A Survey of Optimization Modeling Meets LLMs: Progress and Future Directions

Ziyang Xiao[1], **Jingrong Xie**[1], **Lilin Xu**[1], **Shisi Guan**[1], **Jingyan Zhu**[1], **Xiongwei Han**[2], **Xiaojin Fu**[2], **WingYin Yu**[2], **Han Wu**[2], **Wei Shi**[2], **Qingcan Kang**[2], **Jiahui Duan**[2], **Tao Zhong**[2], **Mingxuan Yuan**[2], **Jia Zeng**[2], **Yuan Wang**[3], **Gang Chen**[1] and **Dongxiang Zhang**[1]*

[1]Zhejiang University [2]Huawei Noah's Ark Lab

[3]School of Business, Singapore University of Social Sciences

{xiaoziyang, zhangdongxiang, cg}@zju.edu.com, {hanxiongwei, rocket.yuwingyin}@huawei.com

Abstract

By virtue of its great utility in solving real-world problems, optimization modeling has been widely employed for optimal decision-making across various sectors, but it requires substantial expertise from operations research professionals. With the advent of large language models (LLMs), new opportunities have emerged to automate the procedure of mathematical modeling. This survey presents a comprehensive and timely review of recent advancements that cover the entire technical stack, including data synthesis and fine-tuning for the base model, inference frameworks, benchmark datasets, and performance evaluation. In addition, we conducted an in-depth analysis on the quality of benchmark datasets, which was found to have a surprisingly high error rate. We cleaned the datasets and constructed a new leaderboard with fair performance evaluation in terms of base LLM model and datasets. We also build an online portal that integrates resources of cleaned datasets, code and paper repository to benefit the community. Finally, we identify limitations in current methodologies and outline future research opportunities.

Input: problem description

Power generation units are grouped into three distinct types, with different characteristics for each type (**power output**, **cost per megawatt hour**, **startup cost**, etc.). A unit can be on or off, with a startup cost associated with **transitioning from off to on**, and power output that can fall anywhere between **a specified minimum and maximum value** when the unit is on. A **5-hour time horizon** is with an expected total power demand for each hour. The model decides which units to turn on, and when, in order to **satisfy demand for each time period**. The model also captures a **reserve requirement**, where the selected power plants must be capable of increasing their output, while still respecting their maximum output, in order to cope with the situation where **actual demand exceeds predicted demand**.

⇩

Output: modeling result

Sets: $i \in I, t \in T = \{1, 2, \ldots, 5\}$

Parameters: $P_i^{min}, P_i^{max}, c_i, c_i^{SU}, D_i, R_t$

Variables: $u_{i,t}, \in \{0,1\}, x_{i,t} \geq 0, y_{i,t} \in \{0,1\}$

Constraints:

Power Limits for On Units: $P_i^{min} u_{i,t} \leq x_{i,t} \leq P_i^{max} u_{i,t}, \quad \forall i \in I, \forall t \in T$

Demand Satisfaction: $\sum_{i \in I} x_{i,t} \geq D_t, \quad \forall t \in T$

Reserve Requirement: $\sum_{i \in I} (P_i^{max} - x_{i,t}) \geq R_t$

Startup Definition: $y_{i,t} \geq u_{i,t} - u_{i,t-1}, \quad \forall i \in I, \forall t \in T$

Objective: *minimize* $\sum_{t=1}^{T} \sum_{i \in I} (c_i x_{i,t} + c_i^{SU} y_{i,t})$

Figure 1: An example of an optimization modeling task. The orange text in the problem description implies domain-specific terminology, and the green text denotes implicit constraints.

1 Introduction

Optimization modeling aims to mathematically model complex decision-making problems that arise from a wide spectrum of industry sectors, including supply chain management [et al., 1997], healthcare resource allocation [Delgado *et al.*, 2022], air traffic flow management [et al., 2000] and portfolio optimization [Mokhtar *et al.*, 2014]. Despite its potential to enhance operational efficiency, there exists an expertise barrier that limits the broader adoption of optimization tools. According to a survey of Gurobi users, 81% of them hold advanced degrees, with nearly half specializing in operations research [Gurobi Optimization, 2023].

To automate the procedure and reduce the dependence on domain-specific modeling experts, NL4Opt (Natural Language for Optimization) [Ramamonjison *et al.*, 2023] has emerged as an attractive but challenging NLP task. Its objective is to translate the text description of an OR problem into math formulations for optimization solvers. Figure 1 illustrates an instance of NL4Opt task. It transforms an input problem text into a formal mathematical model, including variables, constraints, and objective functions. The problem is challenging because the text descriptions of optimization problems often require a large amount of domain-specific knowledge to understand terminologies, such as "megawatt hour", "startup cost", and "5-hour time horizon", highlighted in orange text. Moreover, these descriptions may contain numerous implicit constraints that need to be inferred by human experts. Solving the problem of automatic optimization modeling can enhance time and cost efficiency while enabling access for users without deep optimization expertise.

Recently, large language models (LLMs) offer a promising way to make optimization more accessible. They can understand the complicated text descriptions — identity the optimization objective and extract the decision variables and constraints. Consequently, they automatically build the mathematical model and generate the code. Numerous works have been proposed in this rapidly expanding field:

*Corresponding Author.

- **Domain-specific LLM**. Representative works such as ORLM [Tang *et al.*, 2024] and LLMOPT [Jiang *et al.*, 2024] take advantage of data synthesis and instruction tuning to enhance the capability of base model for optimization modeling.
- **Advanced Inference Framework**: Various reasoning frameworks have emerged, include multi-agent systems (e.g. Chain-of-Experts [Xiao *et al.*, 2024] and OptiMUS [AhmadiTeshnizi *et al.*, 2024]) and chain-of-thought variants (e.g. Tree of Thoughts [Yao *et al.*, 2023], Autoformulation [Astorga *et al.*, 2024]).
- **Benchmark Datasets and Evaluation.** There have been multiple benchmark datasets released, such as IndustryOR [Tang *et al.*, 2024], NL4Opt [Ramamonjison *et al.*, 2023] and MAMO [Huang *et al.*, 2024]. However, these datasets vary significantly in quality, and evaluation methods lack standardization across different studies.

Thus, it is of high necessity to present a just-in-time survey to summarize the progress and indicate possible future research directions. In this paper, we propose the first systematic review of optimization modeling in the era of LLMs. As shown in Figure 3, we present a detailed taxonomy of the various methodologies employed to harness the power of LLMs for optimization modeling. Besides, we noticed that existing benchmark datasets are associated with high error rates and performed data cleaning to enhance quality. We constructed a new leader-board with fair comparison in terms of base model and benchmark datasets, and deliver new insights of performance evaluation. To benefit the community, these datasets and implementation code are accessible from our online portal[1].

2 Background

2.1 Problem Definition

Optimization modeling transforms a problem description in natural language $\mathcal{P}$ into a model $\mathcal{M}$. Mathematically, an optimization model is defined by an objective and a set of constraints, as shown in Equation 1.

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, ..., m \\ & h_j(\mathbf{x}) = 0, \quad j = 1, ..., p \end{aligned} \tag{1}$$

Here, $\mathbf{x}$ is the vector of decision variables, $f(\mathbf{x})$ denotes the objective function, $g_i(\mathbf{x})$ and $h_j(\mathbf{x})$ represent the inequality and equality constraints respectively.

2.2 Abstract Model and Concrete Model

In practice, optimization models can be categorized into two types: *abstract models* and *concrete models*. A model whose parameters are denoted by mathematical symbols called a abstract model, while a model whose parameters are specified by numerical values is called a concrete model. Correspondingly, optimization modeling can be divided into two types:

[1]https://anonymous.4open.science/r/LLM4OR-2781

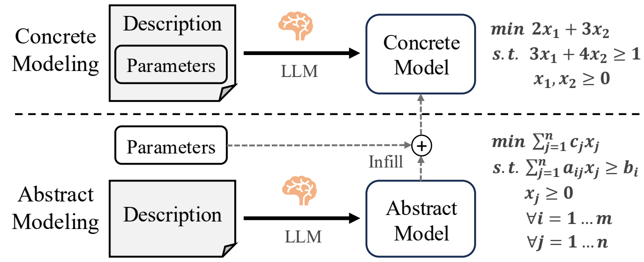

Figure 2: Comparison between concrete and abstract models. The right part illustrates a linear programming formulation example.

concrete modeling and *abstract modeling*, as illustrated in Figure 2. Concrete modeling directly translates a problem description containing numerical parameters into a concrete model. In contrast, abstract modeling follows a model-data separation approach where the problem description only contains the model structure, with parameters are provided separately at a later stage.

3 Technical Stack of Optimization Modeling

This section presents a typical technical stack for applying LLMs to optimization modeling. The pipeline consists of four key steps: (1) data preparation and LLM fine-tuning; (2) inference; (3) benchmarking; and (4) evaluation. Figure 3 shows the representative works in each step of this pipeline.

3.1 Data Synthesis and Fine-tuning

Data Synthesis Methods

It is a common practice to fine-tune language models for specialized domains such as optimization modeling. However, fine-tuning requires a substantial amount of high-quality training data. In the field of optimization modeling, data availability is limited due to the scarcity of problem sources and the high cost of problem annotation. To address this challenge, current approaches employ data synthesis to generate training datasets. Formally, the data synthesis process can be defined as $seed \rightarrow \{\mathcal{P}', \mathcal{M}'\}$, where $\mathcal{P}'$ represents the generated problem description, $\mathcal{M}'$ denotes the corresponding modeling and $seed$ is the seed data of generation process. Depending on the primary focus of the generation process, existing works can be divided into two approaches: problem-centric and model-centric.

Problem-centric The problem-centric approach involves two steps. First, it takes an existing problem $\mathcal{P}$ and generates a new problem $\mathcal{P}'$. Then, it automatically produces the corresponding model $\mathcal{M}'$ using LLMs, with human experts filtering out low-quality annotations. In the first step, OR-Instruct [Tang *et al.*, 2024] devises three primitives to increase the diversity of a problem: modifying constraints and objectives, rephrasing questions for scenario diversity, and adding multiple modeling techniques for linguistic diversity. Besides, the data augmentation pipeline introduced in LLMOPT [Jiang *et al.*, 2024] proposes seven primitives to further enhance diversity by incorporating new instructions on modifying the problem type and scenario. Beyond diversity, Evo-Step-Instruct [Wu *et al.*, 2025] introduces complexity

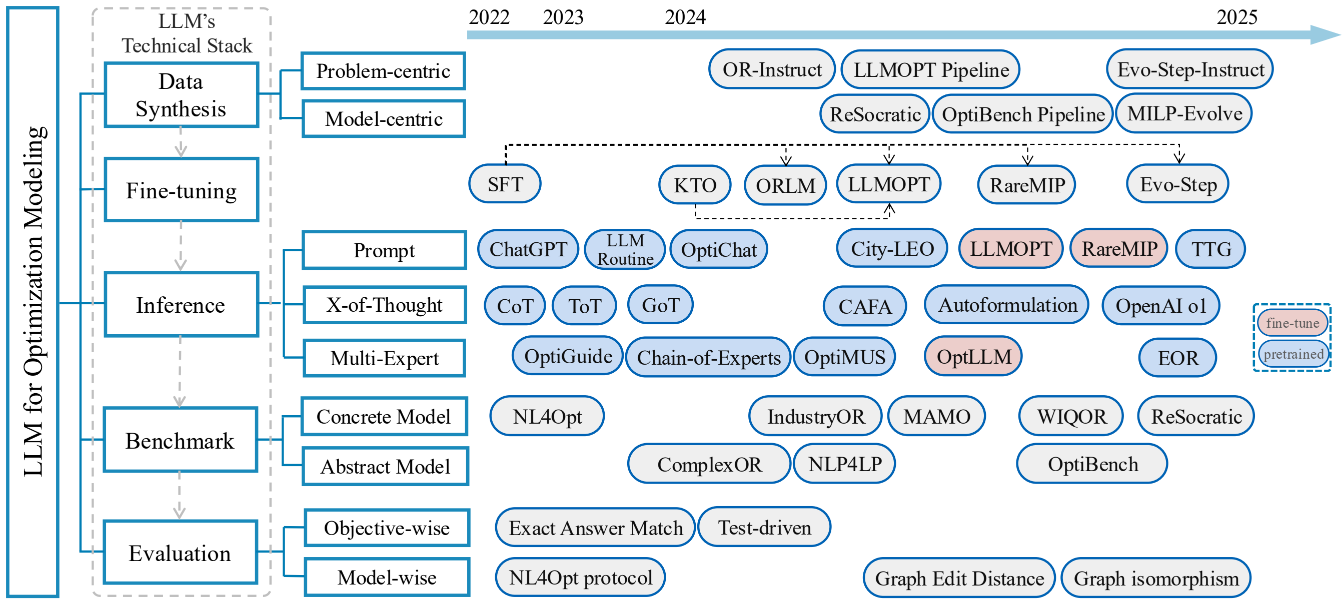

Figure 3: Left: Taxonomy of LLMs-based optimization modeling, organized according to the LLMs' technical stack. Right: Representative works for each category are presented in chronological order. The dashed arrows indicates where later works build upon techniques proposed in earlier studies.

as an additional dimension, along with a method to modify constraints, parameters, and objectives progressively to create more challenging problems. However, the problem-centric approach is limited in its ability to escalate complexity. As the complexity grows, generating a valid solution model becomes more difficult, leading to a higher risk of errors in annotations. To address this, Evo-Step-Instruct employs a sophisticated workflow to filter out unqualified data.

Model-centric The model-centric method adopts a different approach by first generating an augmented model $\mathcal{M}'$ and then crafting a corresponding problem description $\mathcal{P}'$. Compared to problem-centric approach, this methodology provides more fine-grained control over instance types and difficulty while ensuring the labeled model remains solvable. MILP-Evolve [Li *et al.*, 2024] pioneer this approach by using existing model code as input, prompting LLMs to add, delete, or mutate code elements to evolve new models. However, since this work focus solely on generating MILP instances, it does not incorporate the problem description generation step. Similarly, OptiBench [Wang *et al.*, 2024b] prioritizes model code generation but differs by using simple seeds such as model types (e.g., MILPs or MIPs), problem classes (e.g., knapsack problem), and domains (e.g., cargo loading) instead of existing models. This approach enables better control over dataset distribution. After code generation, LLMs transform the solver code into detailed word description. Another work, ReSocratic [Yang *et al.*, 2025], extends this paradigm by defining models as semantically rich formatted demonstrations. Unlike pure code, these demonstrations incorporate structured data for variables, objective functions, and constraints, along with their natural language descriptions, resulting in richer semantic content. ReSocratic employs a multi-step sampling method with LLMs to first generate such documentation, which is then transforms into comprehensive problem descriptions as data points.

Fine-tuning Methods

Once the data is prepared, the next step is to fine-tune open-source LLMs to enhance their optimization modeling capabilities. Fine-tuning typically involves two key steps: model instruction training and model alignment. Existing works [Tang *et al.*, 2024; Wang *et al.*, 2024a; Wu *et al.*, 2025] focus on the first step by applying supervised fine-tuning (SFT) with synthetic data. Meanwhile, LLMOPT [Jiang *et al.*, 2024] introduces Kahneman-Tversky Optimization (KTO) [Ethayarajh *et al.*, 2024], which further aligns model outputs with human preferences and helps mitigate biases. Despite these advancements, there remains a notable gap in research exploring innovative training techniques and paradigms for optimization modeling, highlighting the need for further investigation.

3.2 Inference

During the inference stage, trained LLMs translate the problem description $\mathcal{P}$ into the modeling result $\mathcal{M}$, which can be either executable code or structured documentation. As with other domain-specific tasks, prompt engineering is a straightforward yet effective method for applying LLMs to optimization modeling problems. Moreover, as illustrated in Figure 4, the capabilities of LLMs can be enhanced along two dimensions. One approach involves inference-time scaling, which encourages LLMs to generate additional intermediate reasoning steps (referred to as "X-of-thought"). The other approach scales up the single LLM to the LLM-based multi-agent system (referred to as "multi-expert").

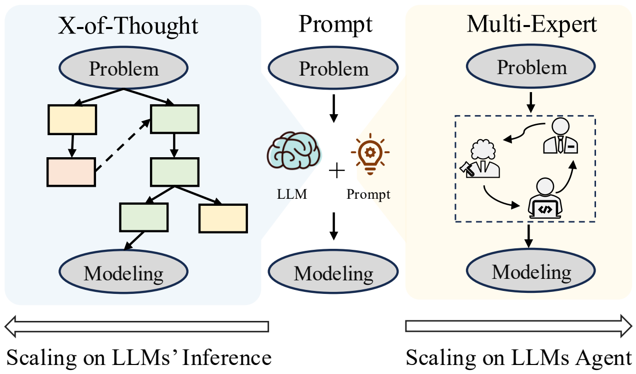

Figure 4: Three types of inference methods.

Prompt At the advent of ChatGPT, NL4Opt [Ramamonjison *et al.*, 2023] pioneers the use of ChatGPT for solving optimization modeling problems. This work introduces a simple prompt template comprising three components: the problem description, task instructions, and format control. Since then, many studies have leveraged LLMs for optimization modeling via prompt engineering, which varies between training-based and training-free approaches.

For training-based approaches, prompts are primarily designed for format control, helping the model generate output that conforms to the training set's label format. For example, ORLM [Tang *et al.*, 2024] prompts the model to first produce a plain-text description of the model and then generate the corresponding code. Similarly, LLMOPT [Jiang *et al.*, 2024] instructs the trained LLM to output a five-element formulation, while RareMIP [Wang *et al.*, 2024a] prompts the model to generate LaTeX code that details the model-building process. Additionally, TTG [JU *et al.*, 2024] uses prompts to produce JSON output, which can be easily parsed into a symbolic model suitable for solvers.

For training-free approaches, the goal shifts toward infusing richer domain knowledge into LLMs through the prompt. For instance, OptiChat [Chen *et al.*, 2023] provides the LLM with step-by-step instructions that mimic the guidance of an optimization expert, thereby equipping the model with domain-specific insights. It also employs few-shot learning by supplying examples of optimization problems paired with expert solutions. Similarly, City-LEO [Jiao *et al.*, 2024] adopts in-context learning techniques to construct its LLM pipeline, and another work [Li *et al.*, 2023b] incorporates prior knowledge into prompt design to further enhance LLM performance on routine tasks.

Although prompt engineering can be rapidly implemented, it only scratches the surface of what LLMs can achieve in tackling complex modeling problems. Much of their potential remains untapped. The following sections introduce two promising directions to unleash this power: X-of-thought and Multi-Agent.

X-of-Thought To enhance the reasoning capabilities of LLMs and tackle increasingly complex optimization modeling problems, researchers have begun exploring LLMs' potential during inference time. The chain-of-thought (CoT) approach [Wei *et al.*, 2022] pioneers LLM reasoning by encouraging the model to think step-by-step, effectively bridging logical gaps during inference. Building on this foundation, Tree of Thoughts (ToT) [Yao *et al.*, 2023] and Graph of Thoughts (GoT) [Besta *et al.*, 2024] further enhance reasoning by employing tree- and graph-structured exploration of intermediate thoughts. Collectively, these approaches are known as "X-of-thought" [Chu *et al.*, 2024]. Although originally designed for general reasoning tasks, these methods have also been successfully applied to optimization modeling [Xiao *et al.*, 2024].

Subsequently, several X-of-thought methods tailored for optimization modeling have emerged. For instance, CAFA [haoxuan deng *et al.*, 2024] defines the inference process as a linear sequence of steps that explicitly captures the reasoning required for modeling. Furthermore, Autoformulation [Astorga *et al.*, 2024] treats the modeling process as a Monte Carlo Tree Search, where each level of the tree corresponds to a specific modeling step—sequentially addressing parameters and decision variables, the objective function, equality constraints, and inequality constraints. This framework integrates an LLM with two key components: (1) a dynamic formulation hypothesis generator responsible for exploring the Monte Carlo Tree, and (2) an evaluator that provides feedback on the correctness of solutions at the leaf nodes.

Recently, OpenAI's o1 [OpenAI, 2024] has attracted significant attention for its exceptional reasoning capabilities in tackling complex problems, including optimization modeling. It explicitly integrates an extended internal chain-of-thought into its inference process, representing a promising direction that merits further investigation.

Multi-Expert Another approach to scaling language models for complex reasoning is the use of multi-agent collaboration systems [Qian *et al.*, 2024]. In the field of optimization modeling, LLMs are adapted to mimic human experts and collaborate to complete the entire modeling process. This system is referred to as multi-expert system. Early examples include OptiMUS [AhmadiTeshnizi *et al.*, 2024] and Chain-of-Experts (CoE) [Xiao *et al.*, 2024]. Both systems predefine a set of LLM-based experts, with two key roles: a formulator for optimization modeling and a programmer for code generation. They differ in how they manage the workflow: OptiMUS uses a predefined workflow to engage experts in collaborative problem-solving, while CoE employs a special expert called the "Conductor" to orchestrate the entire process. Additionally, CoE introduces a system-level reflection mechanism to adjust answers based on external feedback.

Subsequently, the OptiGuide framework [Li *et al.*, 2023a] is proposed with a focus on improving the reliability and readability of modeling results. Specifically, it incorporates a safeguard agent to address potential output errors and an interpreter that generates human-readable explanations of both the modeling results and the solver's solution. Similarly, OptLLM [Zhang *et al.*, 2024a] includes a diagnostic agent that reformulates the modeling output based on internal feedback when code fails syntax tests. Explainable Operations Research (EOR) [Zhang *et al.*, 2025] adopts a similar framework to OptiGuide but focuses on what-if analysis for optimization modeling, in which way it can evaluate the impact of complex constraint changes on decision-making.

Compared to X-of-Thought, the merits of multi-expert methods lie in their interpretable intermediate results and better capability of safeguarding against potential errors hidden in the output, making them a popular direction for future research.

3.3 Benchmarks

To evaluate performance of LLMs-based optimization modeling methods, several benchmarks have been proposed. As discussed in Section 2, these benchmarks can be categorized into two types: concrete modeling and abstract modeling.

Concrete Modeling NL4Opt [Ramamonjison *et al.*, 2023] is the first optimization modeling benchmark proposed in a competition, featuring a test set of 289 instances. However, NL4Opt primarily focuses on simple optimization modeling problems. To address the need for more challenging cases, IndustryOR [Tang *et al.*, 2024] is introduced, consisting of 100 real-world industry cases. IndustryOR covers a variety of problem types—including mixed integer programming and nonlinear integer programming—and features descriptions with or without tabular data, thereby increasing problem complexity. However, IndustryOR suffers from quality control issues, which result in a high error rate. To overcome this limitation, ReSocratic [Yang *et al.*, 2025] introduces a comprehensive framework that applies multiple filters to remove erroneous cases, efficiently improving dataset quality and expanding the test set to 605 instances. While the annotations in these three benchmarks focus solely on providing an objective as final answer, MAMO [Huang *et al.*, 2024] goes a step further by including optimal variable information, offering additional perspectives for evaluating model correctness. Note that MAMO also categorize problems into three classes: EasyLP, ComplexLP and ODE. Our study primarily focuses on the former two categories. All these benchmarks are designed for end-to-end modeling tasks. WIQOR [Parashar *et al.*, 2025], on the other hand, employs what-if analyses to assess performance, providing insights into whether LLMs possess a deeper understanding of the modeling process.

Abstract Modeling ComplexOR [Xiao *et al.*, 2024] is an abstract modeling benchmark introduced in the CoE, containing 37 instances collected from both industrial and academic scenarios. In ComplexOR, numerical parameter values are separated from the problem descriptions. NLP4LP [AhmadiTeshnizi *et al.*, 2024] is another early abstract modeling benchmark, extending the number of instances to 269. Although both datasets are relatively small, the subsequent release of OptiBench [Wang *et al.*, 2024b] offers a larger collection of 816 instances following a model-data separation format.

While most existing research focuses on concrete modeling, it is worth noting that abstract modeling is more common in industrial scenarios, where an abstract model, once constructed, can be reused multiple times with different concrete parameters. However, due to the inherent complexity of abstract modeling, high-quality benchmarks remain scarce.

Dataset	Size	Complexity	Error Rate
NL4Opt	289	5.59	$\geq 26.4\%$
IndustryOR	100	**14.06**	$\geq$ **54.0%**
EasyLP	652	7.12	$\geq 8.13\%$
ComplexLP	211	**13.35**	$\geq 23.7\%$
ReSocratic	605	7.45	$\geq 16.0\%$
NLP4LP	269	5.58	$\geq 21.7\%$
ComplexOR	37	5.98	$\geq 24.3\%$

Table 1: Quality statistics of optimization modeling benchmarks.

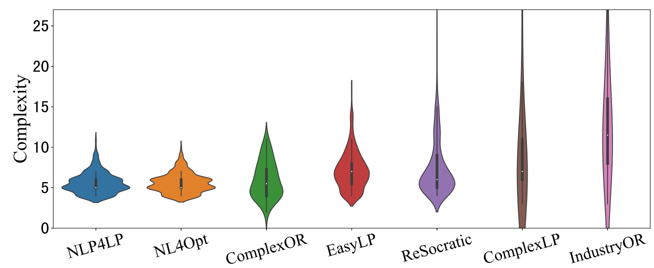

Figure 5: Statistics of complexity distribution for each benchmark visualized using a violin plot. X-axis shows different benchmarks, and y-axis shows the complexity indicator.

Analysis on Benchmarks

To assess the quality of current benchmarks, we conduct an in-depth analysis of them. The results are shown in Table 1 and Figure 5. We evaluate three key statistical features: (1) Data Size: the number of instances in the benchmark's test set; (2) Complexity: for each problem, we first use standard prompting to generate a model and then use the number of variables and constraints in the model to indicate its complexity; (3) Error Rate: to compute this metric, we have 11 human experts manually identify errors in the problems, and each error case is cross-validated by at least three different experts.

According to our results, we obtain several key findings. First, in current benchmarks, the error rate is relatively high. As shown in Table 1, except for EasyLP in MAMO, the error rates of other benchmarks exceed 15%, with IndustryOR even reaching as high as 54%, indicating that these benchmarks are not entirely reliable for evaluation. The errors can be caused by three main factors: (1) logical errors in problem descriptions, such as unbounded constraints; (2) poorly defined parameters that lead to unsolvable models; and (3) incorrect ground truth data. To address these issues, we manually filter all error cases and compile a unified, cleaned collection of optimization modeling benchmarks to facilitate future research.

Takeaway #1: The high error rates in current benchmarks undermine their reliability. We curate a cleaned and unified set of optimization modeling benchmarks to facilitate more accurate evaluation.

Second, our analysis of benchmark complexity reveals that current benchmarks mainly cover simple cases and exhibit

Methods	NL4Opt	IndustryOR	EasyLP	ComplexLP	NLP4LP	ReSocratic	ComplexOR
Standard	61.2%	38.1%	70.3%	57.7%	73.6%	48.4%	42.9%
CoT	62.2%	40.5%	49.5%	42.3%	74.7%	43.6%	39.2%
Chain-of-Experts	66.7%	31.2%	**94.4**%	50.6%	**87.4**%	**71.2**%	**57.1**%
CAFA	68.1%	41.1%	71.2%	44.5%	50.0%	40.1%	46.4%
ORLM-LLaMA-3 8B	**73.8**%	**42.9**%	90.4%	**59.5**%	76.4%	61.8%	50.0%

Table 2: Performance comparison of existing fully open-source methods on cleaned benchmarks in a unified setting (use GPT-4o for training-free methods and use accuracy as metric). All results are reproduced using our standardized evaluation method.

an imbalanced distribution. As shown in Table 1, NL4Opt, NLP4LP, and ComplexOR clearly present low levels of challenge. Figure 5 further shows that most instances concentrate at the simple and medium complexity levels, with instances of complexity greater than 10 being very scarce, which indicates a lack of truly complex cases.

Takeaway #2: Existing benchmarks are dominated by simple and moderate problems, with very few challenging cases. This imbalance highlights the need for more high-complexity benchmarks.

3.4 Evaluation

Evaluating optimization models can be challenging because it is often difficult to determine the correctness of the results. There are two primary approaches exist. The first is objective-wise evaluation, which focuses exclusively on the final objective value produced by the solver. The second is model-wise evaluation, where the generated model is directly compared against a ground truth model.

Objective-wise In objective-wise evaluation, the focus is solely on the correctness of the final objective. This approach originates from mathematical word problems [Cobbe *et al.*, 2021], where LLMs directly generate a final answer and compare it to the ground truth, referred as the exact answer match method. However, in optimization modeling, LLMs produce a model rather than a final answer. To address this, a test-driven method is introduced in Chain-of-Experts (CoE) [Xiao *et al.*, 2024], where a solver takes the generated model (with specified parameters), computes the final objective, and compares it to the ground truth. Subsequent works, including ORLM [Tang *et al.*, 2024], CAFA [haoxuan deng *et al.*, 2024], and Autoformulation [Astorga *et al.*, 2024], adopt this same test-driven method.

Model-wise While objective-wise evaluation is straightforward, it has a notable limitation: a correct objective value does not necessarily guarantee a correct model. To address this, model-wise evaluation is introduced. NL4Opt [Ramamonjison *et al.*, 2023] pioneers a protocol that converts modeling results into a canonical formulation, where the coefficients of the objective function and constraints are extracted into matrices and then are compared with ground truth. Although this method captures model correctness comprehensively, it provides only a binary metric and fails to reflect the degree of correctness, which is essential for fine-grained assessments. To overcome this limitation, a graph-based evaluation method [Xing *et al.*, 2024] is proposed, representing modeling results as a graph and using graph edit distance to produce a continuous correctness score between 0 to 1. Building on this, a modified graph isomorphism testing algorithm [Wang *et al.*, 2024b] offers even more precise evaluation, with theoretical guarantees ensuring the correctness of its comparisons.

Evaluation Result of Existing Methods
In this survey, we observe that the reported evaluation results across existing works often exhibit inconsistencies, making fair comparisons challenging. These discrepancies arise primarily from three factors.

- **Choice of Base Model**: Researchers use different commercial LLMs as base model. For example, Chain-of-Experts employs GPT-3.5, whereas Autoformulation uses GPT4-mini, due to the rapid evolution of LLMs.
- **Dataset Preprocessing Approaches**: Different strategies are used for handling incorrect samples and decimal precision, resulting in varying preprocessing pipelines.
- **Evaluation Metrics**: Metrics also vary: ORLM reports micro and macro average accuracy, whereas Chain-of-Experts focuses on compile error rates.

These factors collectively contribute to the difficulty of establishing a consistent leader-board for optimization modeling methods.

To address the challenge of inconsistent evaluations and create a fair comparison, we adopt a unified setting to assess all fully open-source optimization modeling methods on our cleaned benchmarks. Specifically, we employ the cutting-edge commercial LLM *gpt-4o-2024-08-06* as the base model for all training-free methods. We report accuracy as the evaluation metric, as it is the most widely accepted measure.

Regarding optimization modeling methods, we strive to evaluate every fully open-source approach. However, many methods mentioned in Subsection 3.2 remain closed-source, including LLMOPT [Jiang *et al.*, 2024], RareMIP [Wang *et al.*, 2024a], Autoformulation [Astorga *et al.*, 2024], OptLLM [Zhang *et al.*, 2024a], LLM Routine [Li *et al.*, 2023b], City-LEO [Jiao *et al.*, 2024], and TTG [JU *et al.*, 2024]. Three other methods, including OptiChat [Chen *et al.*, 2023], OptiGuide [Li *et al.*, 2023a], and EOR [Zhang *et al.*, 2025], are interactive and thus not directly comparable to end-to-end approaches. Additionally, OptiMUS [AhmadiTeshnizi *et al.*, 2024] requires a preprocessing step that is unavailable for most benchmarks, leading us to exclude it. For broader comparison, we include two general reasoning strategies, including standard prompting and chain-of-thought prompting, as baselines.

> **Takeaway #3:** The evaluation results reported in existing works lack a unified standard. And the open-source landscape in optimization modeling remains limited.

Table 2 shows the overall results, revealing several key observations. First, Chain-of-Experts and ORLM are two competitive methods in optimization modeling. While Chain-of-Experts works well for simpler tasks, ORLM surpasses it on more complex datasets such as IndustryOR and ComplexLP, indicating that trained models may be more effective in challenging scenarios. Second, contrary to popular belief, CoT does not always yield better results than standard prompting. On certain datasets, it even leads to a noticeable drop in performance, supporting the idea that CoT should be applied selectively [Sprague *et al.*, 2024]. Finally, the performance of CAFA is comparable to CoT. This is likely because CAFA can be seen as a specialized form of CoT prompting.

> **Takeaway #4:** Three key findings: (1) Chain-of-Experts and ORLM emerge as the most competetive frameworks; (2) CoT prompting does not always outperform standard prompting; (3) The performance of CAFA resembles that of a specialized CoT strategy.

4 Online Portal for Optimization Modeling

We develop a website portal that integrates the resources of LLM-based optimization modeling and provides great convenience for researchers to follow the topic. First, we provide the download links for both original and cleaned version of benchmark datasets. Second, we collect and publish the implementation of existing solutions and provide a leader-board to report their performance on the benchmarks. Thirdly, we continue to update the latest research papers on this promising research domain. We believe such an integrated portal brings significant benefit for the community.

5 Challenges and Future Directions

5.1 Reasoning Model for Optimization Modeling

A prominent trend in recent LLM research is enhancing the reasoning capabilities of base models. The release of OpenAI o1 [OpenAI, 2024] demonstrates impressive performance on complex mathematical tasks. However, these advances have not yet been transferred to optimization modeling. One key obstacle is that training a reasoning model heavily relies on long chain-of-thought data, which is expensive and difficult to annotate in the context of optimization modeling. To bridge this gap, Deepseek R1 Zero [et al., 2025] proposed a promising alternative by using pure reinforcement learning for training, enabling LLMs to develop reasoning capabilities without requiring supervised chain-of-thought annotations. This reinforcement learning strategy is also promising for optimization modeling, where the modeling process can be formulated as a Markov Decision Process and solver feedback can be used as reward to train the reasoning model.

5.2 Explainable Modeling Processes

The black-box nature of LLMs, most existing studies treat optimization modeling as an end-to-end process. However, the explainability of this process is also crucial for real-world applications, as it allows experts to effectively debug, modify, and understand the generated models. Recent work like Explainable Operations Research [Zhang *et al.*, 2025] has made progress in this direction by developing methods to evaluate how modeling decisions impact outcomes. More research efforts to develop a trustworthy and user-friendly modeling framework are encouraged.

5.3 Domain Knowledge Injection

The optimization modeling process relies heavily on domain knowledge. As demonstrated by a research [Runnwerth *et al.*, 2020], much of this specialized knowledge, including conception and empirical insights, can be stored in a knowledge graph. Incorporating such domain-specific knowledge into LLMs to aid the modeling process remains a significant challenge. A recent work [Zhang *et al.*, 2024b] uses rule mining to construct training data from knowledge graphs and introduces a learning method to integrate knowledge graphs with LLMs, offering a promising pathway for advancing the field of optimization modeling.

5.4 Human-in-the-Loop Modeling

Existing inference approaches have primarily focused on the modeling capabilities of LLMs and have not explored human intervention during the inference process. Recent research indicates that LLMs can proactively query humans for domain-specific knowledge when needed [Pang *et al.*, 2024]. These characteristics offer an opportunity to open up a new paradigm, human-in-the-loop modeling, where human experts contribute external knowledge, clarifications, and insights at critical points. To develop such a collaborative system, we need to overcome the following challenges. First, effective mechanisms are needed to identify when human intervention is required, since LLMs themselves lack this capability. Second, an effective human-in-the-loop framework should ensure that humans can seamlessly integrate their expertise into the inference process.

6 Conclusion

This survey provides a timely overview of the rapid progress in applying LLMs to optimization modeling. We present a thorough taxonomy of existing works across data synthesis, model fine-tuning, inference approaches, benchmarks, and evaluation methods, offering a structured understanding of the technical stack. We also highlight persisting challenges, particularly in data quality and evaluation protocols, that hinder reliable performance comparisons. To address these gaps, we evaluate current open-source methods on a set of cleaned and standardized benchmarks, revealing several key insights. Building on these findings and the latest advances, we propose promising directions to inspire further research in this emerging field.

References

[AhmadiTeshnizi *et al.*, 2024] Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. Optimus: Scalable optimization modeling with (MI)LP solvers and large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[Astorga *et al.*, 2024] Nicolás Astorga, Tennison Liu, Yuanzhang Xiao, and Mihaela van der Schaar. Autoformulation of mathematical optimization models using llms, 2024.

[Besta *et al.*, 2024] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michał Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, Mar 2024.

[Chen *et al.*, 2023] Hao Chen, Gonzalo E. Constante-Flores, and Can Li. Diagnosing infeasible optimization problems using large language models. *CoRR*, abs/2308.12923, 2023.

[Chu *et al.*, 2024] Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. Navigate through enigmatic labyrinth A survey of chain of thought reasoning: Advances, frontiers and future. pages 1173–1203. Association for Computational Linguistics, 2024.

[Cobbe *et al.*, 2021] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

[Delgado *et al.*, 2022] Erwin J Delgado, Xavier Cabezas, Carlos Martin-Barreiro, Víctor Leiva, and Fernando Rojas. An equity-based optimization model to solve the location problem for healthcare centers applied to hospital beds and covid-19 vaccination. *Mathematics*, 10(11):1825, 2022.

[et al., 1997] Bramel et al. *The Logic of Logistics: Theory, Algorithms, and Applications for Logistics Management*. Springer, 1997.

[et al., 2000] De Matos et al. The application of operational research to european air traffic flow management–understanding the context. *European Journal of Operational Research*, 123(1):125–144, 2000.

[et al., 2025] DeepSeek-AI et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[Ethayarajh *et al.*, 2024] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. KTO: model alignment as prospect theoretic optimization. *CoRR*, abs/2402.01306, 2024.

[Gurobi Optimization, 2023] Gurobi Optimization. 2023 state of mathematical optimization report, 2023.

[haoxuan deng *et al.*, 2024] haoxuan deng, Bohao Zheng, Yirui Jiang, and Trung Hieu Tran. CAFA: Coding as autoformulation can boost large language models in solving linear programming problem. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024.

[Huang *et al.*, 2024] Xuhan Huang, Qingning Shen, Yan Hu, Anningzhe Gao, and Benyou Wang. Mamo: a mathematical modeling benchmark with solvers, 2024.

[Jiang *et al.*, 2024] Caigao Jiang, Xiang Shu, Hong Qian, Xingyu Lu, Jun Zhou, Aimin Zhou, and Yang Yu. LLMOPT: learning to define and solve general optimization problems from scratch. *CoRR*, abs/2410.13213, 2024.

[Jiao *et al.*, 2024] Zihao Jiao, Mengyi Sha, Haoyu Zhang, Xinyu Jiang, and Wei Qi. City-leo: Toward transparent city management using LLM with end-to-end optimization. *CoRR*, abs/2406.10958, 2024.

[JU *et al.*, 2024] Da JU, Song Jiang, Andrew Cohen, Aaron Foss, Sasha Mitts, Arman Zharmagambetov, Brandon Amos, Xian Li, Justine T. Kao, Maryam Fazel-Zarandi, and Yuandong Tian. To the globe (TTG): towards language-driven guaranteed travel planning. *CoRR*, abs/2410.16456, 2024.

[Li *et al.*, 2023a] Beibin Li, Konstantina Mellou, Bo Zhang, Jeevan Pathuri, and Ishai Menache. Large language models for supply chain optimization. *CoRR*, abs/2307.03875, 2023.

[Li *et al.*, 2023b] Ran Li, Chuanqing Pu, Junyi Tao, Canbing Li, Feilong Fan, Yue Xiang, and Sijie Chen. Llm-based frameworks for power engineering from routine to novel tasks, 2023.

[Li *et al.*, 2024] Sirui Li, Janardhan Kulkarni, Ishai Menache, Cathy Wu, and Beibin Li. Towards foundation models for mixed integer linear programming. *CoRR*, abs/2410.08288, 2024.

[Mokhtar *et al.*, 2014] Mazura Mokhtar, A Shuib, and D Mohamad. Mathematical programming models for portfolio optimization problem: A review. *International Journal of Mathematical and Computational Sciences*, 8(2):428–435, 2014.

[OpenAI, 2024] OpenAI. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/, 2024. [Accessed 19-09-2024].

[Pang *et al.*, 2024] Jing-Cheng Pang, Heng-Bo Fan, Pengyuan Wang, Jia-Hao Xiao, Nan Tang, Si-Hang Yang, Chengxing Jia, Sheng-Jun Huang, and Yang Yu. Empowering language models with active inquiry for deeper understanding, 2024.

[Parashar *et al.*, 2025] Aditya Parashar, Natalia Kosilova, and Ari Kobren. WIQOR: A dataset for what-if analysis of operations research problems, 2025.

[Qian *et al.*, 2024] Chen Qian, Zihao Xie, Yifei Wang, Wei Liu, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Scaling large-language-model-based multi-agent collaboration, 2024.

[Ramamonjison *et al.*, 2023] Rindranirina Ramamonjison, Timothy T. L. Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, and Yong Zhang. Nl4opt competition: Formulating optimization problems based on their natural language descriptions. *CoRR*, abs/2303.08233, 2023.

[Runnwerth *et al.*, 2020] Mila Runnwerth, Markus Stocker, and Sören Auer. *Operational Research Literature as a Use Case for the Open Research Knowledge Graph*, page 327–334. Springer International Publishing, 2020.

[Sprague *et al.*, 2024] Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning, 2024.

[Tang *et al.*, 2024] Zhengyang Tang, Chenyu Huang, Xin Zheng, Shixi Hu, Zizhuo Wang, Dongdong Ge, and Benyou Wang. ORLM: training large language models for optimization modeling. *CoRR*, abs/2405.17743, 2024.

[Wang *et al.*, 2024a] Teng Wang, Wing-Yin Yu, Ruifeng She, Wenhan Yang, Taijie Chen, and Jianping Zhang. Leveraging large language models for solving rare MIP challenges. *CoRR*, abs/2409.04464, 2024.

[Wang *et al.*, 2024b] Zhuohan Wang, Ziwei Zhu, Yizhou Han, Yufeng Lin, Zhihang Lin, Ruoyu Sun, and Tian Ding. Optibench: Benchmarking large language models in optimization modeling with equivalence-detection evaluation, 2024.

[Wei *et al.*, 2022] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. 2022.

[Wu *et al.*, 2025] Yang Wu, Yifan Zhang, Yurong Wu, Yuran Wang, Junkai Zhang, and Jian Cheng. Evo-step: Evolutionary generation and stepwise validation for optimizing LLMs in OR, 2025.

[Xiao *et al.*, 2024] Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, and Gang Chen. Chain-of-experts: When llms meet complex operations research problems. OpenReview.net, 2024.

[Xing *et al.*, 2024] Linzi Xing, Xinglu Wang, Yuxi Feng, Zhenan Fan, Jing Xiong, Zhijiang Guo, Xiaojin Fu, Rindra Ramamonjison, Mahdi Mostajabdaveh, Xiongwei Han, Zirui Zhou, and Yong Zhang. Towards human-aligned evaluation for linear programming word problems. In *LREC/COLING*, pages 16550–16556, 2024.

[Yang *et al.*, 2025] Zhicheng Yang, Yiwei Wang, Yinya Huang, Zhijiang Guo, Wei Shi, Xiongwei Han, Liang Feng, Linqi Song, Xiaodan Liang, and Jing Tang. Optibench meets resocratic: Measure and improve LLMs for optimization modeling. In *The Thirteenth International Conference on Learning Representations*, 2025.

[Yao *et al.*, 2023] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. 2023.

[Zhang *et al.*, 2024a] Jihai Zhang, Wei Wang, Siyan Guo, Li Wang, Fangquan Lin, Cheng Yang, and Wotao Yin. Solving general natural-language-description optimization problems with large language models. pages 483–490. Association for Computational Linguistics, 2024.

[Zhang *et al.*, 2024b] Yifei Zhang, Xintao Wang, Jiaqing Liang, Sirui Xia, Lida Chen, and Yanghua Xiao. Chain-of-knowledge: Integrating knowledge reasoning into large language models by learning from knowledge graphs, 2024.

[Zhang *et al.*, 2025] Yansen Zhang, Qingcan Kang, Wing Yin YU, HaileiGong, Xiaojin Fu, Xiongwei Han, Tao Zhong, and Chen Ma. Decision information meets large language models: The future of explainable operations research. In *The Thirteenth International Conference on Learning Representations*, 2025.