

Crosscutting Areas

ORLM: A Customizable Framework in Training Large Models for Automated Optimization Modeling

Chenyu Huang,^a Zhengyang Tang,^{b,c,d} Shixi Hu,^e Ruoqing Jiang,^{f,g} Xin Zheng,^h Dongdong Ge,^{i,*} Benyou Wang,^{b,d,*} Zizhuo Wang^{b,i,*}

^aResearch Institute for Interdisciplinary Sciences, School of Information Management and Engineering, Shanghai University of Finance and Economics, Shanghai 200433, China; ^bSchool of Data Science, The Chinese University of Hong Kong, Shenzhen 518172, China; ^cSchool of Science and Engineering, The Chinese University of Hong Kong, Shenzhen 518172, China; ^dShenzhen Research Institute of Big Data, Shenzhen 512182, China; ^eCardinal Operations, Yangpu District, Shanghai 200433, China; ^fDepartment of Economics, Columbia University, New York, New York 10027; ^gSchool of Economics and Management, Tsinghua University, Beijing 100084, China; ^hDepartment of Economics, Duke University, Durham, North Carolina 27705; ⁱAntai College of Economics and Management, Shanghai Jiao Tong University, Shanghai 200030, China; ^jSchool of Management and Economics, The Chinese University of Hong Kong, Shenzhen 518172, China

*Corresponding authors

Contact: chenyuhuang@stu.sufe.edu.cn,  <https://orcid.org/0009-0009-6130-6285> (CH); zhengyangtang@link.cuhk.edu.cn (ZT); shixi@shanshu.ai (SH); rj2556@columbia.edu (RJ); xin.zheng@duke.edu,  <https://orcid.org/0009-0003-6040-0655> (XZ); ddge@sjtu.edu.cn,  <https://orcid.org/0009-0006-9328-527X> (DG); wangbenyou@cuhk.edu.cn,  <https://orcid.org/0000-0002-1501-9914> (BW); wangzizhuo@cuhk.edu.cn,  <https://orcid.org/0000-0003-0828-7280> (ZW)

Received: August 7, 2024

Revised: January 6, 2025; February 24, 2025;
March 23, 2025

Accepted: April 1, 2025

Published Online in Articles in Advance:
May 8, 2025

Area of Review: Machine Learning and Data
Science

<https://doi.org/10.1287/opre.2024.1233>

Copyright: © 2025 INFORMS

Abstract. Optimization modeling plays a critical role in the application of Operations Research (OR) tools to address real-world problems, yet they pose challenges and require extensive expertise from OR experts. With the advent of large language models (LLMs), new opportunities have emerged to streamline and automate such tasks. However, current research predominantly relies on closed-source LLMs, such as GPT-4, along with extensive prompt engineering techniques. This reliance stems from the scarcity of high-quality training data sets for optimization modeling, resulting in elevated costs, prolonged processing times, and privacy concerns. To address these challenges, our work is the first to propose a viable path for training open-source LLMs that are capable of optimization modeling and developing solver codes, eventually leading to a superior ability for automating optimization modeling and solving. Particularly, we design the OR-INSTRUCT, a semiautomated data synthesis framework for optimization modeling that enables customizable enhancements for specific scenarios or model types. This work also introduces IndustryOR, the first industrial benchmark for evaluating LLMs in solving practical OR problems. We train several 7B-scale open-source LLMs using synthesized data (dubbed ORLMs), which exhibit significantly enhanced optimization modeling capabilities, achieving competitive performance across the NL4Opt, MAMO, and IndustryOR benchmarks. Additionally, our experiments highlight the potential of scaling law and reinforcement learning to further enhance the performance of ORLMs. The workflows and human-machine interaction paradigms of ORLMs in practical industrial applications are also discussed in the paper.

Funding: D. Ge received financial support from the National Natural Science Foundation of China (NSFC) [Grants 72225009, 72394360, and 72394365]. Z. Wang received financial support from the NSFC [Grants NSFC-72425013 and NSFC-72394361] and the Guangdong Provincial Key Laboratory of Mathematical Foundations for Artificial Intelligence [Grant 2023B1212010001]. C. Huang received financial support from the NSFC [Grants 72394360, 72394364, and 72171141]. Z. Tang and B. Wang received financial support from the Shenzhen Science and Technology Program [Grant JCYJ20220818103001002], Shenzhen Doctoral Startup Funding [Grant RCBS20221008093330065], the Tianyuan Fund for Mathematics of the National Natural Science Foundation of China [Grant 12326608], the Shenzhen Key Laboratory of Cross-Modal Cognitive Computing [Grant ZDSYS20230626091302006], and the Shenzhen Stability Science Program 2023.

Supplemental Material: The online appendix is available at <https://doi.org/10.1287/opre.2024.1233>.

Keywords: automated optimization modeling • large language model • synthetic data

1. Introduction

Optimization models have been a critical analytical tool for business decision making, aiding companies in making optimal choices under various complex decision

environments. With the fast development of computing capability (including both the development of algorithm and hardware) and the growing amount of data, optimization models are playing more and more significant

roles in the operations of large corporations. For instance, optimization techniques used by JD.com enable them to handle 10 times the normal volume of orders during peak sales seasons, helping the company decrease its fulfillment expense ratio to a world-leading level of 6.5%, which has resulted in hundreds of millions of dollars in savings (Qin et al. 2022). Baosteel, one of the largest steel companies in China, utilized integer optimization for production planning at its main plant in Shanghai, leading to a significant reduction in carbon monoxide emissions—over 500,000 tons annually—as reported by INFORMS (2013). The U.S. Census Bureau was recognized as a finalist for the 2022 Franz Edelman Award for their innovative use of operations research and analytics in re-engineering field operations for the 2020 U.S. Census. This project utilizes advanced analytics, optimization, and machine learning techniques to automate and optimize the scheduling, workload assignments, and management of field data collection, a process previously done manually (Adams et al. 2023). Despite the huge amount of success stories, the sensitivity of optimization models to environmental changes necessitates frequent adjustments in dynamically evolving business contexts. This, combined with the high level of expertise and substantial labor costs required for optimization modeling, represents a significant bottleneck that has hindered the full potential to be unlocked for optimization techniques.

The advent of large language models (LLMs) gives an emerging hope to the above challenge. As soon as their advent, these pretrained LLMs have exhibited remarkable capabilities in rapid knowledge comprehension and cross-disciplinary learning, demonstrating remarkable success and transformative potential across various fields (Huang et al. 2024b), like medical consultation (Wang et al. 2023c), software development (Zaremba et al. 2021), offering stock market trading strategies (Chen et al. 2022a), improving recommendation systems (Xu et al. 2024), and enhancing marketing research (Wang et al. 2024). In the realm of more abstract mathematical reasoning, research has increasingly incorporated more techniques to enhance the capabilities of large models, yielding promising results. For example, Trinh et al. (2024) train Alphaproof with hundreds of billions of synthetic data used to tackle complex olympiad geometry problems. Similarly, MathScale (Tang et al. 2024) offers a straightforward and scalable method to generate high-quality data for mathematical reasoning, establishing a foundation for further training of large language models in mathematical reasoning capabilities.

In recent years, with the advancements in pretrained language models (PLMs), such as GPT-4 and Claude 3.5, a growing number of studies have explored automating the modeling and solving of optimization

problems expressed in natural language. To produce complete solutions, including executable programs that leverage established solvers, existing approaches often employ prompt engineering techniques and multiagent collaboration frameworks built on proprietary LLMs, such as Chain-of-Experts (Xiao et al. 2023), OptiMUS (AhmadiTeshnizi et al. 2024), and OptiGuide (Li et al. 2023b). However, utilizing online application programming interfaces (APIs) for advanced large models offers no assurances of privacy protection, which is prohibitive in some sensitive industry applications. Additionally, these approaches are heavily reliant on the inherent capabilities of the foundational LLMs themselves, rendering them incapable of addressing the customized needs of users. As illustrated in Figure 1, even a basic optimization modeling problem presents challenges for the cutting-edge foundational large language model, GPT-4. It demonstrates deficiencies in comprehending and transforming logical constraints, which imposes a limitation on methods dependent on these general closed-source LLMs.

In summary, existing large language models for optimization modeling exhibit the following deficiencies:

1. Limited modeling capabilities. Contemporary advanced LLMs frequently exhibit errors in interpreting logical constraints or inadvertently incorporate nonlinear elements, making it difficult to model real-world problems effectively.

2. Insufficient quality of optimization modeling training data. The quality and scale of data are directly proportional to the capability of the model, as described by the *Scaling Law* (Kaplan et al. 2020). However, a shortage in high-quality optimization modeling training data is often a critical challenge, obstructing the advancement of LLMs' modeling ability.

3. Data privacy concerns. Dependence on APIs for closed-source LLMs may lead to data breaches, which hamper the broader adoption and development of LLM technologies in the industrial sector.

4. Test sets are relatively homogeneous. The efficacy of large language models in automatic modeling is predominantly assessed using the data set provided by the NL4Opt competition (Ramamujison et al. 2023). This data set primarily focuses on simple linear programming tasks that exhibit relatively low complexity, narrow scope, and few types compared with real-world applications.

To fill this gap, this paper introduces, for the first time, a new path for training an open-source large language model specifically designed for modeling and solving optimization problems. To address the core of the above challenges, we present a semiautomated framework for the synthesis of high-quality optimization modeling data, aimed at progressively enhancing the specialized modeling capabilities of LLMs. We hope that this approach will inspire future research and serve

Figure 1. (Color online) An Example Is Presented to Illustrate the Failure of Advanced Large Models to Accurately Respond to Seemingly Straightforward Optimization Modeling Questions

The diagram shows a conversation between a User and a large language model (GPT-4). The User poses a toy production problem, which GPT-4 responds to by defining a mathematical model, decision variables, objective function, and constraints. The User's input is on the right, and GPT-4's response is on the left. A bracket on the right side of the User's input group is labeled "Logic constraints".

User:

Toy manufacturers can produce and sell toy trucks, airplanes, boats, and trains. The profit from selling one truck is \$5, one airplane is \$10, one boat is \$8, and one train is \$7. During the production process, 890 units of wood are available. Manufacturing one truck requires 12 units of wood, one airplane requires 20 units of wood, one boat requires 15 units of wood, and one train requires 10 units of wood. There are 500 units of steel available. Manufacturing one airplane requires 3 units of steel, one boat requires 5 units of steel, one train requires 4 units of steel, and one truck requires 6 units of steel.

In actual production operations, the following production conditions must be observed:

1. If trucks are manufactured, trains will not be manufactured.
2. If boats are manufactured, airplanes will also be manufactured.
3. The number of toy boats manufactured cannot exceed the number of toy trains manufactured.

How many types of toys should the manufacturer produce to maximize profit?

GPT-4:

Mathematical Model:
 To solve this problem, we can formulate it as a linear programming problem. We will define the decision variables, objective function, and constraints, then use a solver to find the optimal solution.

Decision Variables:

- x_1 : Number of toy trucks produced.
- x_2 : Number of toy airplanes produced.
- x_3 : Number of toy boats produced.
- x_4 : be the number of toy trains produced.

Objective Function:

- Maximize profit: Profit = $5x_1 + 10x_2 + 8x_3 + 7x_4$

Constraints:

1. Wood constraint: $12x_1 + 20x_2 + 15x_3 + 10x_4 \leq 890$
2. Steel constraint: $6x_1 + 3x_2 + 5x_3 + 4x_4 \leq 500$
3. If trucks are manufactured, trains will not be manufactured: $x_1 \cdot x_4 = 0$
4. If boats are manufactured, airplanes will also be manufactured: $x_3 \leq x_2$
5. The number of toy boats manufactured cannot exceed the number of toy trains manufactured: $x_3 \leq x_4$
6. Non-negativity constraints: $x_1, x_2, x_3, x_4 \geq 0$

Notes. The highlighted responses contain errors, specifically the introduction of nonlinear terms and misunderstandings of logical relationships. This response was generated in June 2024, originating from the web-based GPT-4.

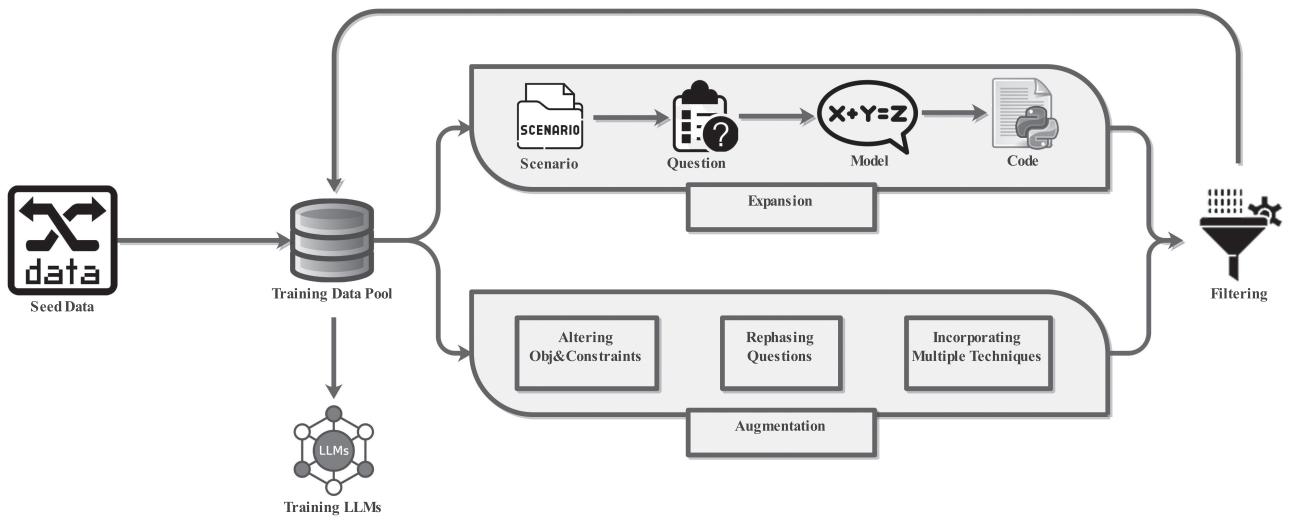
as a foundation for the deeper integration of large models with operations research. Our main contributions can be summarized as follows:

1. To ensure the effectiveness, robustness, and real-world applicability of our model, we have identified four crucial requirements that the training data set must fulfill, as informed by academic research like Alzubaidi et al. (2023), Jiang et al. (2024), and industry insights. Inspired by these requirements, we design and implement OR-INSTRUCT, a semiautomated process for creating synthetic data tailored to optimization modeling. The process uses an iterative bootstrapping algorithm (see Figure 2). Initially, we collect a set of seed industry cases (e.g., 686 cases in our study) and add them to the training data pool. Following this, we use two strategies. One is expansion, which employs GPT-4 and leverages its in-context learning capability to generate data spanning diverse scenarios and question types. Another is the augmentation strategy,

which involves modifying objectives and constraints, rephrasing questions, and incorporating diverse modeling techniques to ensure the difficulty and quality of generation problems. Finally, heuristics are used to automatically filter out obviously low-quality data. This cycle can be repeated for multiple iterations until the training data pool reaches the requisite volume (e.g., 32,481 cases in our study). Notably, this approach is highly customizable; new seed data and scenarios can be seamlessly introduced to adapt the data set to specific needs.

2. To evaluate the efficacy of OR-INSTRUCT, we introduce IndustryOR, the inaugural industrial benchmark tailored for optimization modeling. This benchmark uses data sourced from 13 different industries, covering 5 types of questions across 3 levels of difficulty. For comprehensive evaluation, we also include NL4Opt (Ramamonjison et al. 2023) and MAMO (Huang et al. 2024a) benchmarks.

Figure 2. Overview of OR-INSTRUCT



3. We utilize the synthetic data generated by OR-INSTRUCT to train a series of open-source large language models, each approximately 7 billion parameters in size, including Mistral-7B (Jiang et al. 2023), Deepseek-Math-7B-Base (Shao et al. 2024), and LLaMA-3-8B (AI@Meta 2024). We designate these trained models as Operations Research Language Models (ORLMs) and observe a marked enhancement in their optimization modeling capabilities. Our experimental results show that the best-performing ORLM achieves competitive performance on the NL4Opt, MAMO, and IndustryOR benchmarks. Building upon this foundation, we conduct extensive numerical experiments to analyze the limitations of the ORLM and propose feasible solutions for its improvement. The results highlight the promising prospects of ORLM in advancing the application and development of operations research, while also suggesting potential human-LLM collaboration frameworks. These frameworks demonstrate how ORLM could enhance the productivity of both industry and academia in the field of operations research.

To the best of our knowledge, this work is the first initiative to train open-source LLMs specifically for optimization modeling in real-world industries. The proposed approach significantly reduces the reliance on closed-source LLMs, thus adding significant flexibility to the training process and greatly enhancing privacy preservation for industrial applications. Remarkably, the highest-performing ORLM achieves competitive results on the NL4Opt, MAMO, and IndustryOR benchmarks. This suggests that in the field of optimization modeling, our approach enables ORLMs with approximately 7 billion parameters to outperform many cutting-edge large language models, such as GPT-4, LLaMA-3.1, Qwen 2.5, and related agent-based frameworks.

We also conduct a thorough analysis of the ORLM's results, with numerous ablation experiments validating the effectiveness of our approach. The results show that, although ORLM performs relatively weakly on some challenging data sets, several strategies can still ensure improvements in its performance. For instance, applying scaling laws to further increase the model size and data scale leads to continuous performance enhancements. Additionally, we find that ORLM has weak ranking capabilities for optimal optimization models. Using alignment strategies, such as reinforcement learning, also helps strengthen ORLM's performance.

Furthermore, we want to comment that the proposed data synthesis method, the *OR-INSTRUCT*, has the ability to facilitate customization to address particular needs of larger models, such as supply chain management, scheduling, inventory issues, or enhancing capabilities in linear programming, integer programming, and mixed integer programming. This synthetic data framework robustly addresses the scarcity of training data for optimization modeling and provides a valuable reference for advancing the application of operations research in various industrial sectors with the support of large models. In Section 6.1, we provide a detailed discussion of representative workflows driven by large models.

Finally, we note that almost all of our results, including certain samples from the training data set, industry-level benchmark for optimization modeling, and all the best-performing LLMs that have been fine-tuned, are made open source. We hope that our work can serve as a foundation for subsequent explorations, empowering future researchers to enhance the utility and efficiency of large language models in optimization modeling.

The structure of this paper is organized as follows: Section 2 reviews the literature related to our research.

Section 3 delineates four essential conditions required for the training data and outlines the implementation specifics of the semiautomated synthetic data framework for optimized modeling, OR-INSTRUCT. Section 4 examines the impact of training open-source large models on various test sets and presents the experimental outcomes of customized enhancements in modeling capabilities. Section 5 analyzes the limitations of ORLM through extensive experiments and discusses potential areas for improvement. Section 6 discusses the practical application scenarios of ORLM and explores future research directions based on the conclusions drawn in this paper. We conclude the paper in Section 7.

2. Literature Review

Our research is related to several streams of literature; we will provide a detailed review of each stream below.

Artificial intelligence approach for facilitating solution to operations research problems. With the fast development of artificial intelligence (AI) techniques, there has been a growing amount of research leveraging advanced techniques, such as deep learning, to address challenging operations research (OR) problems traditionally resistant to conventional methods. For example, in the supply chain context, Gijsbrechts et al. (2022) employ deep reinforcement learning to enhance inventory management, demonstrating parity with leading heuristics and approximate dynamic programming approaches. Qi et al. (2023) investigate a data-driven multiperiod inventory replenishment problem with uncertain demand and vendor lead time and propose a one-step end-to-end framework that uses deep learning models to output the suggested replenishment amount directly from input features. In the revenue management field, Aouad and Désir (2022) propose a neural network-based discrete choice model called RUMnets, which combines the theoretical robustness of traditional economic models with the flexibility and learning capabilities of modern artificial intelligence approaches, leading to better predictions and insights into consumer behavior. Wang et al. (2023b) explore the deployment of deep neural networks in making the optimal assortment selection, demonstrating their effectiveness in capturing consumer choice. As for queuing theory, Dai and Gluzman (2022) investigate the deployment of advanced policy gradient methods for controlling multiclass queueing networks, underscoring the superior efficacy of the proximal policy optimization algorithm under diverse load scenarios. For personalized recommendation, Wang et al. (2023a) utilize a novel unsupervised deep learning network and graph embedding techniques to improve recommendation accuracy and diversity by effectively transferring check-in patterns across different geographical contexts. Regarding the general Markov decision process, Zhang et al. (2021) introduce a

model-free reinforcement learning algorithm that attains asymptotically optimal sample complexity for learning an ϵ -optimal policy within discounted Markov decision processes. In addition, these techniques have been used in demand prediction; for example, Lee et al. (2024) propose a Transformer-based methodology for time series conformal prediction, utilizing its decoder to estimate prediction intervals through the quantile prediction of residuals, showcasing enhanced performance over existing conformal prediction techniques. Chen et al. (2020) introduce a novel architecture of convolutional neural networks that significantly improves forecasting by modeling the conditional distribution of future series data directly, rather than relying on traditional autoregressive methods.

AI technology also demonstrates significant strengths in facilitating the implementation of operations research; for instance, Liu et al. (2023) employ reinforcement learning combined with spatial data to optimize the allocation of resources and strategize for disease outbreak responses. Parmentier (2022) introduces a novel paradigm that integrates machine learning with operations research to efficiently approximate complex industrial problems by transforming them into classic, more tractable optimization challenges.

In addition, there have also been recent attempts in using AI methods to empower optimization algorithm design. For instance, deep learning or machine learning can be used to accelerate conventional optimization algorithms for tackling mixed-integer programming problems (e.g., Nair et al. 2020, Chen et al. 2023b). Numerous researchers also explore end-to-end optimization problem-solving strategies based on deep learning methods, motivated by the high structural similarity among many optimization problems, with differences primarily in data distribution (e.g., Khalil et al. 2017, Chen et al. 2022b). Compared with this work, the above stream of works studies the integration of AI approach and traditional OR approaches in solving OR problems. In contrast, the main focus of the present work is in the facilitation and automation of the modeling process of the optimization model, which is different from the above stream.

Automated optimization modeling is an emerging area, which aims to automate mathematical modeling and solving tasks to facilitate efficient decision making. Recently, a growing body of studies have emerged, aiming to bridge the gap between natural language processing and mathematical optimization. One of the pioneering endeavors in this field is the NL4Opt competition, which focuses on formulating mathematical models for optimization problems (Ramamrison et al. 2023). In particular, their work introduces a two-step framework using PLMs and offers a widely used benchmark for intelligence mathematical modeling. Expanding on this, Xiao et al. (2023) introduce Chain-of-Experts, a multiagent

LLM framework to build optimization models that significantly outperforms GPT-4 by incorporating domain-specific knowledge and cooperative reasoning strategies. AhmadiTeshnizi et al. (2024) propose OptiMUS, a robust LLM-based agent designed to formulate and solve mixed-integer linear programming (MILP) problems. This system not only develops mathematical models, but also assists in writing and debugging solver code, showcasing a comprehensive solution that extends beyond simple problem formulation. Similarly, Li et al. (2023b) develop a framework called OptiGuide which leverages LLMs to interpret and explain optimization outcomes in supply chain management, thereby enhancing stakeholder comprehension and trust. In terms of diagnosing and resolving optimization issues, Chen et al. (2023a) introduce OptiChat, a natural language-based system equipped with GPT-4, which assists in identifying and addressing infeasible optimization models. Unlike these methods, our approach involves fine-tuning open-source LLMs and delivering a complete solution using direct prompting, which is more concise and privacy-preserving than multiagent interaction and simply calling API from advanced LLMs like GPT-4.

Synthetic data play a crucial role in enhancing the capabilities of large language models by providing controlled, high-quality data sets that help fill gaps in real-world data, eliminate biases, and improve model robustness. A review of literature on synthetic data and LLMs is referred to Ding et al. (2024). In the field of synthetic data for optimization modeling, traditional methods often require specialized knowledge to convert real-world problems into mathematical models, which limits their accessibility and flexibility. Most research focuses on how to automatically generate mixed-integer linear programming models to overcome these challenges more effectively. For example, the methods proposed by Pawlak and Krawiec (2017) and Sroka and Pawlak (2018) utilize MILP examples for the synthesis of constraints. These methods develop an approach that can generate constraints from examples of both feasible and infeasible solutions, thereby demonstrating the feasibility of automating this conversion process. Further expanding on this theme, Pawlak and O'Neill (2021) leverage evolutionary strategies to generate constraints that improve the adaptability and efficiency of optimization models, thus enhancing the model's flexibility in handling complex scenarios. Prasath and Karande (2023) examine the effectiveness of synthesizing mathematical programs from natural language specifications based on the NL4Opt competition data set, focusing on data synthesis methods using the CodeT5 model with enhancements from data augmentation and post-processing. The study highlights how synthetic data generation, notably through techniques like reverse translation and parameter variation, significantly improves model performance in solving optimization

problems. The synthetic data framework proposed in Li et al. (2023a) leverages LLMs to identify decision variables, classify constraints, and generate MILP models based on NL4Opt problems. This approach significantly outperforms traditional methods, achieving higher accuracy in constraint recognition and model formulation. In contrast to the aforementioned synthetic data approaches, the method proposed in this paper is designed for the general purpose of synthesizing high-quality data for optimization modeling from unstructured natural language descriptions. This allows for the development of synthesis strategies that are not only tailored to specific problems, but also capable of customized data augmentation.

3. Semiautomated Framework for Data Generation

In this section, we present the formal definition of the optimization modeling task. We first discuss some critical requirements for effectively training an open-source LLM to perform this task. Based on these requirements, we design a semiautomated data synthesis framework OR-INSTRUCT which achieves the desired properties.

3.1. Data Requirements

Given an OR problem p described in natural language, optimization modeling (Berry and Houston 1995, AhmadiTeshnizi et al. 2024) involves constructing a mathematical model m that converts the real-world problem into an optimization model with concrete decision variables, objective function, and constraints. Then, for large-scale problems, in order to obtain an optimal solution, we further need to convert the mathematical model into a program c compatible with a specific optimization solver. Hence, an expected training example for this task is usually required in the form of the triplet (p, m, c) , as illustrated in Figure 3. Training a large language model f for this task essentially involves learning a function $f : p \rightarrow (m, c)$ that maps problems to their corresponding mathematical models and solver programs. In this paper, without loss of generality, COPT is employed as the default solver. COPT (Ge et al. 2022), an acronym for Cardinal Optimizer, is able to tackle large-scale optimization problems and achieves good performance in a variety of tasks (Hans 2002). We select the COPT solver because the predominant large language models available in the market have not yet mastered the COPT syntax. This choice underscores the efficiency and extensibility of our framework. However, any well-established solver may be utilized in the training data set construction.

In addition to the desired format, as described above, a desired training data set should demonstrate effectiveness, robustness, and practical applicability to real-world scenarios. In particular:

Figure 3. (Color online) An Expected Training Example for Optimization Modeling Task

An Expected Training Example

Input - Natural Language Problem:
 A company has three transportation options to choose from to transport 25 tons of cargo, namely trucks, airplanes, and ships with costs \$100, \$120, \$130 per ton and capacities of 10, 20, 30 tons respectively. The company can't choose trucks and ships together. How should the company optimize the selection and allocation of these methods to minimize overall costs?

Target - Mathematical Model and Program:

```

## Mathematical Model:
### Variables:
-  $x_1, x_2, x_3$ : 0-1 variables indicating whether trucks, airplanes, and ships are selected, respectively.
-  $y_1, y_2, y_3$ : Non-negative continuous variables indicating the volume of cargo transported by three modes.
### Objectives and Constraints:
Minimize $_{x_i \in \{0,1\}, y_i \geq 0}$   $100y_1 + 120y_2 + 80y_3$ 
subject to  $x_1 + x_2 + x_3 \geq 1$  (At least one transportation)
 $y_1 \leq 10x_1, \quad y_2 \leq 20x_2, \quad y_3 \leq 30x_3$  (Capacity limits)
 $x_1 + x_3 \leq 1$  (Mutual exclusion constraint)
 $y_1 + y_2 + y_3 \geq 25$  (Transportation requirements)

## Program using COPT solver:
1 import copty as cp
2 from copty import COPT
3 # Initialize the optimization environment
4 env = cp.Envr()
5 model = env.createModel("TransportationOptimization")
6 # Define costs and capacities for each mode of transportation
7 costs = {'trucks': 100, 'airplanes': 120, 'ships': 130}
8 capacities = {'trucks': 10, 'airplanes': 20, 'ships': 30}
9 # Define binary and continuous variables for each transportation mode
10 x = {mode: model.addVar(vtype=COPT.BINARY, name=f"x_{mode}") for mode in costs}
11 y = {mode: model.addVar(vtype=COPT.CONTINUOUS, lb=0, name=f"y_{mode}") for mode
12     in costs}
13 # Set the objective function to minimize the total transportation cost
14 model.setObjective(sum(costs[mode] * y[mode] for mode in costs), sense=COPT.
15     MINIMIZE)
16 # Add constraints
17 model.addConstr(x['trucks'] + x['airplanes'] + x['ships'] >= 1, name="
18     AtLeastOneMode")
19 for mode in costs:
20     model.addConstr(y[mode] <= capacities[mode] * x[mode], name=f"Capacity_{mode}")
21 model.addConstr(x['trucks'] + x['ships'] <= 1, name="ModeExclusivity")
22 model.addConstr(sum(y[mode] for mode in capacities) >= 25, name="Volume
23     Requirement")
24 # Solve the model
25 model.solve()
26 # Check the solution status and print the optimal values of the variables
27 if model.status == COPT.OPTIMAL:
28     print("Optimal solution found:")
29     for mode in costs:
30         print(f"{mode}: x = {x[mode].x}, y = {y[mode].x}")

```

Requirement 1. Comprehensive Coverage: The data set should cover: (1) diverse scenarios, such as supply chain optimization, scheduling, inventory management, and transportation logistics; (2) different problem types like linear programming, integer programming, and mixed integer programming; and (3) varying difficulty levels (easy, medium, hard)—we

provide a way to describe the difficulty level of a problem in Online Appendix A.1.

Requirement 2. Environmental Adaptability: In real-world industrial settings, the objectives and constraints of problems often change due to shifts in business goals, market conditions, or resource availability. A good optimization model should be able to adapt to

those changes easily. Therefore, it is vital that the data set includes cases reflecting these dynamic changes.

Requirement 3. Linguistic Diversity: Problems described in natural language often show different syntax, ambiguities, and complexities. For example, one problem might mention “inventory overflow,” whereas another refers to “excess stock”. Including this Linguistic Diversity in the data set could improve the model’s ability to understand varied descriptions.

Requirement 4. Technique Variability: For some challenging problems, there may be multiple modeling techniques, such as linearizing a nonlinear problem by introducing auxiliary variables or introducing the big M term in an integer optimization formulation. Including this variety in the data set allows the model to learn different modeling techniques and approaches.

3.2. OR-INSTRUCT: Toward Training Effective OR LLMs

In the above section, we describe some critical requirements on the data set for an effective training of an LLM to be equipped with good optimization modeling capabilities. However, collecting data on a large scale that aligns with the above requirements presents significant challenges because: (1) such data primarily reside within private industrial cases, and no public data sets sufficiently meet the requirements; (2) training data suitable for optimization modeling are limited and require extensive time to collect; and (3) conventional synthesis methods (for example, Wang et al. 2022) encounter substantial obstacles in generating this type of data, as will be elaborated on below. In the following, to address the above challenges, we design and implement OR-INSTRUCT, a semiautomated process for creating synthetic data tailored to these requirements. The overall pipeline of OR-INSTRUCT is depicted in Figure 2, which iteratively applies two strategies—namely, expansion and augmentation—to the training data pool, followed by the filtering of obviously low-quality data.

3.2.1. Initial Data Set. The quality of initial data plays a crucial role in large model training, providing a strong foundation for diversity and difficulty as a starting point (Marion et al. 2023). Additionally, in this customizable framework designed to enhance modeling capabilities, the category or domain of the optimization problem associated with the initial data is important. We will demonstrate later that when the initial data set is concentrated in a specific type (e.g., scenario or type), it substantially enhances the LLM’s ability to model effectively within that domain.

In our particular case, we start with 686 real-world industry cases collected from some OR textbooks and our previous industrial projects (after proper abstraction and anonymization).

3.2.2. Strategy 1: Expansion for Comprehensive Coverage. Initially, OR-INSTRUCT seeks to generate new data by expanding scenarios and question types from the training data set using a bootstrapping approach powered by GPT-4. The process begins with GPT-4 generating a list of 100 scenarios where optimization models are applied in real-life contexts. In each iteration, three examples are selected from the data pool to serve as in-context references for GPT-4, and one scenario is randomly chosen from the list. Among the three examples, two are sourced from real-world data, whereas the third, if available, is taken from data previously generated by the model. Finally, GPT-4 is prompted to create a new sample based on the three examples within the context of the selected scenario. This iterative approach promotes greater diversity in the resulting data set. The prompting template for this expansion process is detailed in Online Appendix A.2.

It’s important to note that optimization models are naturally abstract, often built on similar logical ideas across various scenarios. In this setting, *Comprehensive Coverage* not only exposes the LLMs to a broad range of situations, but also helps it better grasp the underlying patterns and principles when modeling across these different contexts. However, it falls short of meeting the other requirements, especially concerning varying levels of difficulty. In manually reviewing the difficulty of generated example, of 50 cases, 87% are deemed easy, 13% medium, and none hard, as judged by criteria in Online Appendix A.1. We also provide examples in Online Appendix A.7 for comparing generated easy entries with real-world hard entries. Fortunately, the original seed data already show a diverse range of difficulties. Thus, we can naturally enhance the difficulty diversity by augmenting them as described in the next strategy, thereby further addressing *Comprehensive Coverage*.

3.2.3. Strategy 2: Augmentation for Problem-Solution Diversity. To fulfill the remaining requirements, the OR-INSTRUCT adopts an augmentation strategy for the original seed data, designed to both enhance the diversity and robustness of the data set, while preserving the complexity inherent in optimization modeling problems. This strategy is crafted to mirror potential scenarios that might be encountered in real-world industrial applications, encompassing modifications such as alterations in problem descriptions, model adjustments, or concurrent changes. To address these three scenarios, the implementation of this strategy involves three specific tactics: rephrasing questions, modifying objectives and constraints, and integrating diverse modeling techniques. Together, these tactics aim to broaden the range of problem-solution diversity, thereby enriching the training data set.

a. Altering Objectives and Constraints: The first augmentation involves adding, removing, or replacing objectives and constraints in the problem, along with making necessary adjustments to the mathematical models and programs. Specifically, we start by providing GPT-4 with the original example and ask it to list five potential changes to the objectives and constraints. These suggested changes are then fed back to GPT-4 using a prepared few-shot prompt to modify the problem, model, and programs accordingly. This augmentation is designed to enhance *Environmental Adaptability*. An example is provided in Figure 4, and the corresponding prompting template is shown in Online Appendix A.3.1.

Remark 1. One important point to note is that during the process of altering objectives and constraints, GPT-4 may generate background scenarios that are illogical, such as presenting two contradictory constraints. However, this issue arises from unreasonable problem descriptions. As long as the model can accurately map each statement of the problem to the correct objectives and constraints, it can still yield the desired results when tested on meticulously designed data sets that reflect such issues.

b. Rephrasing Questions: The second augmentation involves modifying the problem description while essentially addressing the same optimization model. This approach enhances the robustness of large language models to different prompts, enabling them to abstract a unified optimization model across various problem descriptions. This process involves instructing GPT-4 to rewrite the target problem, either simplifying or complicating it, while ensuring that the core logic aligns with the solution, including the mathematical model and programs. This augmentation is designed to enhance *Linguistic Diversity*. An example is provided

in Figure 5, and the corresponding prompting template is shown in Online Appendix A.3.2.

c. Incorporating Multiple Modeling Techniques: The third augmentation explores the use of different modeling techniques. We identify five potential techniques from engineers' experiences, such as introducing auxiliary variables or using the Big M method, for GPT-4 to choose from in modifying an objective or constraint in the original mathematical model. This augmentation is designed to enhance *Solution Variability*. An example is provided in Figure 6, and the corresponding prompting template is shown in Online Appendix A.3.3.

3.2.4. Postprocessing and Filtering. At the end of each iteration, OR-INSTRUCT implements correction and filtering measures on the generated examples. First, we apply a regular match correction function to address minor grammatical errors in the programs, which may arise from GPT-4's limited familiarity with the COPT API. Unexecutable programs are discarded, as they clearly represent low-quality data. Notably, in the final iteration, we also eliminate duplicate questions and any examples overlapping with the evaluation benchmarks to ensure that the data set remains uncontaminated.

Based on manual sampling assessments, we find that the accuracy of the synthesized data by expansion operation is about 70% and 75% for the augmentation data, judged by the correctness of both the code and the model. On average, each iteration filters out approximately 39% of the generated examples, with the remaining examples incorporated into our training data pool.

In summary, the above outlines all the key steps of OR-INSTRUCT, and the specific procedures can be found in Figure 2.

Figure 4. (Color online) An Example Illustrating Altering Objectives and Constraints

Altering Objectives and Constraints for Requirement 2

Original:

Q: ... The company can't choose trucks and ships together. Denote the cost ...

Augmented:

Q: ... The company can't choose trucks and ships together. *Due to the special nature of the goods, the company has decided that if trucks are chosen, airplanes must also be selected for transportation.* Denote the cost ...

A: ... New dependency constraint (choosing trucks necessitates choosing airplanes): $x_1 \leq x_2$...

```

1 ...
2 model.addConstr(x['trucks'] <= x['airplanes'], name="New constraint")
3 ...
  
```

Note. Italicized text denotes altered content.

Figure 5. An Example Illustrating Rephrasing Questions

Rephrasing Questions for Requirement 3

Original:
 Q: A company has three transportation options to choose from to transport 25 tons of cargo, namely trucks, airplanes, and ships with costs \$100, \$120, \$130 per ton and capacities of 10, 20, 30 tons respectively. The company can't choose trucks and ships together. How should the company optimize the selection and allocation of these methods to minimize overall costs?

Augmented:
Q: A corporation wants to transport 25 tons of cargo with least cost, and must choose from three transportation modes: trucks, airplanes, and ships. These options cost \$100, \$120, and \$130 per ton, respectively, with capacities of 10, 20, and 30 tons. However, trucks and ships cannot be used together.

Note. Italicized text denotes the question that has been rephrased.

4. Results

In this section, we train the open-source 7b size large language models with synthetic data from OR-INSTRUCT and compare the results with GPT-4, as well as other benchmarks. Then, using the case of mixed integer linear programming as an example, we demonstrate the customized enhancement effect of OR-INSTRUCT by including directionally generated training data.

4.1. Data Generation Using OR-INSTRUCT

In our experiment, we initiate the study with 686 cases derived from previous industry projects, which the team engaged with previously. These cases are abstracted and anonymized appropriately. Throughout the OR-INSTRUCT process, we utilize the proprietary LLM gpt-4-0613 as the standard model. For each cycle, the training data pool undergoes an expansion procedure 20,000 times, and each augmentation operation is applied 6,000 times. Subsequently, an automatic filtration system excludes low-quality entries. After completing two iterations of this framework, we amass a total of 32,481 training examples.

Statistics. Figure 7 presents the statistics of the OR-INSTRUCT data. Of the data, 57% are generated by the expansion operation, 17.2% by the augmentation of altering objectives and constraints, 15.3% by the augmentation of rephrasing questions, and 10.5% by the augmentation of incorporating multiple modeling techniques. For scenario diversity, we have abstracted 16 industries from a total of 1,556 expanded scenarios to facilitate their representation in a plot. Regarding question type diversity, it expands to eight question types. In terms of data quality, the accuracy of correctness is 70% for expansion data and 75% for augmentation data. We find these accuracy levels acceptable in exchange for automatic filtering and more cost-effective data synthesis. Additionally, our preliminary experiments have demonstrated the benefits of incorporating such data.

4.2. Model Finetuning and Inference

This paper employs instruction tuning for open-source large models, a technique designed to enhance the ability of large language models to follow diverse natural language instructions. The primary aim is to improve zero-shot performance on unseen tasks, facilitating

Figure 6. (Color online) An Example Illustrating Incorporating Multiple Modeling Techniques

Incorporating Multiple Modeling Techniques for Requirement 4

Original:
 A: Mutual exclusion constraint (trucks and ships cannot be selected simultaneously): $x_1 + x_3 \leq 1$

Augmented:
A: Mutual exclusion constraint (Using big M method): $x_1 \leq (1 - x_3)M$, where M is a large number

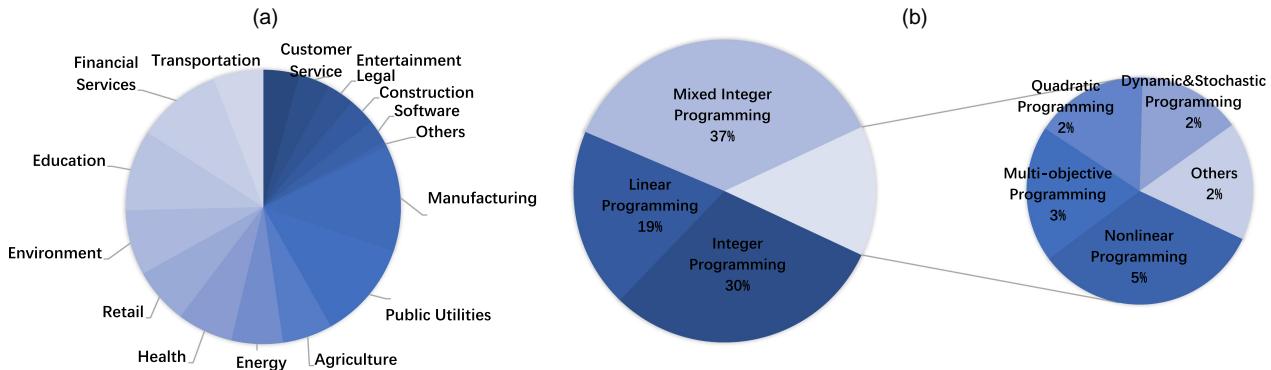
```

1 ...
2 model.addConstr(x['trucks'] <= (1-x['ships'])*M, name="New constraint")
3 ...

```

Note. Italicized text denotes the section that have been remodeled using new techniques.

Figure 7. (Color online) Statistics of the Data Generated by OR-INSTRUCT



Notes. (a) Distribution of industries. (b) Question type.

effective generalization across various instruction-based scenarios. The methodology involves training on a data set of instruction-output pairs, typically consisting of three elements: an instruction (e.g., "Translate this sentence from English to Spanish"), optional context, and the target output. The training can be mathematically represented as:

$$\mathcal{L}(\theta) = - \sum_{(x,y) \in \mathcal{D}} \log P(y | x; \theta),$$

where $\mathcal{L}(\theta)$ is the loss function, x is the instruction input, y is the expected output, and θ denotes the model parameters. We borrow a widely used training framework from open-instruct (Wang et al. 2023d), where we take a natural language OR problem wrapped in an Alpaca-like template (Taori et al. 2023) as the input prompt (see Online Appendix A.4) and treat a complete solution including mathematical models and programs as the target completion.

During training, we compute the loss only over the target completion. We employ the OR-INSTRUCT data set to train several open-source large language models of approximately 7 billion parameters, including Mistral-7B (Jiang et al. 2023), Deepseek-Math-7B-Base (Shao et al. 2024), and LLaMA-3-8B (AI@Meta 2024), which serve as our backbone models. We conduct a grid hyperparameter search for each backbone and present the best-performing hyperparameters in Table 1.¹ The models resulting from this process are referred to as ORLMs.

Regarding the inference techniques during evaluation, it is important to consider that large language models, as probabilistic generative models, exhibit inherent randomness. This randomness can lead to inconsistent outcomes across multiple evaluations. To mitigate the influence of this randomness, we implemented a greedy decoding strategy in a zero-shot scenario, consistently selecting the highest probability output (top-1) as the final result. After obtaining these results, we executed the corresponding code to derive the predicted optimal

values for comparison with the ground truth. Although this greedy approach may reduce the performance of large models on such tasks, it ensures the stability of the output and enables the reproducibility of our findings under identical conditions. We will discuss the impact of employing additional inference techniques on the performance of large models in Section 5.4.

4.3. Evaluation and Baselines

4.3.1. Evaluation Benchmarks and Metrics. We use NL4Opt (Ramamonjison et al. 2023), MAMO (Huang et al. 2024a), and IndustryOR as evaluation benchmarks. NL4Opt is the most widely used benchmark for operations research and includes 289 linear programming problems in its test set. However, NL4Opt only provides mathematical models as targets, which complicates the verification of execution accuracy due to the absence of optimal solutions. To address this, we convert these mathematical models into programs using GPT-4, calculate and check the optimal solutions, and use these as ground truth. MAMO, a concurrent project, evaluates the mathematical modeling capabilities in LLMs. It includes 652 easy and 211 complex linear programming problems, each paired with its corresponding optimal solution, sourced from various academic materials. IndustryOR, which is proposed in this work, is the first industrial benchmark, consisting of 100 real-world OR problems from eight industries. It covers five types of questions, linear programming, integer programming, mixed integer programming, nonlinear programming, and others across three levels of difficulty.

Table 1. Hyperparameters for Training ORLMs

Backbone	BatchSize	LearningRate	Epochs
Mistral-7B	512	3e-6	2
Deepseek-Math-7B-Base	128	2e-5	2
LLaMA-3-8B	64	5e-6	2
Qwen-2.5-7B	64	2e-5	2

We measure performance using execution accuracy, where an executed optimal value that matches any provided ground truth optimal value is considered correct. Compared with NL4Opt, this metric enables a fully automated evaluation and provides greater flexibility for mathematical modeling approaches.

4.3.2. Baselines. To ensure a comprehensive evaluation, we select a diverse set of models from previous methods for comparison. We include tag-BART (Kani and Gangwar 2022), which secured first place in the NeurIPS competition (Ramamorjison et al. 2022). Additionally, we consider methods that utilize proprietary LLMs. The “standard” prompting method involves prompting a proprietary LLM to produce mathematical programs, serving as a fundamental baseline. We also incorporate complex prompt engineering methods, such as Reflexion (Shinn et al. 2023), Chain-of-Experts (Xiao et al. 2023), and OptiMUS (AhmadiTeshnizi et al. 2024). These methods employ agents to refine both mathematical models and programs, achieving good performances on NL4Opt. We report their performance based on GPT-3.5 and GPT-4, respectively. Note that we implement the standard prompting on IndustryOR using the toolkit released by Chain-of-Experts (Xiao et al. 2023).

In addition, we introduce the most well-known open-source large models for comparison, including LLaMA-3.1-Instruct (Grattafiori et al. 2024), DeepSeek-v2-Chat (DeepSeek AI 2024), DeepSeek-R1 (Guo et al. 2025), Qwen2-Instruct (Yang et al. 2024), and Mistral-Nemo (Mistral AI 2024). Each model is tasked with constructing optimization models and generating code to solve the test set problems.

We also assess the difference between the performance of ORLM and human-level performance. Given that constructing and solving optimization models requires a solid foundation in operations research and programming skills, we recruit eight highly capable senior undergraduate students from relevant fields and eight experts in the domain, to evaluate four test sets: NL4OPT, MAMO-EasyLP, MAMO-ComplexLP, and IndustryOR. Each test set is evaluated by two senior undergraduate students and two experts. Each participant randomly selects 70 questions from their corresponding test set and is required to construct an optimized model and calculate the optimal solution. Participants are allowed to use any programming language; however, to ensure fairness in comparison with large models, they are only permitted to submit a finalized version of their code for testing and cannot make further modifications. Finally, the accuracy rates for the student and expert groups on each test set are obtained by averaging the results.

The results are presented in Table 2. First, it is clear that methods based on LLMs generally outperform the

PLM-based best method (tag-BART) in the NL4Opt test. This suggests that PLMs have limited generalization capabilities. For proprietary LLMs, as the mathematical reasoning capability increases from GPT-3.5 to GPT-4, we observe that performance has advanced across all prompt engineering methods.

Our best-performing ORLMs, like ORLM-LLaMA-3 and ORLM-Qwen2.5, achieve state-of-the-art performance across four benchmarks with respect to GPT-4 and other open-source LLMs,² surpassing Standard prompting based on GPT-4 by 42.2% in micro average³ and 55.4% in macro average.⁴ A comparison with human evaluation results reveals that ORLMs outperform senior undergraduate students on simpler problems (NL4Opt/MAMO-EasyLP) and approach expert-level performance. However, for more complex problems (IndustryOR/MAMO-ComplexLP), ORLMs still underperform relative to the average student, indicating that their problem-solving capabilities for complex tasks remain inadequate. We will analyze the underlying reasons for this in Section 5, highlighting that ORLMs still have the potential to surpass student performance and approach expert-level solutions on more challenging problems.

Although ORLMs demonstrate strong performance in optimization modeling tasks, we also evaluate their efficacy on general large language model benchmarks after domain-specific fine-tuning. In particular, we conduct a rigorous evaluation of the LLaMA-3-8B model (pre-fine-tuning) and the ORLM-LLaMA-3-8B model (post-fine-tuning) following the open-instruct (Wang et al. 2023d) on benchmarks from various domains, including mathematics (GSM8K), code (HumanEval), and general knowledge (MMLU, BBH, TydiQA); the results are listed in Table 3.

The results reveal that ORLM-LLaMA-3-8B exhibits improved performance in the mathematics and coding domains, which aligns with expectations, as the OR-INSTRUCT data emphasize mathematical modeling and coding with solver assistance. However, the model exhibits a slight decline in general domain performance. Overall, the average performance across all domains shows no substantial deviation.

4.3.3. A Comparative Study on Problem-Solving Efficiency. To verify the effectiveness of ORLM in human-AI collaboration, we conduct a comparative experiment to demonstrate its impact on algorithm engineers in solving practical problems. Specifically, we recruit a group of 30 people, consisting of experts and senior undergraduate students, dividing them evenly into two groups, A and B. Each group comprises seven experts and eight undergraduate students, with comparable proficiency levels. We design seven problems with varying difficulty levels, ranging from textbook exercises to those approximating industrial complexity.

Table 2. Comparison of Performance on the NL4Opt, MAMO, and IndustryOR Benchmarks

Method/Model	Size	NL4Opt (%)	MAMO EasyLP (%)	MAMO ComplexLP (%)	IndustryOR (%)	Micro Avg (%)	Macro Avg (%)
Methods based on PLMs							
tag-BART	140/400M	47.9 ^a	—	—	—	—	—
Methods based on GPT-3.5							
Standard	Unknown	42.4 ^a	—	—	—	—	—
Reflexion	Unknown	50.7 ^a	—	—	—	—	—
Chain-of-Experts	Unknown	58.9 ^a	—	—	—	—	—
Methods based on GPT-4							
Standard	Unknown	47.3 ^a	66.5 ^a	14.6 ^a	28.0	50.2	39.1
Reflexion	Unknown	53.0 ^a	—	—	—	—	—
Chain-of-Experts	Unknown	64.2 ^a	—	—	—	—	—
OptiMUS	Unknown	78.8 ^a	—	—	—	—	—
Standard prompting based on open-source LLMs							
LLaMA-3.1-Instruct	405B	38.7	35.1	20.8	13.0	31.5	26.9
DeepSeek-V2-Chat	236B	66.5	60.5	32.7	16.0	53.1	43.9
Qwen2-Instruct	72B	72.6	79.9	29.0	18.0	64.4	49.8
DeepSeek-R1-Distill	32B	80.4	69.1	45.4	33.0	64.8	56.9
Mistral-Nemo	12B	14.6	19.4	3.7	7.0	14.6	11.1
ORLMs based on open-source LLMs							
ORLM-Mistral	7B	84.4	81.4	32.0	27.0	68.8	56.2
ORLM-Deepseek-Math	7B	86.5	82.2	37.9	33.0	71.2	59.9
ORLM-LLaMA-3	8B	85.7	82.3	37.4	38.0	71.4	60.8
ORLM-Qwen2.5	7B	86.1	85.2	44.1	25	73.7	60.1
Human evaluation							
Senior undergraduates	—	80.4	84.9	53.1	44.0	75.2	65.6
Experts	—	94.3	90.4	78.9	76.0	85.0	88.2

Note. The highest results are highlighted in bold.

^aValues that are directly copied from original papers, with blanks where data were not reported.

Group A is tasked with independently modeling and programming the solutions without the aid of ORLM, whereas Group B utilizes ORLM for problem-solving and programming. We assess the effectiveness of ORLM based on two dimensions: accuracy and solution time. The results are presented in Figure 8. We acknowledge that the sample size of the experiment is limited due to the difficulty of finding suitable subjects; nevertheless, we perform statistical analysis on the limited data and find that the comparison is still statistically significant. In the following, we detail our statistical analysis.

Table 4 shows the results of our statistical analysis. Specifically, for each metric (difference in accuracy for total/experts/students, difference in time for total/experts/students), we first select an appropriate statistical test based on the distributional properties of the data, as indicated in the third column of the table. For solution time of each subgroup, we find that the distribution successfully passes the normality test and meets

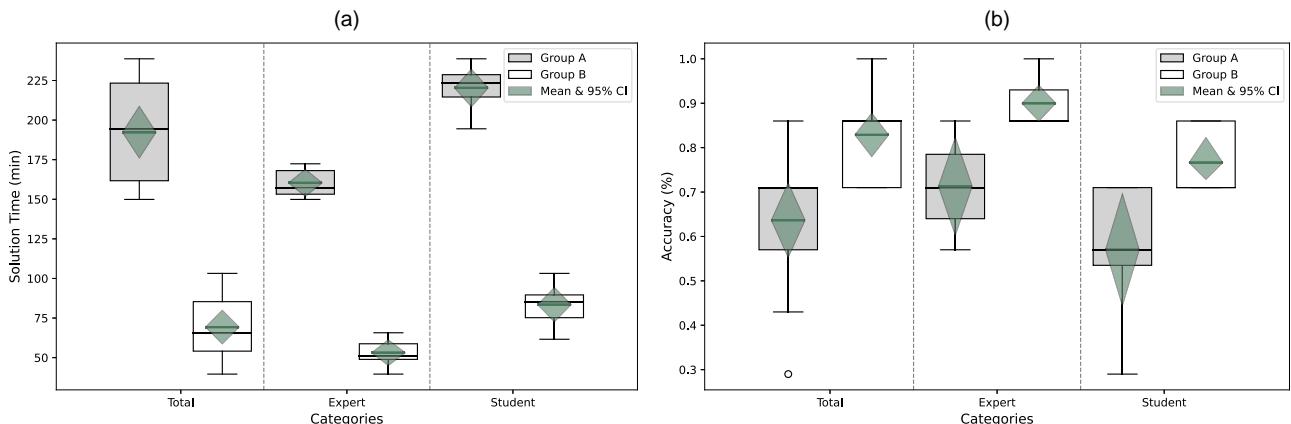
the assumption of homogeneity of variance. Consequently, we employ a standard *t*-test for statistical analysis. In contrast, for accuracy, the distribution does not conform to normality; therefore, we utilize the Mann-Whitney *U* test as a nonparametric alternative (see Hogg et al. 2019 for reference). Subsequently, we compute the test statistics and the corresponding *p*-value for each test, which are reported in the 4th and 5th columns, respectively. We also calculate the 95% confidence interval for the difference between the two groups based on each test and present the results in the 6th column.

To further demonstrate the robustness of our results, the last two columns provide additional analyses, including the effect size (Hedge's *g*) and the statistical power (Cohen 2013). Notably, Hedge's *g* is independent of sample size, making it a more objective measure of the generalizability and strength of the statistical significance. Meanwhile, statistical power serves as an indicator of the reliability and accuracy of our results. The

Table 3. Comparison of Accuracy Rate Between LLaMA-3-8B and ORLM-LLaMA-3-8B Across Multiple Benchmark Domains

Model	GSM8K (%)	HumanEval (%)	MMLU (%)	BBH (%)	TydiQA (%)
LLaMA-3-8B	56.5	67.7	65.2	63.7	21.1
ORLM-LLaMA-3-8B	58.0	70.6	64.6	61.9	21.9

Figure 8. (Color online) A Comparative Study on Problem-Solving Efficiency



Notes. (a) Comparison of solution time. (b) Comparison of accuracy.

detailed testing procedures and statistical indicators are provided in Online Appendix A.6.

For Table 4, we can see that at a 95% confidence level, the use of ORLM significantly improves both solution time and accuracy across all populations (total, expert, and student samples). The confidence intervals suggest that, after implementing ORLM, accuracy improves by approximately 10%–25%. Additionally, the total time saved in solving optimization modeling problems is 1.8–2.2 hours, with experts saving approximately 1.5–2 hours and students saving around 2–2.5 hours. The absolute value of Hedge's $g > 1$ indicates a substantial effect size, highlighting a large difference between experimental and control groups. Additionally, statistical power > 0.8 , ensuring a high probability of detecting a true effect and minimizing Type II error risk.

In addition, we collect feedback from engineers on their experience using ORLM, as shown in Table 5. None of the engineers rate ORLM as "useless" or "minimal help." Instead, the majority provide highly positive evaluations ("significant help" or "highly effective"), indicating that ORLM is both efficient and effective in assisting their work. These demonstrate the significant potential of ORLM to enhance the application of operations research in industry through human-

AI collaboration. We will further discuss the application value of ORLM in Section 6.1.

4.4. Customization Enhancement

In this subsection, we illustrate the capability of our framework to enhance large models in specific areas by generating targeted types of optimization modeling data. Specifically, we focus on the scenario of mixed integer linear programming. We provide an empirical example demonstrating how the quality of responses improves with training. Initially, we use a base data set comprising 10,000 general modeling cases. To enhance the ORLM's proficiency in MILP, we selected 50 fundamental MILP questions and their answers for inclusion in the seed data pool. After several iterative rounds, this process yields 2,000 tailored training data points for MILP. We will subsequently assess the impact of incorporating these 2,000 data points into the training data set by comparing the ORLM's performance before and after their addition.

As depicted in Figure 9, in response to the scenario presented in Figure 1, GPT-4 inaccurately interprets logical relationships as quantitative ones and includes nonlinear terms during the modeling process, as shown in the highlighted area, adversely affecting solver

Table 4. Summary of Tests and Effect Sizes

Type	Test	Method	Statistic Testing				
			Stat	p-value	Confidence interval	Hedge's g	Stat power
Total	Accuracy	Mann-Whitney U test	192.00	0.001	(0.14, 0.24)	1.46	0.97
	Time	Mann-Whitney U test	0.00	0.000	(−134.02, −112.35)	−4.43	1.00
Expert	Accuracy	Mann-Whitney U test	44.00	0.008	(0.10, 0.25)	1.81	0.87
	Time	t-test	−22.50	0.000	(−117.61, −96.85)	−11.26	1.00
Student	Accuracy	Mann-Whitney U test	56.5	0.006	(0.14, 0.26)	1.56	0.82
	Time	t-test	−20.27	0.000	(−151.29, −122.34)	−9.58	1.00

Note. For the criteria used in selecting between the *Mann-Whitney U test* and the *independent samples t-test*, please refer to Online Appendix A.6.

Table 5. Feedback from Participants on ORLM Effectiveness

Category	Useless	Minimal help	Moderate help	Significant help	Highly effective
Percentage (%)	0.0	0.0	13.3	60.0	26.7

performance. A similar mistake is observed in the general ORLM, which also misinterprets logical relationships as quantitative. In contrast, the targeted-trained ORLM accurately transforms the problem into a correct mathematical model by utilizing indicator variables and the Big M method. This showcases the robust generalization capability of OR-INSTRUCT. Furthermore, OR-INSTRUCT not only facilitates customized improvements across different modeling types, but also demonstrates potential for targeted enhancements within specific industrial sectors.

5. Analysis and Discussion

5.1. Detailed Comparison of ORLM vs GPT-4 on IndustryOR

To assess the optimization modeling capabilities across different levels of difficulty and question types, we

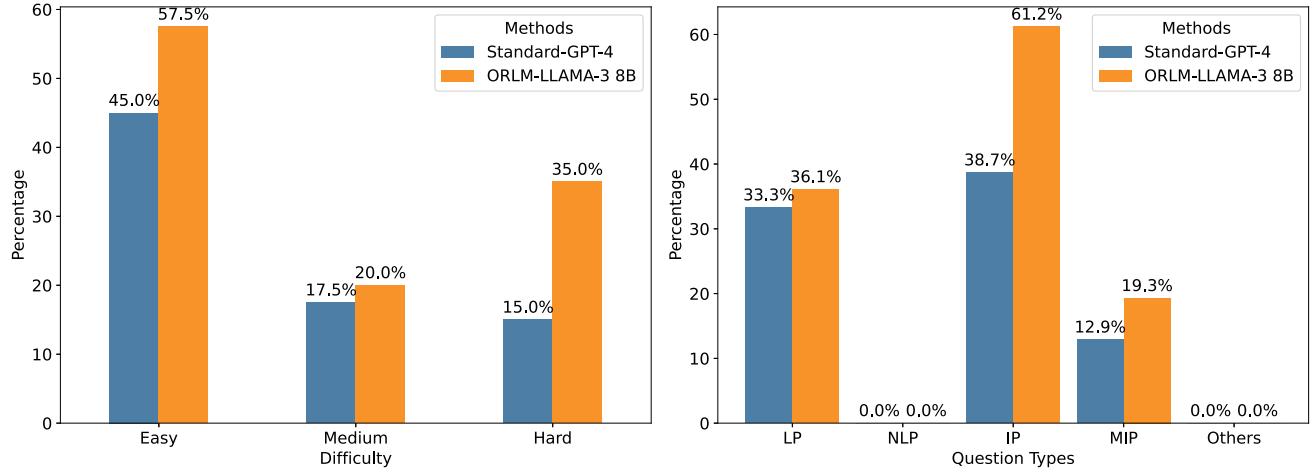
compare the accuracy rate of ORLM-LLaMA-3-8B and Standard-GPT-4 on IndustryOR, as shown in Figure 10. ORLM-LLaMA-3-8B shows superior performance over Standard-GPT-4 across all difficulty levels, especially in the hard category. Regarding different question types, ORLM-LLaMA-3-8B outperforms Standard-GPT-4 in linear programming, integer programming, and mixed-integer programming. Both models perform poorly in nonlinear programming and other rare question types. This is partly due to the fact that the seed data set collected contains too few examples of nonlinear and other types of optimization modeling, resulting in a low percentage of that type in the training data set, and partly due to the inherent complexity and scarcity of these types of questions. Overall, the OR-INSTRUCT data prove effective in enhancing *Comprehensive Coverage* across various question types and difficulty levels.

Figure 9. (Color online) Comparison of ORLM Before and After Adding Specific Data for Training and GPT-4 Responses; the Code Part Is Omitted Here

GPT-4	General ORLM	Enhanced ORLM
<p>Mathematical Model: To solve this problem, we can formulate it as a linear programming problem. We will define the decision variables, objective function, and constraints, then use a solver to find the optimal solution.</p> <p>Decision Variables:</p> <ul style="list-style-type: none"> ▪ x_1: Number of toy trucks produced. ▪ x_2: Number of toy airplanes produced. ▪ x_3: Number of toy boats produced. ▪ x_4: be the number of toy trains produced. <p>Objective Function:</p> <ul style="list-style-type: none"> ▪ Maximize profit: Profit = $5x_1 + 10x_2 + 8x_3 + 7x_4$ <p>Constraints:</p> <ol style="list-style-type: none"> 1. Wood constraint: $12x_1 + 20x_2 + 15x_3 + 10x_4 \leq 890$ 2. Steel constraint: $6x_1 + 3x_2 + 5x_3 + 4x_4 \leq 500$ 3. If trucks are manufactured, trains will not be manufactured: $x_1 \cdot x_4 = 0$ 4. If boats are manufactured, airplanes will also be manufactured: $x_3 \leq x_2$ 5. The number of toy boats manufactured cannot exceed the number of toy trains manufactured: $x_3 \leq x_4$ 6. Non-negativity constraints: $x_1, x_2, x_3, x_4 \geq 0$ <p>Non-negativity Constraints: $x_1, x_2, x_3, x_4 \geq 0$, and they should be integers to ensure practical integer solutions.</p>	<p>Mathematical Model: To solve the optimization problem for the toy manufacturer, we constructed a linear programming model and used the 'copypt' library to solve it. This model helps us maximize profit while meeting production conditions and resource constraints. Below are the detailed mathematical model and the solution script:</p> <p>Decision Variables:</p> <ul style="list-style-type: none"> ▪ x_1: Number of toy trucks produced. ▪ x_2: Number of toy airplanes produced. ▪ x_3: Number of toy boats produced. ▪ x_4: Number of toy trains produced. <p>Objective Function:</p> <ul style="list-style-type: none"> ▪ Maximize profit: $Z = 5x_1 + 10x_2 + 8x_3 + 7x_4$ ▪ This function calculates the total profit based on the profit and quantity of each toy. <p>Constraints:</p> <ol style="list-style-type: none"> 1. Wood usage limit: $12x_1 + 20x_2 + 15x_3 + 10x_4 \leq 890$ 2. Steel usage limit: $6x_1 + 3x_2 + 5x_3 + 4x_4 \leq 500$ 3. Production conditions: 4. If trucks are produced, trains will not be produced: $x_1 \cdot x_4 \leq 1$ 5. If boats are produced, airplanes will also be produced: $x_3 \leq x_2$ 6. The number of toy boats produced cannot exceed the number of toy trains produced: $x_3 \leq x_4$ <p>Non-negativity Constraints: $x_1, x_2, x_3, x_4 \geq 0$, and they should be integers to ensure practical integer solutions.</p>	<p>Mathematical Model: To help the toy manufacturer maximize profit, we need to construct an integer programming model to determine the optimal production mix. Below is the model construction:</p> <p>Decision Variables:</p> <ul style="list-style-type: none"> ▪ x_T: Number of toy trucks to be produced. ▪ x_P: Number of toy planes to be produced. ▪ x_S: Number of toy ships to be produced. ▪ x_R: Number of toy trains to be produced. ▪ y_T: Whether to produce toy trucks ($1 = Yes, 0 = No$). ▪ y_P: Whether to produce toy planes ($1 = Yes, 0 = No$). ▪ y_S: Whether to produce toy ships ($1 = Yes, 0 = No$). ▪ y_R: Whether to produce toy trains ($1 = Yes, 0 = No$). <p>Objective Function: Maximize total profit: $Z = 5x_T + 10x_P + 8x_S + 7x_R$</p> <p>Constraints:</p> <ol style="list-style-type: none"> 1. Wood usage limit: $12x_T + 20x_P + 15x_S + 10x_R \leq 890$ 2. Steel usage limit: $6x_T + 3x_P + 5x_S + 4x_R \leq 500$ 3. Production condition 1: $y_T + y_R \leq 1$ (If trucks are produced, trains will not be produced) 4. Production condition 2: $y_P \geq y_S$ (If ships are produced, planes will also be produced) 5. Production condition 3: $x_S \leq x_R$ (The number of toy ships produced cannot exceed the number of toy trains produced) 6. Logical relationship for production: <ol style="list-style-type: none"> a) $x_T \leq M \cdot y_T$ b) $x_P \leq M \cdot y_P$ c) $x_S \leq M \cdot y_S$ d) $x_R \leq M \cdot y_R$ <p>Where M is a sufficiently large number to ensure that if a toy type is not selected for production, the corresponding production quantity is zero.</p>

Note. Incorrect responses are highlighted.

Figure 10. (Color online) Accuracy Rate of ORLM and GPT-4 on IndustryOR Across Different Difficulty Levels and Question Types



Note. IP, integer programming; MIP, mixed IP; NLP, nonlinear programming.

5.2. Ablation Study

5.2.1. Ablation Study on Data Synthesis Strategy. We first assess whether the data generated by OR-INSTRUCT truly lead to improvements in the Seed data, particularly in terms of the performance of the ORLM. Particularly, we compare the performance of fine-tuning on seed data alone with the results of fine-tuning on OR-INSTRUCT-generated data based on the same seed data. We observe that, with the exception of minor variations in simpler linear programming cases (MAMO-EasyLP), the OR-INSTRUCT-generated data yield a significant performance boost for ORLM compared to only using the seed data, as shown in Table 6.

5.2.2. Ablation Study on OR-INSTRUCT Augmentations. To further verify the effectiveness of the augmentations in OR-INSTRUCT, we conduct detailed ablation experiments to study the effects of altering objectives and constraints, rephrasing questions, and incorporating multiple modeling techniques. Specifically, we construct four distinct data sets, each comprising 3,000 instances, drawn from the OR-INSTRUCT data but employing different augmentation strategies. These data sets are then utilized to train the LLaMA-3-8B model, maintaining consistent hyperparameters across all experimental conditions. The results are presented in Table 7. Training data with all three augmentations (denoted as Full Augmentations) achieves a base

performance of 68.6% in micro average and 55.7% in macro average. Removing any of the three augmentations from the base setting leads to a performance drop across all benchmarks, both in micro and macro averages. Rephrasing questions seems slightly more important than the other two. Overall, the results show that all three augmentations contribute to general performance, proving their effectiveness in enhancing the *data diversity*.

5.2.3. Ablation Study on Question Types. ORLM can be tailored to specific domains within the field of optimization, depending on its initial training data set. *A key consideration in this process is whether incorporating data from other domains is necessary.* This investigation aims to provide valuable guidance for implementing ORLM in specialized areas.

We use linear programming as a case study, comparing ORLM's performance when trained solely on linear programming data versus a mixed data set including other problem types. Specifically, we construct two training sets, each comprising 7,049 instances: one set exclusively consisting of pure linear programming (LP) problems and the other comprising a diverse mixture of problem types.⁵ We fine-tune the Meta-LLaMA 8B model on both data sets, ensuring that hyperparameter settings remain consistent across experiments. The test performance is summarized in Table 8.

Table 6. Performance Comparison with Varying Data Sizes

Base model	Data size	NL4Opt (%)	MAMO EasyLP (%)	MAMO ComplexLP (%)	IndustryOR (%)	Micro Avg (%)	Macro Avg (%)
LLaMA-3-8B	32,481 (Full data)	85.7	82.3	37.4	38.0	71.4	60.8
LLaMA-3-8B	686 (seed data)	75.1	82.8	29.4	20.0	67.0	51.8

Table 7. Ablation Study on OR-INSTRUCT Augmentations

Method	NL4Opt (%)	MAMO EasyLP (%)	MAMO ComplexLP (%)	IndustryOR (%)	Micro Avg (%)	Macro Avg (%)
Full augmentations	78.3	80.6	43.1	21.0	68.6	55.7
w/o Altering Obj&Const	77.5	79.2	36.4	20.0	66.4	53.2
w/o Rephrasing Questions	74.2	77.3	41.1	15.0	65.1	51.9
w/o Multiple Modeling	78.3	78.0	38.8	18.0	66.2	53.2

As illustrated in Table 8, even with identical test sizes, the inclusion of mixed problem types significantly enhances the performance of the ORLM-LLaMA-3-8B across various data sets. Notably, when examining the tests focused on linear programming, specifically MAMO-EasyLP and MAMO-ComplexLP, it is evident that integrating additional problem types is more effective in enhancing the ORLM’s performance on challenging linear programming problems. A plausible explanation for this phenomenon is that the incorporation of a broader variety of problems effectively strengthens the generalization ability of the ORLM-LLaMA-3-8B, allowing it to capture a wider modeling capacity and consequently improve modeling performance (see Dong et al. 2023).

5.3. Scaling Laws

The concept of scaling laws is a key element in the development of large models. It refers to the phenomenon that the performance of large language models improves as their size and complexity increase. In this section, we will examine the impact of scaling laws on ORLMs from two dimensions: training data and model size.

From the perspective of training data, we begin with seed data and subsequently randomly sample a predetermined number of training data from the data generated by OR-INSTRUCT. Using the LLaMA-3-8B model as the base, we conduct training while keeping all other hyperparameters constant (as in Table 1), in order to observe the performance of ORLM-LLaMA-3-8B on various test sets. Regarding model size, we focus on the Qwen large model series, as it offers a more refined range of model sizes that are compatible with our hardware limitations (0.5B, 1.5B, 3B, 7B, 14B). We fix the training data set to be all data generated by OR-INSTRUCT. Under the condition of identical training hyperparameters, we observe the performance of different Qwen2.5 model sizes on various test sets. In our study, we use the overall macro average and micro average

across four test sets as metrics to evaluate the performance of the ORLMs, as visualized in Figure 11. More detailed results are provided in Online Appendix A.5.2.

The figure clearly shows an upward trend in ORLM’s accuracy on the test sets as either the model size or the data volume increases. However, the impact of these two types of scaling laws on ORLM’s performance differs. Specifically, increasing the training set size results in fluctuating performance gains from the macro average metric, this is because for simple test data sets (NL4Opt and MAMO-easyLP), ORLM quickly reaches a performance bottleneck and begins to fluctuate, causing instability in its accuracy across the entire data set. For micro average metric, the overall performance continues to exhibit a monotonic increase with the expansion of training data, approximately following a power-law trend. This indicates that increasing the amount of training data remains beneficial for improving the model’s capabilities.

Regarding the scaling law for model size, it is evident that increasing the model size significantly enhances ORLM’s accuracy. This increase not only yields steady incremental gains, but also follows a power-law trend, consistent with the findings reported by Kaplan et al. (2020). This indicates that the intrinsic capability of the base model establishes a higher performance ceiling for ORLMs in optimization modeling tasks.

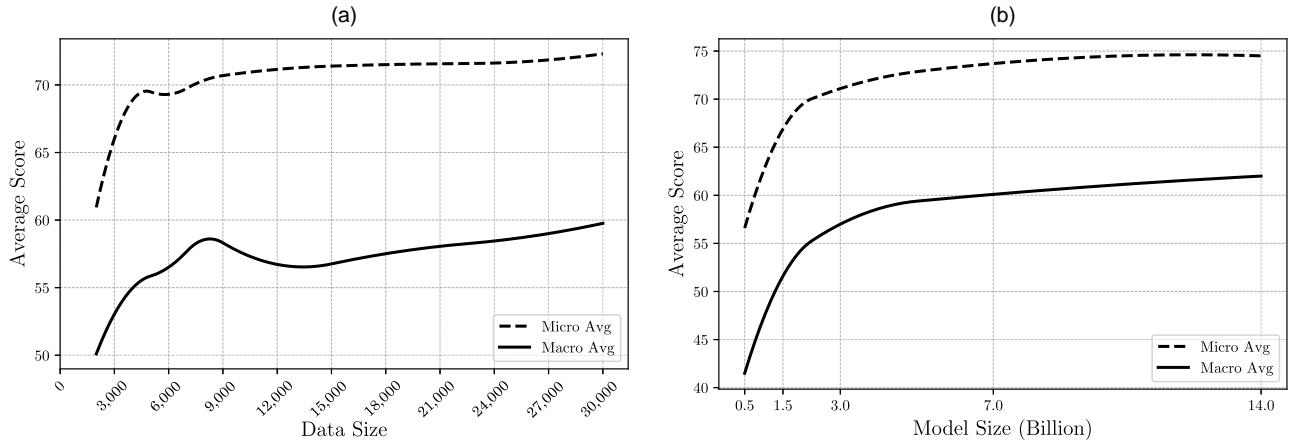
5.4. Inference Techniques

In previous model evaluations, we employed a greedy inference strategy to ensure the replicability and fairness of the results. However, using greedy sampling entails always selecting the word with the highest probability from the large model’s output during each generation. Although this approach guarantees deterministic results, it also limits the model’s diversity and creativity. To address this, in this section, we will utilize more common sampling techniques to assess the performance improvements of the ORLMs. Specifically, we adjust the following two parameters: First, the *temperature*

Table 8. Performance of ORLM-LLaMA-3-8B on Different Data Sets

Data set	Data size	NL4Opt (%)	MAMO EasyLP (%)	MAMO ComplexLP (%)	IndustryOR (%)	Micro Avg (%)	Macro Avg (%)
LP only	7,049	82.0	77.4	26.1	20.0	65.4	51.4
Mixed	7,049	86.5	81.2	34.6	21.0	69.8	55.8

Figure 11. Scaling Law Validation Across Different Data and Model Sizes



Notes. (a) Different sizes of training sets. (b) Different sizes of base models.

parameter controls the diversity of the generated text. Higher temperature values increase the randomness and creativity of the generated content, whereas lower values tend to produce more deterministic text. Second, the *top-P* parameter restricts the range of vocabulary choices considered by the model during generation. A higher *top-P* value means the model will consider a broader set of candidate words, thus enhancing the diversity of the generated content. To avoid excessive randomness that may lead to significant fluctuations in the accuracy of the generated results, we apply *Pass@k* ($k = 2, 4, 8$) strategy to enhance stability. Specifically, for each query, the large model generates top k candidate answers, and as long as at least one of the top k answers is correct, it is considered to have answered this question correctly (*Pass@k*). This approach helps improve the consistency and stability of the generated outcomes. The test results are detailed in Table 9.

Significant improvements have been observed in ORLMs across four test sets, with the overall micro average of *Pass@8* increasing by 17.09% relative to the previous version, whereas the macro average rose by 24.34%. A comparison with Table 2 reveals that, in terms of both macro average and micro average accuracy, the ORLM under the *Pass@8* strategy has already surpassed the

accuracy of senior undergraduate students and is approaching that of experts. This highlights the significant potential of ORLMs to serve as a viable alternative to human engineers. The high *Pass@8* score indicates that the correct answer is often found within the top eight solutions. However, the model's greedy decoding strategy remains insufficiently effective to consistently prioritize the optimal answer. The notable progress between the *Pass@1* and *Pass@8* scores suggests that although the model has the potential to generate correct solutions, its ranking capability is currently inadequate to reliably place the optimal solution in the first position. Fortunately, reinforcement learning may offer a promising solution to bridge the gap between *Pass@1* and *Pass@8*; we will discuss it in the future work (see Section 6.2).

5.5. Limitations Analysis

Building upon the previous discussion and results, this section concludes with an analysis of the current limitations of ORLM. First, using ORLM-LLaMA-3-8B as a case study, we present a statistical breakdown of error types under the greedy inference strategy, as shown in Table 10.

In the table, the "Overall Percentage" represents the occurrence frequency of each error type across all test

Table 9. Performance of ORLM-LLaMA-3-8B on Different Decoding Methods

Model	Inference settings	NL4Opt (%)	MAMO EasyLP (%)	MAMO ComplexLP (%)	IndustryOR (%)	Micro Avg (%)	Macro Avg (%)
ORLM-LLaMA-3-8B	<i>Pass@1</i> , temp = 0, top- p = 1.0	85.7	82.3	37.4	38.0	71.4	60.8
ORLM-LLaMA-3-8B	<i>Pass@2</i> , temp = 0.7, top- p = 0.95	88.6	83.7	49.8	40.0	75.6	65.5
ORLM-LLaMA-3-8B	<i>Pass@4</i> , temp = 0.7, top- p = 0.95	91.4	85.9	56.9	44.0	78.9	69.6
ORLM-LLaMA-3-8B	<i>Pass@8</i> , temp = 0.7, top- p = 0.95	93.0	88.4	72.1	49.0	83.6	75.6

Table 10. Error Frequency Across Different Data Sets

Error type	NL4Opt (%)	MAMO Easy (%)	MAMO Complex (%)	IndustryOR (%)	Overall percentage (%)	Error percentage (%)
Code error	1.04	0.00	16.59	31.00	5.51	19.77
Model error	12.46	17.64	45.97	31.00	22.28	80.23

sets, while the "Error Percentage" indicates the proportion of each error type within the total number of errors. The data reveal that ORLM has demonstrated considerable proficiency in utilizing COPT, achieving a pass rate of approximately 95% across more than 1,000 test cases. Consequently, the primary challenges lie in the optimization modeling phase of the process.

To analyze the types of errors occurring in ORLM modeling, we randomly selected 100 problems from those where ORLM failed. Following manual evaluation and statistical analysis, and after excluding errors caused by coding issues, we categorized the modeling-related errors into three primary types, as detailed in Table 11. Notably, the predominant cause of ORLM's modeling errors is "Low Model Completeness," which accounts for more than half of the observed errors. Further analysis reveals that this issue stems from the limitations in the expressive and learning capabilities of 8B-scale models. These models often produce overly simplified outputs in complex scenarios, reducing the intricacy of optimization modeling problems even when such scenarios are included in the training set. This highlights the importance of fine-tuning larger-scale models, a conclusion supported by our scaling law experiments.

Although ORLM still has certain limitations, as demonstrated by previous experimental results, we recognize substantial opportunities for improvement through the application of scaling laws, different inference techniques, and others. ORLM is still promising in addressing complex optimization modeling problems, even certain real-world industrial problems within specific domains. In the next section, we will demonstrate how ORLM can be applied to practical scenarios such as industrial production, education, and other real-world applications.

6. Future Direction

In previous sections, we conduct a detailed analysis of ORLM's strengths and limitations within the field of

optimization modeling. In this section, we build on the above experimental results to discuss the practical applications of ORLM and potential directions for future improvement.

6.1. Potential Applications

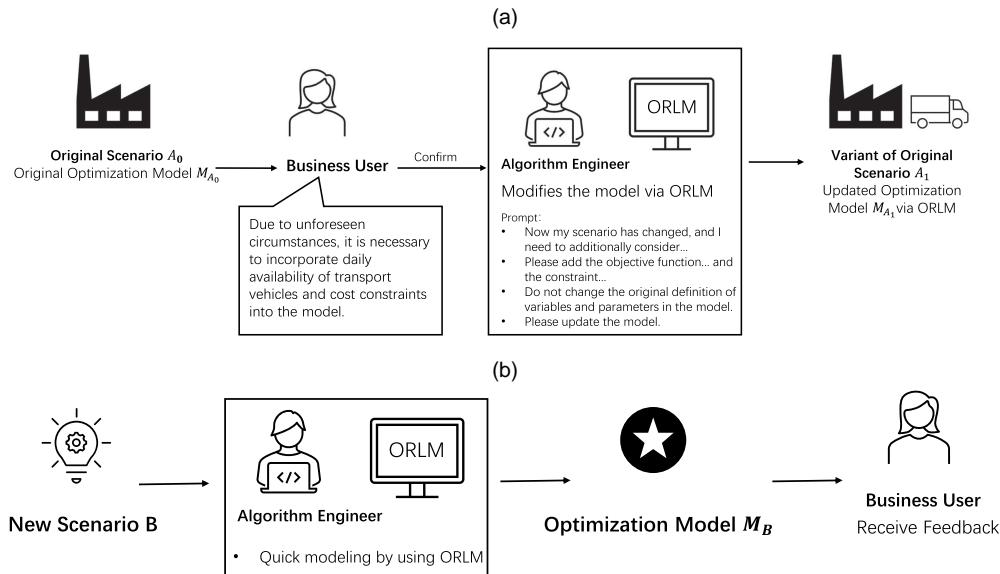
Large language models are increasingly influential across various industries, and the rise of privatized and domain-specific LLMs presents new topics for management. This work represents the first attempt to fine-tune open-source LLMs specifically for optimization modeling. The ORLM not only surpasses the performance of cutting-edge proprietary models, but also ensures privacy, security, and stability for commercial applications. Additionally, its 7B size allows for private deployment on personal computers with only 16GB of VRAM, making operations research more accessible to a wider audience. We hope that this work could shed light on the deep integration of large language models with operations research and management science. Potential applications are listed as follows:

1. Revolutionizing the role of operations research in industry: In recent years, operations research has rapidly permeated various industrial sectors, supporting companies in making more informed and effective decisions. However, optimization modeling tasks in industry often encounter two critical challenges: (1) Optimization models are typically not robust enough to withstand changes in external conditions. When the environment shifts, models must be quickly adjusted to help companies make timely decisions under new circumstances. (2) Optimization projects typically rely on business experts to specify goals and constraints, whereas algorithm engineers are responsible for developing and solving the models. For many specialized industrial problems, communication between these two roles can be costly. Algorithm engineers may struggle to grasp the intricacies of business logic, and

Table 11. Analysis of Main Mistakes in Optimization Modeling

Modeling error type	Description	Proportion (%)
Semantic misunderstanding	Misunderstand the problem that the optimization model needs to solve.	13.40
Errors in objective/constraint translation	Correctly understand the requirements but make errors when formulating the optimization model.	30.30
Low model completeness	Ignore some constraints or implicit conditions of the real problem, simplifying the complex problem.	56.30

Figure 12. Utilize ORLM for Adaptive Modeling in Enterprise Decision Making



Notes. (a) In the course of enterprise development, the constraints and objectives of operational contexts continually evolve, necessitating swift adjustments to existing algorithmic models to meet business demands. (b) For newly arising decision-making problems, enterprises can leverage ORLM for rapid modeling and experiment with small data samples. This approach enables enterprises to respond swiftly to market changes.

business experts often lack technical modeling expertise.

The introduction of ORLM could help address many of these challenges. Particularly, companies with extensive training data can customize ORLMs to their domain based on OR-INSTRUCT. As demonstrated in Tables 2 and 3, ORLM significantly enhances optimization modeling capabilities, while retaining the broad knowledge acquired during pretraining. This allows it to quickly grasp industry-specific terminology and facilitate effective communication between business experts and algorithm engineers. ORLM operates with remarkable efficiency, generating an initial solution within seconds that both engineers and business teams can refer to. Given ORLM's strong performance on Pass@8 metrics, it can also produce multiple solutions, supporting selection and comparison, which accelerates project iterations. Two representative workflows are illustrated in Figure 12.

Moreover, ORLM can seamlessly modify existing optimization models by adding new business constraints, creating preliminary models, and generating initial solution code, which engineers can then validate. This capability not only reduces delivery costs, but also improves accuracy on complex tasks.

Beyond these capabilities, ORLM has the potential to leverage Retrieval-Augmented Generation and similar technologies to incorporate internal company documents, evolving into a domain-specific optimization expert well-versed in the company's field. This feature allows ORLM to support knowledge transfer and

updates, transforming traditional project reports into valuable data assets for future model training. In addition to project delivery, ORLM can also be applied in training junior algorithm engineers in modeling techniques, further enhancing its value within the enterprise.

2. Operations research education: The outstanding performance of ORLM in fundamental optimization modeling problems makes it sufficiently capable of addressing the modeling challenges encountered by beginners in operations research. Instructors can utilize ORLM for efficient instructional guidance, allowing students to quickly derive reference solutions and processes for unfamiliar operations research modeling problems, thereby enhancing teaching efficiency and quality.

3. Mathematical modeling competitions: As one of the most influential competitions, mathematical modeling contests aim to cultivate students' abilities to solve problems using mathematical knowledge. Since 2023, several mathematical modeling competitions have begun to acknowledge the significant role of generative AI in modeling, permitting its use as an aid (e.g., MCM 2024). The capabilities of ORLM position it as a powerful tool in such competitions.

4. Reducing the learning cost of solvers: Solvers are a core component of optimization modeling problems, yet the diverse programming syntax of different solvers significantly increases the learning cost for users, hindering their widespread adoption. The emergence of ORLM marks a pivotal shift in this regard. Notably, Table 10 indicates that ORLM rarely encounters errors

due to coding issues (95% accuracy rate), even with less common syntax such as that specific to COPT. This suggests that algorithm engineers can confidently develop reliable operations research models and rely on ORLM to translate them into corresponding solver code, alleviating the need to extensively learn solver syntax.

6.2. Future Research Directions

In Section 5, we conduct extensive experiments to analyze the current limitations of ORLM and to investigate the underlying causes. Based on our findings, we propose directions for future improvements, calling for community attention and collaboration to drive progress in this field:

- **Incorporating sophisticated techniques:** In Section 5.4, we observe that although ORLM can generate correct answers, it struggles with ranking them effectively. Incorporating reinforcement learning techniques could help narrow the gap between Pass@1 and Pass@8 scores. As noted by Sutton and Barto (2018), one of the primary goals of reinforcement learning is to align the policy with the Bellman optimality policy, effectively optimizing ranking quality. In this context, relevant methods like reinforcement learning from human feedback (RLHF), direct preference optimization, and Kahneman-Tversky optimization aim to minimize the disparity between the model's current greedy decoding policy and the Bellman optimality policy, thereby enhancing the model's ranking accuracy and moving Pass@1 closer to Pass@8. In addition, approaches such as multiagent collaboration and internal chain of thought structures similar to o1 (OpenAI 2024) could further enhance the performance of current large models in optimization modeling tasks.

- **Data set construction:** Although this paper presents a data synthesis strategy, it is insufficient for reinforcement learning techniques like RLHF, which require training data in the form of a preference list. For each question, multiple responses must be ranked according to preference. Furthermore, in the context of optimization problems, incorporating actual optimal solutions into the training set would enable a wider range of techniques. Constructing such data sets may require users' feedback or the development of new data synthesis methods to better meet these needs.

- **Data mining and exploration:** As shown in Figure 11, we have demonstrated that scaling laws significantly benefit ORLM. However, for a fixed model size, the performance gains from additional data become increasingly marginal at later stages. This raises the question of whether it is possible to identify an optimized, minimal data set or develop efficient data synthesis strategies that can balance ORLM's performance with the associated training costs. Existing studies (e.g., Chen et al. 2023c) indicate that only a small portion of a complete data set is often sufficient to achieve

comparable performance. Therefore, developing a tailored data exploration strategy for optimization problem modeling, which minimizes the reliance on large-scale data resources for enhancing large models, represents an important direction for future research.

7. Concluding Remarks

In this paper, we propose a new path for the training of open-source large language models specifically tailored for optimization modeling. We characterize four critical requirements essential for the training data set of optimization modeling LLMs and develop OR-INSTRUCT, a semiautomated framework for generating synthetic data that meets these specific needs. Additionally, we introduce the IndustryOR benchmark, the first of its kind in the industry. We utilize the OR-INSTRUCT data to train open-source LLMs with approximately 7 billion parameters, which significantly enhances their optimization modeling capabilities, achieving competitive performance across all test data sets.

This paper also identifies the current limitations of ORLM, such as its overly simplistic output for complex problems, weak ability to rank optimal solutions, and poor learning capability from data. Our experiments suggest that ORLM can be further enhanced by leveraging scaling laws or incorporating methods such as reinforcement learning, potentially reaching expert-level performance. We hope these findings provide valuable insights for the future advancement of ORLM's applications in industry and open promising avenues for subsequent research.

Finally, we note that during the revision process of this paper, numerous high-performing large models have emerged, such as DeepSeek-R1 and Grok-3. Although these new models demonstrate impressive capabilities, further enhancing their performance in optimization modeling requires additional refinement. The framework proposed in this paper offers a valuable reference for such improvements, including synthesizing high-quality data to strengthen modeling comprehension or considering reinforcement learning strategies to align ranking capabilities. By leveraging our framework as a promising starting point, the inherent strengths of large models can be more effectively translated into optimization modeling capabilities.

Acknowledgments

The authors thank the area editor, the associate editor, and two referees for their diligent reading and thoughtful comments that significantly improved the paper. Chenyu Huang and Zhengyang Tang contributed equally to this work. We publicly release all relevant code, models, and data sets associated with this work:

- Code repository: <https://github.com/Cardinal-Operations/ORML>.
- Model and data sets: <https://huggingface.co/CardinalOperations>.

Endnotes

- ¹ Prior to the final fine-tuning, we conducted a grid search with learning rates within {2e-5, 5e-5, 7e-5, 3e-6, 5e-6, 7e-6}, batch sizes within {64, 128, 256, 512}, and training epochs within {1, 2, 3} to determine the optimal settings.
- ² Because of the severe scarcity of API resources, we are unable to conduct large-scale modeling tests on many state-of-the-art LLMs, such as DeepSeek-R1 (671B).
- ³ Micro average aggregates the outcomes of all classes to compute the metrics, heavily weighing the classes with more instances. This method is effective for evaluating the overall effectiveness of the model, particularly in data sets where some classes significantly outnumber others.
- ⁴ Macro average calculates individual metrics for each class without regard to class frequency and then averages these metrics, treating all classes with equal importance. This approach highlights model performance on minority classes and is beneficial for assessing model fairness across diverse class distributions.
- ⁵ This data set comprises 35% linear programming, 40% mixed integer programming, and 25% other types.

References

- Adams T, Ferrucci A, Carvalho P, Em S, Whitley B, Cecchi R, Hicks T, et al. (2023) Advanced analytics drives reengineering of field operations for the 2020 U.S. Census. *INFORMS J. Appl. Anal.* 53(1):47–58.
- AhmadiTeshnizi A, Gao W, Udell M (2024) OptiMUS: Scalable optimization modeling with (MI)LP solvers and large language models. Preprint, submitted February 15, <https://arxiv.org/abs/2402.10172>.
- AI@Meta (2024) Llama 3 model card. Accessed May 1, 2024, https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.
- Alzubaidi L, Bai J, Al-Sabaawi A, Santamaría J, Albahri AS, Al-Dabbagh BSN, Fadhel MA, et al. (2023) A survey on deep learning tools dealing with data scarcity: Definitions, challenges, solutions, tips, and applications. *J. Big Data* 10(1):46.
- Aouad A, Désir A (2022) Representing random utility choice models with neural networks. Preprint, submitted July 26, <https://arxiv.org/abs/2207.12877>.
- Berry J, Houston K (1995) *Mathematical Modelling* (Elsevier Ltd., Oxford, UK).
- Chen H, Constante-Flores GE, Li C (2023a) Diagnosing infeasible optimization problems using large language models. Preprint, submitted August 23, <https://arxiv.org/abs/2308.12923>.
- Chen Y, Kelly BT, Xiu D (2022a) Expected returns and large language models. Preprint, submitted November 22, <https://ssrn.com/abstract=4416687>.
- Chen Y, Gao W, Ge D, Ye Y (2023b) Pre-trained mixed integer optimization through multi-variable cardinality branching. Preprint, submitted May 21, <https://arxiv.org/abs/2305.12352v1>.
- Chen Y, Kang Y, Chen Y, Wang Z (2020) Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing* 399:491–501.
- Chen T, Chen X, Chen W, Heaton H, Liu J, Wang Z, Yin W (2022b) Learning to optimize: A primer and a benchmark. *J. Machine Learn. Res.* 23(189):1–59.
- Chen H, Zhang Y, Zhang Q, Yang H, Hu X, Ma X, Yanggong Y, Zhao J (2023c) Maybe only 0.5% data is needed: A preliminary exploration of low training data instruction tuning. Preprint, submitted May 16, <https://arxiv.org/abs/2305.09246>.
- Cohen J (2013) *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed. (Routledge, Abingdon, UK).
- Dai JG, Gluzman M (2022) Queueing network controls via deep reinforcement learning. *Stochastic Systems* 12(1):30–67.
- DeepSeek AI (2024) DeepSeek-v2: A strong, economical, and efficient mixture-of-experts language model. Accessed June 6, 2024, <https://github.com/deepseek-ai/DeepSeek-V2>.
- Ding B, Qin C, Zhao R, Luo T, Li X, Chen G, Xia W, Hu J, Luu AT, Joty S (2024) Data augmentation using LLMs: Data perspectives, learning paradigms and challenges. Preprint, submitted July 2, <https://arxiv.org/abs/2403.02990>.
- Dong G, Yuan H, Lu K, Li C, Xue M, Liu D, Wang W, Yuan Z, Zhou C, Zhou J (2023) How abilities in large language models are affected by supervised fine-tuning data composition. Preprint, submitted October 9, <https://arxiv.org/abs/2310.05492v1>.
- Ge D, Huangfu Q, Wang Z, Wu J, Ye Y (2022) Cardinal Optimizer (COPT) user guide. Accessed June 6, 2024, <https://guide.coap.online/copt/en-doc>.
- Gijsbrechts J, Boute RN, Van Mieghem JA, Zhang DJ (2022) Can deep reinforcement learning improve inventory management? Performance on lost sales, dual-sourcing, and multi-echelon problems. *Manufacturing Service Oper. Management* 24(3):1349–1368.
- Grattafiori A, Dubey A, Jauhri A, Pandey A, Kadian A, Al-Dahle A, Letman A, et al. (2024) The Llama 3 herd of models. Preprint, submitted November 23, <https://arxiv.org/abs/2407.21783>.
- Guo D, Yang D, Zhang H, Song J, Zhang R, Xu R, Zhu Q, et al. (2025) DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. Preprint, submitted January 22, <https://arxiv.org/abs/2501.12948>.
- Hans M (2002) Benchmark for optimization software. Accessed June 6, 2024, <http://plato.asu.edu/bench.html>.
- Hogg R, Craig A, McKean J (2019) *Introduction to Mathematical Statistics*, 8th ed. (Pearson, London).
- Huang X, Shen Q, Hu Y, Gao A, Wang B (2024a) Mamo: A mathematical modeling benchmark with solvers. Preprint, submitted June 30, <https://arxiv.org/abs/2405.13144v2>.
- Huang K, Wang Y, Zhu F, Chen X, Xing C (2024b) *Beyond AI: ChatGPT, Web3, and The Business Landscape of Tomorrow* (Springer, Cham, Switzerland).
- INFORMS (2013) 2013 Franz Edelman finalists. Accessed June 6, 2024, https://www.informs.org/content/download/279469/2673366/file/2013edelman_finalWEB.pdf.
- Jiang B, Xie Y, Hao Z, Wang X, Mallick T, Su WJ, Taylor CJ, Roth D (2024) A peek into token bias: Large language models are not yet genuine reasoners. Preprint, submitted October 4, <https://arxiv.org/abs/2406.11050>.
- Jiang AQ, Sablayrolles A, Mensch A, Bamford C, Chaplot DS, Casas D, Bressand F, et al. (2023) Mistral 7b. Preprint, submitted October 10, <https://arxiv.org/abs/2310.06825>.
- Kani N, Gangwar N (2022) Tagged input and decode all-at-once strategy. Accessed May 1, 2024, <https://github.com/MLPgroup/nl4opt-generation>.
- Kaplan J, McCandlish S, Henighan T, Brown T, Chess B, Child R, Gray S, Radford A, Wu J, Amodei D (2020) Scaling laws for neural language models. Preprint, submitted January 23, <https://arxiv.org/abs/2001.08361>.
- Khalil E, Dai H, Zhang Y, Dilkina B, Song L (2017) Learning combinatorial optimization algorithms over graphs. *Adv. Neural Inform. Processing Systems* 30:6351–6361.
- Lee J, Xu C, Xie Y (2024) Transformer conformal prediction for time series. Preprint, submitted June 8, <https://arxiv.org/abs/2406.05332>.
- Li Q, Zhang L, Mak-Hau V (2023a) Synthesizing mixed-integer linear programming models from natural language descriptions. Preprint, submitted November 26, <https://arxiv.org/abs/2311.15271>.
- Li B, Mellouk K, Zhang B, Pathuri J, Menache I (2023b) Large language models for supply chain optimization. Preprint, submitted July 13, <https://arxiv.org/abs/2307.03875>.

- Liu Z, Clifton J, Laber EB, Drake J, Fang EX (2023) Deep spatial Q-learning for infectious disease control. *J. Agricultural Biological Environmental Statist.* 28(4):749–773.
- Marion M, Üstün A, Pozzobon L, Wang A, Fadaee M, Hooker S (2023) When less is more: Investigating data pruning for pre-training LLMs at scale. Preprint, submitted September 8, <https://arxiv.org/abs/2309.04564>.
- Mistral AI (2024) Frontier AI in your hands—Mistral NeMo. *Mistral AI News*. Accessed May 1, 2024, <https://mistral.ai/news/mistral-nemo/>.
- Nair V, Bartunov S, Gimeno F, Von Glehn I, Lichocki P, Lobov I, O'Donoghue B, et al. (2020) Solving mixed integer programs using neural networks. Preprint, submitted December 23, <https://arxiv.org/abs/2012.13349v1>.
- OpenAI (2024) OpenAI o1 technical report. *OpenAI Technical Reports*. Accessed December 5, 2024, <https://openai.com/index/openai-o1-system-card/>.
- Parmentier A (2022) Learning to approximate industrial problems by operations research classic problems. *Oper. Res.* 70(1):606–623.
- Pawlak TP, Krawiec K (2017) Automatic synthesis of constraints from examples using mixed integer linear programming. *Eur. J. Oper. Res.* 261(3):1141–1157.
- Pawlak TP, O'Neill M (2021) Grammatical evolution for constraint synthesis for mixed-integer linear programming. *Swarm Evolutionary Comput.* 64:100896.
- Prasath G, Karande S (2023) Synthesis of mathematical programs from natural language specifications. Preprint, submitted March 30, <https://arxiv.org/abs/2304.03287>.
- Qi M, Shi Y, Qi Y, Ma C, Yuan R, Wu D, Shen ZJ (2023) A practical end-to-end inventory management model with deep learning. *Management Sci.* 69(2):759–773.
- Qin H, Xiao J, Ge D, Xin L, Gao J, He S, Hu H, Carlsson JG (2022) JD.com: Operations research algorithms drive intelligent warehouse robots to work. *INFORMS J. Appl. Anal.* 52(1):42–55.
- Ramamonjiso R, Li H, Yu TTL, He S, Rengan V, Banitalabi-Dehkordi A, Zhou Z, Zhang Y (2022) Augmenting operations research with auto-formulation of optimization models from problem descriptions. *Proc. 2022 Conf. Empirical Methods Natural Language Processing EMNLP 2022 Industry Track* (Association for Computational Linguistics, Stroudsburg, PA), 29–62.
- Ramamonjiso R, Yu T, Li R, Li H, Carenini G, Ghaddar B, He S, et al. (2023) NLAOpt competition: Formulating optimization problems based on their natural language descriptions. *NeurIPS 2022 Competition Track* (PMLR, New York), 189–203.
- Shao Z, Wang P, Zhu Q, Xu R, Song J, Zhang M, Li Y, Wu Y, Guo D (2024) DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. Preprint, submitted April 27, <https://arxiv.org/abs/2402.03300>.
- Shinn N, Labash B, Gopinath A (2023) Reflexion: An autonomous agent with dynamic memory and self-reflection. Preprint, submitted March 20, <https://arxiv.org/abs/2303.11366v1>.
- Sroka D, Pawlak TP (2018) One-class constraint acquisition with local search. *Proc. Genetic Evolutionary Comput. Conf.* (Association for Computing Machinery, New York), 363–370.
- Sutton RS, Barto AG (2018) *Reinforcement Learning: An Introduction*, 2nd ed. (MIT Press, Cambridge, MA).
- Tang Z, Zhang X, Wan B, Wei F (2024) MathScale: Scaling instruction tuning for mathematical reasoning. Preprint, submitted March 5, <https://arxiv.org/abs/2403.02884>.
- Taori R, Gulrajani I, Zhang T, Dubois Y, Li X, Guestrin C, Liang P, Hashimoto TB (2023) Stanford alpaca: An instruction-following Llama model. Accessed July 17, 2024, https://github.com/tatsu-lab/stanford_alpaca.
- Trinh TH, Wu Y, Le QV, He H, Luong T (2024) Solving olympiad geometry without human demonstrations. *Nature* 625(7995): 476–482.
- Wang X-J, Liu T, Fan W (2023a) TGVx: Dynamic personalized POI deep recommendation model. *INFORMS J. Comput.* 35(4): 786–796.
- Wang M, Zhang DJ, Zhang H (2024) Large language models for market research: A data-augmentation approach. Preprint, submitted December 26, <https://arxiv.org/abs/2412.19363v1>.
- Wang H, Cai Z, Li X, Talluri K (2023b) A neural network based choice model for assortment optimization. Preprint, submitted August 10, <https://arxiv.org/abs/2308.05617>.
- Wang H, Liu C, Xi N, Qiang Z, Zhao S, Qin B, Liu T (2023c) Huatuo: Tuning LLaMA model with Chinese medical knowledge. Preprint, submitted April 14, <https://arxiv.org/abs/2304.06975>.
- Wang Y, Kordi Y, Mishra S, Liu A, Smith NA, Khashabi D, Hajishirzi H (2022) Self-instruct: Aligning language models with self-generated instructions. Preprint, submitted December 20, <https://arxiv.org/abs/2212.10560v1>.
- Wang Y, Ivison H, Dasigi P, Hessel J, Khot T, Chandu KR, Wadden D, et al. (2023d) How far can camels go? Exploring the state of instruction tuning on open resources. Preprint, submitted October 30, <https://arxiv.org/abs/2306.04751>.
- Xiao Z, Zhang D, Wu Y, Xu L, Wang YJ, Han X, Fu X, et al. (2023) Chain-of-experts: When LLMs meet complex operations research problems. *Twelfth Internat. Conf. Learn. Representations (ICLR, Appleton, WI)*.
- Xu L, Zhang J, Li B, Wang J, Cai M, Zhao WX, Wen JR (2024) Prompting large language models for recommender systems: A comprehensive framework and empirical analysis. Preprint, submitted January 10, <https://arxiv.org/abs/2401.04997v1>.
- Yang A, Yang B, Hui B, Zheng B, Yu B, Zhou C, Li C, et al. (2024) Qwen2 technical report. Preprint, submitted September 10, <https://arxiv.org/abs/2407.10671>.
- Zaremba W, Brockman G, OpenAI (2021) OpenAI codex. *OpenAI Blog*. Accessed August 1, 2024, <https://openai.com/blog/openai-codex/>.
- Zhang Z, Zhou Y, Ji X (2021) Model-free reinforcement learning: From clipped pseudo-regret to sample complexity. *Internat. Conf. Machine Learn. (JMLR, Cambridge, MA)*, 12653–12662.

Chenyu Huang is a PhD student in management science and engineering at the Research Institute for Interdisciplinary Sciences, Shanghai University of Finance and Economics. His main research interests include large language models for operations research, revenue management, and robust optimization.

Zhengyang Tang is a PhD student in the School of Science and Engineering at the Chinese University of Hong Kong, Shenzhen. His research focuses on advancing large language models with an emphasis on mathematical reasoning, optimization modeling, and developing scalable oversight techniques to improve AI systems' performance and reliability.

Shixi Hu serves as the head of the OR platform team at Cardinal Operations. She obtained her master's degree in management science and engineering from Shanghai University of Finance and Economics. Her primary responsibilities include integrating operations research with large language models and driving product development that leverages both technologies for enterprise applications.

Ruoqing Jiang is a PhD candidate in economics at Columbia University and will join the School of Economics and Management at Tsinghua University as an assistant professor. His research interests lie at the intersection of econometrics and statistical machine learning, with applications to high-dimensional data and decision making under uncertainty.

Xin Zheng is a master's student in the Departments of Computer Science and Economics at Duke University. Her primary research interests include interpretable machine learning and the application of generative models in economic research.

Dongdong Ge is a professor and the dean of the Institute of Intelligent Computing at the Shanghai Jiao Tong University. His

main research interests lie in large-scale optimization algorithms and their applications in supply chain and intelligent industry. He is also the chief scientist and a co-founder of Cardinal Operations (shanshu.ai) and leads the development of mathematical programming solver COPT.

Benyou Wang is an assistant professor in the School of Data Science at the Chinese University of Hong Kong, Shenzhen. His research interests focus on developing specialized large language models for healthcare (HuatuoGPT), multilingual AI systems (AceGPT, Phoenix), reasoning capabilities in natural language

processing, and multimodal learning approaches that integrate text with visual information for medical applications.

Zizhuo Wang is a professor at the School of Data Science and the School of Management and Economics (by courtesy) at The Chinese University of Hong Kong, Shenzhen. He is also the co-founder and CTO of Cardinal Operations. His primary research interests include data-driven decision making and pricing and revenue management. He is also the director of Guangdong Key Laboratory of Mathematical Foundation for Artificial Intelligence.

Copyright of Operations Research is the property of INFORMS: Institute for Operations Research & the Management Sciences and its content may not be copied or emailed to multiple sites without the copyright holder's express written permission. Additionally, content may not be used with any artificial intelligence tools or machine learning technologies. However, users may print, download, or email articles for individual use.