SCIQ			LogPai		
Alg	Baselines	OptLLM	Savings	Baselines	OptLLM	Savings	Baselines	OptLLM	Savings	Baselines	OptLLM	Savings	Baselines	OptLLM	Savings
NSGA-II	30.84	2.35	92.38%	26.63	16.91	36.50%	4.78	3.40	28.87%	28.70	20.14	29.83%	11.93	1.83	84.66%
MOPSO	51.15	5.10	90.03%	36.17	32.99	8.79%	14.63	1.96	86.60%	80.30	41.11	48.80%	3.10	2.07	33.23%
MOEA/D	16.21	1.54	90.50%	25.74	17.40	32.40%	6.67	0.62	90.70%	44.78	10.42	76.73%	2.01	1.83	8.96%
RNSGA-II	26.44	3.07	88.39%	39.82	18.40	53.79%	16.15	0.71	95.60%	71.15	10.45	85.31%	11.17	1.83	83.62%
SMS-EMOA	28.62	2.83	90.11%	43.34	16.51	61.91%	16.71	0.69	95.87%	77.91	9.53	87.77%	9.77	1.83	81.27%
MOEA/D-GEN	35.44	4.81	86.43%	28.15	19.19	31.83%	14.81	2.31	84.40%	66.96	32.08	52.09%	3.79	1.83	51.72%

TABLE V
COMPARISONS OF SOLUTION SETS FROM ALL ALGORITHMS IN TERMS OF IGD, Δ, AND TIME

Dataset	AGNEWS			COQA			HEADLINES			SCIQ			LogPai		
Metrics	IGD	Δ	Time	IGD	Δ	Time	IGD	Δ	Time	IGD	Δ	Time	IGD	Δ	Time
OptLLM	**0.13**	**0.55**	**6.15**	**0.27**	**0.79**	**7.64**	**0.08**	**0.53**	**7.71**	**0.14**	**0.58**	**10.01**	**0.19**	**0.95**	**18.14**
NSGA-II	22.23	1.00	30.16	16.60	1.00	36.46	4.52	1.00	40.35	26.01	1.00	50.53	10.90	1.00	129.20
MOPSO	13.13	0.99	33.85	16.06	0.98	39.93	5.26	0.99	43.93	30.07	0.99	51.68	0.38	1.00	126.01
MOEA/D	13.97	0.99	33.91	23.26	1.00	39.65	4.44	0.99	43.79	44.32	1.00	50.97	1.05	0.98	127.24
RNSGA-II	23.85	0.99	30.43	34.87	0.99	35.10	15.02	0.99	42.78	63.54	0.99	50.76	10.17	1.00	128.11
SMS-EMOA	25.97	0.99	30.60	39.30	0.99	36.14	15.85	0.99	39.78	72.74	0.99	50.61	8.78	1.00	130.22
MOEA/D-GEN	9.12	1.09	35.06	23.91	1.16	39.54	4.69	1.07	44.80	42.52	1.13	45.47	0.75	1.07	124.00

In OptLLM$_{w/o\ o}$, we consider an LLM capable of processing a query correctly if the predicted probability of a correct response exceeds 0.5. Solutions are then generated by assigning each job to the LLM predicted to process it correctly at the lowest cost. As illustrated in Figure 5, this approach successfully provides low-cost solutions but fails to identify solutions with high accuracy, indicating that the prediction component alone is insufficient for achieving optimal performance. The results presented in Table VI demonstrate the crucial role of the optimization module in OptLLM, as it significantly improves both the quality and diversity of the generated solutions. By facilitating the discovery of more high-accuracy solutions, the optimization process enhances the overall performance and versatility of the OptLLM framework.

In OptLLM$_{w/o\ r}$, we use the predicted accuracy output by the base model to evaluate candidate solutions during the optimization phase. The difference between OptLLM$_{w/o\ r}$ and OptLLM is minimal. Further analysis reveals that when the prediction model has high accuracy, the performance of OptLLM$_{w/o\ r}$ is similar to that of OptLLM. This similarity is due to the smaller difference between the predicted accuracy and the robust-aware predicted accuracy when the prediction model performs well. In other words, the robust-aware function is useful when the predicted accuracy is suboptimal.

Figure 5 provides an example using the AGNEWS dataset. It clearly shows that OptLLM$_{w/o\ o}$ generates substantially fewer solutions and fails to find solutions with higher accuracy. Although OptLLM$_{w/o\ r}$ performs better than OptLLM$_{w/o\ o}$, OptLLM, which incorporates a robust-aware function, excels in searching for solutions with higher accuracy.

> *In response to RQ2, both robust-aware prediction function and heuristic optimization significantly enhance the overall capability of the OptLLM.*

C. RQ3: Effect of Hyper-Parameter Settings

1) The grid parameter GN: In Table VII, we show IGD, Δ, average runtime, and average size of the solution set produced by OptLLM under various settings of the GN parameter, which controls objective distance during the optimization component's destruction phase. Analysis of the table shows that increasing the GN value leads to a larger number of solutions generated by OptLLM but at the cost of increased computation time. While the solution quality, as indicated by IGD and Δ, remains similar across different GN settings.

2) The training data size of prediction component: Table VI-B presents the results of OptLLM under different training data sizes for the prediction component. As the size of the training data increases, the Δ metric consistently increases, while the IGD shows a slight decrease overall. Consequently, we choose to use 1% of the dataset for training, striking

TABLE VI
ABLATION STUDY OF OPTLLM

Metrics	AGNEWS			COQA			HEADLINES			SCIQ			LogPai		
	IGD	Δ	Time	IGD	Δ	Time	IGD	Δ	Time	IGD	Δ	Time	IGD	Δ	Time
OptLLM	**0.13**	**0.55**	**6.15**	0.27	**0.79**	7.64	**0.08**	**0.53**	**7.71**	**0.14**	**0.58**	**10.01**	**0.19**	**0.95**	**18.14**
OptLLM$_{w/o\ r}$	0.13	0.78	7.23	0.24	0.80	10.02	0.08	0.79	11.23	0.28	0.77	10.33	0.20	0.97	19.49
OptLLM$_{w/o\ a}$	0.23	0.86	14.38	**0.12**	1.1	16.74	0.1	0.87	18.41	0.40	1.08	27.60	0.19	1.21	500.32

TABLE VII
COMPARISON OF OPTLLM WITH DIFFERENT SETTINGS OF GN (N: NUMBER OF GENERATED SOLUTIONS)

Dataset	AGNEWS				COQA				HEADLINES				SCIQ				LogPai			
	IGD	Δ	Time	N	IGD	Δ	Time	N	IGD	Δ	Time	N	IGD	Δ	Time	N	IGD	Δ	Time	N
GN=10	0.17	0.82	**2.29**	14	**0.27**	**0.80**	**2.13**	20	0.10	0.85	**3.49**	16	0.18	0.80	**4.81**	18	**0.14**	**0.83**	7.78	20
GN=50	0.13	0.80	6.35	71	0.27	0.80	7.64	99	0.08	0.79	8.85	81	0.14	0.77	10.01	92	0.19	0.95	18.14	100
GN=100	0.13	0.80	11.82	144	0.30	0.81	13.52	199	0.08	0.78	15.65	162	0.14	0.77	20.45	189	0.20	0.96	37.87	200
GN=200	**0.13**	**0.79**	18.25	**291**	0.33	0.82	23.42	**398**	**0.08**	**0.77**	25.35	**327**	**0.13**	**0.77**	35.72	**387**	0.18	0.95	63.60	**400**

TABLE VIII
RESULTS UNDER DIFFERENT TRAINING DATA SIZES

Dataset	AGNEWS		COQA		HEADLINES		SCIQ		LogPai	
size	IGD	Δ	IGD	Δ	IGD	Δ	IGD	Δ	IGD	Δ
1%	0.13	**0.65**	0.23	**0.66**	0.08	**0.61**	**0.13**	0.61	0.19	**0.87**
5%	0.11	0.71	0.19	0.67	0.07	0.71	0.14	0.63	0.18	1.01
10%	0.10	0.73	0.17	0.69	0.06	0.74	0.14	0.63	0.18	1.09
20%	**0.09**	0.74	**0.15**	0.68	**0.06**	0.73	0.14	0.63	**0.16**	1.10

TABLE IX
ACCURACY OF PREDICTION MODEL WITH DIFFERENT α

Dataset	AGNEWS	COQA	HEADLINES	SCIQ	LogPai
base model	0.21	0.45	0.44	0.25	**0.46**
$\alpha = -1$	0.14	**0.50**	0.41	0.17	0.39
$\alpha = -0.5$	0.19	0.48	0.45	0.24	0.43
$\alpha = 0$	0.24	0.47	0.46	0.31	0.45
$\alpha = 0.5$	0.26	0.43	0.47	0.35	0.43
$\alpha = 1$	**0.27**	0.40	**0.48**	**0.37**	0.38

a balance between performance and the cost of acquiring labelled data.

3) The robustness parameter α: Table IX presents the accuracy of the predictions on different datasets when using different values of the robustness parameter α. The base model represents the accuracy of the classification model trained directly on the training data without using bootstrap sampling. The impact of the robustness parameter α on prediction accuracy varies across datasets. While some datasets (AGNEWS, HEADLINES, SCIQ) benefit from higher values of α, others (COQA, LogPai) achieve better performance with lower or zero values of α.

> *In response to RQ3, OptLLM allows for the adjustment of GN to a larger value to generate more alternative solutions, while it prefers a relatively small training size for decreased Δ and lower costs in labelled data acquisition. Moreover, the effect of the robustness parameter α on prediction accuracy is found to differ among datasets.*

VII. RELATED WORK

LLMs have been adopted to address a variety of tasks and have shown numerous potential breakthroughs [18], [40]–[42]. This has driven the demand for optimizing the performance and cost of LLM-based tasks. Zong et al. [43] conducted an empirical study assessing the annotation, training, and inference costs of various LLMs in text classification, offering guidance for selecting the most suitable model in real-world scenarios. Chen et al. [10] introduced FrugalGPT, an algorithmic framework that adaptively selects suitable LLMs for different queries to reduce cost and improve accuracy. Specifically, it sends queries sequentially to available LLMs until reaching a predefined performance threshold defined by a scoring function. Similarly, Sakota et al. [8] developed FORC, a framework that allocates queries to suitable LLMs by predicting cost and performance of each candidate LLM on every query. Notably, these prediction models often require substantial training data (e.g., 50%), incurring high label collection cost. Unlike the above work, we formulate the problem as a multi-objective optimization and use less data (i.e., 1%) to construct the prediction model. We focus on the one-time allocation of each query to a suitable LLM, aiming to achieve a trade-off between cost and accuracy.

VIII. CONCLUSION

The substantial cost associated with leveraging LLMs in real-world scenarios poses a significant obstacle to their widespread adoption. In this paper, we propose OptLLM, an effective and efficient framework that automatically assigns queries to suitable LLMs. OptLLM offers a diverse set of non-dominated solutions, from which users can select based on their budget and performance requirements. These solutions span the range from the highest expected accuracy to the lowest cost alternatives. Our experimental results demonstrate that OptLLM outperforms other baseline methods in terms of the effectiveness and efficiency of query assignments.

The source code of OptLLM and all experimental results are available at https://github.com/superyue72/OptLLM.

REFERENCES

[1] E. Kasneci, K. Seßler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Günnemann, E. Hüllermeier *et al.*, "Chatgpt for good? on opportunities and challenges of large language models for education," *Learning and individual differences*, vol. 103, p. 102274, 2023.

[2] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

[3] S. M. Xie, A. Raghunathan, P. Liang, and T. Ma, "An explanation of in-context learning as implicit bayesian inference," *arXiv preprint arXiv:2111.02080*, 2021.

[4] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer, "Rethinking the role of demonstrations: What makes in-context learning work?" in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022, pp. 11 048–11 064.

[5] J. Liu, C. S. Xia, Y. Wang, and L. Zhang, "Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation," *arXiv preprint arXiv:2305.01210*, 2023.

[6] C. S. Xia, Y. Wei, and L. Zhang, "Automated program repair in the era of large pre-trained language models," in *Proceedings of the 45th International Conference on Software Engineering (ICSE)*, 2023.

[7] C. S. Xia and L. Zhang, "Keep the conversation going: Fixing 162 out of 337 bugs for $0.42 each using chatgpt," *arXiv preprint arXiv:2304.00385*, 2023.

[8] M. Šakota, M. Peyrard, and R. West, "Fly-swat or cannon? cost-effective language model choice via meta-modeling," *arXiv preprint arXiv:2308.06077*, 2023.

[9] (2023) How much does it cost to use gpt models? gpt-3 pricing explained. [Online]. Available: https://neoteric.eu/blog/how-much-does-it-cost-to-use-gpt-models-gpt-3-pricing-explained/

[10] L. Chen, M. Zaharia, and J. Zou, "Frugalgpt: How to use large language models while reducing cost and improving performance," *arXiv preprint arXiv:2305.05176*, 2023.

[11] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.

[12] H. Wu, M. Wu, W. Peng, S. Chen, and Z. Feng, "Its: Improved tabu search algorithm for path planning in uav-assisted edge computing systems," in *Proceedings of the 2023 IEEE International Conference on Web Services (ICWS)*. IEEE, 2023, pp. 340–349.

[13] F. U. Haq, D. Shin, and L. Briand, "Efficient online testing for dnn-enabled systems using surrogate-assisted and many-objective optimization," in *Proceedings of the 44th international conference on software engineering (ICSE)*, 2022, pp. 811–822.

[14] R. Cheng, Y. Jin, M. Olhofer *et al.*, "Test problems for large-scale multiobjective and many-objective optimization," *IEEE transactions on cybernetics*, vol. 47, no. 12, pp. 4108–4121, 2016.

[15] C. He, R. Cheng, and D. Yazdani, "Adaptive offspring generation for evolutionary large-scale multiobjective optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 2, pp. 786–798, 2020.

[16] OpenAI, "Gpt-4 technical report," 2023.

[17] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[18] V.-H. Le and H. Zhang, "Log parsing: How far can chatgpt go?" in *Proceedings of the 38th (2023) IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023.

[19] K. Deb, K. Sindhya, and J. Hakanen, "Multi-objective optimization," in *Decision sciences*. CRC Press, 2016, pp. 161–200.

[20] A. Ramirez, J. R. Romero, and S. Ventura, "A survey of many-objective optimisation in search-based software engineering," *Journal of Systems and Software*, vol. 149, pp. 382–395, 2019.

[21] M. Cheikh, B. Jarboui, T. Loukil, and P. Siarry, "A method for selecting pareto optimal solutions in multiobjective optimization," *Journal of Informatics and mathematical sciences*, vol. 2, no. 1, pp. 51–62, 2010.

[22] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability engineering & system safety*, vol. 91, no. 9, pp. 992–1007, 2006.

[23] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust optimization*. Princeton university press, 2009, vol. 28.

[24] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, 2015.

[25] S. Reddy, D. Chen, and C. D. Manning, "Coqa: A conversational question answering challenge," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 249–266, 2019.

[26] A. Sinha and T. Khandait, "Impact of news on the commodity market: Dataset and results," in *Advances in Information and Communication: Proceedings of the 2021 Future of Information and Communication Conference (FICC), Volume 2*. Springer, 2021, pp. 589–601.

[27] J. Welbl, N. F. Liu, and M. Gardner, "Crowdsourcing multiple choice science questions," *arXiv preprint arXiv:1707.06209*, 2017.

[28] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "Tools and benchmarks for automated log parsing," in *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 121–130.

[29] Z. A. Khan, D. Shin, D. Bianculli, and L. Briand, "Guidelines for assessing the accuracy of log message template identification techniques," in *Proceedings of the 44th International Conference on Software Engineering (ICSE)*, 2022, pp. 1095–1106.

[30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[31] C. C. Coello and M. S. Lechuga, "Mopso: A proposal for multiple objective particle swarm optimization," in *Proceedings of the 2002 Congress on Evolutionary Computation (CEC)*, vol. 2. IEEE, 2002, pp. 1051–1056.

[32] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.

[33] J. Chen, V. Nair, and T. Menzies, "Beyond evolutionary algorithms for search-based software engineering," *Information and Software Technology*, vol. 95, pp. 281–294, 2018.

[34] K. Deb and J. Sundar, "Reference point based multi-objective optimization using evolutionary algorithms," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, 2006, pp. 635–642.

[35] N. Beume, B. Naujoks, and M. Emmerich, "Sms-emoa: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007.

[36] Z. Wang, Y.-S. Ong, and H. Ishibuchi, "On scalable multiobjective test problems with hardly dominated boundaries," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 217–231, 2018.

[37] C. Audet, J. Bigeon, D. Cartier, S. Le Digabel, and L. Salomon, "Performance indicators in multiobjective optimization," *European journal of operational research*, vol. 292, no. 2, pp. 397–422, 2021.

[38] J. Blank and K. Deb, "pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.

[39] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2019, pp. 2623–2631.

[40] Z. Jiang, J. Liu, Z. Chen, Y. Li, J. Huang, Y. Huo, P. He, J. Gu, and M. R. Lyu, "Llmparser: A llm-based log parsing framework," *arXiv preprint arXiv:2310.01796*, 2023.

[41] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: A systematic literature review," *arXiv preprint arXiv:2308.10620*, 2023.

[42] D. Sobania, M. Briesch, C. Hanna, and J. Petke, "An analysis of the automatic bug fixing performance of chatgpt," *arXiv preprint arXiv:2301.08653*, 2023.

[43] S. Zong, J. Seltzer, K. Cheng, J. Lin *et al.*, "Which model shall i choose? cost/quality trade-offs for text classification tasks," *arXiv preprint arXiv:2301.07006*, 2023.