Crossroads of Optimization under Uncertainty: How to Choose the Optimal Model

Chengxi She[♠*], **Zhiqiang Chen**[♠], **Xingyu Lu**[♣], **Caihua Chen**[♠†], **Piaoyang Zhao**[♠], **Xuedong Wang**[♠],
[♠]Nanjing University, Nanjing, China
[♣]Ant Group, Hangzhou, China
522023150104@smail.nju.edu.cn, sing.lxy@antgroup.com, chchen@nju.edu.cn

Abstract

To address two correlated question in Optimization under Uncertainty (OuU): ***Expertise Threshold*** and ***Selection Conundrum***, we propose LLM4OuU, a multi-agent framework that automates both the modeling and solving of six distinct types of uncertainty models and generates mapping pairs to explore the potential relationship between optimization problems and optimal models. Firstly, we decompose the complex modeling process into five sequential steps and design specialized LLM agents combining high-level domain expertise. Secondly, we introduce a hybrid dataset spanning various industries based on Retrieval-Augmented Generation (RAG) to benchmark performance. Extensive experiments demonstrate that LLM4OuU achieves superior performance compared to baselines, even reaching up to 99% on specific model types. Finally, we establish a mapping from problem features to optimal models, with correlation analysis revealing that not only data scale but also the specific scenario significantly influence model selection. **Data and code** are available: https://github.com/ChengxiSHE/LLM4OuU.git.

1 Introduction

Optimization under Uncertainty (OuU) has found significant success in various critical applications, such as transportation (Barnhart et al., 2003; Kleywegt et al., 2002), supply chain (Lambert and Cooper, 2000), finance (Fabozzi et al., 2010), and healthcare (Bertsimas et al., 2008). Despite its crucial practicality, OuU remains less widely adopted globally compared with related fields like machine learning (ML). This disparity stems mainly from ***Expertise Threshold***. In the past, solving each OuU problem requires separately manual modeling, algorithm design, data processing, and coding, typically requiring specialized expertise, i.e. optimization and statistics. In contrast, the widespread availability of ML packages has made ML widely accessible to everybody.

Furthermore, a follow-up question arises: ***Selection Conundrum***. There are several distinct approaches and models emerging, such as Sample Average Approximation (SAA), Robust Optimization (RO), and Distributionally Robust Optimization (DRO). Each of these methods has its own unique advantages and disadvantages, along with inherent limitations in practical applicability. Thus, how to select appropriate uncertainty models according to different scenarios among various models? The primary limitation lies in the lack of mapping mechanism from features of problem to the optimal model, which caused by the scarcity of substantial, high-quality mapping pairs.

Recently, the advancement of large language models (LLMs) presents novel and effective prescriptions, which exhibit fabulous performance in text summary, question answering, automated modeling, and coding (Zhang et al., 2024; Ramamonjison et al., 2023; Nam et al., 2024). Specifically, LLMs enable automated modeling of OR problems by leveraging their extensive knowledge base and customized prompts (Romera-Paredes et al., 2024). However, existing works mainly focus on deterministic optimization problem, while the core challenge we tackle lies in the more complex paradigm of data-driven optimization under uncertainty.

Therefore, we construct a multi-agent automated modeling framework applicable to various OuU models, and further generate numerous mapping pairs to explore the potential relationship between optimization problems and optimal models. Our key contributions are summarized as follows:

- We propose LLM4OuU, a specialized multi-agent framework for automated modeling of OuU. To the best of our knowledge, this is the first LLMs-based framework capable of han-

[*]Work done during the academic internship at Ant Group.
[†]Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2026, pages 3508–3539

dling the modeling process for complex data-driven optimization under uncertainty. It incorporates six specialized sub-agents in model transformation covering the full spectrum of mainstream uncertainty paradigms (SAA, RO, DRO).

- We construct a comprehensive dataset covering various industries as benchmark. By integrating over 2,000 OR-related papers via RAG with structured numerical data from open-source platforms, we generate high-quality problem descriptions containing five essential modeling elements. This pipeline with self-correcting repair and retry mechanism ensures fabulous automated modeling performance, even reaching up to 99% on specific model types.
- We generate numerous high-quality mapping pairs datasets and quantitatively validate the intrinsic mapping relationship between problem features and optimal uncertainty models. We provide empirical evidence that not only is *Data Scale* a dominant factor affecting the model selection process, but *Industry* and *Scenario* also contribute to this procedure.

2 Related Work

The related work is divided into two main aspects: (1) optimization under uncertainty, and (2) leveraging LLM modeling and solving OR problems.

2.1 Optimization under Uncertainty

OuU is a critical paradigm for addressing real-world systems where parameters are subject to fluctuations and data limitations (Sadana et al., 2025). Decision-makers have to infer uncertain parameters from relevant side information before make decisions. In this field, existing literature typically classifies the prevailing approaches into three primary categories: SAA is a classical stochastic programming approach that relies on the law of large numbers (Kleywegt et al., 2002). RO focuses on worst-case performance to address the sensitivity of optimal solutions to parameter perturbations (Bertsimas et al., 2011). DRO serves as a bridge between SAA and RO by considering a family of distributions characterized by an ambiguity set (Mohajerin Esfahani and Kuhn, 2018; Rahimian and Mehrotra, 2019). However, different methods have its own characteristic and limitation, while different parameter settings are also optional under different models (Keith and Ahner, 2021). Thus, how to select optimal uncertainty model under different scenarios remains a challenging problem.

2.2 LLM for OR

LLMs empowered by well-designed strategies, like Chain-of-thought (CoT) (Wei et al., 2022), Self-Consistency (Wang et al., 2022), Progressive-hint (Zheng et al., 2023) have demonstrated remarkable proficiency across a wide range of NLP tasks. Specifically, researchers leveraged LLMs to implement automatic modeling of OR problems as transforming a business description into a mathematical optimization problem. One of the representative related studies was the NL4Opt problem (Ramamonjison et al., 2022). Subsequent extensive works focused on well-designed and structural multi-agent framework like Chain-of-Experts (CoE) (Xiao et al., 2023), Optimus (AhmadiTeshnizi et al., 2023, 2024), OPRO (Yang et al., 2023), NL2OPT (Mostajabdaveh et al., 2024), ORLM (Huang et al., 2024). However, almost all recent works of LLM4OR only focus on simple linear programming, lacking exploration of more complex data-driven OR paradigm like OuU. This gap stems from the fact that resolving such complex tasks not only requires mastery of higher-level domain expertise and modeling techniques but also demands seamless execution across the entire workflow.

3 Preliminary

In this section, we introduce three classical paradigms for handling uncertainty. The standard assumptions and corresponding tractable reformulations are collected in Appendix A. All these contexts are used for prompt design.

3.1 Notation

Let $x \in \mathcal{X} \subseteq \mathbb{R}^d$ denote the decision variable and $\xi \in \mathbb{R}^m$ denote the uncertain parameter. We write $\Xi \subseteq \mathbb{R}^m$ for the (known) support of ξ; in this paper we assume $\Xi = \{\xi \in \mathbb{R}^m : C\xi \leq d\}$ is a nonempty polyhedron. For a given x, the loss or evaluation function is denoted $h(x, \xi)$.

3.2 Three paradigms for optimization under uncertainty

Stochastic optimization (SO). When a probabilistic model for ξ is believed available, one typically optimizes an expected objective or a risk functional. The canonical expected-value problem

is

$$\min_{x\in\mathcal{X}} \mathbb{E}_{\mathbb{P}}[\, h(x,\xi)\,], \tag{1}$$

where $\mathbb{P}$ denotes the true (unknown) distribution of ξ supported on Ξ. In practice $\mathbb{P}$ is estimated from data; a common data-driven approximation is the sample-average approximation (SAA). Given i.i.d. samples $\{\hat{\xi}_i\}_{i=1}^n \subset \Xi$, the SAA surrogate of (1) replaces the expectation by the empirical mean:

$$\min_{x\in\mathcal{X}} \frac{1}{n}\sum_{i=1}^{n} h(x,\hat{\xi}_i). \tag{2}$$

Robust optimization (RO). When only bounds or structural information about ξ are available (no trusted distribution), robust optimization minimizes worst-case performance over a deterministic uncertainty set $\mathcal{U} \subseteq \Xi$:

$$\min_{x\in\mathcal{X}} \sup_{\xi\in\mathcal{U}} h(x,\xi). \tag{3}$$

Different choices of $\mathcal{U}$ encode different modeling attitudes (conservative vs. permissive) and lead to different computational forms. We consider three common uncertainty sets $\mathcal{U} \subseteq \Xi$.

- **Box uncertainty.**

$$\mathcal{U}_{\text{box}} = \{\xi \in \mathbb{R}^m | \underline{\xi} \le \xi \le \bar{\xi}\}, \tag{4}$$

  where $\underline{\xi}, \bar{\xi}$ provide lower/upper bounds per coordinate.
- **Budget uncertainty.** A piecewise bounded model that limits simultaneous deviations:

$$\mathcal{U}_{\Gamma} = \Big\{\xi | \xi = \hat{\xi} + \Delta z,\ 0 \le z_i \le 1,\ \sum_{i=1}^{m} z_i \le \Gamma\Big\}, \tag{5}$$

  where Δ is a vector of maximal deviations and Γ controls the budget of deviation.
- **Ellipsoidal uncertainty.**

$$\mathcal{U}_{\text{ellip}} = \{\xi | \xi - \mu)^{\top}\Sigma^{-1}(\xi - \mu) \le \rho^2\}, \tag{6}$$

  which models correlated, norm-bounded perturbations (parameterized by center μ, shape Σ, radius ρ).

Distributionally robust optimization (DRO). DRO interpolates between SO and RO by hedging against a family (ambiguity set) $\mathcal{P}$ of probability distributions supported on Ξ:

$$\min_{x\in\mathcal{X}} \sup_{\mathbb{P}\in\mathcal{P}} \mathbb{E}_{\mathbb{P}}[\, h(x,\xi)\,]. \tag{7}$$

Choices of $\mathcal{P}$ reflect different statistical/robustness assumptions (e.g., proximity to an empirical measure). We consider two widely used ambiguity sets $\mathcal{P}$.

- **Kullback–Leibler (KL) divergence ball.** Centering at a nominal distribution $\mathbb{P}_0$ (often the empirical $\hat{\mathbb{P}}_n$), and radius η,

$$\mathcal{P}_{\text{KL}} = \big\{\mathbb{P} \ll \mathbb{P}_0 | D_{\text{KL}}(\mathbb{P}\|\mathbb{P}_0) \le \eta\big\}, \tag{8}$$

  where $\mathbb{P} \ll \mathbb{P}_0$ indicates that $\mathbb{P}$ is absolutely continuous with respect to $\mathbb{P}_0$.
- **Wasserstein ball.** With ground cost $c(\cdot,\cdot)$ and radius ε,

$$\mathcal{P}_W = \big\{\mathbb{P} \in \mathcal{M}(\Xi) | W_c(\mathbb{P},\mathbb{P}_0) \le \varepsilon\big\}. \tag{9}$$

4 Methodology

In this section, we introduce the process of data generation and our multi-agent automated modeling framework LLM4OuU, and the overflow is illustrated in Figure 1.

4.1 Data Generation

In contrast to previous LLM4OR research, LLM4OuU requires not only natural language descriptions of optimization under uncertainty but also structured domain-specific numerical data, as illustrated in Appendix B. To generate optimization problem descriptions more aligned with uncertain scenarios, we implement a systematic four-step approach, as detailed below:

(1) Knowledge Base Construction. We first collect over 2,000 OR-related papers from arXiv, which are depicted in Appendix B. From these papers, we extracted core content including abstract, introduction, and conclusion, which are then processed to establish a retrieval vector database.

(2) Numerical Data Processing. We integrate numerical data from two reliable sources: open-source ML platforms (e.g., Kaggle) and real-world business cases derived from our corporate collaborations.Then, we perform rigorous pre-processing, including outlier removal and missing value imputation.

(3) Problem Description Generation. We then employ pre-defined prompts to guide LLMs in generating natural language descriptions of uncertain optimization problems. In this process, the previously constructed retrieval vector database and the preprocessed numerical data are used as dual-input to enhance the authenticity and accuracy of the generated content.

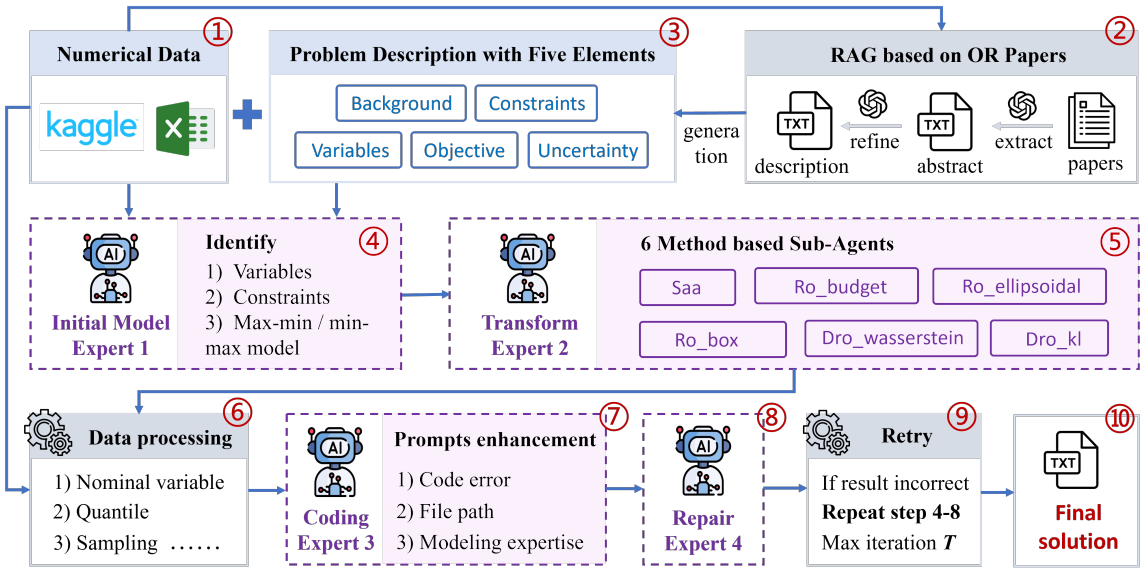


Figure 1: The overall workflow of multi-agent framework LLM4OuU: (1) we analyze over 2000 papers in OR and employ Retrieval-Augmented Generation (RAG) to construct a retrieval library; (2) we generate problem description with five elements combining retrieval library when given numerical data sourced from open-access website; (3) we decompose the automated modeling process into five distinct phases: problem formulation, problem transformation, data processing, coding, and repair, and each phase is executed by a separate well-designed agent; (4) we construct 6 specialized sub-agents to handle the transformation of different types of uncertainty models as different models require distinct theorems and modeling techniques; (5) we implement prompts enhancement based on domain knowledge and a retry mechanism; and (6) ultimately output the optimal decision.

(4) Problem Description Decomposition. Research indicates that the categorization and standardized formatting of problem descriptions significantly improve the performance of automated modeling (Jiang et al., 2024). Accordingly, we decomposed the generated natural language descriptions into five distinct sub-elements to facilitate subsequent modeling tasks.

4.2 Multi-agent Framework

As illustrated in Figure 1, we decompose the automated modeling of OuU into a structured multi-agent framework, denoted as LLM4OuU, which consists of four core agents: (1) initial problem formulation, (2) uncertainty model transformation via six specialized sub-agents, (3) coding, and (4) repair. The output trajectory of each agent is denoted by π. The specific process is depicted in Algorithm 1.

Initial Modeling Expert. Given a specific problem description with five elements $g \in G$, initial prompt p_{ini}, and one template shot s_{ini}, the output of initial model $\mathcal{O}_{\text{ini}}$ consisting of varaiables, constraints, and min-max/max-min model is formulated as

$$\mathcal{O}_{\text{ini}} \sim \pi_{\text{ini}}\left(\mathcal{O}_{\text{ini}} \mid g, p_{\text{ini}}, s_{\text{ini}}\right). \quad (10)$$

Transforming Expert. We design corresponding prompts and templates for 6 uncertainty model in total. Given a specific problem description with five elements $g \in G$, different modeling selection c^i_{model}, model transformation prompt p^i_{tra}, initial model obtained in the previous step $\mathcal{O}_{\text{ini}}$, and template shots designed for each different model s^i_{tra}, the output of transferred model $\mathcal{O}_{\text{tra}}$ is formulated as

$$\mathcal{O}_{\text{tra}} \sim \pi_{\text{tra}}\left(\mathcal{O}_{\text{tra}} \mid g, c^i_{\text{model}}, p^i_{\text{tra}}, \mathcal{O}_{\text{ini}}, s^i_{\text{tra}}\right). \quad (11)$$

Coding Expert. Given a specific problem description $g \in G$, transferred model $\mathcal{O}_{\text{tra}}$, coding prompt p_{cod}, and template shots s_{cod}, numerical data $d \in D$, the output of executable code $\mathcal{O}_{\text{cod}}$ is formulated as

$$\mathcal{O}_{\text{cod}} \sim \pi_{\text{cod}}\left(\mathcal{O}_{\text{cod}} \mid g, \mathcal{O}_{\text{tra}}, p_{\text{cod}}, s_{\text{cod}}, D\right). \quad (12)$$

Additionally, we integrate a golden ratio search algorithm into the invocation pipeline of coding expert to efficiently determine optimal hyper-

Algorithm 1 LLM4OuU

Input: Description $g \in \mathcal{G}$, prompts $p \in P$, template $s \in S$, model candidate set $c \in C$, max repair steps K, max iteration T
Output: final solution r
initialize $k \leftarrow 0$
for candidate $c = 1$ to C **do**
  for iteration $t = 1$ to T **do**
    step 1 generates O_{ini}
    step 2 generates $\mathcal{O}_{\text{tra}}$
    step 3 generates O_{cod}
    step 4 $r^k_{c,t}, e^k_{c,t} \leftarrow \text{EXECUTE}(O_{\text{cod}})$
    while $e^k_{c,t} \neq \emptyset$ **and** $k < K$ **do**
      repeat step 1-4
      $k \leftarrow k + 1$
    end while
    Let $r_{c,t} \leftarrow r^k_{c,t}$
  end for
  Let $r_c \leftarrow r_{c,t}$
end for
return r_c

parameters in certain methods like the Wasserstein radius.

Repair Expert. When an error occurs during code execution, we provide feedback on the error message $e \in \mathcal{E}$ along with the code $\mathcal{O}_{\text{cod}}$ and repair prompt p_{rep}, and output the updated code $\mathcal{O}_{\text{rep}}$, which is formulated as

$$\mathcal{O}_{\text{rep}} \sim \pi_{\text{rep}}\left(\mathcal{O}_{\text{rep}} \mid e, \mathcal{O}_{\text{cod}}, p_{\text{rep}}\right). \quad (13)$$

4.3 Domain Knowledge Enhancement

We categorize common errors encountered during testing and incorporate them into the prompts of each agent: (1) code error: errors that may affect code execution, such as 'exit()' function, etc. (2) Loading error: errors occurring in the loading data file, such as hallucinatory file path, invalid escape character, etc. (3) Expertise error: fundamental errors related to the field of OR, such as the suitability of different solvers (Gurobi, Pulp, and MOSEK) for solving different types of models.

5 Experiment

In this section, we present extensive experiments conducted on our dataset consisting of 100 samples covering 11 types of industries. We compare LLM4OuU with wide-adopted reasoning methods from two main evaluation metrics. Also, we conduct ablation analysis to demonstrate the indispensability of domain knowledge. Finally, we construct high-quality mapping pairs from optimization problem to optimal model, and perform regression test and correlation analysis.

5.1 Experimental Settings

Our dataset consists of natural language descriptions of optimization problems with five decomposed elements and structured numerical data. Numerical data is divided into training part and testing part, which used for out-of-sample test. Following prior multi-agent modeling framework, we set repair steps $k = 1$ during code execution. We leverage APIs for calling with temperature 0.

We evaluate the framework from two complementary metrics: (1) Completion Rate indicates whether LLM4OuU can automatically perform end-to-end modeling from questions and data to the final decision; (2) Accuracy indicates the consistency between the final decision made by LLM4OuU and the optimal decision labeled by human experts.

5.2 Effectiveness Analysis

We evaluate the overall modeling performance of LLM4OuU among 6 types of uncertainty models. The statistics is averaged over 3 trials and the main results are shown in Table 1. DeepSeek-R1 exhibits remarkably highest performance across all 6 uncertainty models with completion rate ranging from **76%** to **96%**. Also, the completion rate rises by 8%-18% as the retry iteration T increases from 1 to 4. As demonstrated in Table 2, despite optimal solution variations from sampling strategy differences compared with labels annotated by human experts, most results of models maintain accuracy consistently within 50%–60%. The results are significantly better when appropriately relaxing the evaluation budget. For instance, accuracy improves to 56%–79% when b=0.2.

5.3 Ablation Analysis

To isolate the contribution of key design components, we conduct ablation studies focusing on three factors: (1) $w/o_{\text{shot\_modeling}}$ or $w/o_{\text{shot\_coding}}$: whether containing modeling prompt template or coding prompt template designed specifically for each uncertainty model, (2) w/o_{enhance}: whether containing enhanced prompt based on priori experience; (3) w/o_{repair}: whether containing code repair agent. The results are merged in Table 1, 2. The

	SAA	RO[1]	RO[2]	RO[3]	DRO[4]	DRO[5]
Standard$_a$	5%	3%	7%	2%	0%	3%
Chain-of-Thought$_a$	4%	2%	4%	3%	0%	0%
Progressive-hint$_a$	7%	5%	5%	4%	5%	3%
LLM4OuU$_a$	**96%**	**87%**	**86%**	**82%**	**84%**	**85%**
LLM4OuU$_b$	85%	78%	77%	42%	21%	64%
LLM4OuU$_c$	94%	76%	84%	78%	73%	82%
Retry						
LLM4OuU$_a$ ($T = 2$)	98%	91%	88%	86%	88%	89%
LLM4OuU$_a$ ($T = 3$)	99%	93%	90%	88%	91%	90%
LLM4OuU$_a$ ($T = 4$)	**99%**	**94%**	**92%**	**90%**	**92%**	**92%**
Ablation						
$w/o_{\text{shot\_modeling}}$	89%	82%	81%	58%	66%	71%
$w/o_{\text{shot\_coding}}$	81%	77%	73%	68%	72%	69%
w/o_{enhance}	91%	83%	79%	78%	80%	80%
w/o_{repair}	95%	84%	82%	80%	81%	77%

Table 1: Completion rate. Superscripts denote the uncertainty/ambiguity-set settings ([1]: box, [2]: budget, [3]: ellipsoidal, [4]: Wasserstein, [5]: KL). Subscripts denote the API setting ($_a$: DeepSeek-R1, $_b$: Qwen3-max, $_c$: DeepSeek-V1).

	SAA	RO[1]	RO[2]	RO[3]	DRO[4]	DRO[5]
$b = 0.01$	70%	60%	53%	53%	50%	37%
$b = 0.01$ ($w/o_{\text{shot\_modeling}}$)	44%	58%	20%	28%	33%	33%
$b = 0.01$ ($w/o_{\text{shot\_coding}}$)	25%	56%	36%	36%	47%	28%
$b = 0.05$	74%	64%	60%	55%	56%	46%
$b = 0.1$	75%	64%	61%	56%	60%	46%
$b = 0.2$	79%	66%	62%	58%	73%	56%

Table 2: Accuracy using DeepSeek-R1 with $T = 1$, where b denotes the budget used in the accuracy calculation.

completion rate reduces by 7%-32% and accuracy by 6%-33% without modeling-related expertise, and this pattern remains consistent when coding-related expertise, i.e. are removed. These findings fully demonstrate the critical importance of our domain-expertise-driven prompt design strategy.

5.4 Mapping Pair Construction

We annotate four feature labels $x_1, ..., x_4 \in X$ (*Industry*, *Scenario*, *Sample*, and *Data Scale*) for each sample. *Industry* reflects the broader macro context in which they operate, as explicitly defined by strategic national policy documents, such as Agriculture, forestry, and fisheries, etc **(11 types in total)**. *Scenario* reflects a narrower application context to which optimization problem belongs defined by domain experience such as supply chain, finance, etc **(10 types in total)**. *Sample* represents the number of sample points included in the historical data and *Data Scale* represents the classification based on the number of sample points **(8 types in total)**. The specific principle for classification is illustrated in Appendix C.

Leveraging the LLM4OuU, we calculate the optimal decisions of six distinct models for each sample data. Based on the min/max properties of the problem, we select the best-performing model as the response variable y for the corresponding problem. Thus, we have mapping pairs (X_d, y_d) for each $d \in D$.

5.5 Correlation Analysis

Based on the high-quality mapping pairs (X, y) constructed, we aim to uncover the implicit relationship between problem features x and optimal model y. We employ a multi-class Logistic Regression model to fit these pairs and utilize statistical tests including Chi-square test, ANOVA, and Odds Ratio (OR) analysis to interpret the significance and directionality of these correlations. The main results are illustrated in Appendix C.

Feature Significance Analysis. First, to determine which problem features significantly influence the selection of the optimal uncertainty model, we conduct Chi-square tests for categorical variables and ANOVA for numerical variables. As presented in Table 3, *Data Scale* exhibits the highest statistical significance ($\chi^2 = 104.85, p < 0.001$), suggesting that the volume of available historical data is the dominant factor driving model selection. *Industry* and *Scenario* also show statistical significance ($p < 0.05$), indicating that specific domain

Feature	χ^2	p	Sig.
Data Scale (x_4)	104.85	< 0.001	***
Industry (x_1)	28.58	0.005	**
Scenario (x_2)	28.08	0.031	*
Sample (x_3)	4.79	0.010	*

Table 3: Statistical significance of problem features on model selection. Note: ***, **, and * denote significance at the 0.1%, 1%, and 5% levels, respectively.

Model	Top Feature	Coef.	OR
DRO	$data\_scale\_a$	1.35	**3.87**
	$data\_scale\_b$	1.11	3.02
RO	$data\_scale\_d$	1.22	**3.38**
	$scenario\_C$	0.51	1.66
SAA	$data\_scale\_h$	1.43	**4.19**
	$Sample$	0.54	1.72

Table 4: Top Predictive Features and OR for Each Model Class. High OR values indicate a strong positive correlation between the feature and the optimal model choice.

contexts play a secondary but important role.

Regression Analysis. The classification accuracy of our logistic regression model reaches **76.67%** on the test set (with a 5-fold cross-validation score of $77.14\% \pm 7.0\%$), confirming that these features possess strong predictive power for model selection. More interestingly, the classification accuracy will drop to around 50% when reducing the number of categories in the *Data Scale* to three, which further underscores the importance of this feature.

Model Preference and Data Scale. To further explore how these features influence the choice between SAA, RO, and DRO, we analyze the Odds Ratios (OR) derived from the logistic regression coefficients. The Permutation Importance analysis from our experiment aligns with the regression results, identifying specific data scale levels as the most informative features. The analysis reveals a distinct transition in model preference correlated with data volume, as detailed in Table 4.

The analysis reveals a distinct transition in model preference correlated with data volume: in **Small Data Regime**, the model exhibits a strong preference for DRO ($OR = 3.87$), which effectively mitigates distribution deviations and overfitting by constructing an ambiguity set. As the data sample increases to **Medium Data Regime**, the preference shifts towards RO ($OR = 3.38$), offering a balance between capturing parameter uncertainty and maintaining computational efficiency. Finally, for **Large Data Regime**, the optimal choice converges to SAA ($OR = 4.19$), empirically validating the law of large numbers.

6 Conclusion

In this paper, we propose a multi-agent framework empowered by LLMs named LLM4OuU for multi-types uncertainty models automated modeling. LLM4OuU decomposes complex modeling workflow into initial formulation, model-specific transformation, data loading, iterative coding and repair, which bridges the gap between natural language problems with historical data and optimal decisions. We construct a comprehensive benchmark with 100 data samples from various industries. Extensive experiments verify the effectiveness of our approach, with performance even reaching up to 99% on specific model types. More significantly, we provide interpretability of mapping relationship from problem features to optimal model. This insight offers a prescriptive guidance for future data-driven decision-making.

7 Limitations

This work has two main limitations: (1) LLM4OuU could benefit from exploring a wider range of model variants. (2) The correlation analysis could benefit from incorporating a broader and more granular set of feature types to further reveal the underlying mapping relationships.

Acknowledgments

This work was supported by Ant Group Research Fund.

References

Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. 2023. Optimus: Optimization modeling using mip solvers and large language models. *arXiv preprint arXiv:2310.06116*.

Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. 2024. Optimus: Scalable optimization modeling with (mi) lp solvers and large language models. *arXiv preprint arXiv:2402.10172*.

Cynthia Barnhart, Peter Belobaba, and Amedeo R Odoni. 2003. Applications of operations research in the air transport industry. *Transportation science*, 37(4):368–391.

Dimitris Bertsimas, Margrét V Bjarnadóttir, Michael A Kane, J Christian Kryder, Rudra Pandey, Santosh Vempala, and Grant Wang. 2008. Algorithmic prediction of health-care costs. *Operations Research*, 56(6):1382–1392.

Dimitris Bertsimas, David B Brown, and Constantine Caramanis. 2011. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501.

Frank J Fabozzi, Dashan Huang, and Guofu Zhou. 2010. Robust portfolios: contributions from operations research and finance. *Annals of operations research*, 176(1):191–220.

Chenyu Huang, Zhengyang Tang, Dongdong Ge, Shixi Hu, Ruoqing Jiang, Benyou Wang, Zizhuo Wang, and Xin Zheng. 2024. Orlm: A customizable framework in training large models for automated optimization modeling. *arXiv e-prints*, pages arXiv–2405.

Caigao Jiang, Xiang Shu, Hong Qian, Xingyu Lu, Jun Zhou, Aimin Zhou, and Yang Yu. 2024. Llmopt: Learning to define and solve general optimization problems from scratch. *arXiv preprint arXiv:2410.13213*.

Andrew J Keith and Darryl K Ahner. 2021. A survey of decision making and optimization under uncertainty. *Annals of Operations Research*, 300(2):319–353.

Anton J Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. 2002. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on optimization*, 12(2):479–502.

Douglas M Lambert and Martha C Cooper. 2000. Issues in supply chain management. *Industrial marketing management*, 29(1):65–83.

Peyman Mohajerin Esfahani and Daniel Kuhn. 2018. Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations. *Mathematical Programming*, 171(1):115–166.

Mahdi Mostajabdaveh, Timothy T Yu, Rindranirina Ramamonjison, Giuseppe Carenini, Zirui Zhou, and Yong Zhang. 2024. Optimization modeling and verification from problem specifications using a multi-agent multi-stage llm framework. *INFOR: Information Systems and Operational Research*, 62(4):599–617.

Daye Nam, Andrew Macvean, Vincent Hellendoorn, Bogdan Vasilescu, and Brad Myers. 2024. Using an llm to help with code understanding. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13.

Hamed Rahimian and Sanjay Mehrotra. 2019. Distributionally robust optimization: A review. *arXiv preprint arXiv:1908.05659*.

Rindranirina Ramamonjison, Haley Li, Timothy T Yu, Shiqi He, Vishnu Rengan, Amin Banitalebi-Dehkordi, Zirui Zhou, and Yong Zhang. 2022. Augmenting operations research with auto-formulation of optimization models from problem descriptions. *arXiv preprint arXiv:2209.15565*.

Rindranirina Ramamonjison, Timothy Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, and 1 others. 2023. Nl4opt competition: Formulating optimization problems based on their natural language descriptions. In *NeurIPS 2022 Competition Track*, pages 189–203. PMLR.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, and 1 others. 2024. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475.

Utsav Sadana, Abhilash Chenreddy, Erick Delage, Alexandre Forel, Emma Frejinger, and Thibaut Vidal. 2025. A survey of contextual optimization methods for decision-making under uncertainty. *European Journal of Operational Research*, 320(2):271–289.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, and 1 others. 2023. Chain-of-experts: When llms meet complex operations research problems. In *The Twelfth International Conference on Learning Representations*.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*.

Yang Zhang, Hanlei Jin, Dan Meng, Jun Wang, and Jinghua Tan. 2024. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. *arXiv preprint arXiv:2403.02901*.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*.

Appendices

A Tractable Reformulations of Worst-Case Objectives

In this section, we derive tractable finite-dimensional convex reformulations for worst-case objective functions arising in robust optimization and distributionally robust optimization (DRO), which are designed to guide LLMs for model transformation. We present general, tractable reformulations of the objective function, followed by specific reformulations as Linear Programming (LP), Second-Order Cone Programming (SOCP), and Exponential Cone Programming (ECP) for the affine case, $h(x, \xi) = a(x)^\top \xi + b(x)$, where any constraints that depend solely on x, i.e., independent of ξ, are excluded.

Common Assumptions

Throughout this section, we impose the following standard assumptions:

Assumption 1.

1. (**Convexity**) For any feasible decision $x \in \mathcal{X}$, the function $-h(x, \xi)$ is convex in ξ.
2. (**Lower semicontinuity**) For any fixed x, $-h(x, \xi)$ is lower semicontinuous in ξ.
3. (**Properness**) For each x, the function $-h(x, \cdot)$ is proper: it never takes the value $-\infty$ and is not identically $+\infty$.
4. (**Feasibility**) All uncertainty sets $\mathcal{U}$ and ambiguity sets $\mathcal{P}$ considered below are nonempty.
5. (**Integrability for DRO**) For distributionally robust problems, $h(x, \xi)$ is integrable with respect to all $\mathbb{P} \in \mathcal{P}$.

Preliminaries and Notation

Let $x \in \mathcal{X} \subseteq \mathbb{R}^z$ denote the decision variable and $\xi \in \Xi \subseteq \mathbb{R}^m$ denote the random parameter, where the support set is the polyhedron $\Xi = \{\xi \in \mathbb{R}^m : \ C\xi \leq d\}, C \in \mathbb{R}^{p \times m}, \ d \in \mathbb{R}^p$. We assume access to n historical samples $\{\hat{\xi}_i\}_{i=1}^n$, inducing an empirical distribution $\hat{\mathbb{P}}_n = \frac{1}{n} \sum_{i=1}^n \delta_{\hat{\xi}_i}$. We analyze the worst-case objective functions:

$$\text{(Robust)} \quad \sup_{\xi \in \mathcal{U}} h(x, \xi), \tag{14}$$

$$\text{(DRO)} \quad \sup_{\mathbb{P} \in \mathcal{P}} \mathbb{E}_{\mathbb{P}}[h(x, \xi)]. \tag{15}$$

For a set $\mathcal{U}$, let $\sigma_{\mathcal{U}}(y) \coloneqq \sup_{\xi \in \mathcal{U}} y^\top \xi$ denote its support function. For a norm $\|\cdot\|$, let $\|\cdot\|_*$ denote its dual norm.

A.1 Sample Average Approximation (SAA)

Proposition 1 **(SAA General Formulation)**. *Let $\hat{\mathbb{P}}_n$ be the empirical distribution supported on $\{\hat{\xi}_i\}_{i=1}^n$. The sample average $\mathbb{E}_{\hat{\mathbb{P}}_n}[h(x, \xi)]$ is exactly computable as:*

$$\mathbb{E}_{\hat{\mathbb{P}}_n}[h(x, \xi)] = \frac{1}{n} \sum_{i=1}^n h(x, \hat{\xi}_i). \tag{16}$$

Corollary 1 **(Affine Case $\rightarrow$ LP)**. *Assume $h(x, \xi) = a(x)^\top \xi + b(x)$. The sample average is:*

$$\mathbb{E}_{\hat{\mathbb{P}}_n}[h(x, \xi)] = a(x)^\top \left(\frac{1}{n} \sum_{i=1}^n \hat{\xi}_i \right) + b(x). \tag{17}$$

Proof of Corollary 1 Substituting the affine form into Proposition 1 and exploiting linearity:

$$\mathbb{E}_{\hat{\mathbb{P}}_n}[h(x,\xi)] = \frac{1}{n}\sum_{i=1}^{n}\left(a(x)^\top \hat{\xi}_i + b(x)\right) = a(x)^\top\left(\frac{1}{n}\sum_{i=1}^{n}\hat{\xi}_i\right) + b(x). \tag{18}$$

□

A.2 Robust Optimization with Box Uncertainty

Proposition 2 **(General Box Reformulation)**. *Let* $\mathcal{U}_{\text{box}} = \{\xi \in \mathbb{R}^m \mid \underline{\xi} \le \xi \le \overline{\xi}\}$. *The worst-case objective is given by:*

$$\sup_{\xi\in\mathcal{U}_{\text{box}}\cap\Xi} h(x,\xi) = \sup_{\xi}\left\{h(x,\xi) | \underline{\xi} \le \xi \le \overline{\xi}, C\xi \le d\right\} \tag{19}$$

Corollary 2 **(Affine Case → LP)**. *Assume* $h(x,\xi) = a(x)^\top\xi + b(x)$. *The worst-case objective value over the box uncertainty set is:*

$$\sup_{\xi\in\mathcal{U}_{\text{box}}\cap\Xi} h(x,\xi) = b(x) + \min_{s^+,s^-,\mu\ge 0} \overline{\xi}^\top s^+ - \underline{\xi}^\top s^- + d^\top\mu \quad \text{s.t.} \quad s^+ - s^- + C^\top\mu = a(x). \tag{20}$$

Proof of Corollary 2 Write the primal LP as stated. Form the Lagrangian with multipliers $s^+ \ge 0$ for $\xi \le \overline{\xi}$, $s^- \ge 0$ for $-\xi \le -\underline{\xi}$, and $\mu \ge 0$ for $C\xi \le d$. Group ξ-terms to obtain dual feasibility $a(x) - s^+ + s^- - C^\top\mu = 0$. The dual objective is $\overline{\xi}^\top s^+ - \underline{\xi}^\top s^- + d^\top\mu$. Strong duality holds since primal is feasible and bounded. □

A.3 Robust Optimization with Budget Uncertainty

Proposition 3 **(General Budget Reformulation)**. *Consider the uncertainty set* $\mathcal{U}_{\text{budget}} = \{\xi = \underline{\xi} + Dz \mid z \in [0,1]^m,\ \mathbf{1}^\top z \le \Gamma\}$, *where* $D = \operatorname{diag}(\overline{\xi} - \underline{\xi})$. *The worst-case objective is:*

$$\sup_{\xi\in\mathcal{U}_{\text{budget}}\cap\Xi} h(x,\xi) = \sup_{z}\left\{h(x,\underline{\xi}+Dz) | z\in[0,1]^m, \mathbf{1}^\top z \le \Gamma, C(\underline{\xi}+Dz) \le d\right\} \tag{21}$$

Corollary 3 **(Affine Case → LP)**. *Assume* $h(x,\xi) = a(x)^\top\xi + b(x)$. *The worst-case objective value is:*

$$\sup_{\xi\in\mathcal{U}_{\text{budget}}\cap\Xi} h(x,\xi) = b(x) + a(x)^\top\underline{\xi} + \min_{\lambda\ge 0,\ s\ge 0,\ \mu\ge 0}\{\lambda\Gamma + \mathbf{1}^\top s + d^\top\mu - \mu^\top C\underline{\xi} | \lambda\mathbf{1} + s + D^\top C^\top\mu \ge c\}, \tag{22}$$

Proof of Corollary 3 The substitution $\xi = \underline{\xi} + Dz$ yields the expression:

$$\sup_{\xi\in\mathcal{U}_{\text{budget}}\cap\Xi} y^\top\xi = b(x) + a(x)^\top\underline{\xi} + \sup_{z}\{(D^\top y)^\top z \mid 0 \le z \le \mathbf{1},\ \mathbf{1}^\top z \le \Gamma, C(\underline{\xi}+Dz) \le d\}. \tag{23}$$

The inner supremum is a linear program $\max_z\{c^\top z \mid 0 \le z \le \mathbf{1},\ \mathbf{1}^\top z \le \Gamma, C(\underline{\xi}+Dz) \le d\}$, where $c = D^\top y$. By strong duality of linear programming, this primal maximum equals the dual minimum, which is the expression displayed in the corollary. □

A.4 Robust Optimization with Ellipsoidal Uncertainty

Proposition 4 **(General Ellipsoidal Reformulation)**. *Let* $\mathcal{U}_{\text{ellip}} = \{\xi \mid \xi = \xi_0 + \Sigma^{1/2}u,\ \|u\|_2 \le \rho\}$. *The worst-case objective is given by:*

$$\sup_{\xi\in\mathcal{U}_{\text{ellip}}\cap\Xi} h(x,\xi) = \sup_{u}\left\{h(x,\xi_0+\Sigma^{1/2}u) \;\middle|\; \|u\|_2 \le \rho,\ C\Sigma^{1/2}u \le d - C\xi_0\right\}. \tag{24}$$

Corollary 4 **(Affine Case → SOCP)**. *Assume* $h(x,\xi) = a(x)^\top\xi + b(x)$. *The worst-case objective value is:*

$$\sup_{\xi\in\mathcal{U}_{\text{ellip}}\cap\Xi} h(x,\xi) = \inf_{\lambda,\gamma\ge 0} b(x) + a(x)^\top\xi_0 + \lambda\rho + (d - C\xi_0)^\top\gamma \quad \text{s.t.} \quad \left\|\Sigma^{1/2}(a(x) - C^\top\gamma)\right\|_2 \le \lambda. \tag{25}$$

Proof of Corollary 4 Parameterize $\xi = \xi_0 + \Sigma^{1/2}u$. The constraints become $\|u\|_2 \leq \rho$ and linear inequalities $C(\xi_0 + \Sigma^{1/2}u) \leq d$. The objective is linear in u. This is a convex optimization problem with an SOC constraint and linear constraints; standard duality produce an SOCP.

□

A.5 DRO with Kullback–Leibler (KL) Divergence

Assumption 2.

1. (**Exponential integrability**) For all $\tau > 0$, $\mathbb{E}_{\mathbb{P}_0}\left[e^{h(x,\xi)/\tau}\right] < \infty$.
2. (**Absolute continuity**) Measures in $\mathcal{P}_{\mathrm{KL}}$ must be absolutely continuous w.r.t. $\mathbb{P}_0$.

Proposition 5 **(General KL Reformulation)**. *Let $\mathcal{P}_{\mathrm{KL}} = \{\mathbb{P} \mid D_{\mathrm{KL}}(\mathbb{P}\|\mathbb{P}_0) \leq \eta\}$. The worst-case expectation is given by the Donsker-Varadhan dual:*

$$\sup_{\mathbb{P}\in\mathcal{P}_{\mathrm{KL}}} \mathbb{E}_{\mathbb{P}}[h(x,\xi)] = \inf_{\tau>0}\left\{\tau\eta + \tau\log\mathbb{E}_{\hat{\mathbb{P}}_n}\left[\exp(h(x,\xi)/\tau)\right]\right\}. \tag{26}$$

Corollary 5 **(Affine Case → ECP)**. *Let $\mathbb{P}_0 = \hat{\mathbb{P}}_n$ and $h(x,\xi) = a(x)^\top\xi + b(x)$. The worst-case expectation is:*

$$\sup_{\mathbb{P}\in\mathcal{P}_{\mathrm{KL}}} \mathbb{E}_{\mathbb{P}}[h(x,\xi)] = \inf_{\tau>0}\left\{\tau\eta + \tau\log\left(\frac{1}{n}\sum_{i=1}^{n}\exp\left(\frac{a(x)^\top\hat{\xi}_i + b(x)}{\tau}\right)\right)\right\}. \tag{27}$$

Proof of Corollary 5 Substituting the affine form into Proposition 5:

$$\sup_{\mathbb{P}\in\mathcal{P}_{\mathrm{KL}}} \mathbb{E}_{\mathbb{P}}[h(x,\xi)] = \inf_{\tau>0}\left\{\tau\eta + \tau\log\mathbb{E}_{\hat{\mathbb{P}}_n}\left[\exp(a(x)^\top\xi + b(x)/\tau)\right]\right\}. \tag{28}$$

□

A.6 Wasserstein DRO

Assumption 3.

1. (**Lipschitz condition**) There exists $\gamma > 0$ such that $h(\cdot,\tilde{\xi}) - h(\cdot,\xi) \leq \gamma\|\tilde{\xi} - \xi\|, \quad \forall\xi,\tilde{\xi} \in \Xi$.
2. (**light-tailed condition**) There exists an exponent $a > 1$ such that

$$A := \mathbb{E}^{\mathbb{P}}\left[\exp(\|\xi\|^a)\right] = \int_{\Xi}\exp(\|\xi\|^a)\mathbb{P}(\mathrm{d}\xi) < \infty. \tag{29}$$

Proposition 6 **(General Wasserstein Reformulation)**. *Let $\mathcal{P}_{\mathrm{W}} = \{\mathbb{P} \in \mathcal{M}(\Xi) \mid W_c(\mathbb{P},\hat{\mathbb{P}}_n) \leq \varepsilon\}$, where $c(\xi,\zeta) = \|\xi - \zeta\|$, using Kantorovich duality on the support Ξ, the objective is:*

$$\sup_{\mathbb{P}\in\mathcal{P}_{\mathrm{W}}} \mathbb{E}_{\mathbb{P}}[h(x,\xi)] = \inf_{\lambda\geq 0}\left\{\lambda\varepsilon + \frac{1}{n}\sum_{i=1}^{n}\sup_{\xi\in\mathbb{R}^m}\left(h(x,\xi) - \lambda\|\xi - \hat{\xi}_i\| \;\middle|\; C\xi \leq d\right)\right\}. \tag{30}$$

Corollary 6 **(Affine Case → LP or SOCP)**. *Let $h(x,\xi) = a(x)^\top\xi + b(x)$. The worst-case expectation is the optimal value of the following problem:*

$$\sup_{\mathbb{P}\in\mathcal{P}_{\mathrm{W}}} \mathbb{E}_{\mathbb{P}}[h(x,\xi)] = \inf_{\lambda,\gamma_1,\ldots,\gamma_n} \lambda\varepsilon + b(x) + \frac{1}{n}\sum_{i=1}^{n}\left(a(x)^\top\hat{\xi}_i + (d - C\hat{\xi}_i)^\top\gamma_i\right), \tag{31}$$

subject to

$$\|a(x) - C^\top\gamma_i\|_* \leq \lambda, \quad \forall i = 1,\ldots,n, \tag{32}$$

$$\lambda \geq 0, \quad \gamma_i \geq 0, \quad \forall i = 1,\ldots,n. \tag{33}$$

Classification:

- *If $q = 1$ (L_1 norm), dual norm is L_∞. Constraints are linear. $\to$ LP.*
- *If $q = \infty$ (L_∞ norm), dual norm is L_1. Constraints are linear. $\to$ LP.*
- *If $q = 2$ (L_2 norm), dual norm is L_2. Constraints are SOC. $\to$ SOCP.*

Proof of Corollary 6 Starting from Proposition 6 we have

$$\sup_{\mathbb{P}\in\mathcal{P}_{\mathrm{W}}} \mathbb{E}_{\mathbb{P}}[h(x,\xi)] = \inf_{\lambda\geq 0}\left\{\lambda\varepsilon + \frac{1}{n}\sum_{i=1}^{n}\sup_{\xi\in\mathbb{R}^m}\left(h(x,\xi) - \lambda\|\xi-\hat{\xi}_i\| \;\middle|\; C\xi\leq d\right)\right\}. \tag{34}$$

For the affine integrand $h(x,\xi) = a(x)^\top\xi + b(x)$ the inner supremum becomes (we suppress the dependence on x in a, b for brevity)

$$\sup_{\xi}\{a^\top\xi + b - \lambda\|\xi-\hat{\xi}_i\| \mid C\xi\leq d\}. \tag{35}$$

Perform the translation $u = \xi - \hat{\xi}_i$ (so $\xi = \hat{\xi}_i + u$ and the constraint $C\xi\leq d$ becomes $Cu\leq d - C\hat{\xi}_i$). Thus

$$\sup_{\xi\,:\,C\xi\leq d}\left(a^\top\xi - \lambda\|\xi-\hat{\xi}_i\|\right) = a^\top\hat{\xi}_i + \sup_{u\,:\,Cu\leq r_i}\left(a^\top u - \lambda\|u\|\right), \tag{36}$$

where we set $r_i := d - C\hat{\xi}_i$.

We now dualize the constrained maximization over u. Introduce Lagrange multipliers $\gamma_i \geq 0$ for the linear constraints $Cu\leq r_i$. The Lagrangian (for the inner sup) is

$$\mathcal{L}(u,\gamma_i) = a^\top u - \lambda\|u\| + \gamma_i^\top(r_i - Cu) = \gamma_i^\top r_i + (a - C^\top\gamma_i)^\top u - \lambda\|u\|. \tag{37}$$

Hence by strong duality / minimax we get

$$\sup_{u\,:\,Cu\leq r_i}\left(a^\top u - \lambda\|u\|\right) = \inf_{\gamma_i\geq 0}\left\{\gamma_i^\top r_i + \sup_{u\in\mathbb{R}^m}\left((a - C^\top\gamma_i)^\top u - \lambda\|u\|\right)\right\}. \tag{38}$$

The remaining unconstrained supremum is a support function of the norm ball; using the dual norm $\|\cdot\|_*$ we have the identity

$$\sup_{u\in\mathbb{R}^m}\left(w^\top u - \lambda\|u\|\right) = \begin{cases} 0, & \text{if } \|w\|_*\leq\lambda, \\ +\infty, & \text{otherwise,}\end{cases} \tag{39}$$

where $w := a - C^\top\gamma_i$. Applying this identity yields

$$\sup_{u\,:\,Cu\leq r_i}\left(a^\top u - \lambda\|u\|\right) = \inf_{\gamma_i\geq 0}\left\{\gamma_i^\top r_i \;\middle|\; \|a - C^\top\gamma_i\|_*\leq\lambda\right\}. \tag{40}$$

Combining with the constant terms we obtain for each i

$$\sup_{\xi\,:\,C\xi\leq d}\left(a^\top\xi + b - \lambda\|\xi-\hat{\xi}_i\|\right) = b + a^\top\hat{\xi}_i + \inf_{\gamma_i\geq 0}\left\{\gamma_i^\top(d - C\hat{\xi}_i) \;\middle|\; \|a - C^\top\gamma_i\|_*\leq\lambda\right\}. \tag{41}$$

Substituting back into the outer infimum over $\lambda\geq 0$ and exchanging the finite sums and infima (which is justified here because all are finite-dimensional convex problems) yields the stated reformulation

$$\sup_{\mathbb{P}\in\mathcal{P}_{\mathrm{W}}} \mathbb{E}_{\mathbb{P}}[h(x,\xi)] = \inf_{\lambda\geq 0,\ \gamma_1,\ldots,\gamma_n\geq 0}\left\{\lambda\varepsilon + b + \frac{1}{n}\sum_{i=1}^{n}\left(a^\top\hat{\xi}_i + (d - C\hat{\xi}_i)^\top\gamma_i\right) \;\middle|\; \|a - C^\top\gamma_i\|_*\leq\lambda,\ \forall i\right\}, \tag{42}$$

which is exactly the formulation in the statement of the corollary.

□

B Dataset

In this section, we detail the numerical data collection and problem description data generation process.

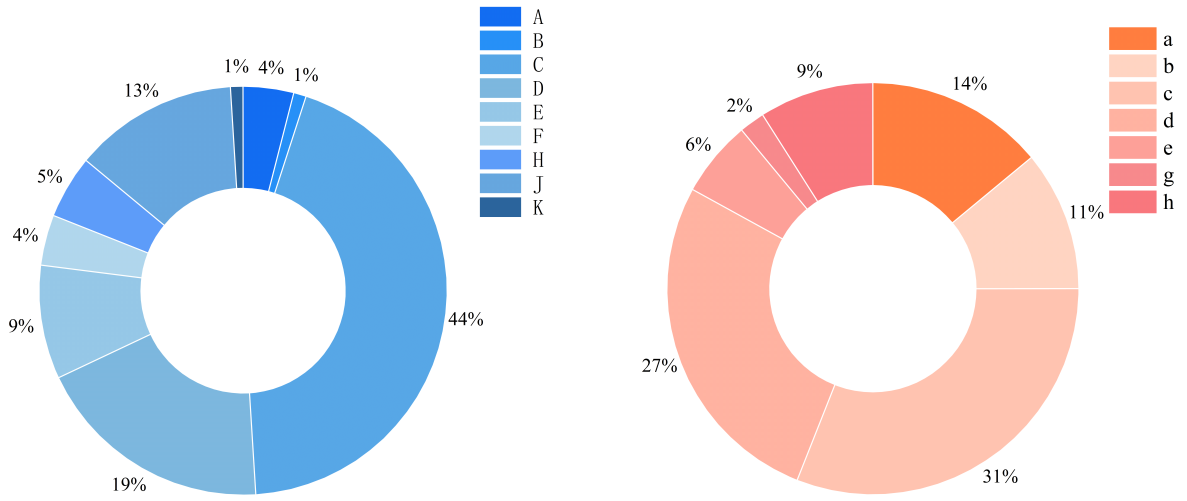
B.1 Numerical Data Collection And Processing

Firstly, select data from open-source ML platforms like Kaggle for various scenarios. Secondly, extract the required uncertain decision variable columns. Thirdly, reshape extracted decision data into a matrix format based on the time step of the potential optimization problem. Fourthly, perform smoothing or substitution process on outliers such as missing values and zero values. Lastly, identify problem feature and rename the data file based on features (*Industry*, *Scenario*, *Sample*, and *Data Scale*).

Raw Data Crawling. We select open-source data websites concluding Kaagle, UCI machine learning, Mendeley data, Figshare, Github, input scenario keywords for retrieval, and crawl the corresponding datasets.

Decision Variable Extraction. Such website data is typically used for machine learning tasks, which containing both feature variables and response variables. However, we only need to extract the data columns required for the uncertain optimization problem. For instance, we assume that we are facing an uncertain electricity production problem where we need to decide daily power generation capacity based on daily electricity consumption. Therefore, we need to extract the "consumption" column from the table without other feature variables as historical empirical data for the optimization problem.

Feature Classification. We provide four categories when processing dataset including (*Industry*, *Scenario*, *Sample*, and *Data Scale*).



(a) Distribution of *Industry*

(b) Distribution of *Data Scale*

Figure 2: Statistics of the data collected

(1) *Industry* classification follows the detailed classification of major industries in China's National Economy with appropriate screening and consolidation: A-Agriculture, Forestry, Animal Husbandry, and Fisheries; B-Manufacturing; C-Electricity Production and Supply; D-Wholesale, Retail, and Business Services; E-Transportation; F-Accommodation and Food Services; G-Information Technology Services; H-Finance; I-Education; J-Healthcare; K-Public Utilities and Management.

(2) *Scenario*: 01-Manufacturing; 02-Energy; 03-Supply Chain and Logistics; 04-Finance; 05-Transportation; 06-Healthcare; 07-Communication Networks; 08-Facility Planning; 09-Education; 10-Commerce and Trade.

(3) *Sample* represents the sample points in historical data.

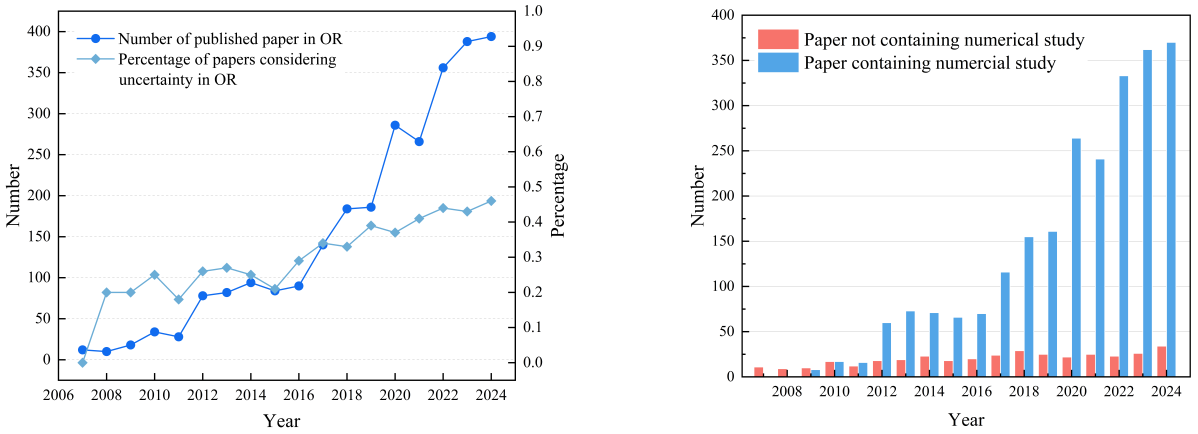
(4) *Data Scale* represents the classification based on *Sample*, which is an artificially assigned criterion: a-$<$ 100; b-100 $\sim$ 200; c-200 $\sim$ 300; d-300 $\sim$ 400; e-400 $\sim$ 500; f-500 $\sim$ 700; g-700 $\sim$ 1000; h-$>$ 1000.

Each data is uniformly saved in (.xlsx) format. The name of folder follows the aforementioned category rules. For example, 001_01_a_B denotes (No._*Scenario_Data Scale_Industry*). We collect 100 sample data in total, the statistical results are as shown in Figure 2.

B.2 Problem Description Generation

To construct a high-quality benchmark dataset that aligns with professional OR standards, we implement a systematic three-stage data generation pipeline. This pipeline bridges the gap between abstract numerical data and specific business scenarios by leveraging domain knowledge and LLMs.

Knowledge Base Construction. To ensure the generated problem descriptions adhere to academic rigor and industry logic, we first established a specialized domain knowledge base. We collect over 2,000 OR-related papers from arXiv, covering a wide range of industries and optimization scenarios. Recognizing that full papers contain extraneous information (such as complex proofs), we process these documents to extract core content. Specifically, we focus on the Abstract, Introduction, and Conclusion sections, which typically contain the most relevant information regarding problem definitions and modeling logic. Then, the extracted contents are processed and stored in a retrieval vector database, serving as the foundational external knowledge source for the subsequent generation steps. As shown in Figure 3, we conduct trend analysis and it's easily to find that optimization under uncertainty is increasingly becoming a research focus.



(a) Published papers in OR

(b) Distribution of Theoretical and empirical papers

Figure 3: Statistics of the paper collected

Retrieval-Augmented Generation (RAG) Mechanism We utilize Retrieval-Augmented Generation (RAG) to generate descriptions that are not only grammatically correct but also contextually authentic to real-world OR problems. The generation process utilizes dual inputs: the structured numerical data and the retrieval vector database. Based on the features of the input numerical data, the system retrieves relevant academic papers from the vector database. Then, we employ LLMs to conclude relevant papers as input context.

Structured Problem Description Generation We leverage LLMs guided by domain-specific prompts to transform the numerical data and retrieved context into standardized natural language descriptions. To facilitate better automated modeling, we constraint the generated description into five distinct sub-elements rather than a single unstructured block of text. The specific prompts are illustrated below.

Data Generation Prompt

[System]
You are a helpful assistant specialized in optimizaiton problem under uncertainty description creation.

[User]
I will provide you a data description in real world operation environment: `{description}`,
and relevant content from published papers: `{content_rag}`.

Based on the data description I provided, you need to conceptualize an uncertain optimization scenario and generate an uncertain optimization problem description. The specific requirements are as follows:

1. The description should consist of five sentences, describing the problem background, the decision variable, the objective function, the constraints, and the uncertain parameters of the optimization problem separately. Each sentence should not exceed 50 words.
2. You need to describe the objective function as a linear function with respect to the decision variable.
3. The parameters involved in the optimization problem modeling need to be specifically defined, like weighting coefficients in the objective function, etc.
4. Keep your problems simple, for instance, limit decision variables to one and constraints to no more than one, etc.
5. Constraints need to be directly related to decision variables. If an indirect mapping relationship exists, the specific relationship must be detailed in the problem description to ensure the model's feasibility.
6. Uncertain parameters need to be set in the objective function; please do not include them in the constraints.
7. Please refrain from making predictions involving uncertain parameters; we only have historical empirical data for the corresponding parameters.

I will provide you with a reference format:
- The company needs to reserve materials in the warehouse, and the specific quantity depends on the possible consumption of corresponding materials each month.
- The decision variable is the monthly inventory level of materials, and it needs to develop a plan for inventory levels over the next 12 months.
- The decision need to meet supply demand at the lowest possible inventory cost over the next 12 months, including holding cost and shortage cost, and both have the same weight coefficient of 1 in the objective function.
- The monthly inventory level should be set to non-negative values.
- The monthly material demand quantity is uncertain.

I will give you 1 billion dollars if you strictly follow the above requirements and format to generate the uncertain optimization problem description based on the data description I provided.

C Correlation Analysis

This section provides the detailed results of the statistical analysis performed to understand the correlation between problem features and the selection of the optimal method, defined as the response variable $best\_method\_category$ (DRO, RO, or SAA). We employe a multi-class Logistic Regression model, Chi-square tests (χ^2), ANOVA, and Permutation Importance, focusing on the influence of problem characteristics such as *Industry*, *Sample*, *Scenario*, *Data Scale*.

C.1 Multiclass Model Performance Evaluation

We evaluate the performance of the multi-class Logistic Regression model to assess its capability in predicting the optimal OuU category. The overall test set accuracy achieved is 0.77, and the 5-fold cross-validation accuracy is 0.7714 ± 0.0700.

Table 5: Multi-class Model Evaluation Results

Category	Precision	Recall	F1-score	Support
DRO	0.74	0.88	0.80	16
RO	0.71	0.56	0.63	9
SAA	1.00	0.80	0.89	5
Overall Accuracy		0.77		
Macro Avg	0.82	0.74	0.77	30
Weighted Avg	0.77	0.77	0.76	30

Interpretation of Table 5: The model exhibits strong predictive power, especially for SAA, where a perfect precision (1.00) indicates that every instance predicted to be SAA is correct. The overall accuracy of 77% validates the hypothesis that problem features are indeed effective indicators for selecting the optimal uncertainty optimization approach.

C.2 Feature Significance Tests

To confirm that the selected features are statistically relevant to the choice of the optimal OuU model, we perform significance tests for both categorical and numerical variables.

C.2.1 Categorical Variable χ^2 Test

Table 6 presents the χ^2 test results for the categorical variables. The null hypothesis of independence between each feature and the response variable is strongly rejected for all three features.

Table 6: Categorical Variable Chi-square Test Results

Categorical Feature	χ^2	**Degrees of Freedom (DF)**	p**-value**
Data Scale	104.852985	12	6.20×10^{-17}
Industry	28.582807	12	4.54×10^{-3}
Scenario	28.084340	16	3.09×10^{-2}

Interpretation of Table 6: The extremely low p-values for all three categorical features (especially *Data Scale*) confirm a statistically significant association with the optimal model category.

C.2.2 Numerical Variable Significance Test (ANOVA)

Table 7 shows the ANOVA test result for the numerical variable *Sample*.

Table 7: Numerical Variable Significance Test (ANOVA) for *Sample*

Numerical Feature	**Test Method**	**Statistic**	P**-value**
Sample	ANOVA	4.786819	0.010409

Interpretation of Table 7: With a p-value of 0.010409, the numerical feature *Sample* is also found to be statistically significant at the $\alpha = 0.05$ level. This is a crucial finding, as it quantitatively confirms that the magnitude of data availability is a key determinant in selecting an appropriate OuU model.

C.3 Univariate Logistic Regression Results

Table 8 presents the top 10 features from the univariate Logistic Regression analysis. The results highlight the independent effect of each feature on the odds of being the optimal model category.

Table 8: Univariate Logistic Regression Results (Top 10)

Feature	**Coef.** (β)	**Std.Err**	**Z-value**	**OR**	**OR 95% Low**	**OR 95% High**	p**-value**
$data\_scale\_d$	-0.69472	0.22066	-3.14837	0.49921	0.32393	0.76934	**0.00164**
$data\_scale\_a$	1.23926	0.54267	2.28365	3.45306	1.19200	10.00304	**0.02239**
$data\_scale\_c$	0.45952	0.21329	2.15449	1.58331	1.04235	2.40502	**0.03120**
$data\_scale\_h$	-1.06359	0.53123	-2.00214	0.34521	0.12187	0.97787	**0.04527**
$data\_scale\_b$	1.09255	0.54771	1.99478	2.98188	1.01923	8.72384	**0.04607**
$data\_scale\_e$	-0.86287	0.53610	-1.60951	0.42195	0.14754	1.20671	0.10751
$scenario\_H$	-0.33291	0.24168	-1.37748	0.71684	0.44638	1.15117	0.16836
$scenario\_E$	-0.25651	0.21004	-1.22123	0.77375	0.51263	1.16786	0.22200
$industry$	-0.24214	0.20542	-1.17878	0.78495	0.52479	1.17407	0.23849
$sample$	-1.53017	1.52348	-1.00439	0.21650	0.01093	4.28812	0.31519

Interpretation of Table 8: The results emphasize the central importance of the *Data Scale*. The majority of the statistically significant predictors ($p < 0.05$) are different levels of $data\_scale$ (e.g., d, a, c, h, b). This strongly suggests that the size, quality, or distributional features of the data used to represent uncertainty is the most dominant individual driver of optimal OuU model selection.

C.4 Multivariate Feature Coefficients and Permutation Importance

The multivariate Logistic Regression provides coefficients (β) and Odds Ratios (OR) for specific features associated with each category (DRO, RO, SAA), while Permutation Importance offers a global ranking of feature relevance.

C.4.1 Multivariate Logistic Regression Feature Coefficients

Table 9 shows the top five features with the highest Odds Ratios for each predicted category, highlighting which problem features strongly increase the likelihood of one category being optimal, relative to the others.

Table 9: Multivariate Logistic Regression Feature Coefficients

Category	Feature	Coefficient (β)	Odds Ratio (OR)
DRO	$data\_scale\_a$	1.3537	3.8717
	$data\_scale\_b$	1.1063	3.0233
	$scenario\_D$	0.6399	1.8964
	$scenario\_K$	0.5025	1.6529
	$data\_scale\_c$	0.3869	1.4724
RO	$data\_scale\_d$	1.2191	3.3842
	$scenario\_C$	0.5086	1.6629
	$data\_scale\_e$	0.4717	1.6028
	$scenario\_A$	0.4556	1.5771
	$data\_scale\_g$	0.4346	1.5443
SAA	$data\_scale\_h$	1.4335	4.1934
	$sample$	0.5411	1.7179
	$scenario\_H$	0.4960	1.6421
	$data\_scale\_e$	0.4364	1.5471
	$scenario\_D$	0.1997	1.2210

Interpretation of Table 9: The OR results reveal specific feature-to-model mappings. For instance, SAA is most strongly predicted by $sample\_scale\_h$ ($OR \approx 4.19$), and, importantly, by the numerical feature $scale$ ($OR \approx 1.72$), which aligns with the theoretical understanding that SAA is favored in large-data regimes. Conversely, DRO is strongly associated with $sample\_scale\_a$ ($OR \approx 3.87$) and b ($OR \approx 3.02$), suggesting that these specific sample properties potentially related to small or ambiguous data favor the robust properties of DRO.

C.4.2 Permutation Importance

Permutation Importance measures the drop in model accuracy when a feature's values are randomly shuffled, providing a robust, global measure of its predictive power.

Table 10: Feature Permutation Importance Ranking

Feature	Mean Importance	Standard Deviation
$data\_scale\_d$	0.133333	0.074536
$data\_scale\_h$	0.086667	0.026667
$data\_scale\_e$	0.063333	0.017951
$industry$	0.060000	0.013333
$data\_scale\_c$	0.033333	0.021082
$data\_scale\_g$	0.026667	0.013333
$data\_scale\_b$	0.023333	0.026034
$scenario\_C$	0.023333	0.033500
$data\_scale\_a$	0.020000	0.037118
$scenario\_E$	0.016667	0.022361

Interpretation of Table 10: The permutation importance results strongly confirm the multivariate analysis. The various levels of the $data\_scale$ feature (d, h, e) occupy the top ranks, indicating that these are the most critical determinants of the model's overall prediction accuracy.

D Prompts with Domain Knowledge for Each Agent

In this section, we detail the specific prompts for each agent. We design tailored prompts and corresponding shots for different methods.

Initial Modeling Prompt

[System]
You are an expert in operation research and optimization.
You are now facing a optimization problem under uncertainty. Now you need to model uncertain optimization problem.
You need to do:
1. Analyze optimization problem, historical data and identify the uncertain parameter.
2. Construct initial model with Latex language for formulation, which need to contain the initial min-max/ max-min model.
3. When establishing an optimization model, it is necessary to consider the realistic constraints in the problem, such as inventory cannot be negative.
4. Please output three information: Variables, Constraints, Min-max or max-min objective.

[Task & Template Input]
Optimization problem: `{description}`

[Template Shot - Output]

```
### 1. Variables
- $x_t$ is the decision variable: inventory level in month $t$.
### 2. Constraints
- Non-negative inventory: $x_t \geq 0$.
### 3. Min-max / max-min objective
$$
\min_{{\substack{{x_t \geq 0}} \\ t=1,2,\dots,T}}
\max_{{\substack{{d_t \in \mathcal{{U}} \\ t=1,2,\dots,T}}
\sum_{{t=1}}^{{T}} |x_t - d_t|
$$
where:
- $|x_t - d_t|$ represents the inventory cost (carrying cost or shortage cost) in month $t$,
- $\mathcal{{U}}$ denotes the uncertainty set,
- $T$ denotes the total number of months.
```

Model Transformation Prompt for Box RO

[System]
You are an expert in operation research and optimization.
You are now facing a optimization problem under uncertainty. You have conducted problem anaysis and modeled initial mathematical model. Now you need to transfer the initial model based on box uncertainty set. You have conducted historical data analysis and obtained the lower and upper bounds of box uncertainty set for each decision step.
You need to do: 1. Analyze optimization problem, initial mathematical model and boundary of box uncertainty set.
2. Transfer the initial RO model based on box uncertainty set, and give the final optimized model obtained.
3. Please output three information: the construction uncertainty set, the specific model conversion process, and the final optimized model obtained.

[Task & Template Input]
Optimization problem: `{description}`
Initial mathematical RO model: `{initial_model}`

[Template Shot - Output]

```
### 1. Uncertainty set construction
Since the uncertainty set (the box-shaped set formed by the demand interval $[l_t, u_t]$) is
box-shaped and the objective function is separable, the above min-max problem can be decomposed into
independent problems for each month, where the “sum of maximums” equals the “maximum of sums”:
$$
\max_{{\substack{{d_t \in [l_t, u_t]}} \\ t=1,2,\dots,T}}
\sum_{{t=1}}^{{T}} |x_t - d_t|
=
\sum_{{t=1}}^{{T}} \max_{{d_t \in [l_t, u_t]}} |x_t - d_t|
$$
### 2. Model Transformation
For each $t$, compute $\max_{{d_t \in [l_t, u_t]}} |x_t - d_t|$: Since the absolute value function is
convex, its maximum value within the interval must occur at one of the interval endpoints.
Thus, this maximum can be expressed as:
$$
\max_{{d_t \in [l_t, u_t]}} |x_t - d_t|
=
\max\left\{{ u_t - x_t, x_t - l_t }}\right\
$$
Thus, the original min-max problem transforms into an optimization problem involving only the
```

```
decision variable $x_t$:
$$
\min_{{\substack{{x_t \geq 0 \ t=1,2,\dots,T}}}}
\sum_{{t=1}}^{{T}} \max\left\{{ u_t - x_t, x_t - l_t }}\right\
$$
To transform this nonlinear model containing $\max$ functions into a linear programming problem, we
introduce auxiliary variables $y_t$ (corresponding to the worst-case cost in month $t$), such that:
$$
y_t \geq u_t - x_t,
\quad
y_t \geq x_t - l_t
\quad (\forall \, t=1,2,\dots,T)
$$
At this point, $y_t$ can equivalently replace $\max\left\{{ u_t - x_t, x_t - l_t }}\right\$,
completing the linearization of the model.
### 3. Final Linear Programming Model LaTeX Code:
$$
\min \quad \sum_{{t=1}}^{{T}} y_t
$$
subject to:
\[
y_t \geq u_t - x_t & \forall \, t = 1, \ldots, T \\
y_t \geq x_t - l_t & \forall \, t = 1, \ldots, T \\
x_t \geq 0 & \forall \, t = 1, \ldots, T \\
y_t \geq 0 & \forall \, t = 1, \ldots, T
\]
where $y_t$ is automatically non-negative (guaranteed by the constraints), but for completeness,
we explicitly include the constraint $y_t \geq 0$.
```

Model Transformation Prompt for Budget RO

[System]
You are an expert in operation research and optimization.
You are now facing a optimization problem under uncertainty. You have conducted problem anaysis and modeled initial mathematical RO model. Now you need to transfer the initial RO model based on budget uncertainty set. You have conducted historical data analysis and obtained the mean and variance of the empirical data.
You need to do:
1. Analyze optimization problem, initial mathematical model, mean and variance of the empirical data, and parameter omega of ebudget uncertainty set.
2. Transfer the initial RO model based on budget uncertainty set, and give the final optimized model obtained.
3. Please output two information: the specific model conversion process, and the final optimized model obtained.

[Task & Template Input]
Optimization problem: `{description}`
Initial mathematical RO model: `{initial_model}`

[Template Shot - Output]

```
### 1. Definition of of Budget Uncertainty Set
The budget uncertainty set is typically expressed as:
\[
\mathcal{{U}} = \left\{{ d_t = \bar{{d}}_t + \hat{{d}}_t \zeta_t \ \middle| \ |\zeta_t| \leq 1,
\ \sum_{{t=1}}^T |\zeta_t| \leq \Gamma \right\}}
\]
where:
- \(\bar{{d}}_t\) is the nominal demand (the mean value) of empirical data),
- \(\hat{{d}}_t\) is the maximum demand deviation (magnitude of uncertainty),
- \(\zeta_t\) is the perturbation variable,
- \(\Gamma\) is the budget parameter (\(0 \leq \Gamma \leq T\)).
### 2. Reformulation of the Inner Maximization Problem
For fixed inventory decisions \(x_t\), the inner problem is to maximize the total cost over the
uncertainty set \(\mathcal{{U}}\):
\[
\max_{{d_t \in \mathcal{{U}}}} \sum_{{t=1}}^T |x_t - d_t|
\]
Substituting \(d_t = \bar{{d}}_t + \hat{{d}}_t \zeta_t\) and letting \(c_t = x_t - \bar{{d}}_t\),
the inner problem becomes:
\[
\max_{{\zeta_t}} \sum_{{t=1}}^T |c_t - \hat{{d}}_t \zeta_t|
```

```
\]
subject to the constraints \(|\zeta_t| \leq 1\) and \(\sum_{{t=1}}^T |\zeta_t| \leq \Gamma\).
However, subject to the budget constraint, the adversary can only choose to set \(\zeta_t = \pm 1\)
(i.e., increase the cost by \(\hat{{d}}_t\)) for at most \(\Gamma\) periods. For the remaining
periods, \(\zeta_t\) must be set to 0 (cost is \(|x_t - \bar{{d}}_t|\)).
Thus, the inner maximum is equivalent to (This is a theoretical conclusion that can be used):
\[
\sum_{{t=1}}^{{T}} |x_t - \bar{{d}}_t| + \max_{{S \subseteq \{{1, \ldots, T\}}, |S| \leq \Gamma}}
\sum_{{t \in S}} \hat{{d}}_t
\]
where \(S\) is the set of selected periods. The term \(\max_S \sum_{{t \in S}} \hat{{d}}_t\) is the
sum of the largest \(\Gamma\) values of \(\hat{{d}}_t\).
Let \(\hat{{d}}_{{[1]}} \geq \hat{{d}}_{{[2]}} \geq \cdots \geq \hat{{d}}_{{[T]}}\)
denote the ordered deviations. Then:
\[
\max_S \sum_{{t \in S}} \hat{{d}}_t = \sum_{{t=1}}^{{\Gamma}} \hat{{d}}_{{[t]}}
\]
Therefore, the inner maximum value is:
\[
\sum_{{t=1}}^{{T}} |x_t - \bar{{d}}_t| + \sum_{{t=1}}^{{\Gamma}} \hat{{d}}_{{[t]}}
\]
### 3. Reformulation of the Outer Minimization Problem
Substituting the result of the inner problem, the original problem transforms into:
\[
\min_{{x_t \geq 0}} \left[ \sum_{{t=1}}^T |x_t - \bar{{d}}_t| +
\sum_{{t=1}}^\Gamma \hat{{d}}_{{[t]}} \right]
\]
The second term, \(\sum_{{t=1}}^\Gamma \hat{{d}}_{{[t]}}\), is a constant independent of the
decision variables \(x_t\). Thus, the optimization problem simplifies to:
\[
\min_{{x_t \geq 0}} \sum_{{t=1}}^T |x_t - \bar{{d}}_t|
\]
(4) Final Transformed RO Model
Considering the budget uncertainty set, the robust optimization model is equivalent to the
following deterministic model:
\[
\min_{{x_t \geq 0}} \sum_{{t=1}}^T |x_t - \bar{{d}}_t|
\]
where \(\bar{{d}}_t\) is the nominal demand.
```

Model Transformation Prompt for Ellipsoidal RO

[System]
You are an expert in operation research and optimization.
You are now facing a optimization problem under uncertainty.
You have conducted problem anaysis and modeled initial mathematical RO model. Now you need to transfer the initial RO model based on ellipsoidal uncertainty set. You have conducted historical data analysis and obtained the mean and variance for each decision step.
You need to do:
1. Analyze optimization problem, initial mathematical model, mean and variance of the empirical data, and parameter omega of ellipsoidal uncertainty set.
2. Transfer the initial RO model based on ellipsoidal uncertainty set, and give the final optimized model obtained.
3. Please output two information: the definition of uncertainty set, the specific model conversion process, and the final optimized model obtained.

[Task & Template Input]
Optimization problem: `{description}`
Initial mathematical RO model: `{initial_model}`

[Template Shot - Output]

```
### 1. Definition of of Ellipsoidal Uncertainty Set
An ellipsoidal uncertainty set is used, defined as follows:
\[
\mathcal{{U}} = \left\{{ {{d}} \in \mathbb{{R}}^T: \sum_{{t=1}}^{{T}}
\frac{{(d_t - \bar{{d}}_t)^2}}{{\sigma_t^2}} \leq \Omega^2 \right\}}
\]
Where:
\item $\bar{{d}}_t$ is the nominal consumption (mean value),
```

```
\item $\sigma_t$ is the standard deviation of empirical data,
\item $\Omega$ is a parameter controlling the conservatism level (typically $\Omega \geq 0$).
### 2. Treatment of the Inner Maximization Problem
The inner problem is:
\[
\max_{{d_t \in \mathcal{{U}}}} \sum_{{t=1}}^{{T}} |x_t - d_t|
\]
The maximum value of the inner problem is (This conclusion can be used directly):
\[
\max_{{d_t \in \mathcal{{U}}}} \sum_{{t=1}}^{{T}} |x_t - d_t| = \sum_{{t=1}}^{{T}} |x_t -
\bar{{d}}_t| + \Omega \sqrt{{\sum_{{t=1}}^{{T}} \sigma_t^2}}
\]
### 3. Transformation of the Outer Minimization Problem
Substituting the result of the inner problem into the outer problem:
\[
\min_{{x_t \geq 0}} \left[ \sum_{{t=1}}^{{T}} |x_t - \bar{{d}}_t| + \Omega
\sqrt{{\sum_{{t=1}}^{{T}} \sigma_t^2}} \right]
\]
Here, $\Omega \sqrt{{\sum_{{t=1}}^{{T}} \sigma_t^2}}$ is a constant independent of the decision
variables $x_t$. Therefore, the minimization problem is equivalent to:
\min \sum_{{t=1}}^{{T}} |z - d_t|
### 4. Final Transformed RO Model
The transformed model is a deterministic optimization problem:
\[
\min_{{x_t}} \sum_{{t=1}}^{{T}} |x_t - \bar{{d}}_t|
\]
subject to $x_t \geq 0, \quad t = 1, 2, \ldots, T$
where $\bar{{d}}_t$ is the nominal consumption (mean value).
```

Model Transformation Prompt for Wasserstein DRO

[System]
You are an expert in operation research and optimization.
You are now facing a optimization problem under uncertainty. You have conducted problem analysis and modeled initial mathematical DRO model. Now you need to transfer the initial DRO model into tractable form based on wasserstein ambiguity set using the strong duality theory.
You need to do:
1. Analyze optimization problem and initial mathematical model.
2. Transfer the initial DRO model into tractable form based on wasserstein ambiguity set using the strong duality theory, and give the final optimized model obtained.
3. Please output six information: definition of ambiguity set, strong duality application, inner problem calculation, tractable model conversion, the final optimized model, and the empirical sample mean approximation form.

[Task & Template Input]
Optimization problem: `{description}`
Initial mathematical DRO model: `{initial_model}`

[Template Shot - Output]

```
### 1. Ambiguity set definition
The initial model is:
\[
\min_{{x_t \geq 0}} \max_{{P \in \mathcal{{P}}}} \mathbb{{E}}_P
\left[ \sum_{{t=1}}^T |x_t - d_t| \right],
\]
where $\mathcal{{P}}$ is the Wasserstein ambiguity set with radius $\theta$ around a
reference distribution $P_0$.
The cost function $\sum_{{t=1}}^T |x_t - d_t|$ represents the total inventory cost.
### 2. Strong Duality Application
For the Wasserstein ambiguity set with L1 norm as the cost function (i.e., $c(\zeta, d)
= \|\zeta - d\|_1 = \sum_{{t=1}}^T |\zeta_t - d_t|$),
the inner max problem can be expressed using strong duality:
\[
\max_{{P \in \mathcal{{P}}} \mathbb{{E}}_P \left[ \sum_{{t=1}}^T |x_t - d_t| \right] =
\inf_{{\lambda \geq 0}} \left\{{ \lambda \theta + \mathbb{{E}}_{{P_0}}
\left[ \sup_\zeta \left\{{ \sum_{{t=1}}^T |x_t - \zeta_t| - \lambda
\sum_{{t=1}}^T |\zeta_t - d_t| \right\}} \right] \right\}}.
\]
### 3. Inner Supremum Calculation
```

```
The inner supremum
\[
\sup_\zeta \left\{{ \sum_{{t=1}}^T |x_t - \zeta_t| - \lambda \sum_{{t=1}}^T |\zeta_t - d_t| \right\}}
\]
is separable by time period $t$. For each $t$, we compute:
\[
\sup_\zeta \left\{{ |x_t - \zeta_t| - \lambda |\zeta_t - d_t| \right\}}.
\]
Using the triangle inequality, for $\lambda \geq 1$, this supremum equals $|x_t - d_t|$, achieved at
$\zeta_t = d_t$. Thus, the inner expression simplifies to $\sum_{{t=1}}^T |x_t - d_t|$.
### 4. Tractable Model Conversion
The duality expression becomes:
\[
\inf_{{\lambda \geq 0}} \left\{{ \lambda \theta + \mathbb{E}}_{{P_0}} \left[ \sum_{{t=1}}^{{T}}
|x_t - d_t| \right] \right\}} = \inf_{{\lambda \geq 1}} \left\{{ \lambda \theta +
\mathbb{E}}_{{P_0}} \left[ \sum_{{t=1}}^{{T}} |x_t - d_t| \right] \right\}},
\]
since for $\lambda < 1$, the supremum is infinite. The infimum is achieved at $\lambda = 1$, yielding:
\[
\theta + \mathbb{E}}_{{P_0}} \left[ \sum_{{t=1}}^{{T}} |x_t - d_t| \right].
\]
The worst-case expected cost is
$\theta + \mathbb{E}}_{{P_0}} \left[ \sum_{{t=1}}^{{T}} |x_t - d_t| \right]$.
Since $\theta$ is constant, minimizing this is equivalent to minimizing
$\mathbb{E}}_{{P_0}} \left[ \sum_{{t=1}}^{{T}} |x_t - d_t| \right]$ subject to $x_t \geq 0$.
### 5. Final Tractable Model
The tractable form of the DRO model is:
\[
\min_{{x_t \geq 0}} \mathbb{E}}_{{P_0}} \left[ \sum_{{t=1}}^{{T}} |x_t - d_t| \right],
\]
where:
\item $x_t$ is the inventory level in month $t$ (decision variable),
\item $d_t$ is the random demand in month $t$,
\item $P_0$ is the reference distribution for demands,
\item $T$ is the total number of months.
### 6. Empirical Sample Mean Approximation Form
This model can be solved using sample average approximation (SAA) if $P_0$ is empirical. Specifically,
with historical demand samples $\{{d_t^{{(i)}}\}}_{{i=1}}^N$ for each $t$, the objective becomes:
\[
\min_{{x_t \geq 0}} \frac{{1}}{{N}} \sum_{{i=1}}^N \sum_{{t=1}}^T |x_t - d_t^{{(i)}}|.
\]
The optimal solution sets each $x_t$ to the median of the demand distribution for period $t$ under
$P_0$, as the median minimizes the expected absolute deviation.
```

Model Transformation Prompt for KL DRO

[System]
You are an expert in operation research and optimization.
You are now facing a optimization problem under uncertainty. You have conducted problem anaysis and modeled initial mathematical DRO model. Now you need to transfer the initial DRO model into tractable form based on Kullback-Leibler (KL) divergence ambiguity set using the strong duality theory.
You need to do:
1. Analyze optimization problem and initial mathematical model.
2. Transfer the initial DRO model into tractable form based on Kullback-Leibler (KL) divergence ambiguity set using the strong duality theory, and give the final optimized model obtained.
3. Please output two information: the specific model conversion process, and the final optimized model obtained.

[Task & Template Input]
Optimization problem: `{description}`
Initial mathematical DRO model: `{initial_model}`

[Template Shot - Output]

```
### 1. Assumptions and notation
- Time horizon: \( t = 1, \ldots, T \). Decision variables \( x_t \geq 0 \).
- Loss per scenario (vector of demands) \( d^s = (d_1^s, \ldots, d_T^s) \) and sample space \( s \in
S \). Define the per-scenario loss
\[
L(x, d^s) = \sum_{{t=1}}^T |x_t - d_t^s|.
```

\]
- Nominal distribution \(P_0 \) on scenarios. Two common choices: \textbf{{Empirical discrete}}:
\(P_0 \) supported on \(s = 1, \ldots, N \) with probabilities \(p_s \) (often \(p_s = 1/N \)).
- KL ambiguity set of radius \(\rho \):
\[
\mathcal{{P}}_{{KL}} = \{{P \ll P_0 : D_{{KL}}(P\|P_0) \leq \rho\}}.
\]
- DRO problem:
\[
\min_{{x \geq 0}} \sup_{{P \in \mathcal{{P}}_{{KL}}}} \mathbb{{E}}_P [L(x, D)].
\]
2. Key duality / reformulation (exponential-tilt representation)
Use the well-known KL-duality (Donsker-Varadhan / exponential-tilt) identity:
for any measurable random variable \(Z\),
\[
\sup_{{P: D_{{\text{{KL}}}}(P\|P_{{0}})\leq\rho}} \mathbb{{E}}_{{P}}[Z] = \inf_{{\lambda >0}}
\left\{{ \lambda\rho + \lambda\log\mathbb{{E}}_{{P_{{0}}}}[e^{{Z/\lambda}}] \right\}}.
\]
Apply that with \(Z = L(x,D) = \sum_{{t}}|x_{{t}}-D_{{t}}|\). Therefore the DRO objective becomes
\[
\sup_{{P\in\mathcal{{P}}_{{\text{{KL}}}}}} \mathbb{{E}}_{{P}}[L(x,D)] = \inf_{{\lambda>0}}
\left\{{ \lambda\rho + \lambda\log\mathbb{{E}}_{{P_{{0}}}} \left[e^{{\frac{{1}}{{\lambda}}
\sum_{{i=1}}^{{T}}|x_{{i}}-D_{{i}}|}} \right] \right\}}.
\]
Interchanging min over \(x\) and inf over \(\lambda>0\) (valid since \(\lambda\) appears nicely and
the right-hand expression is convex in \(x\)) yields the saddle reformulation
\[
\min_{{x\geq 0}} \inf_{{\lambda>0}} \left\{{ \lambda\rho + \lambda\log\mathbb{{E}}_{{P_{{0}}}}
\left[e^{{\frac{{1}}{{\lambda}} \sum_{{i=1}}^{{T}}|x_{{i}}-D_{{i}}|}} \right] \right\}}
\]
or equivalently
\[
\min_{{x\geq 0,\,\lambda>0}} \left\{{ \lambda\rho + \lambda\log\mathbb{{E}}_{{P_{{0}}}}
\left[e^{{\frac{{L(x,D)}}{{\lambda}}}} \right] \right\}}.
\]
This is the standard finite-dimensional reformulation: the infinite-dimensional sup over
distributions becomes an optimization over scalar \(\lambda>0\) plus an expectation
under the known \(P_{{0}}\).
3. Practical discrete (empirical) implementation
If \(P_0 \) is empirical with scenarios \(s = 1, \ldots, N \) and probabilities \(p_s \)
(commonly \(p_s = \frac{{1}}{{N}} \)), then
\[
\mathbb{{E}}_{{P_0}} \left[e^{{L(x,D)/\lambda}} \right] =
\sum_{{s=1}}^N p_s e^{{L(x,d^s)/\lambda}}.
\]
Introduce auxiliary per-scenario variables \(y_s \) to represent the scenario loss:
\[
y_s \geq L(x,d^s) = \sum_{{t=1}}^T |x_t - d_t^s|, \quad s = 1, \ldots, N.
\]
Then the DRO is equivalently
\[
& \min_{{x,\lambda,y}} \lambda \rho + \lambda \log \left(
\sum_{{s=1}}^N p_s e^{{y_s/\lambda}} \right) \\
& \text{{s.t.}} \quad y_s \geq \sum_{{t=1}}^T |x_t - d_t^s|, \quad s = 1, \ldots, N, \\
& \qquad x_t \geq 0, \quad t = 1, \ldots, T, \\
& \qquad \lambda > 0.
\]
4. Linearizing the absolute values
Replace each \(|x_{{t}}-d_{{t}}^{{s}}|\) by an auxiliary \(a_{{t,s}}\geq 0\):
& a_{{t,s}} \geq x_{{t}}-d_{{t}}^{{s}}, \\
& a_{{t,s}} \geq -(x_{{t}}-d_{{t}}^{{s}}), \\
& y_{{s}} \geq \sum_{{t=1}}^{{T}}a_{{t,s}}, \\
& a_{{t,s}} \geq 0.
So the full implementable form is
\min_{{\begin{{subarray}}{{c}}x,\lambda,y,a{{subarray}}}} \lambda\rho+\lambda\log\biggl(
\sum_{{s=1}}^{{N}}p_{{s}}\,e^{{y_{{s}}/\lambda}}\biggr)
s.t.
\[

```
& a_{{t,s}}\geq x_{{t}}-d_{{t}}^{{s}}, && \forall t,s \\
& a_{{t,s}}\geq -(x_{{t}}-d_{{t}}^{{s}}), && \forall t,s \\
& y_{{s}}\geq \sum_{{t=1}}^{{T}}a_{{t,s}}, && \forall s \\
& x_{{t}}\geq 0, ~ a_{{t,s}}\geq 0, ~ \lambda\geq \varepsilon > 0.
\]
```

Code Generation Prompt for SAA

[System]
You are a Python programmer working in the fields of operations research and optimization.
You are proficient in using the third-party solving library like gurobipy. You will receive a specific problem description, data, and a mathematical model based on SAA method. Your goal is to write a usable Python program.
You need to do:
1. Analyze optimization problem and empirical data matrix.
2. Generate the solving code for the SAA optimization model.
3. Print the final solution of optimization problems.
4. Evaluate the solution with real test data from an excel file with the path: `{test_dir}`, and print the evaluation results.
Note:
1. Ensure the data loading path in your code is consistent with the provided `{train_dir}` and `{test_dir}`. Please strictly import empirical data from `{train_dir}` and real test data from `{test_dir}` , and do not fabricate data yourself.Additionally, strictly use "/" as the file path separator.
2. Please note that the table has headers, so do not set header=None when reading it.
3. Don't output any content containing 'exit()', which will lead to program crash.
4. Please give the complete code, including the code using "'python"'

[Task & Template Input]
The empirical data is from `{train_dir}` and real test data is from `{test_dir}`.
Empirical data description is as follows:`{data_description}`.
Specific task: `{description}`.
SAA model: `{math_model}`.

[Template Shot - Output]

```python
import pickle
import pandas as pd
import numpy as np
from gurobipy import Model, GRB, quicksum
import os
# === Step 1: Load data ===
train_dir = "{train_dir}"
test_dir = "{test_dir}"
# Load training data
historical_data = pd.read_excel(train_dir) # Please note that the table has headers.
demand_samples = historical_data.values
# Load test data
real_data = pd.read_excel(test_dir) # Please note that the table has headers.
real_demands = real_data.values.flatten().tolist()
T = len(demand_samples)     # Number of time steps (days)
S = len(demand_samples[0])  # Number of samples per time step
# === Step 2: Create optimization model ===
model = Model("SAA_Power_Generation_Optimization")
model.Params.OutputFlag = 1
# === Step 3: Define variables ===
g = model.addVars(T, vtype=GRB.CONTINUOUS, lb=0.0, ub=500.0, name="g")
u = model.addVars(T, S, vtype=GRB.CONTINUOUS, lb=0.0, name="u")
# === Step 4: Add constraints ===
for t in range(T):
    for s in range(S):
        d_ts = demand_samples[t][s]
        model.addConstr(u[t, s] >= d_ts - g[t], name=f"shortage_{{t}}_{{s}}")
# === Step 5: Define the objective function ===
generation_cost = quicksum(50 * g[t] for t in range(T))
shortage_penalty = quicksum(100 * u[t, s] for t in range(T) for s in range(S))
total_cost = (generation_cost + shortage_penalty) / S
model.setObjective(total_cost, GRB.MINIMIZE)
# === Step 6: Optimize ===
model.optimize()
# === Step 7: Extract and save solution ===
```

```
if model.status == GRB.OPTIMAL:
    optimal_generation = []
    for t in range(T):
        gen_value = g[t].X
        optimal_generation.append(gen_value)
        print(f"Day {{t+1}}: {{gen_value:.2f}} MW")
    # === Step 8: Evaluate with real historical data ===
    test_data_path = "{test_dir}"
    try:
        # Load test data
        test_df = pd.read_excel(test_data_path)
        # Assuming the last column contains the real demand data
        last_column_name = test_df.columns[-1]
        real_demands = test_df[last_column_name].tolist()
        # Ensure we have enough test data
        if len(real_demands) < T:
         print(f"Warning: Test data has only {{len(real_demands)}} days, but we need {{T}} days")
            real_demands = real_demands + [0] * (T - len(real_demands))
        elif len(real_demands) > T:
            real_demands = real_demands[:T]
        # Calculate actual cost with real demands
        total_actual_cost = 0
        print("\n=== EVALUATION WITH REAL HISTORICAL DATA ===")
        print("Day | Generation | Real Demand | Shortage | Daily Cost")
        print("-" * 55)
        for t in range(T):
            generation = optimal_generation[t]
            real_demand = real_demands[t]
            shortage = max(0, real_demand - generation)
            daily_cost = 50 * generation + 100 * shortage
            total_actual_cost += daily_cost
        print("-" * 55)
        print(f"Total actual cost: ${{total_actual_cost:.2f}}")
    except FileNotFoundError:
        print(f"Test data file not found: {{test_data_path}}")
    except Exception as e:
        print(f"Error loading test data: {{e}}")
else:
    print("Optimization failed. Status:", model.status)
```

Code Generation Prompt for Box RO

[System]

You are a Python programmer working in the fields of operations research and optimization.
You are proficient in using the third-party solving library like gurobipy. You will receive a specific task description, data, and a mathematical model based on robust optimization method. Your goal is to write a usable Python program.
You need to do:
1. Analyze optimization problem, historical data and mathematical model based on RO model.
2. Generate the solving code for the RO model.
3. Print the final solution of optimization problems.
4. After obtaining the final decision, evaluate it using real data. This requires you to explicitly implement the following processing flow in your code: table data extraction → list conversion → inputting actual values and final optimal solution into the objective function for calculation.
5. Print the real objective value.
6. Save the solution, actual value, corresponding decision loss for each decision step, and the real objective value as a text file format in `{result_dir}`.
Note:
1. Ensure the data loading path in your code is consistent with the provided `{train_dir}` and `{test_dir}`. Please strictly import empirical data from `{train_dir}` and real test data from `{test_dir}` , and do not fabricate data yourself.Additionally, strictly use "/" as the file path separator.
2. Ensure strict consistency between imported modules and code usage. For example, if GRB is used in the code, the import statement must contain 'from gurobipy import GRB'.
3. Please note that the table has headers, so do not set header=None when reading it.
4. Don't output any content contating 'exit()', which will lead to program crash.
5. Please give the complete code, including the code using "'python"'

[Task & Template Input]

The empirical data is from {train_dir} and real test data is from {test_dir}.
Empirical data description is as follows:{data_description}.
Specific task: {description}.
Initial mathematical RO model: {initial_model}.
Transferred RO model: {math_model}.

[Template Shot - Output]

```python
import gurobipy as gp
from gurobipy import GRB
import pandas as pd
import os
from utils import save_txt, extract_python_code, safe_exec, safe_exec_with_flag, calculate_bounds
# Load Data
real_data = pd.read_excel("{test_dir}").values.flatten().tolist()
T = len(real_data)  # Number of months
l_t = {{lower}}  # Lower bounds for each month
u_t = {{upper}}  # Upper bounds for each month
# Model initialization
model = gp.Model("RobustInventoryManagement")
# Decision variables
x = model.addVars(T, lb=0, name="x")  # Inventory levels
y = model.addVars(T, lb=0, name="y")  # Auxiliary variables for worst-case cost
# Objective function: minimize sum of worst-case costs
model.setObjective(gp.quicksum(y[t] for t in range(T)), GRB.MINIMIZE)
# Constraints for worst-case cost calculation
# for t in range(T):
#     model.addConstr((y[t] >= u_t[t] - x[t] for t in range(T)), name="upper_bound_cost")
#     model.addConstr((y[t] >= x[t] - l_t[t] for t in range(T)), name="lower_bound_cost")
model.addConstrs((y[t] >= u_t[t] - x[t] for t in range(T)), name="upper_bound_cost")
model.addConstrs((y[t] >= x[t] - l_t[t] for t in range(T)), name="lower_bound_cost")
# Solve the model
model.optimize()
# Check solution status
if model.status == GRB.OPTIMAL:
    print("Optimal solution found:")
else:
    print("No optimal solution found")
# Extract solution
solution = [x[t].X for t in range(T)]
# Evaluation
real_objective_value = 0
evaluation_results = []
for t in range(len(real_data)):
    inventory = solution[t]
    demand = real_data[t]
    loss = abs(inventory - demand)
    evaluation_results.append(loss)
    real_objective_value += loss
```

Code Generation Prompt for Budget RO

[System]
You are a Python programmer working in the fields of operations research and optimization.
You are proficient in using the third-party solving library like gurobipy. You will receive a specific task description, data, and a mathematical model based on robust optimization method. Your goal is to write a usable Python program.
You need to do:
1. Analyze optimization problem, historical data and mathematical model based on RO model.
2. Generate the solving code for the RO model.
3. Print the final solution of optimization problems.
4. After obtaining the final decision, evaluate it using real data. This requires you to explicitly implement the following processing flow in your code: table data extraction → list conversion → inputting actual values and final optimal solution into the objective function for calculation.
5. Print the real objective value.
6. Save the solution, actual value, corresponding decision loss for each decision step, and the real objective value as a text file format in {result_dir}.

Note:
1. Ensure the data loading path in your code is consistent with the provided {train_dir} and {test_dir}. Please strictly import empirical data from {train_dir} and real test data from {test_dir} , and do not fabricate data yourself.Additionally, strictly use "/" as the file path separator.
2. Ensure strict consistency between imported modules and code usage. For example, if GRB is used in the code, the import statement must contain 'from gurobipy import GRB'.
3. Please note that the table has headers, so do not set header=None when reading it.
4. Don't output any content contating 'exit()', which will lead to program crash.
5. Please give the complete code, including the code using "'python"'

[Task & Template Input]
We already have calculated parameters {d_bar} needed in budget uncertainty set, and {omega} is a parameter of budget uncertainty set. The empirical data is from {train_dir} and real test data is from {test_dir}.
Empirical data description is as follows:{data_description}.
Specific task: {description}.
Initial mathematical RO model: {initial_model}.
Transferred RO model: {math_model}.

[Template Shot - Output]

```python
import gurobipy as gp
import numpy as np
from gurobipy import GRB
import pandas as pd
import os
# Load Data
historical_data = pd.read_excel("{train_dir}")
real_data = pd.read_excel("{test_dir}").values.flatten().tolist()
T = len(real_data)  # Number of time periods
# Calculate nominal demand (mean of historical data)
d_bar = {d_bar}
# Model initialization
model = gp.Model("RobustInventoryManagement")
# Budget parameter
Gamma = 2
# Decision variables
x = model.addVars(T, lb=0, name="x")
# Auxiliary variables for absolute values
y = model.addVars(T, lb=0, name="y")
# Constraints for absolute value linearization
for t in range(T):
    model.addConstr(y[t] >= x[t] - d_bar[t], name=f"abs_pos_{{t}}")
    model.addConstr(y[t] >= d_bar[t] - x[t], name=f"abs_neg_{{t}}")
# Objective: minimize sum of absolute deviations
model.setObjective(gp.quicksum(y[t] for t in range(T)), GRB.MINIMIZE)
# Solve the model
model.optimize()
if model.status == GRB.OPTIMAL:
    print("\n=== OPTIMAL SOLUTION FOUND ===")
    # Extract solution
    x_opt = [x[t].X for t in range(T)]
    obj_value = model.objVal
    # Evaluate with real historical data
    real_objective_value = 0
    decision_losses = []
    for t in range(T):
        actual_demand = real_data[t]
        decision_loss = abs(x_opt[t] - actual_demand)
        decision_losses.append(decision_loss)
        real_objective_value += decision_loss
    print(f"\nReal objective value (sum of absolute deviations): {{real_objective_value:.2f}}")
    # Save results
    filename = os.path.join("{result_dir}")
    with open(filename, 'w') as f:
        f.write("Month\tInventory Level\tActual Demand\tLoss\n")
        for t in range(T):
          f.write(f"{{t+1}}\t{{x_opt[t]:.2f}}\t{{real_data[t]}}\t{{decision_losses[t]:.2f}}\n")
        f.write(f"\nReal Objective Value: {{real_objective_value:.2f}}\n")
    print(f"\nResults saved to: {{filename}}")
```

```
else:
    print("No optimal solution found")
```

Code Generation Prompt for Ellipsoidal RO

[System]
You are a Python programmer working in the fields of operations research and optimization.
You are proficient in using the third-party solving library like gurobipy. You will receive a specific task description, data, and a mathematical model based on robust optimization method. Your goal is to write a usable Python program.
You need to do:
1. Analyze optimization problem, historical data and mathematical model based on RO model.
2. Generate the solving code for the RO model.
3. Print the final solution of optimization problems.
4. After obtaining the final decision, evaluate it using real data. This requires you to explicitly implement the following processing flow in your code: table data extraction → list conversion → inputting actual values and final optimal solution into the objective function for calculation.
5. Print the real objective value.
6. Save the solution, actual value, corresponding decision loss for each decision step, and the real objective value as a text file format in `{result_dir}`.
Note:
1. Ensure the data loading path in your code is consistent with the provided `{train_dir}` and `{test_dir}`. Please strictly import empirical data from `{train_dir}` and real test data from `{test_dir}` , and do not fabricate data yourself.Additionally, strictly use "/" as the file path separator.
2. Ensure strict consistency between imported modules and code usage. For example, if GRB is used in the code, the import statement must contain 'from gurobipy import GRB'.
3. Please note that the table has headers, so do not set header=None when reading it.
4. Don't output any content contating 'exit()', which will lead to program crash.
5. Please give the complete code, including the code using "'python"'

[Task & Template Input]
We already have calculated parameters `{d_bar}` needed in budget uncertainty set, and `{omega}` is a parameter of budget uncertainty set. The empirical data is from `{train_dir}` and real test data is from `{test_dir}`.
Empirical data description is as follows:`{data_description}`.
Specific task: `{description}`.
Initial mathematical RO model: `{initial_model}`.
Transferred RO model: `{math_model}`.

[Template Shot - Output]

```python
import gurobipy as gp
import numpy as np
from gurobipy import GRB
import pandas as pd
import os
# Load Data
historical_data = pd.read_excel("{train_dir}")
real_data = pd.read_excel("{test_dir}").values.flatten().tolist()
T = len(real_data)  # Number of time periods
# Calculate nominal demand (mean of historical data)
d_bar = {d_bar}
# Model initialization
model = gp.Model("RobustInventoryManagement")
# Decision variables
x = model.addVars(T, lb=0, name="x")
# Auxiliary variables for absolute values |x_t - d_bar|
abs_diff = model.addVars(T, name="abs_diff")
# Constraints for absolute value linearization
for t in range(T):
    model.addConstr(abs_diff[t] >= x[t] - d_bar[t])
    model.addConstr(abs_diff[t] >= d_bar[t] - x[t])
# Objective: minimize sum of absolute deviations
objective = gp.quicksum(abs_diff[t] for t in range(T))
model.setObjective(objective, GRB.MINIMIZE)
# Solve the model
model.optimize()
if model.status == GRB.OPTIMAL:
    print("\n=== OPTIMAL SOLUTION FOUND ===")
```

```
    # Extract solution
    optimal_x = [x[t].X for t in range(T)]
    optimal_obj_value = model.objVal
    # Evaluate with real historical data
    real_objective_value = 0
    decision_losses = []
    for t in range(T):
        actual_demand = real_data[t]
        decision_loss = abs(optimal_x[t] - actual_demand)
        decision_losses.append(decision_loss)
        real_objective_value += decision_loss
    print(f"\nReal objective value (sum of absolute deviations): {{real_objective_value:.2f}}")
    # Save results
    filename = os.path.join("{result_dir}")
    with open(filename, 'w') as f:
        f.write("Month\tInventory Level\tActual Demand\tLoss\n")
        for t in range(T):
        f.write(f"{{t+1}}\t{{optimal_x[t]:.2f}}\t{{real_data[t]}}\t{{decision_losses[t]:.2f}}\n")
        f.write(f"\nReal Objective Value: {{real_objective_value:.2f}}\n")
print(f"\nResults saved to: {{filename}}")
else:
    print("No optimal solution found")
```

Code Generation Prompt for Wasserstein DRO

[System]
You are a Python programmer working in the fields of operations research and optimization.
You are proficient in using the third-party solving library like gurobipy. You will receive a specific task description, data, and a mathematical model based on robust optimization method. Your goal is to write a usable Python program.
You need to do:
1. Analyze optimization problem, historical data and mathematical model based on DRO model.
2. Generate the solving code for the DRO model.
3. Print the final solution of optimization problems.
4. After obtaining the final decision, evaluate it using real data. This requires you to explicitly implement the following processing flow in your code: table data extraction → list conversion → inputting actual values and final optimal solution into the objective function for calculation.
5. Print the real objective value.
6. Save the solution, actual value, corresponding decision loss for each decision step, and the real objective value as a text file format in `{result_dir}`.
Note:
1. Ensure the data loading path in your code is consistent with the provided `{train_dir}` and `{test_dir}`. Please strictly import empirical data from `{train_dir}` and real test data from `{test_dir}` , and do not fabricate data yourself.Additionally, strictly use "/" as the file path separator.
2. Ensure strict consistency between imported modules and code usage. For example, if GRB is used in the code, the import statement must contain 'from gurobipy import GRB'.
3. Please note that the table has headers, so do not set header=None when reading it.
4. Don't output any content contating 'exit()', which will lead to program crash.
5. Please give the complete code, including the code using "'python"'

[Task & Template Input]
We already have calculated the parameters needed in wasserstein ambiguity set, where `{mean}` represent the mean of empirical data for each decision step. The parameter(radius) needed in wasserstein ambiguity set is `{theta}`. The empirical data is from `{train_dir}` and real test data is from `{test_dir}`.
Empirical data description is as follows:`{data_description}`.
Specific task: `{description}`.
Initial mathematical DRO model: `{initial_model}`.
Transferred DRO model: `{math_model}`.

[Template Shot - Output]

```python
import gurobipy as gp
import numpy as np
from gurobipy import GRB
import pandas as pd
import os
# Load Data
historical_data = pd.read_excel("{train_dir}")
```

```
real_data = pd.read_excel("{test_dir}").values.flatten().tolist()
T, N = historical_data.shape
l_t = {{lower}}  # Lower bounds for each month
u_t = {{upper}}  # Upper bounds for each month
# Calculate mean for each timestep
mu = np.mean(historical_data, axis=1)
# Calculate lambda_t for each period
lambda_t = []
for t in range(T):
    if u_t[t] != l_t[t]:
        lambda_val = (u_t[t] - mu[t]) / (u_t[t] - l_t[t])
        lambda_val = max(0, min(1, lambda_val))
    else:
        lambda_val = 0.5
    lambda_t.append(lambda_val)
# Create model
model = gp.Model("DRO_Inventory_Optimization")
# Decision variables
x = model.addVars(T, lb=0.0, name="inventory")  # Inventory levels
a = model.addVars(T, lb=0.0, name="a")  # Auxiliary variables for |x_t - l_t|
b = model.addVars(T, lb=0.0, name="b")  # Auxiliary variables for |x_t - u_t|
# Objective function: minimize sum_t [lambda_t * a_t + (1 - lambda_t) * b_t]
objective = gp.quicksum(lambda_t[t] * a[t] + (1 - lambda_t[t]) * b[t] for t in range(T))
model.setObjective(objective, GRB.MINIMIZE)
# Constraints for absolute values
for t in range(T):
    model.addConstr(a[t] >= x[t] - l_t[t], f"a_lower_{{t}}")
    model.addConstr(a[t] >= l_t[t] - x[t], f"a_upper_{{t}}")
    model.addConstr(b[t] >= x[t] - u_t[t], f"b_lower_{{t}}")
    model.addConstr(b[t] >= u_t[t] - x[t], f"b_upper_{{t}}")
# Solve the model
model.setParam('OutputFlag', 1)
model.optimize()
# Check solution status
if model.status == GRB.OPTIMAL:
    print("\n=== OPTIMAL SOLUTION FOUND ===")
    # Extract solution
    x_opt = np.array([x[t].X for t in range(T)])
    obj_value = model.objVal
    # Evaluate with real historical data
    if real_data.shape[1] > 1:
        # If multiple samples, take mean for each timestep
        real_demand = np.mean(real_data, axis=1)
    else:
        # If single column, flatten
        real_demand = real_data.flatten()
    real_obj_value = np.sum(np.abs(x_opt - real_demand))
    decision_losses = np.abs(x_opt - real_demand)
    print("\nDecision losses (|x_t - d_t|):")
    for t in range(T):
        print(f"Month {{t+1}}: {{decision_losses[t]}}")
    print(f"\nObjective value based on real data: {{real_obj_value}}")
else:
    print("No optimal solution found")
```

Code Generation Prompt for KL DRO

[System]
You are a Python programmer working in the fields of operations research and optimization.
You are proficient in using the third-party solving library like Mosek. (Note that you need to use Mosek to implement the DRO model solving code if the model is convex but not linear). You will receive a specific task description, data, and a mathematical model based on distributionally robust optimization method. Your goal is to write a usable Python program.
You need to do:
1. Analyze optimization problem, historical data and mathematical model based on DRO model.

2. Generate the solving code for the DRO model (Golden section search need to be used to find the best parameter, which is the parameter needed in (KL) divergence ambiguity set DRO)
3. Print the final solution of optimization problems.
4. After obtaining the final decision, evaluate it using real data. This requires you to explicitly implement the following processing flow in your code: table data extraction → list conversion → inputting actual values and final optimal solution into the objective function for calculation.
5. Print the real objective value.
6. Save the solution, actual value, corresponding decision loss for each decision step, and the real objective value as a text file format in `{result_dir}`.
Note:
1. Ensure the data loading path in your code is consistent with the provided `{train_dir}` and `{test_dir}`. Please strictly import empirical data from `{train_dir}` and real test data from `{test_dir}` , and do not fabricate data yourself.Additionally, strictly use "/" as the file path separator.
2. Please use Golden section search to find the best parameter, which is the parameter needed in (KL) divergence ambiguity set DRO.
3. Please note that the table has headers, so do not set header=None when reading it.
4. Don't output any content contating 'exit()', which will lead to program crash.
5. Please give the complete code, including the code using "'python"'

[Task & Template Input]
We already have calculated the parameters needed in wasserstein ambiguity set, where `{mean}` represent the mean of empirical data for each decision step. The parameter(radius) needed in wasserstein ambiguity set is `{theta}`. The empirical data is from `{train_dir}` and real test data is from `{test_dir}`.
Empirical data description is as follows:`{data_description}`.
Specific task: `{description}`.
Initial mathematical DRO model: `{initial_model}`.
Transferred DRO model: `{math_model}`.

[Template Shot - Output]

```python
import cvxpy as cp
import numpy as np
import pandas as pd
import math
import os
# 1. Load data
train_dir = "{train_dir}"
test_dir = "{test_dir}"
historical_data = pd.read_excel(train_dir)
real_data = pd.read_excel(test_dir)
real_demands = real_data.values.flatten().tolist()
d_samples = historical_data.values
T, N = d_samples.shape
rho = 0.1
# 2. Define a function: solve DRO problem
def solve_for_lambda(lambda_val):
    x = cp.Variable(T, nonneg=True)
    #  L_s = sum_t |x_t - d_t^s|
    scenario_losses = [cp.sum(cp.abs(x - d_samples[:, s])) for s in range(N)]
    L = cp.hstack(scenario_losses)
    # KL-dual objective:
    objective = lambda_val * rho + lambda_val * cp.log_sum_exp(L / lambda_val - np.log(N))
    prob = cp.Problem(cp.Minimize(objective))
    prob.solve(solver=cp.MOSEK, verbose=False)
    if prob.status not in ["optimal", "optimal_inaccurate"]:
        return np.inf, None
    return prob.value, x.value
# 3. Golden section search for best
def golden_section_search(a=1e-3, b=10.0, tol=1e-2, max_iter=20):
    gr = (math.sqrt(5) + 1) / 2
    c = b - (b - a) / gr
    d = a + (b - a) / gr
    val_c, _ = solve_for_lambda(c)
    val_d, _ = solve_for_lambda(d)
    iter_count = 0
    while abs(b - a) > tol and iter_count < max_iter:
        iter_count += 1
        if val_c < val_d:
            b, d = d, c
```

```
            c = b - (b - a) / gr
            val_d = val_c
            val_c, _ = solve_for_lambda(c)
        else:
            a, c = c, d
            d = a + (b - a) / gr
            val_c = val_d
            val_d, _ = solve_for_lambda(d)
    lambda_star = (a + b) / 2
    best_val, x_star = solve_for_lambda(lambda_star)
    return lambda_star, best_val, x_star
# 4. Perform search
lambda_star, dro_value, x_star = golden_section_search()
print("\n========== Optimal Results ==========")
print(f"Optimal: {{lambda_star:.4f}}")
print(f"DRO objective value: {{dro_value:.4f}}")
print(f"Optimal decision: {{x_star}}")
# 5. Evaluate using real demand
real_losses = np.abs(x_star - real_demands)
total_loss = float(np.sum(real_losses))
print("\n========== Evaluation on Real Data ==========")
for i, v in enumerate(real_losses, start=1):
    print(f"t={{i}}: |x - d| = {{v:.4f}}")
# 6. Save results
result_dir = "results"
os.makedirs(result_dir, exist_ok=True)
result_path = os.path.join(result_dir, "dro_result.txt")
with open(result_path, "w", encoding="utf-8") as f:
    f.write("===== DRO Solution Report =====\n")
    f.write(f"Optimal: {{lambda_star:.6f}}\n")
    f.write(f"DRO Objective Value: {{dro_value:.6f}}\n\n")
    f.write("Per-step decisions (x_t):\n")
    for i, val in enumerate(x_star, 1):
        f.write(f"t={{i}}: x_t={{val:.4f}}\n")
    f.write("\nReal evaluation:\n")
    for i, v in enumerate(real_losses, 1):
        f.write(f"t={{i}}: |x - d| = {{v:.4f}}\n")
    f.write(f"\nTotal Real Objective: {{total_loss:.4f}}\n")
print(f"\n Results saved to {{result_path}}")
```