NLData2Opt: Benchmarking Parameter Derivation and Prediction for Modeling Optimization Problems from Natural Language and Data

Lijia Liu[1], **Kyohei Atarashi**[1], **Jiyi Li**[2],
Koh Takeuchi[1], **Shunji Umetani**[3], **Hisashi Kashima**[1]
[1]Kyoto University [2]Hokkaido University [3]Recruit Co., Ltd.

Abstract

Formulating optimization problems from raw inputs of problem descriptions and data is time-consuming and costly. While Large Language Models (LLMs) offer a promising way to automate this process, existing benchmarks overestimate their capabilities by relying on idealized inputs where all data is relevant and complete. This diverges from reality, where data-centric reasoning is required to extract, compute, and predict parameters from noisy sources. To address this, we introduce NLData2Opt, a benchmark to evaluate LLMs on the multi-stage reasoning inherent in practical optimization workflows. NLData2Opt decomposes problems into problem descriptions and data, and requires models to (1) extract relevant parameters from data containing misleading attributes, (2) compute missing parameters derived by transforming existing data attributes, and (3) predict unavailable future values from historical observations. Evaluation results show substantial performance decline relative to existing benchmarks, indicating that benchmark progress does not directly translate to formulation accuracy in practical optimization workflows. NLData2Opt is available at https://github.com/sumarivyn/NLData2Opt.

1 Introduction

Modeling real world problems through mathematical optimization to find optimal solutions has many applications, such as task scheduling (Cheng et al., 2024; Pallagani et al., 2024), vehicle routing (Baty et al., 2024; Luo et al., 2025), and healthcare planning (Tripathi et al., 2024; Tang et al., 2024a). Recent works explore using Large Language Models (LLMs) to automatically translate natural language inputs into optimization models, making the time-consuming and costly formulation process more accessible. In this setting, the natural language input generally consists of two components that summarize the optimization problem: a problem description and its corresponding data. The problem description defines the objective and constraints, and data provides the concrete parameters required to formulate the problem. A *parameter* denotes a known attribute specified in the data, while a *variable* denotes a decision quantity whose value is determined by solving the optimization problem.

Current benchmarks designed to evaluate LLMs on optimization modeling generally assume clean and well-structured inputs. Specifically, they assume complete and relevant problem descriptions and data are provided. For example, NL4Opt (Ramamonjison et al., 2023), Mamo (Huang et al., 2024), OptiGuide (Li et al., 2023a) and OPT-Engine (Chen et al., 2026) assume a single unified natural language input can be directly converted into an optimization model, and explicitly provide all parameters required for the formulation within the problem description. NLP4LP (AhmadiTeshnizi et al., 2023), ComplexOR (Wang et al., 2025) and ORThought (Yang et al., 2026) introduce problems with significantly longer natural language descriptions. Advanced benchmarks like IndustryOR (Tang et al., 2024b) shift towards problems from different industrial sectors, while OptiVerse (Zhang et al., 2026) covers complex domains such as stochastic programming. Optibench (Yang et al., 2025a) embed linear and nonlinear tabular data in the problem description to measure LLMs' end-to-end problem solving abilities.

However, despite recent progress, current benchmarks have two fundamental limitations. Firstly, data is directly embedded in the problem description, and the model's task is to locate and map them into an optimization formulation. This diverges from real-world modeling, in which the construction of the objective and constraints is usually a separate process from data collection, due to the scale and independent storage of the data. Secondly, it is assumed that every parameter is provided in the data in a ready-to-use form. This is rarely guaranteed with noisy and incomplete real-

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 51908–51933
October 24-29, 2026 ©2026 Association for Computational Linguistics

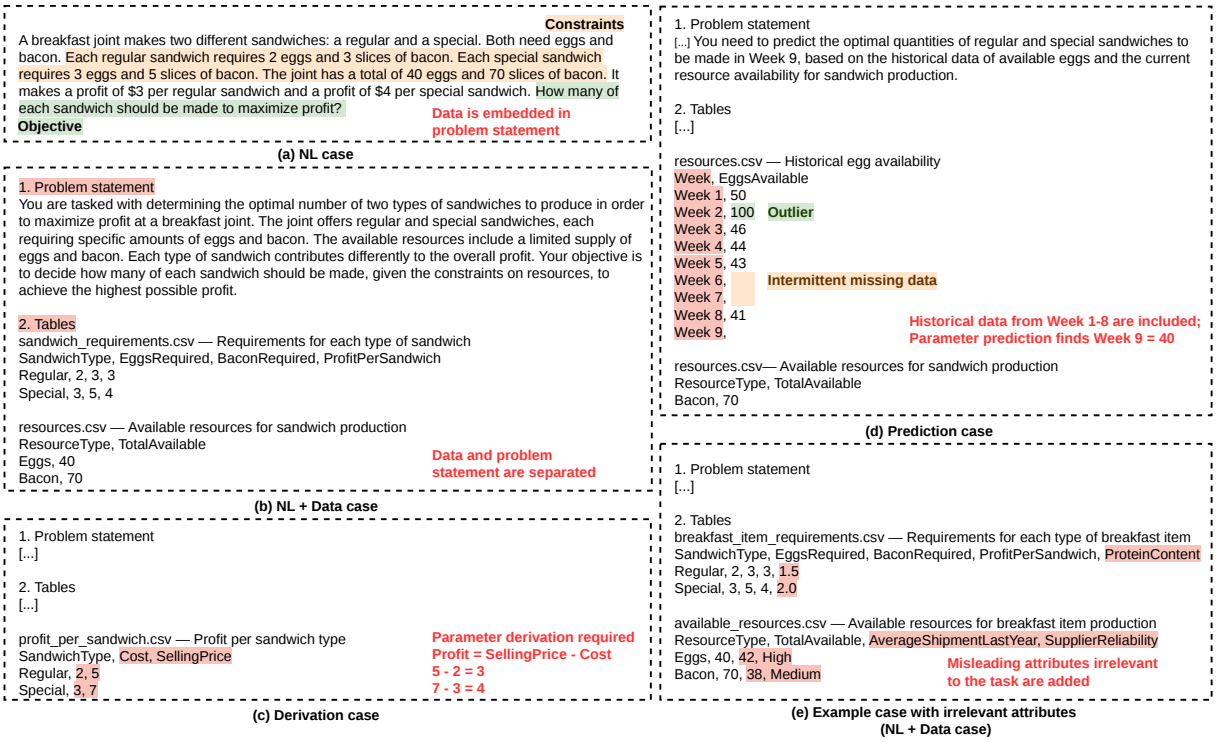

Figure 1: An optimization task in NLData2Opt with its components.

world data (Bengio et al., 2021). Since raw data typically contains several pieces of contextually related information, we have to identify through *parameter extraction* what information is actually necessary to solve the problem (Shi et al., 2023). Analysts must also frequently perform *parameter derivation*, which involves transforming multiple existing data attributes into a single usable value, such as calculating net profit from revenue and expenses. In cases such as demand forecasting or production scheduling, a predict-then-optimize pipeline is commonly adopted, where *parameter prediction* estimates unavailable future data from historical trends, and the predicted values are subsequently used in deterministic planning models (Elmachtoub and Grigas, 2022; Mandi et al., 2024). Due to these two limitations, it is difficult to evaluate LLMs' optimization modeling capabilities in real-world applications with existing benchmarks.

To address this, we introduce NLData2Opt, a novel benchmark that incorporates multi-stage reasoning through controlled and systematic tasks, as illustrated in Figure 1. Tasks are decomposed into two structurally separate components: a natural language problem description and a collection of CSV-style data tables. Noisy and incomplete data is reflected in the need to extract, compute, and predict parameters. Under the NLData2Opt framework, an LLM needs to fulfill three challenges to complete the optimization formulation: (1) extract relevant parameters from data containing misleading attributes, (2) compute missing parameters derived by transforming existing data attributes, and (3) predict unavailable future values from historical observations. We comprehensively evaluate state-of-the-art methods and ten open-weight and proprietary LLMs. Results show that current LLMs perform competitively in simplified tasks but achieve limited success on NLData2Opt.

Contributions. We make three contributions: (1) We introduce NLData2Opt, a benchmark that evaluates whether LLMs can perform multi-stage reasoning for optimization formulation, including model–data separation, information retrieval, implicit derivation, and future-value prediction. (2) We propose an automatic construction pipeline that transforms NLP4LP and Mamo tasks while preserving the feasible region and optimal objective value. (3) We provide comprehensive evaluations of prompting, agent-based, and fine-tuned baselines, which reveal substantial gaps between structural correctness and functional executability.

2 Related Work

Modeling an optimization problem into an accurate formulation is often time-consuming and requires domain expertise. To enhance this process, recent work has explored automating the formulation process, reducing the time and expertise needed.

Chain-of-thought variants include Tree of Thoughts (Yao et al., 2023) and Autoformulation (Astorga et al., 2025), while multi-agent systems include CoE (Wang et al., 2025), OptLLM (Xiao et al., 2024) and EoR (Yang et al., 2025b). Training-free approaches (Chen et al., 2024; Jiao et al., 2024; Li et al., 2023b) provide step-by-step instruction and incorporate prior knowledge into prompt design to improve performance. For learning-based methods, several existing works explore applying SFT with synthetic data (Tang et al., 2024b; Wang et al., 2024; Wu et al., 2025) or adopt reinforcement learning frameworks (Chen et al., 2025; Jiang et al., 2025b), while LLaMoCo (Ma et al., 2024) incorporates a contrastive learning-based warm-up procedure to enhance convergence behavior.

Techniques to improve formulation quality include highlighting named entities with XML-like tags (Gangwar and Kani, 2023). LLMOPT (Jiang et al., 2025a) constructs a five-element formulation and reports an 11.08% average improvement in solving accuracy over SOTA methods. NEDTree (Hu et al., 2026) and LEAN-LLM-OPT (Liang et al., 2026) extend to handle complex nonlinear and large-scale industrial formulations. During inference, format control such as five-element formulation, LaTeX code, or JSON outputs (Jiang et al., 2025a; JU et al., 2024; Wang et al., 2024) assists models to parse the problem into symbolic optimization models.

3 NLData2Opt

This section introduces NLData2Opt, a benchmark for evaluating LLMs on formulating optimization problems in realistic data conditions. We introduce the class of optimization problems considered, then outline the NLData2Opt task structures, and finally provide a validity criterion to ensure ground-truth solution validity.

3.1 Mixed Integer Linear Program

A mathematical optimization problem aims to find an optimal solution of decision variables that minimizes the objective function while satisfying all constraints. In this paper, we focus on modeling Linear Program (LP) and Mixed Integer Linear Programs (MILP), which is one of the most popular optimization problems having a wide variety of industrial applications. Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m_+$, $\boldsymbol{c} \in \mathbb{R}^n$, and a subset $I \subseteq N = \{1, \ldots, n\}$, the corresponding MILP $\mathcal{P}$ is given by

$$\begin{array}{lll} \min & \boldsymbol{c}^\top \boldsymbol{x} & \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}, & \\ & x_j \geq 0, & j \in N, \\ & x_j \in \mathbb{R}, & j \in N \setminus I, \\ & x_j \in \mathbb{Z}, & j \in I, \end{array} \tag{1}$$

where we call LP in case of $I = \emptyset$. Currently, many branch-and-cut solvers have been developed for MILP, where input data is written in the standard MPS (Mathematical Programming System) or LP formats. Alternatively, mathematical optimization problems can be specified via several modeling languages such as AMPL (Fourer et al., 1990) and GAMS (McCarl et al., 2012).

3.2 Task Definition in NLData2Opt

Given a natural language problem description $\mathcal{P}$ and a set of data $\mathcal{T}$, an LLM is required to formulate the task in AMPL. We define $\mathcal{T} = \mathcal{T}_{rel} \cup \mathcal{T}_{irr}$, where $\mathcal{T}_{rel}$ contains all parameters necessary for the optimization formulation, and $\mathcal{T}_{irr}$ denotes the attributes irrelevant to the problem. For all cases except the base case, we consider both $\mathcal{T}_{irr} = \emptyset$ and settings where $\mathcal{T}_{irr}$ contains irrelevant attributes. The objective is to produce an executable formulation that yields a solution to the optimization problem. We denote the generated AMPL model file and data file by $\mathcal{M}$ and $\mathcal{D}$, respectively.

Each instance consists of the following four standard cases, expanded into nine prompt settings.

NL Case (base case). The problem is expressed entirely in natural language, with all numerical parameters embedded in the text. The LLM directly generates $(\mathcal{M}, \mathcal{D})$ from $\mathcal{P}$:

$$\mathcal{F}(\mathcal{P}) \rightarrow (\mathcal{M}, \mathcal{D}).$$

NL+Data Case. The problem description $\mathcal{P}$ and dataset $\mathcal{T}$ are provided separately. The LLM generates

$$\mathcal{F}(\mathcal{P}, \mathcal{T}) \rightarrow (\mathcal{M}, \mathcal{D}).$$

Derivation Case. In addition to the separation of the problem description and data, each $k \in \mathbf{k}$ in a

subset of parameters $\mathbf{k} \subset \mathcal{T}_{rel}$ is defined implicitly through intermediate quantities Q_k. The LLM must compute the parameters via an algebraic functional relationship:

$$k = \phi(Q_k), \quad \forall k \in \mathbf{k},$$

before constructing the optimization model. This transformation preserves the feasible region and the optimal solution while introducing derivation-oriented reasoning. The optimization problem is formulated as:

$$\mathcal{F}(\mathcal{P}, (\mathcal{T} \setminus \mathbf{k}) \cup \bigcup_{k \in \mathbf{k}} Q_k) \rightarrow (\mathcal{M}, \mathcal{D}).$$

Prediction Case. In addition to the separation of the problem description and data, a subset of parameters $\mathbf{t} \subset \mathcal{T}_{rel}$ is masked from the data tables. Instead, for each missing parameter $t \in \mathbf{t}$, a sequence of historical observations $H_t = \{h_1, \ldots, h_n\}$ ($n \geq 7$) is provided. We denote the full set of historical data as $\mathcal{H}$. The LLM must first perform a forecasting step to estimate the missing values:

$$\psi(\mathcal{H}) \rightarrow \hat{\mathbf{t}},$$

where $\hat{\mathbf{t}}$ is the set of predicted values for the masked parameters. The optimization problem is formulated as:

$$\mathcal{F}(\mathcal{P}, (\mathcal{T} \setminus \mathbf{t}) \cup \bigcup_{t \in \mathbf{t}} H_t)) \rightarrow (\mathcal{M}, \mathcal{D}).$$

3.3 Validity Criterion

To maintain the validity of the benchmark, every transformation must satisfy the following criterion. Let $S^*(\mathcal{O})$ denote the optimal objective value of an optimization problem $\mathcal{O}$. Let $\mathcal{O}_{\text{orig}}$ be the original problem and $\mathcal{O}_{\text{trans}}$ its transformed counterpart. Each transformation $\Psi \in \{\text{Base}, \text{Prediction}, \text{Derivation}\}$ is required to preserve the original feasible region and optimal solution. Accordingly, we ensure that

$$S^*(\mathcal{O}_{\text{trans}}) = S^*(\mathcal{O}_{\text{orig}}), \tag{2}$$

which guarantees that the ground-truth solution of the optimization problem remains invariant under all transformations.

4 Automatic Construction of NLData2Opt

4.1 Separation of Problem Description and Data

Each task in NLData2Opt, except for the NL case, is decomposed into two structurally separate components to simulate practical optimization workflows:

Problem description. A structured natural language description that specifies the objective and constraints, but does not include any concrete numerical values. This component guides the LLM to construct the AMPL model file.

Data. A collection of CSV-style tables that contain numerical parameters. Each table includes a name, a brief description and multiple rows of numerical or categorical data. Provided independently of the problem description, this component guides the LLM to construct the AMPL data file.

The transformation process for the prediction and derivation cases follows a multi-stage pipeline, as illustrated in Figure 2. In the single-parameter prediction case, we first augment the data with historical observations. An AMPL Formulator samples from predefined prompt templates to produce controlled temporal patterns that systematically evaluate whether models can predict parameters and integrate them into optimization formulations. Patterns such as seasonality, abrupt changes, intermittent missing data and outliers are then introduced to simulate richer temporal dynamics. Finally, the problem description is revised to explicitly specify the parameter to be predicted and the corresponding future time period.

The multi-parameter prediction case extends this procedure by applying the same transformation to multiple target parameters, incorporating historical data and prediction requirements for each parameter independently.

For derivation case, directly transforming existing optimization problems to require intermediate calculations in a single step proved challenging for GPT-4o. Instead, we adopt a two-stage transformation strategy. The Context Generator first revises the task at the natural language level to introduce intermediate quantities from which the target parameter p can be computed. Only after this modification is the revised task separated into a problem description and corresponding data tables. To further encourage valid derivation-based optimization formulations, we employ few-shot prompting during generation. Detailed prompt templates are provided in Appendix III-IV.

4.2 Generating Gold Solutions in AMPL

To construct gold-standard solutions for NLData2Opt, we adopt AMPL as the modeling language for mathematical programming. This choice

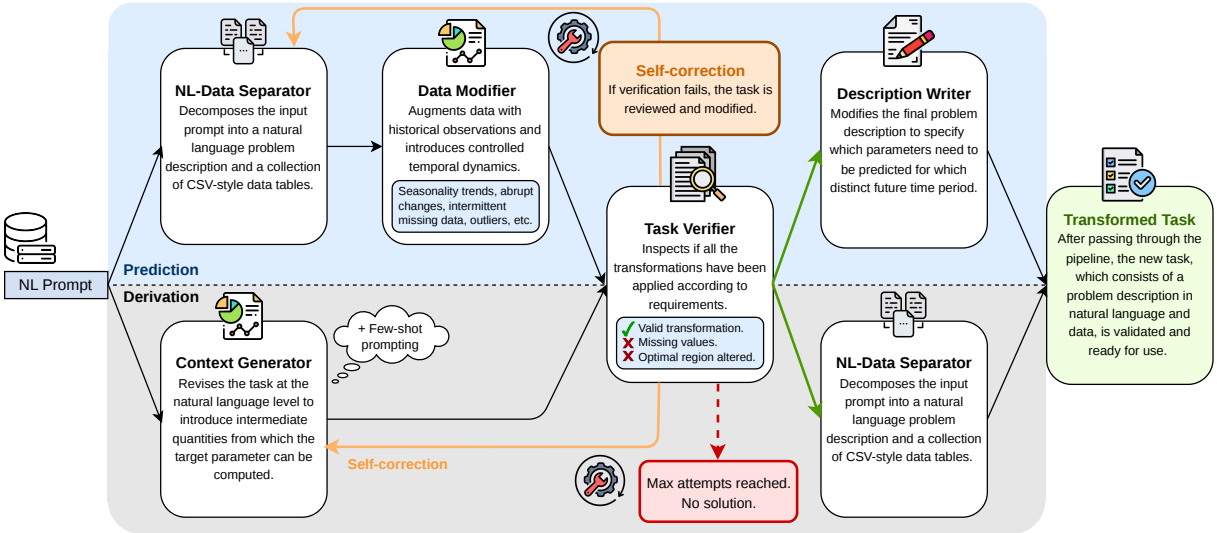

Figure 2: Transformation pipeline for prediction and derivation cases.

```
Model:
set SANDWICHES;                                          Sets

param EggsRequired {SANDWICHES};                         Parameters
param BaconRequired {SANDWICHES};
param Profit {SANDWICHES};

param TotalEggs;
param TotalBacon;

var Make {s in SANDWICHES} >= 0, integer;

s.t. EggLimit:                                           Constraints
   sum {s in SANDWICHES} EggsRequired[s] * Make[s] <= TotalEggs;

s.t. BaconLimit:
   sum {s in SANDWICHES} BaconRequired[s] * Make[s] <= TotalBacon;

maximize TotalProfit:                                    Decision variables
   sum {s in SANDWICHES} Profit[s] * Make[s];            Objective
```

```
Data:
set SANDWICHES := Regular Special;                       Sets

param EggsRequired :=                                    Parameters
Regular 2
Special 3;

param BaconRequired :=
Regular 3
Special 5;

param Profit :=
Regular 3
Special 4;

param TotalEggs := 40;

param TotalBacon := 70;
```

Figure 3: AMPL formulation of an example optimization task with model–data separation.

is motivated by several considerations. (1) AMPL explicitly separates the *model*, which details the sets, parameters, decision variables, objectives, and constraints, from the *data*, which defines the specific set and parameter values, as shown in Figure 3. This encourages more compact and readable model definitions, while mirroring practical scenarios where large-scale data is stored separately. (2) Treating data as a separate component allows modifications without altering the underlying model structure, allowing the same model to be evaluated on different datasets or vice versa (Fourer et al., 1990). (3) AMPL is a declarative language based on algebraic representations of objectives and constraints, which is natural and familiar to practitioners in large-scale mathematical programming.

To generate AMPL formulations, we adopt a CoE framework, as illustrated in Figure 4. Our implementation builds on prior work (Wang et al., 2025), with prompts adapted to produce AMPL files. The system consists of multiple specialized experts, each responsible for a specific subtask. A Problem Classifier first identifies the optimization problem class (e.g., transshipment or max-flow). If a standard template is available, it is passed directly to the AMPL Formulator; otherwise, a Model Constructor and a Data Extractor generate the formulation from scratch. The resulting code is then refined through an iterative Code Runtime and Self-correction loop. Detailed prompt templates for each expert are provided in Appendix I.

4.3 Evaluation Metrics

Since generated formulations may be partially correct, we adopt three complementary metrics to evaluate outputs from different perspectives.

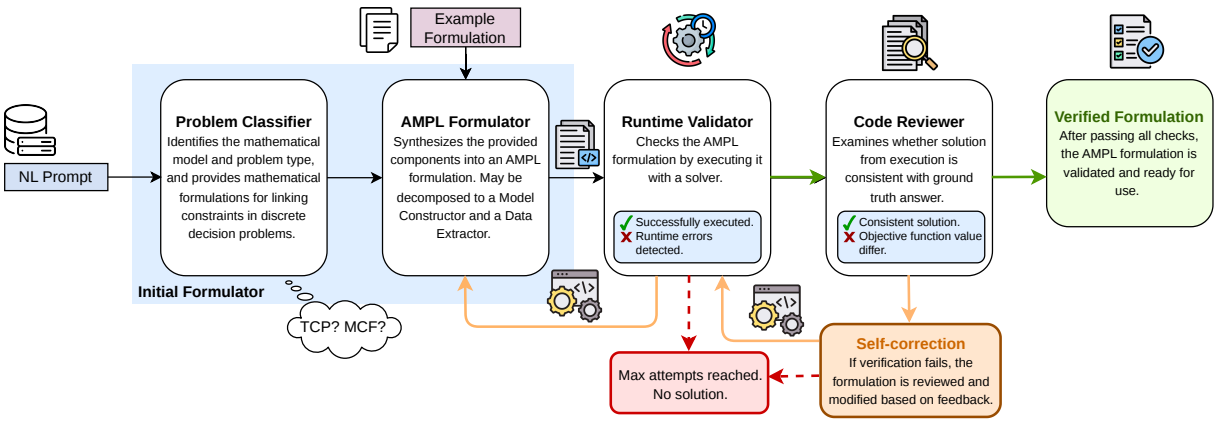

Figure 4: Ground-truth AMPL formulation construction framework.

Entity F1 Score (EF1). Structural correctness is measured using an entity-level F1 score. Each AMPL model is represented as a multiset $\mathcal{E}$ of entity types drawn from

$$\Omega = \{\text{set}, \text{parameter}, \text{variable}, \text{constraint}, \text{objective}\}.$$

Let $\mathcal{E}_{\text{cand}}$ and $\mathcal{E}_{\text{gt}}$ denote the multisets extracted from the candidate and ground-truth models, respectively. The number of true positives is computed as $\text{TP} = \sum_{\omega \in \Omega} \min\big(\text{mult}(\mathcal{E}_{\text{cand}}, \omega), \text{mult}(\mathcal{E}_{\text{gt}}, \omega)\big)$, where $\text{mult}(\mathcal{E}, \omega) := |\{e \in \mathcal{E} \mid e = \omega\}|$ denotes the multiplicity of entity type ω in $\mathcal{E}$. Precision and recall are defined as $\text{P} = \frac{\text{TP}}{|\mathcal{E}_{\text{cand}}|}$ and $\text{R} = \frac{\text{TP}}{|\mathcal{E}_{\text{gt}}|}$, and the F1 score becomes $\text{F1} = \frac{2\text{PR}}{\text{P}+\text{R}}$. This captures structural correctness independent of variable or constraint naming.

Execution-Based Correctness (EBC). Functional correctness is evaluated by executing each model–data pair using the AMPL interpreter with the CBC solver. If execution fails, the score is set to 0. Otherwise, a smoothed correctness score is computed:

$$\text{Correctness} = \begin{cases} \max\big(0, 1 - \left|\frac{O-A}{A}\right|\big) & \text{if } A \neq 0, \\ 1 & \text{if } O = A = 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where O denotes the objective value from the candidate formulation and A is the ground-truth value.

LLM-Based Evaluation Score (LLM-ES). We employ GPT-4o-mini to perform a holistic evaluation of candidate formulations. The evaluator compares the constraints specified in the AMPL model and the parameter values specified in the data against the corresponding components of the gold-standard formulation, assigning a score on a 0-10 scale. The full evaluation prompt is provided in Appendix V.

4.4 Quality Control

We rigorously validate each generated AMPL formulation through a combination of execution-based and semantic checks. Every formulation is executed to detect runtime or modeling errors, and the resulting objective value is compared against the ground-truth solution. In addition, we employ LLM-based reviewers to perform holistic verification of formulation soundness. When validating different cases of natural language problem descriptions and data pairs, these reviewers ensure that all transformation rules are satisfied and that each transformed instance preserves the original feasible region and optimal solution.

5 Experiments

We develop NLData2Opt and comprehensively evaluate SOTA methods to assess their ability to formulate and solve realistic optimization problems from natural language and data. The observed performance decline highlights the challenges posed by NLData2Opt and indicates that current methods are not yet capable of fully addressing such tasks.

5.1 Experimental Setup

Benchmarks. We construct NLData2Opt from two existing benchmarks: NLP4LP (AhmadiTeshnizi et al., 2023) and Mamo (Huang et al., 2024). NLP4LP consists of 269 LP and MILP tasks collected from optimization textbooks and lecture notes. Mamo is divided into two subsets: Easy LP, which contains 647 high school-level MILP

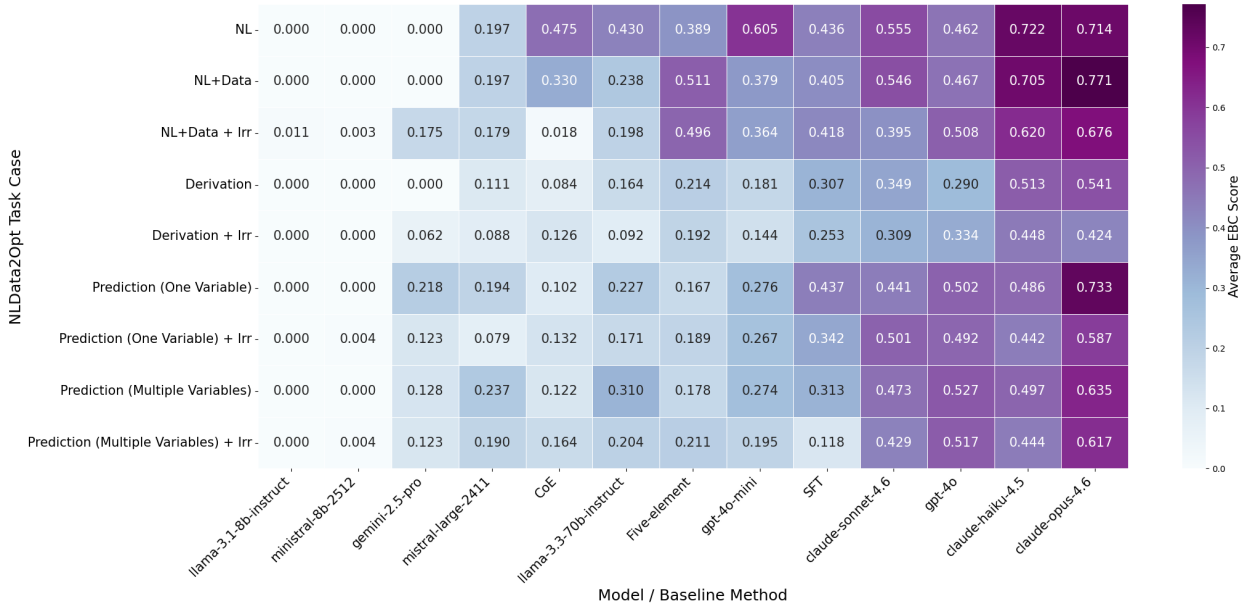

Figure 5: **Average EBC scores across task cases.** The heatmap illustrates functional correctness achieved on the nine task cases of NLData2Opt (rows, cases with irrelevant attributes are marked with "+ Irr"), when evaluated with an LLM model or baseline method (columns). The columns are ranked to reflect the worst to best performing models. Higher saturation represents a higher EBC score. While models with smaller capacity did not generate executable code, stronger models only reached scores as high as 0.77, highlighting the challenges of the tasks.

problems, and Complex LP, which includes 210 undergraduate-level tasks integrating both LP and MILP formulations.

Baselines. We evaluate NLData2Opt using a diverse set of methods. Proprietary LLMs evaluated under direct prompting include GPT-4o-mini (`gpt-4o-mini-2024-07-18`), GPT-4o (`gpt-4o-2024-11-20`), Gemini 2.5 Pro, Claude Haiku 4.5 (`claude-haiku-4-5-20251001`), Claude Sonnet 4.6 and Claude Opus 4.6. Open-weight models include Llama-3.1-8B-Instruct, Ministral-3-8B-2512, Llama-3.3-70B-Instruct and Mistral-Large-2411. We use a temperature of 0.1 and limit each model to a single attempt per task. Generated formulations are normalized to AMPL `.mod`/`.dat` format using the reformatting prompt in Appendix V and executed using the AMPL interpreter with the CBC solver. Non-executable outputs are assigned zero EBC.

In addition, we evaluate several structured baselines: the five-element formulation approach proposed by LLMOPT using GPT-4o-mini, CoE with its default backbone GPT-3.5-turbo, and a SFT baseline trained on Qwen3-1.7B.

5.2 Results and Analysis

Figure 5 reports EBC across nine prompt settings: NL, NL+Data, derivation, two prediction cases, and four cases with irrelevant attributes. Comprehensive evaluation results for all metrics (EF1, EBC and LLM-ES) are summarized in Tables 7-15 in Appendix VIII, averaged over a 15% subset per dataset using a random seed of 42. Additionally, we report full results in Tables 16-18 for three representative models: claude-opus-4.6, gpt-4o-mini, and llama-3.3-70b-instruct. Overall, current LLMs and other baseline methods achieve limited success on NLData2Opt, with subset-based evaluation showing trends consistent with the full benchmark.

Impact of model–data separation. Transitioning from NL (Table 7) to NL+Data cases (Table 8), LLMs struggle with mapping declared variables in the model to instantiated values in the data. Consequently, we observe a general decrease in EBC, even when EF1 remains relatively stable. However, we note that all proprietary models achieve higher EBC on Mamo Complex for NL+Data case. As Mamo Complex contains more complicated tasks, extracting data into separate tables might have provided more clarity and structure.

Qualitative analysis shows that LLMs tend to embed coefficients directly into constraints in the NL case. In contrast, in the NL+Data case, LLMs use a more elegant approach by defining the parameters in the model file and recording the corresponding values in the data file. While both methods yield the

same solution, parameterizing is more desirable as it allows for higher scalability and generalizability.

Impact of irrelevant attributes. Our experiments illustrate that the performance drop is not simply due to noise. Rather, when multiple conceptually related pieces of information are present, models struggle to identify the correct data. In our initial experiments, adding irrelevant attributes did not significantly affect results. However, after introducing more semantically similar attributes, we observe performance declines across all three metrics. Error analysis (Figure 6) further confirms this, revealing that irrelevant attributes increase undefined entities errors, including a rise from 36.3% to 47.6% for the derivation case. This confirms that the presence of semantically similar yet irrelevant data hinders the model's ability to maintain a mapping between declared variables and instantiated data.

Derivation case analysis. The derivation cases (Table 10) receive drops in EBC across baselines. Table 4 reports the percentage of correctly identified models, revealing that correct constraint structures are relatively low. This indicates that models struggle to maintain structural consistency when parameters are not explicitly provided.

Qualitative analysis further supports that models often struggle during optimization formulation that requires parameter derivation. When a total quantity must be derived from multiple components, models may fail to aggregate, generating multiple false constraints instead. Moreover, even when derivation is performed correctly, models introduce irrelevant constraints that unnecessarily restrict the feasible region. This suggests that models struggle to integrate intermediate derived results into correct downstream optimization constraints.

Prediction case analysis. Performance drops in both cases, though one-variable prediction (Table 12) does not show significant EBC improvement over multiple-variable cases (Table 14). Table 5 shows that even if an exact match is not achieved, predictions often lie in the ground truth value's vicinity. Simple forecasting and oracle-prediction baselines (Tables 1-3) confirm that forecasting errors alone cannot explain the observed performance gap caused by formulation errors.

Qualitative analysis shows that models occasionally predict incorrect values, such as directly copying from the second-to-last time period. Others correctly predict the required parameters but fail to incorporate them into the formulation. Across both derivation and prediction cases, common AMPL execution errors include undefined entities and undeclared data assignments. Therefore, even when arithmetic or forecasting is correct, integrating intermediate results into downstream constraints remains a key challenge in multi-step tasks.

5.3 Human Alignment

Although human evaluation remains the gold standard, evaluating NLData2Opt presents scalability challenges. Each task requires the evaluation of nine distinct cases, and assessing AMPL formulations requires domain expertise. Consequently, comprehensive human evaluation on the entire benchmark across baselines is far too expensive.

To provide a scalable and cost-effective alternative, we adopt an LLM-as-a-judge framework using the GPT-4o-mini-based LLM-ES metric. To verify this approach, we manually evaluated 108 candidate formulations generated by Claude Opus 4.6. The instructions used were identical to the LLM judge's prompt (Appendix V). The evaluation yields a Spearman's rank correlation coefficient of $\rho = 0.39$ ($p = 3.90 \times 10^{-6}$) and a Pearson linear correlation coefficient of $r = 0.30$ ($p = 3.90 \times 10^{-4}$). These results indicate a positive alignment with human judgments, consistent with prior work on LLM-based evaluators (Bavaresco et al., 2025; Schroeder and Wood-Doughty, 2024). Results suggest that evaluating specialized programming is more challenging than evaluating open-ended natural language generation. While LLM-ES provides a scalable signal, its moderate correlation with human judgments suggests it should be interpreted with caution and used alongside other execution-based metrics.

6 Conclusion

We introduced NLData2Opt, a benchmark designed to evaluate optimization modeling under realistic data conditions. NLData2Opt reflects practical workflows by separating data from problem descriptions and incorporating irrelevant yet misleading attributes. In addition, it requires the derivation of missing parameters and prediction of future values that are common in applied optimization but largely absent from prior benchmarks.

Through comprehensive evaluations, we demonstrate that current LLMs and baseline methods achieve only limited success on NLData2Opt. These findings underscore the need for more robust approaches to optimization problem formulation.

Limitations

NLData2Opt utilizes LLM-based transformations for benchmark construction, which may contain hallucinated information, temporal inconsistencies, or other logical distortions. They may also introduce synthetic parameters that differ systematically from real-world data distributions. Common mitigation strategies include human evaluation, comparison between real and synthetic data, error analysis, and execution-based validation. In NLData2Opt, we reduce these risks through execution-based checks of generated optimization formulations and LLM-based validation of transformed instances against predefined consistency criteria. Nevertheless, automated validation cannot guarantee that all semantic or logical errors are detected, and certain distortions may remain in the benchmark.

Because NLData2Opt is derived from existing datasets such as NLP4LP and Mamo, some source instances may have appeared in the pretraining data of evaluated models. Our transformations substantially modify the original tasks, but potential data contamination cannot be fully ruled out.

Our conclusions on industrial realism and general optimization are limited to the LP/MILP task families, data perturbations, and AMPL-based formulation setting studied in this work, and should not be interpreted as directly representing the full complexity of industrial optimization problems.

Finally, while NLData2Opt requires LLMs to predict or derive parameters before generating formal optimization formulations, this workflow may emphasize formalization ability over deeper conceptual reasoning. For example, an LLM may correctly construct an AMPL formulation using a predicted parameter without fully understanding how that parameter affects the feasible region or optimal solution. Future benchmarks could evaluate conceptual understanding and decision-making independently from formulation ability, enabling more nuanced assessments of mathematical reasoning.

Ethical considerations

NLData2Opt is constructed by adapting open-source datasets NLP4LP and Mamo. Both datasets are licensed under CC BY NC 4.0. We respect their creators' ownership and intellectual property, and our use of these artifacts is strictly consistent with their intended purpose of evaluating the mathematical modeling capabilities of LLMs.

NLData2Opt is a benchmarking dataset designed to evaluate LLMs on optimization tasks. The tasks presented focus solely on mathematical optimization modeling and do not contain personal or sensitive data. We plan to release NLData2Opt under the same CC BY NC 4.0 license, allowing only non-commercial use for research purposes.

AI assistants were utilized to support prompt template development and improve the clarity of this paper. We maintain full responsibility for the content, experimental design, and conclusions presented.

Acknowledgments

This work was supported by JST CREST Grant Number JPMJCR21D1.

References

Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. 2023. Optimus: Optimization modeling using MIP solvers and large language models. *CoRR*, abs/2310.06116.

Nicolás Astorga, Tennison Liu, Yuanzhang Xiao, and Mihaela van der Schaar. 2025. Autoformulation of mathematical optimization models using llms. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net.

Léo Baty, Kai Jungel, Patrick S. Klein, Axel Parmentier, and Maximilian Schiffer. 2024. Combinatorial optimization-enriched machine learning to solve the dynamic vehicle routing problem with time windows. *Transportation Science*, 58(4):708–725.

Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, Esam Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, André F. T. Martins, Philipp Mondorf, Vera Neplenbroek, Sandro Pezzelle, Barbara Plank, David Schlangen, Alessandro Suglia, Aditya K. Surikuchi, Ece Takmaz, and Alberto Testoni. 2025. Llms instead of human judges? A large scale empirical study across 20 NLP evaluation tasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 238–255. Association for Computational Linguistics.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. 2021. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421.

Hao Chen, Gonzalo E. Constante-Flores, and Can Li. 2024. Diagnosing infeasible optimization problems using large language models. *INFOR Inf. Syst. Oper. Res.*, 62(4):573–587.

Yitian Chen, Cheng Cheng, Yinan Sun, Zi Ling, and Dongdong Ge. 2026. Opt-engine: Benchmarking the limits of llms in optimization modeling via complexity scaling. *CoRR*, abs/2601.19924.

Yitian Chen, Jingfan Xia, Siyu Shao, Dongdong Ge, and Yinyu Ye. 2025. Solver-informed RL: grounding large language models for authentic optimization modeling. *CoRR*, abs/2505.11792.

Yuqing Cheng, Zhiying Cao, Xiuguo Zhang, Qilei Cao, and Dezhen Zhang. 2024. Multi objective dynamic task scheduling optimization algorithm based on deep reinforcement learning. *J. Supercomput.*, 80(5):6917–6945.

Junhyuk Choi, Sohhyung Park, Chanhee Cho, Hyeonchu Park, and Bugeun Kim. 2026. Diagnosing the reliability of llm-as-a-judge via item response theory. *CoRR*, abs/2602.00521.

Adam N. Elmachtoub and Paul Grigas. 2022. Smart "predict, then optimize". *Manag. Sci.*, 68(1):9–26.

Robert Fourer, David M Gay, and Brian W Kernighan. 1990. Ampl: A mathematical programming language. *Management Science*, 36(5):519–554.

Neeraj Gangwar and Nickvash Kani. 2023. Highlighting named entities in input for auto-formulation of optimization problems. In *Intelligent Computer Mathematics - 16th International Conference, CICM 2023, Cambridge, UK, September 5-8, 2023, Proceedings*, volume 14101 of *Lecture Notes in Computer Science*, pages 130–141. Springer.

Zhijing Hu, Yufan Deng, Haoyang Liu, and Changjun Fan. 2026. Ned-tree: Bridging the semantic gap with nonlinear element decomposition tree for LLM nonlinear optimization modeling. *CoRR*, abs/2604.01588.

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

Caigao Jiang, Xiang Shu, Hong Qian, Xingyu Lu, Jun Zhou, Aimin Zhou, and Yang Yu. 2025a. LLMOPT: learning to define and solve general optimization problems from scratch. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Xia Jiang, Yaoxin Wu, Minshuo Li, Zhiguang Cao, and Yingqian Zhang. 2025b. Large language models as end-to-end combinatorial optimization solvers. *CoRR*, abs/2509.16865.

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

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

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

Ran Li, Chuanqing Pu, Junyi Tao, Canbing Li, Feilong Fan, Yue Xiang, and Sijie Chen. 2023b. Llm-based frameworks for power engineering from routine to novel tasks. *Preprint*, arXiv:2305.11202.

Kuo Liang, Yuhang Lu, Jianming Mao, Shuyi Sun, Chunwei Yang, Congcong Zeng, Xiao Jin, Hanzhang Qin, Ruihao Zhu, and Chung-Piaw Teo. 2026. Large-scale optimization model auto-formulation: Harnessing LLM flexibility via structured workflow. *CoRR*, abs/2601.09635.

Fu Luo, Xi Lin, Yaoxin Wu, Zhenkun Wang, Tong Xialiang, Mingxuan Yuan, and Qingfu Zhang. 2025. Boosting neural combinatorial optimization for large-scale vehicle routing problems. In *International Conference on Representation Learning*, volume 2025, pages 14754–14780.

Zeyuan Ma, Hongshu Guo, Jiacheng Chen, Guojun Peng, Zhiguang Cao, Yining Ma, and Yue-Jiao Gong. 2024. Llamoco: Instruction tuning of large language models for optimization code generation. *CoRR*, abs/2403.01131.

Jayanta Mandi, James Kotary, Senne Berden, Maxime Mulamba, Victor Bucarey, Tias Guns, and Ferdinando Fioretto. 2024. Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *J. Artif. Intell. Res.*, 80:1623–1701.

Bruce McCarl, Alex Meeraus, Paul van der Eijk, Michael Bussieck, Steven Dirkse, and Franz Nelissen. 2012. Mccarl expanded gams user guide version 23.6. *GAMS Development, Washington, DC*.

Vishal Pallagani, Bharath C. Muppasani, Kaushik Roy, Francesco Fabiano, Andrea Loreggia, Keerthiram Murugesan, Biplav Srivastava, Francesca Rossi, Lior Horesh, and Amit P. Sheth. 2024. On the prospects of incorporating large language models (llms) in automated planning and scheduling (APS). In *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2024, Banff, Alberta, Canada, June 1-6, 2024*, pages 432–444. AAAI Press.

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

Kayla Schroeder and Zach Wood-Doughty. 2024. Can you trust LLM judgments? reliability of llm-as-a-judge. *CoRR*, abs/2412.12509.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, Proceedings of Machine Learning Research, pages 31210–31227. PMLR.

Paul Mingzheng Tang, Kenji Kah Hoe Leong, Nowshad Shaik, and Hoong Chuin Lau. 2024a. Automated conversion of static to dynamic scheduler via natural language. *CoRR*, abs/2405.06697.

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

Satvik Tripathi, Rithvik Sukumaran, and Tessa S Cook. 2024. Efficient healthcare with large language models: optimizing clinical workflow and enhancing patient care. *Journal of the American Medical Informatics Association*, 31(6):1436–1440.

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

Zihan Wang, Rui Pan, Jiarui Yao, Robert Csordas, Linjie Li, Lu Yin, Jiajun Wu, Tong Zhang, Manling Li, and Shiwei Liu. 2025. Chain-of-experts: Unlocking the communication power of mixture-of-experts models. *CoRR*, abs/2506.18945.

Yang Wu, Yifan Zhang, Yurong Wu, Yuran Wang, Junkai Zhang, and Jian Cheng. 2025. Evo-step: Evolutionary generation and stepwise validation for optimizing llms in or.

Ziyang Xiao, Dongxiang Zhang, Xiongwei Han, Xiaojin Fu, Wing Yin Yu, Tao Zhong, Sai Wu, Yuan Wang, Jianwei Yin, and Gang Chen. 2024. Enhancing LLM reasoning via vision-augmented prompting. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Beinuo Yang, Qishen Zhou, Junyi Li, Chenxing Su, Panagiotis Angeloudis, and Simon Hu. 2026. Orthought: Benchmarking and automating logistics optimization modeling via structured llm reasoning. *Artificial Intelligence for Transportation*, 6:100059.

Zhicheng Yang, Yiwei Wang, Yinya Huang, Zhijiang Guo, Wei Shi, Xiongwei Han, Liang Feng, Linqi Song, Xiaodan Liang, and Jing Tang. 2025a. Optibench meets resocratic: Measure and improve llms for optimization modeling. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Zhicheng Yang, Yiwei Wang, Yinya Huang, Zhijiang Guo, Wei Shi, Xiongwei Han, Liang Feng, Linqi Song, Xiaodan Liang, and Jing Tang. 2025b. Optibench meets resocratic: Measure and improve llms for optimization modeling. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Xinyu Zhang, Boxuan Zhang, Yuchen Wan, Lingling Zhang, YiXing Yao, Bifan Wei, Yaqiang Wu, and Jun Liu. 2026. Optiverse: A comprehensive benchmark towards optimization problem solving. *CoRR*, abs/2604.21510.

Appendix

We provide the details of the exact templates utilized in the automated-generation of NLData2Opt that were omitted in the main text. GPT-4o-mini (`gpt-4o-mini-2024-07-18`) and GPT-4o (`gpt-4o-2024-11-20`) were used with a temperature of 0.1. While GPT-4o-mini primarily constructed NL+Data and prediction cases for NLP4LP, GPT-4o was employed for more complex tasks, including derivative cases within NLP4LP and task instances that were challenging for GPT-4o-mini to process effectively.

I Templates for AMPL formulation construction

To construct high-quality and executable gold-standard solutions for NLData2Opt, we utilize a CoE framework adapted to produce AMPL code. This pipeline decomposes the complex modeling task into subtasks handled by respective experts. The initial formulator proposed an initial AMPL formulation, and consists of the following experts:
Problem Classifier: Identifies the precise mathematical model and objective (e.g., transshipment or max-flow) while ensuring all numerical inputs are treated as generic parameters. It also provides mathematical formulations for linking constraints in discrete decision problems. If a standard template is available, it is passed directly to the AMPL

Formulator.

AMPL Formulator: Synthesizes the provided components into an executable .mod file, adhering to the blueprint and ensuring no numerical data values are included in the model logic.

Model Constructor: Collaborates with the Data Extractor to generate the formulation from scratch if a standard template is unavailable.

Data Extractor: Extracts data from the base case and synthesizes the final components into AMPL .mod and .dat files.

After executing the generated AMPL formulation, the following experts revise the initial formulation through an iterative loop:

Debugger: Identifies the source of numerical errors, determining if the discrepancy between predicted and expected values arises from model logic or data errors.

Code Reviewer: Conducts a detailed analysis of the previous AMPL formulation, identifying whether the discrepancy arises from model logic or data errors, and providing a corrected proposed fix.

Code Modifier: Implements the proposed modifications by the Code Reviewer to ensure the final AMPL code accurately models the optimization problem.

Listing 1: Prompt template for Problem Classifier

```
As the Optimization Modeler and Knowledge
    Retriever, your primary role is to create a
    complete, mathematically precise blueprint
    for the model, ensuring all necessary
    constraints are included. Your output MUST
    be generic, using parameters for all
    numerical inputs (costs, demands, capacities
    ).

### 1. Model Type & Objective:
Identify the precise mathematical model (e.g.,
    TSP, Transshipment Problem, Min-Cost Flow)
    and the objective (Minimize/Maximize).

### 2. Variables and Sets:
Define all necessary Sets (e.g., CITIES, I, J)
    and Variables (e.g., binary x[i,j],
    continuous flow[i,j], auxiliary u[i]).

### 3. Binary Linking Constraints Check:
For any problem involving discrete decisions (
    TSP, Fixed-Charge, etc.), you MUST provide
    the precise mathematical formulation for the
     **LINKING CONSTRAINTS** necessary to
    connect the binary variables to the
    continuous variables.
   * Example for Fixed-Charge: Define the
       constraint ensuring flow is zero if a
       facility is closed (using generic
       parameters like 's[j]' and 'y[j]').
   * Example for TSP: Define the Miller-Tucker-
       Zemlin (MTZ) Subtour Elimination
       Constraint, including all required index
        exclusions (e.g., i != 1 and j != 1)
       and using 'N' or 'card(CITIES)' for the
       scaling factor.

Problem Description: {q}

Output only the complete, mathematically precise
     list of sets, variables, objective, and
    constraints, strictly adhering to the
    instructions above, and using generic
    parameter names only. Do not include any
    specific numerical data.
"""
```

Listing 2: Prompt template for Model Formulator

```
You are a highly specialized AMPL Coder. Your
    sole function is to synthesize the provided
    components into a complete, executable AMPL
    model (.mod) file.

---
### 1. Model Blueprint (From Peer):
You must strictly adhere to the model type,
    parameters, and constraints listed here:
{constraint_output}

### 2. Synthesis Instructions:
1. Define all sets and parameters using generic
    AMPL syntax (e.g., 'param cost {{SETS}};').
2. You must not include any data assignment
    operators (:=) or data values in the .mod
    file. All data belongs in the .dat file.
3. Define all variables, Objective function, and
     Constraints, integrating the complex
    linking constraints provided in the
    blueprint.

Return only the AMPL model code for the .mod
    file. Do not include explanations or
    comments anywhere.
```

Listing 3: Prompt template for Data Extractor

```
You are the final, highly specialized AMPL Coder.
     Your sole function is to synthesize the
    provided components into a complete,
    executable AMPL model (.mod) and data (.dat)
     file.
Pay attention to the units required in the final
     output.

The optimization problem is the following:
{q}


The model file is already attempted by your peer:

{mod}

Your job is to create the model and data files
    for the optimization problem.

Return both the AMPL model and data code. Do not
     include explanations or comments anywhere.
```
```
1.mod:
```

```
[.mod file content here]

2.dat:
[.dat file content here]
```

Listing 4: Prompt template for Debugger

```
As a specialized Optimization Code Reviewer,
    your responsibility is to identify the
    source of the numerical error in the
    provided AMPL model.
Here is the optimization problem:
```
{q}
```

Here is the previous AMPL code:
```
{ampl}
```

Return in a few short lines why the AMPL code
    gave the wrong answer of {predicted}, when
    the correct answer should be {expected}.
Propose a fix briefly.
```

Listing 5: Prompt template for Code Reviewer

```
As a specialized Optimization Code Reviewer,
    your responsibility is to identify the
    source of the numerical error in the
    provided AMPL model.
Your analysis must determine if the discrepancy
    between the PREDICTED value ({predicted})
    and the EXPECTED value ({expected}) is due
    to a modeling error or a data error.
Now, here is the problem:
```
{q}
```

Here is the AMPL code for review:
```
{ampl}
```

Your response should follow this format:
1. Diagnosis of Error Source
[State clearly whether the error is in the Model
     Logic (.mod) or the Data (.dat). If the
    code is correct, state that the expected
    value is likely flawed due to source data.]

2. Code Review: Model Logic (.mod)
[Review structure, variable definitions, and
    constraints. Check for MTZ exclusions,
    capacity links, and correct Objective
    function.]

3. Proposed Fix
[If the error is in the model, provide the
    corrected AMPL code for the .mod file. If
    the error is in the data, specify which
    parameter(e.g., unit error, transcription
    error).]

If both the model and the data file is correct,
    and the error lies in the optimization
    problem, output 1 in the final line.
```

Listing 6: Prompt template for Code Modifier

```
As a specialized Optimization Code Reviewer,
    your responsibility is to improve the
    current AMPL model.
Here is the optimization problem:
```
{q}
```

Here is the previous AMPL code:
```
{ampl}
```

Here is the comment and proposed fix by your
    peer:
```
{comment}
```

Modify the AMPL code such that it models the
    question accurately, based on the comment
    and proposed fix by your peer.
Return only the new AMPL code. Do not include
    explanations or comments anywhere.
Use the exact format below:
```
1.mod:
[.mod file content here]

2.dat:
[.dat file content here]
```
```

II Templates for model–data separation

To systematically separate the natural language description from its associated data, we employ a pipeline consisting of a NL-Data Separator and a Verifier.

Model-Data Separator: Transforms a given optimization problem into two distinct components: a problem statement in natural language and a dataset in CSV format. The problem statement presents a structured task description and specifies the optimization objective without including any numerical values. All quantitative information is captured exclusively in the dataset, which contains only the values explicitly provided in the original problem.

Verifier: Ensures the integrity of the model–data separation by checking that all numerical values required to solve the optimization problem are present in the data file.

Listing 7: Prompt template for NL-Data Separator

```
The following is an optimization problem. Please
     construct:
```

```
1. A problem statement in natural language,
    beginning with "You are..." and following a
    structured task description. It should
    clearly explain what the objective is, but
    must NOT contain any specific numerical
    values. No concrete quantities, figures, or
    units of measurement should be mentioned.

2. A dataset composed of multiple CSV-style
    tables that define a specific parameters.
  - All variables referenced in the
      optimization problem must be represented
      in the tables.
  - All table cells should be numeric, do not
      use placeholders.
  - Only include values explicitly mentioned in
       the problem.

The optimization problem is as follows:
{ques}

Please construct a problem specification and
    tabular dataset according to the above
    requirements, and provide the results the
    following format:
```
1. Problem statement
  [Write the problem specification in plain
      English.]

2. Tables
table1_name.csv - Description
Column1, Column2, ...
value1, value2, ...
...

table2_name.csv - Description
Column1, Column2, ...
value1, value2, ...
...

```
```

Listing 8: Prompt template for Verifier

```
You are asked to compare an optimization problem
     and its corresponding tabular datasets.
The optimization problem is as follows:
{ques}

The tabular datasets corresponding are as
    follows:
{dat}

Please check if the tabular datasets contain all
     values needed to solve the optimization
    problem.
If this is satisfied, return 1. If not, return 0.
     Return 1 or 0 only.
```

III Templates for prediction case construction

To construct prediction cases for NLData2Opt, we transform NL+Data cases into tasks that require LLMs to predict future parameter values from historical data. To construct tasks that require different numbers of parameters to be predicted, we implement a pipeline that consists of the following components:

AMPL Formulator: Modifies the data of a task from NL+Data case to create prediction cases. It handles single or multiple targets, and supports increasing and stable trends. Historical data are generated for each target, with the final time period containing the true value. After passing the Verifier, this value is subsequently blanked out in the historical data table using automated scripts. Based on these patterns, seasonality, abrupt changes, intermittent missing data, and outliers are introduced to simulate richer temporal dynamics.

Verifier: Inspects the new optimization problem if all data in the NL+Data case are included. In addition, historical data are provided for a certain number of parameters, with the final time period containing the value to be predicted. Two separate verifiers are used for single variable and multi-variable prediction, respectively.

Description Writer: Generates a natural language problem description from the tabular datasets. In addition to clearly stating the overall optimization objective, it explicitly specifies which variables must be predicted at which exact time period.

Listing 9: Prompt template for AMPL Formulator (increasing case, single variable)

```
The following are a problem description and its
    data tables:
```
{base}
```

Modify the tabular datasets following the below
    requirements:
- Choose one value in the tabular datasets of
    the optimization problem as the prediction
    target.
- Provide historical data of at least 8 time
    periods that show a realistic trend with
    small random noise.
- The historical data show an increasing trend
    that approaches the true target value
    naturally, without overshooting it.
- The historical trend must stop immediately at
    the true target value.
- The time period should have its own column,
    with realistic names and values.
- Do not remove or rename any existing columns
    or rows in the original tables. However, the
     single true target value must be visibly
    blanked out exactly where it originally
    appeared.

Provide the results the following format, do not
     add comments:
```

```
```
1. Problem statement
  [The original problem statement]

2. Tables
table1_name.csv - Description
Column1, Column2, ...
value1, value2, ...
...

table2_name.csv - Description
Column1, Column2, ...
value1, value2, ...
...

```
```

Listing 10: Prompt template for AMPL Formulator (stable case, multiple variables)

```
The following are a problem description and its
     data tables:
```
{base}
```

Modify the tabular datasets following the below
     requirements:
- The given problem description and tables
     already contain one value that has been
     blanked out as an existing prediction target
     . Keep that historical data table unchanged.
- Select one additional value in the tabular
     datasets to serve as a second new prediction
      target. Blank it out exactly where it
     originally appeared, without removing or
     renaming any columns or rows.
- For the new prediction target, provide at
     least 8 time periods of historical data in a
      separate table.
- The historical series for the new target must
     show natural small random fluctuations
     around the prediction target while
     maintaining integer constraints when
     necessary.
- The historical data should hover both slightly
      above and below the new prediction target,
     centering roughly on it.
- The final row for all historical data should
     have the target value exactly, do not leave
     it blank.
- The time period should have its own column,
     with realistic names and values.
- Do not remove or rename any existing data
     aside from blanking out the two true target
     values.

The final sentence in the problem description
     should specify which two parameters need to
     be predicted at which time period. Phrase it
      as "you need to predict...".

Provide the results the following format, do not
      add comments:
```
1. Problem statement
  [Rewrite the original problem statement]

2. Tables
table1_name.csv - Description
Column1, Column2, ...
value1, value2, ...
...

table2_name.csv - Description
Column1, Column2, ...
value1, value2, ...
...

```
```

Listing 11: Prompt template for Verifier (single variable)

```
You are asked to verify if the given tabular
     datasets satisfy certain rules.

Optimization problem:
{ques}

Tabular datasets:
```
{dat}
```

Check if the data tables satisfy the following
     rules:
1. Historical data is only provided for 1
     variable in the data tables.
2. All variables mentioned in the optimization
     problem appear in the main data tables
     except for one missing value.
3. A historical table provides the correct value
      for one variable, with the value in the
     final time period appearing in the question.

Output "1" if all rules are satisfied, otherwise
      output "0". Do not provide any extra text.
```

Listing 12: Prompt template for Verifier (multiple variable)

```
You are asked to verify if the given tabular
     datasets satisfy certain rules.

Optimization problem:
{ques}

Tabular datasets:
```
{dat}
```

Check if the data tables satisfy the following
     rules:
1. Historical data ending at the same numerical
     value as the optimization problem are
     provided for at least 2 variables. Ignore
     textual differences when comparing numerical
      values.
2. All variables mentioned in the optimization
     problem appear in the main data tables
     except for those in the historical tables.

Output "1" if all rules are satisfied, otherwise
      output "0". Do not provide any extra text.
```

Listing 13: Prompt template for Description Writer

```
The following is an optimization problem, with
    its corresponding tabular datasets. You are
    asked to write a problem description.

Optimization problem:
{ques}

Tabular datasets:
{dat}

Please write a problem description in natural
    language, beginning with "You are..." and
    following a structured task description.
Do not include any specific numerical values. No
     concrete quantities, figures, or units of
    measurement should be mentioned.
It should clearly explain what the overall
    objective of the optimization problem is.
Specify which parameters needs to be predicted
    at which time period. Phrase it as "you need
     to predict...". Explicitly mention the **
    exact time** to be predicted as it appears
    of the provided tables. Do not paraphrase it
     as "the final row" or "the next time".

Provide only the problem statement, do not add
    comments.
```

IV Templates for derivation case construction

To generate derivation cases for NLData2Opt, we transform straightforward optimization problems into new problems that require intermediate calculations. Because directly transforming existing optimization problems to require intermediate calculations in a single step proved challenging for LLMs, we adopt a two-stage transformation strategy. The components of the pipeline are as follows:

Context Generator: Produces a rewritten optimization problem where exactly one numerical quantity is expressed via meaningful intermediate quantities, without changing the total value or feasible region. Example transformations are provided to encourage the LLM to generate valid decompositions. Example transformations are provided to guide the LLM toward valid decompositions. Two context generators are used sequentially, with context generator 2 employed after context generator 1 to process questions that were not successfully transformed according to the Verifier.

Verifier: Inspects the new optimization problem if it maintains the same feasible region and optimal solution as the problem before the transformation. In addition, it makes sure the transformation successfully introduced intermediate quantities to require derivation-based manipulation by making sure one numerical quantity is decomposed into possible components.

NL-Data Separator: Separates the model from its associated data for the new optimization problem. Compared to the standard NL+Data Separator used for the NL+Data case, this version specifies to only include the intermediate quantities when both aggregated parameters and intermediate quantities are provided, ensuring that derivation-based reasoning is required while modeling the optimization problem.

Listing 14: Prompt template for Context Generator 1

```
Rewrite the given optimization problem so that
    exactly one numerical variable is optionally
     expressed in terms of meaningful
    intermediate components, without changing
    the total value or the feasible region of
    the problem. All other parts of the problem
    must remain unchanged. The decomposition
    should reflect derivation-oriented reasoning
    , such as computing profit from revenue and
    costs, or breaking down total production
    time into separate tasks. Return only the
    rewritten problem.
---
The optimization problem is as follows:
A school needs to provide lunch for at least 300
     students.

Rewrite:
A school needs to provide lunch for at least 150
     boys and 150 girls.
---
The optimization problem is as follows:
It makes a profit of $3 per regular sandwich and
     a profit of $4 per special sandwich.

Rewrite:
A regular sandwich costs $1 to make and has a
    selling price of $4. A special sandwich
    costs $3 to make and has a selling price of
    $7.
---
The optimization problem is as follows:
{ques}

Rewrite:
```

Listing 15: Prompt template for Context Generator 2

```
You are an Optimization Problem Decomposer. Your
     task is to take a straightforward
    optimization problem and rewrite it into a
    new, more detailed narrative.

You must follow these rules:

1. You *must* take the simple numbers (constants
    ) in the problem (e.g., 10, 6, 150) and
    break them down arithmetically into
    nontrivial new components.
* FAILURE: A trivial split like '100' -> '50+50'
     is a **complete failure**.
* GOOD EXAMPLE: A good split adds new concepts:
```

```
    '4' -> 'a 1-unit prep fee and 3 units of
    application time'.

2. You must rewrite *only* the parts of the
    sentence that contain the numbers you are
    decomposing. **Do not** add extra,
    unnecessary sentences or rephrase parts of
    the problem that are not being changed.

3. Simply rephrasing the problem in new words or
     translating it into its algebraic form (
    like "minimize 10x + 6y") is a **complete
    failure**. The rewrite *must* contain the
    arithmetic decomposition from Rule 1.

4. The final, rewritten problem must be *
    mathematically identical* to the original.
    The objective function and all constraints
    must simplify to the *exact same equations*.

5. The rewritten problem should preserve the
    narrative style, sentence structure, and
    context around the numbers. It should read
    like a natural problem statement, not a
    formula or explanation.

---
**Example 1: Objective Function**
The optimization problem is as follows:
A company earns a profit of $5 per product.

Rewrite:
Each product sells for $15 and costs $10 to
    produce.

---
**Example 2: Simple Constraint**
The optimization problem is as follows:
A factory produces 800 units per day.

Rewrite:
The factory operates for 10 hours a day. It
    spends 2 hours setting up machines and then
    produces 100 units per hour for the
    remaining 8 hours.

---
**Example 3: Constraint with Multipliers**
The optimization problem is as follows:
Four times the man-hours spent on task X plus
    those spent on task Y should not exceed 16

Rewrite:
Each unit of task X contributes 4 man-hours,
    consisting of 2 hours of planning and 2
    hours of execution; each unit of task Y
    contributes 1 man-hour of work. Then,
    summing these contributions, the total man-
    hours 4X + Y cannot exceed 16.

---
**TASK**
The optimization problem is as follows:
{ques}

Rewrite:
```

Listing 16: Prompt template for Verifier

```
Compare the two optimization problems below and
    determine if the rewritten one correctly
    follows the transformation rules:

1. The rewritten problem must have the same
    feasible region and optimal solution as the
    original problem after the transformation.
2. At least one numerical quantity from the
    original problem is split into meaningful
    components.

Original problem:
{ques}

Rewritten problem:
{new}

Return 1 if both rules are satisfied, else 0.
    Only return 1 or 0.
If 0, in the next paragraph, change the
    rewritten problem to fulfill the rules.
```

Listing 17: Prompt template for NL-Data Separator

```
The following is an optimization problem. Please
     construct:
1. A problem statement in natural language
  - Begin with "You are..." and following a
      structured task description.
  - Include only those constraints that are not
       already evident or numerically encoded
      in the dataset tables.
  - Do not include any numbers, units, or
      placeholders anywhere in the text.

2. A dataset composed of multiple CSV-style
    tables that define a specific parameters.
  - All variables referenced in the
      optimization problem must be represented
      in the tables.
  - All table cells should be numeric, do not
      use placeholders.
  - Only include values explicitly mentioned in
       the problem.
  - When both aggregated parameters and
      intemediate parameters are provided, only
       include the intermediate parameters.

The optimization problem is as follows:
{ques}

Please construct a problem specification and
    tabular dataset according to the above
    requirements, and provide the results the
    following format:
```
1. Problem statement
  [Write the problem specification in plain
      English.]

2. Tables
table1_name.csv - Description
Column1, Column2, ...
value1, value2, ...
...

table2_name.csv - Description
Column1, Column2, ...
value1, value2, ...
```
```

```
...

```
```

V Templates for LLM evaluation

In this section, instructions templates to assess the accuracy of AMPL formulation are introduced. Given that LLM outputs may contain conversational headers or omit the target `.mod`/`.dat` markers, we utilize the template detailed in Listing 18 to extract the AMPL formulation and ensure the files are in a compatible format with the CBC solver.

To perform a holistic evaluation, in addition to the EF1 and EBC, we employ LLM-based evaluation with the template provided in Listing 19. The rating of 0-10 is then normalized to a score between 0 and 10, which is consistent with the other two metrics.

Listing 18: Prompt template for reformatting generated messy formulation

```
You are given an output. Extract the AMPL
    formulation if possible. For the data file,
    leave #empty if empty, but do not omit the
    "2.dat" marker.

Output:
{AMPL_Formulation}

Return strictly in this format:
```1.mod:
<AMPL model>

2.dat:
<AMPL data>```

If the text does not contain tangible AMPL
    formulation, return:
``` 1.mod:
#empty

2.dat:
#empty ```
```

Listing 19: Prompt template for evaluating overall correctness

```
Does this .mod and .dat pair represent the same
    optimization problem as the ground-truth .
    mod and .dat? Rate this solution from 0 to
    10:
.mod file:
{mod1_content}
ground truth .mod file:
{mod2_content}

.dat file:
{dat1_content}
ground truth .dat file:
{dat2_content}

Specifically, evaluate:
- Whether the constraints in the candidate .mod
    file are equivalent to those in the ground-
    truth .mod file,
- Whether the parameter values in the candidate .
    dat file exactly match those in the ground-
    truth .dat file.
Return an explanation. The final line should be
    the rating.
```

VI Controlled Prediction Baselines

We implement two additional controlled baselines to separate forecasting difficulty from formulation difficulty, focusing on three representative models: claude-opus-4.6 (Table 1), gpt-4o-mini (Table 2), and llama-3.3-70b-instruct (Table 3). In the simple forecasting baseline, the LLM-generated predictions are replaced with estimates from a linear trend extrapolation over historical series data, while the remaining formulation process remains unchanged. In the oracle-prediction baseline, the predicted values are replaced with the ground-truth future values, thereby removing forecasting error and allowing us to examine errors arising from incorporating these values into the final optimization formulation.

We observe that replacing the LLM-generated predictions with the simple forecasting baseline generally reduces EBC. However, using oracle-prediction values yields only small improvements over the original LLM baseline. This suggests that the forecasting capability of modern LLMs is already sufficient for many tasks and that forecasting errors alone cannot explain the observed performance gap.

The oracle-prediction baseline generally improves EBC and LLM-ES only slightly. For example, EBC of Llama 3.3 70B Instruct on the One-Variable Prediction case increases from 0.1824 to just 0.1847 under oracle prediction. This indicates that formulation errors remain persistent even when forecasting difficulty is isolated. In particular, lower-capacity models continue to struggle with incorporating predicted values into the optimization instance and constructing the corresponding formulation correctly.

VII Further Analysis

Table 4 summarizes the percentage of tasks where the optimization model is correctly identified for the derivation case. Here, a correct model is when the constraint relation types ($\leq$, $\geq$, $=$) and objective function type (maximize or minimize) exactly
```

Case	Simple Forecasting			Oracle Prediction			LLM		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
One-Variable	0.8665	0.6144	0.5377	0.8654	0.7305	0.5858	0.8836	0.7357	0.5572
One-Variable + Irr	0.8551	0.4796	0.4655	0.8436	0.6538	0.5018	0.8441	0.5331	0.4602
Multi-Variable	0.8739	0.5810	0.5552	0.8643	0.6257	0.5886	0.8626	0.6225	0.5300
Multi-Variable + Irr	0.8424	0.4972	0.4482	0.8325	0.5898	0.4769	0.8683	0.6393	0.4671

Table 1: Performance breakdown for Claude Opus 4.6 across prediction baselines (Simple Forecasting, Oracle Prediction, and LLM).

Case	Simple Forecasting			Oracle Prediction			LLM		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
One-Variable	0.8968	0.2213	0.5390	0.8548	0.2224	0.6043	0.3788	0.2386	0.5368
One-Variable + Irr	0.9208	0.1671	0.4878	0.8765	0.1945	0.4716	0.4259	0.2555	0.4443
Multi-Variable	0.9535	0.2229	0.5239	0.8582	0.2798	0.5830	0.3837	0.2680	0.5522
Multi-Variable + Irr	0.8714	0.1512	0.4692	0.8500	0.1533	0.4833	0.3419	0.2071	0.4434

Table 2: Performance breakdown for GPT-4o Mini across prediction baselines (Simple Forecasting, Oracle Prediction, and LLM).

match the ground truth formulation, regardless of the values of the parameters.

Table 5 analyzes the performance of the prediction step prior to optimization. It reports percentage of tasks where future prediction exactly matches the ground truth value (EC) and relative correctness (RC). We find that most methods obtain over 30% on the EC metric and over 60% on the RC metric.

To further investigate the failure modes of the generated optimization models, we categorize AMPL execution errors encountered across all models and datasets. Figure 6 visualizes the proportional distribution of the error types across the nine distinct task cases, while Table 6 reorganizes failures into tool-specific syntax errors and structural errors.

Across task cases, maintaining structural consistency between the model and the data is demonstrated to be a significant challenge for LLMs. We identify three primary types of errors that are prevalent across tasks:

- **Undefined Entities:** A structural error where the model (`.mod`) attempts to use a set, parameter, or variable in a constraint or objective function that has not been formally declared or initialized.
- **Undeclared Data Assignments:** A structural error where the data file (`.dat`) attempts to assign values for parameters or sets that do not exist in the corresponding `.mod` file.
- **Syntax Errors:** Fundamental formatting violations, such as missing semicolons or incorrect assignment operators, prevent the AMPL compiler from parsing the scripts. This includes dimension mismatches such as improper set index, reflecting a failure of mathematical indexing as well as grammar.

Beyond compile-time failures, semantic modeling errors cannot be caught during compilation but manifest as discrepancies in the final objective value. As shown in Table 6, the exact objective match rate is consistently lower than the compilation success rate, which indicates that many generated formulations are executable but still contain semantic modeling errors that lead to incorrect solutions.

VIII Comprehensive Evaluation Results

The experimental results reported in Tables 7-15 are based on a randomly sampled 15% subset of the full benchmark, selected using a random seed of 42. The results are categorized across all cases, datasets, and methods to provide a comprehensive overview of baseline performance. We report the three evaluation metrics: entity F1 score (EF1), execution-based correctness (EBC) and LLM-based evaluation score (LLM-ES). To assess the variance and stability of LLM-ES, we measure self-consistency by evaluating each candidate formulation with prompt variations. Specifically, we apply character-level errors in high-attention words, insert sentence-level line breaks, and paraphrase certain verbs and adjectives (Choi et al., 2026). We report the mean score and standard deviation across four evaluation runs with different prompt varia-

Case	Simple Forecasting			Oracle Prediction			LLM		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
One-Variable	0.8756	0.0962	0.4874	0.8007	0.1847	0.5626	0.3486	0.1824	0.4932
One-Variable + Irr	0.8004	0.0733	0.4308	0.8046	0.0999	0.4493	0.1557	0.1180	0.4176
Multi-Variable	0.8087	0.1387	0.5107	0.8681	0.2397	0.5739	0.4154	0.2825	0.5045
Multi-Variable + Irr	0.6693	0.1066	0.4305	0.7159	0.1425	0.4543	0.2925	0.2008	0.3960

Table 3: Performance breakdown for Meta LLaMA 3.3 70B Instruct across prediction baselines (Simple Forecasting, Oracle Prediction, and LLM).

Model/Method	**NLP4LP**	**Mamo E.**	**Mamo C.**
GPT-4o-mini	0.000	0.000	0.000
GPT-4o	0.425	0.367	0.467
Gemini-2.5-Pro	0.000	0.067	0.000
Claude Haiku 4.5	0.600	0.587	0.593
Claude Sonnet 4.6	0.573	0.533	0.307
Claude Opus 4.6	0.567	0.547	0.440
Llama-3.1-8B-Instruct	0.250	0.165	0.129
Ministral-3-8B-2512	0.125	0.052	0.097
Llama-3.3-70B-Instruct	0.250	0.300	0.129
Mistral-Large-2411	0.300	0.067	0.194
Five-element formulation	0.525	0.794	0.419
CoE	0.450	0.309	0.129
SFT	0.375	0.845	0.355

Table 4: Percentage of correctly identified optimization models on derivation case.

tions.

To assess whether the subset is representative of the full benchmark, we evaluated claude-opus-4.6 (Table 16), gpt-4o-mini (Table 17), and llama-3.3-70b-instruct (Table 18) on the complete benchmark. The full-benchmark results exhibit trends consistent with those reported on the 15% subset. In particular, the relative ranking of models is preserved, with Claude Opus 4.6 outperforming GPT-4o Mini and Llama 3.3 70B Instruct. Similarly, the comparative difficulty of different task settings are preserved: task cases involving derivation, prediction, and irrelevant attributes remain more challenging than the base NL and NL+Data cases. These observations suggest that the subset-based evaluation is representative of overall model performance.

Model/Method	One-variable				Multiple-variables			
	Standard		Irr. attributes		Standard		Irr. attributes	
	EM	SC	EM	SC	EM	SC	EM	SC
GPT-4o Mini	0.433	0.779	0.322	0.711	0.371	0.526	0.287	0.506
GPT-4o	0.611	0.892	0.400	0.845	0.408	0.560	0.258	0.511
Gemini 2.5 Pro	0.078	0.219	0.022	0.125	0.102	0.146	0.069	0.107
Claude Haiku 4.5	0.556	0.972	0.478	0.961	**0.603**	0.744	0.521	0.718
Claude Sonnet 4.6	**0.667**	**0.980**	**0.567**	0.942	0.590	**0.745**	**0.600**	**0.745**
Claude Opus 4.6	0.533	0.959	0.511	**0.958**	0.571	0.739	0.563	0.738
LLaMA 3.1 8B Inst.	0.344	0.615	0.400	0.620	0.330	0.458	0.217	0.386
Ministral 8B	0.278	0.441	0.189	0.411	0.260	0.388	0.150	0.287
LLaMA 3.3 70B Inst.	0.422	0.846	0.333	0.820	0.335	0.522	0.196	0.452
Mistral Large	0.522	0.894	0.411	0.813	0.409	0.571	0.280	0.501
Five-element	0.389	0.661	0.411	0.752	0.327	0.503	0.303	0.491
CoE	0.367	0.638	0.356	0.700	0.413	0.508	0.284	0.435
SFT	0.544	0.680	0.533	0.749	0.469	0.547	0.411	0.497

Table 5: Percentage of Exact Matches (EM) and continuous Smoothed Correctness (SC) across four prediction cases.

Case	Compilation Success	Exact Match Rate	Syntax Error	Structural Error
NL	51%	23%	11%	43%
NL + Data	45%	22%	24%	46%
NL + Data + Irr	48%	25%	16%	46%
Derivation	42%	8%	20%	50%
Derivation + Irr	43%	10%	12%	49%
Prediction (1-Var)	37%	7%	17%	54%
Prediction (1-Var) + Irr	35%	12%	8%	63%
Prediction (Multi-Var)	41%	8%	16%	52%
Prediction (Multi-Var) + Irr	38%	11%	13%	58%

Table 6: Failure type breakdown across all task cases.

Figure 6: **Distribution of AMPL execution errors across nine task cases.** The breakdown highlights the relative frequency of specific compilation and logical errors, such as undefined entities, undeclared data assignments, and syntax errors.

Model/Method	NLP4LP			Mamo Easy			Mamo Complex		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
GPT-4o Mini	0.582	**0.818**	0.568 ± 0.056	0.729	0.867	0.481 ± 0.055	0.809	0.130	0.548 ± 0.061
GPT-4o	**0.968**	0.150	0.667 ± 0.047	0.687	0.767	0.383 ± 0.050	0.870	0.468	0.670 ± 0.034
Gemini 2.5 Pro	0.413	0.000	0.282 ± 0.056	0.370	0.000	0.203 ± 0.059	0.301	0.000	0.390 ± 0.047
Claude Haiku 4.5	0.588	0.702	0.463 ± 0.051	0.825	0.845	$\mathbf{0.638 \pm 0.060}$	0.865	0.619	0.625 ± 0.043
Claude Sonnet 4.6	0.880	0.450	0.403 ± 0.039	0.850	0.600	0.610 ± 0.046	**0.905**	0.616	0.596 ± 0.042
Claude Opus 4.6	0.898	0.650	0.413 ± 0.037	**0.890**	0.750	0.610 ± 0.039	0.872	**0.741**	0.611 ± 0.042
LLaMA 3.1 8B Inst.	0.776	0.000	0.538 ± 0.051	0.602	0.000	0.324 ± 0.052	0.690	0.000	0.542 ± 0.049
Ministral 8B	0.898	0.000	0.626 ± 0.051	0.645	0.000	0.356 ± 0.053	0.824	0.000	0.601 ± 0.029
LLaMA 3.3 70B Inst.	0.796	0.075	0.493 ± 0.053	0.651	**0.906**	0.367 ± 0.056	0.794	0.308	0.609 ± 0.039
Mistral Large	0.922	0.075	0.588 ± 0.061	0.654	0.100	0.347 ± 0.041	0.867	0.417	0.650 ± 0.041
Five-element	0.872	0.193	0.738 ± 0.013	0.691	0.625	0.132 ± 0.018	0.827	0.349	$\mathbf{0.700 \pm 0.024}$
CoE	0.581	0.602	$\mathbf{0.747 \pm 0.035}$	0.672	0.733	0.132 ± 0.007	0.577	0.091	0.249 ± 0.035
SFT	0.902	0.507	0.732 ± 0.022	0.726	0.748	0.183 ± 0.014	0.817	0.052	0.367 ± 0.022

Table 7: Comprehensive metrics for performance on NL case.

Model/Method	NLP4LP			Mamo Easy			Mamo Complex		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
GPT-4o Mini	0.931	0.400	0.421 ± 0.049	0.700	0.345	0.419 ± 0.049	0.876	0.393	0.529 ± 0.049
GPT-4o	0.940	0.150	0.645 ± 0.049	0.670	0.639	0.361 ± 0.056	0.887	0.612	$\mathbf{0.658 \pm 0.049}$
Gemini 2.5 Pro	0.427	0.000	0.317 ± 0.046	0.239	0.000	0.147 ± 0.044	0.327	0.000	0.396 ± 0.061
Claude Haiku 4.5	0.896	**0.733**	0.644 ± 0.045	0.734	0.753	$\mathbf{0.543 \pm 0.059}$	0.872	0.630	0.625 ± 0.042
Claude Sonnet 4.6	**0.945**	0.290	0.515 ± 0.050	0.670	0.553	0.495 ± 0.062	0.879	0.794	0.596 ± 0.042
Claude Opus 4.6	0.938	0.699	0.598 ± 0.051	0.679	**0.762**	0.513 ± 0.061	0.885	**0.853**	0.611 ± 0.042
LLaMA 3.1 8B Inst.	0.778	0.000	0.530 ± 0.056	0.600	0.000	0.294 ± 0.049	0.650	0.000	0.522 ± 0.047
Ministral 8B	0.885	0.000	0.596 ± 0.047	0.664	0.000	0.382 ± 0.063	0.805	0.000	0.615 ± 0.044
LLaMA 3.3 70B Inst.	0.846	0.025	0.593 ± 0.053	0.657	0.499	0.385 ± 0.059	0.786	0.191	0.620 ± 0.043
Mistral Large	0.916	0.100	0.657 ± 0.050	0.658	0.239	0.357 ± 0.055	0.837	0.253	0.628 ± 0.038
Five-element	0.934	0.400	$\mathbf{0.839 \pm 0.026}$	0.684	0.600	0.165 ± 0.005	**0.922**	0.534	0.583 ± 0.024
CoE	0.724	0.516	0.698 ± 0.005	0.634	0.381	0.122 ± 0.011	0.745	0.092	0.388 ± 0.030
SFT	0.905	0.483	0.699 ± 0.040	**0.702**	0.603	0.168 ± 0.010	0.830	0.129	0.458 ± 0.012

Table 8: Comprehensive metrics for performance on NL+Data case.

Model/Method	NLP4LP			Mamo Easy			Mamo Complex		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
GPT-4o Mini	0.737	0.266	0.421 ± 0.049	0.600	0.490	0.419 ± 0.049	0.811	0.336	0.529 ± 0.049
GPT-4o	0.723	0.627	0.451 ± 0.047	0.627	0.513	0.403 ± 0.053	0.801	0.385	0.558 ± 0.066
Gemini 2.5 Pro	0.879	0.273	0.347 ± 0.055	0.635	0.219	0.299 ± 0.059	0.800	0.032	0.411 ± 0.055
Claude Haiku 4.5	0.824	0.613	0.438 ± 0.047	0.667	**0.697**	0.461 ± 0.056	0.875	0.551	0.554 ± 0.043
Claude Sonnet 4.6	0.861	0.223	0.355 ± 0.074	0.701	0.284	0.503 ± 0.049	0.876	0.678	0.539 ± 0.055
Claude Opus 4.6	**0.912**	**0.655**	0.412 ± 0.050	0.683	0.586	0.480 ± 0.059	0.872	**0.787**	0.551 ± 0.045
LLaMA 3.1 8B Inst.	0.720	0.000	0.371 ± 0.039	0.537	0.000	0.363 ± 0.046	0.593	0.032	0.418 ± 0.049
Ministral 8B	0.801	0.000	0.417 ± 0.060	0.563	0.010	0.398 ± 0.061	0.736	0.000	0.483 ± 0.050
LLaMA 3.3 70B Inst.	0.606	0.187	0.398 ± 0.045	0.528	0.305	0.420 ± 0.048	0.852	0.101	0.473 ± 0.052
Mistral Large	0.539	0.132	0.453 ± 0.056	0.540	0.120	0.416 ± 0.048	0.826	0.284	0.503 ± 0.052
Five-element	0.852	0.633	$\mathbf{0.710 \pm 0.014}$	**0.742**	0.440	$\mathbf{0.543 \pm 0.007}$	0.821	0.415	$\mathbf{0.629 \pm 0.010}$
CoE	0.569	0.033	0.031 ± 0.016	0.507	0.014	0.254 ± 0.022	**0.935**	0.006	0.245 ± 0.029
SFT	0.812	0.495	0.666 ± 0.017	0.542	0.580	0.431 ± 0.009	0.821	0.180	0.526 ± 0.006

Table 9: Comprehensive metrics for performance on NL+Data case with irrelevant attributes.

Model/Method	NLP4LP			Mamo Easy			Mamo Complex		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
GPT-4o Mini	0.901	0.259	0.476 ± 0.053	0.637	0.151	0.289 ± 0.035	0.730	0.132	0.433 ± 0.040
GPT-4o	**0.913**	0.125	0.553 ± 0.062	0.609	0.328	0.310 ± 0.050	0.792	0.416	0.502 ± 0.052
Gemini 2.5 Pro	0.378	0.000	0.318 ± 0.043	0.222	0.000	0.131 ± 0.045	0.282	0.000	0.372 ± 0.049
Claude Haiku 4.5	0.782	**0.542**	0.486 ± 0.041	**0.696**	**0.560**	0.459 ± 0.040	0.818	0.438	0.511 ± 0.037
Claude Sonnet 4.6	0.864	0.289	0.416 ± 0.034	0.646	0.215	0.408 ± 0.038	0.845	0.542	0.543 ± 0.042
Claude Opus 4.6	0.866	0.500	0.441 ± 0.043	0.637	0.514	0.410 ± 0.031	0.861	**0.609**	$\mathbf{0.556 \pm 0.039}$
LLaMA 3.1 8B Inst.	0.752	0.000	0.476 ± 0.053	0.520	0.000	0.289 ± 0.035	0.608	0.000	0.433 ± 0.040
Ministral 8B	0.862	0.000	0.548 ± 0.058	0.580	0.000	0.300 ± 0.046	0.694	0.000	0.465 ± 0.046
LLaMA 3.3 70B Inst.	0.787	0.000	0.506 ± 0.057	0.557	0.345	$\mathbf{0.318 \pm 0.044}$	0.700	0.148	0.526 ± 0.054
Mistral Large	0.898	0.050	0.575 ± 0.042	0.627	0.092	0.309 ± 0.059	0.715	0.192	0.514 ± 0.051
Five-element	0.901	0.314	$\mathbf{0.793 \pm 0.012}$	0.574	0.133	0.072 ± 0.028	**0.897**	0.194	0.342 ± 0.011
CoE	0.639	0.186	0.499 ± 0.033	0.491	0.034	0.086 ± 0.010	0.632	0.032	0.271 ± 0.024
SFT	0.894	0.364	0.612 ± 0.009	0.675	0.216	0.136 ± 0.036	0.815	0.342	0.206 ± 0.019

Table 10: Comprehensive metrics for performance on derivation case.

Model/Method	NLP4LP			Mamo Easy			Mamo Complex		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
GPT-4o Mini	0.672	0.194	0.389 ± 0.041	0.499	0.190	0.353 ± 0.040	0.654	0.048	0.453 ± 0.037
GPT-4o	0.685	0.444	0.396 ± 0.052	0.537	0.283	0.388 ± 0.034	0.695	0.275	0.447 ± 0.042
Gemini 2.5 Pro	**0.914**	0.128	0.290 ± 0.065	**0.708**	0.025	0.209 ± 0.063	0.800	0.033	0.373 ± 0.052
Claude Haiku 4.5	0.766	**0.510**	0.398 ± 0.042	0.653	**0.478**	0.394 ± 0.051	0.827	0.357	0.491 ± 0.033
Claude Sonnet 4.6	0.795	0.225	0.341 ± 0.038	0.561	0.276	0.381 ± 0.047	**0.851**	0.426	$\mathbf{0.493 \pm 0.049}$
Claude Opus 4.6	0.804	0.315	0.355 ± 0.039	0.643	0.417	$\mathbf{0.408 \pm 0.049}$	0.844	**0.540**	0.484 ± 0.038
LLaMA 3.1 8B Inst.	0.610	0.000	0.345 ± 0.043	0.483	0.000	0.328 ± 0.028	0.442	0.000	0.379 ± 0.038
Ministral 8B	0.717	0.000	0.368 ± 0.044	0.489	0.000	0.340 ± 0.044	0.600	0.000	0.423 ± 0.041
LLaMA 3.3 70B Inst.	0.696	0.096	0.392 ± 0.040	0.421	0.160	0.359 ± 0.039	0.574	0.020	0.428 ± 0.045
Mistral Large	0.668	0.150	0.398 ± 0.042	0.480	0.034	0.382 ± 0.039	0.714	0.079	0.439 ± 0.040
Five-element	0.652	0.320	$\mathbf{0.493 \pm 0.016}$	0.512	0.110	0.366 ± 0.012	0.648	0.145	0.390 ± 0.017
CoE	0.600	0.024	0.085 ± 0.008	0.697	0.342	0.218 ± 0.016	0.673	0.012	0.191 ± 0.019
SFT	0.542	0.280	0.393 ± 0.015	0.412	0.195	0.295 ± 0.012	0.621	0.285	0.327 ± 0.016

Table 11: Comprehensive metrics for performance on derivation case with irrelevant attributes.

Model/Method	NLP4LP			Mamo Easy			Mamo Complex		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
GPT-4o Mini	0.326	0.240	0.531 ± 0.058	0.399	0.385	0.471 ± 0.044	0.442	0.204	0.559 ± 0.048
GPT-4o	0.676	0.554	$\mathbf{0.575 \pm 0.049}$	0.546	0.548	0.505 ± 0.066	0.710	0.404	0.548 ± 0.041
Gemini 2.5 Pro	0.405	0.322	0.472 ± 0.043	0.297	0.293	0.385 ± 0.057	0.113	0.038	0.424 ± 0.046
Claude Haiku 4.5	0.891	0.355	0.528 ± 0.057	0.709	0.473	0.468 ± 0.044	**0.902**	0.629	0.574 ± 0.056
Claude Sonnet 4.6	0.769	0.164	0.439 ± 0.039	0.712	0.557	0.505 ± 0.054	0.844	0.602	0.531 ± 0.043
Claude Opus 4.6	**0.909**	**0.720**	0.557 ± 0.050	0.683	**0.721**	0.448 ± 0.065	0.896	**0.759**	$\mathbf{0.582 \pm 0.049}$
LLaMA 3.1 8B Inst.	0.719	0.000	0.456 ± 0.044	0.630	0.000	0.426 ± 0.045	0.586	0.000	0.435 ± 0.042
Ministral 8B	0.598	0.000	0.507 ± 0.060	0.569	0.000	0.530 ± 0.043	0.693	0.000	0.484 ± 0.048
LLaMA 3.3 70B Inst.	0.361	0.216	0.466 ± 0.039	0.334	0.367	0.502 ± 0.056	0.336	0.098	0.526 ± 0.048
Mistral Large	0.387	0.240	$\mathbf{0.576 \pm 0.057}$	0.113	0.137	0.521 ± 0.062	0.371	0.205	0.542 ± 0.052
Five-element	0.904	0.100	0.471 ± 0.016	0.683	0.100	0.514 ± 0.012	0.811	0.300	0.372 ± 0.002
CoE	0.832	0.133	0.560 ± 0.002	0.705	0.142	0.575 ± 0.018	0.771	0.031	0.423 ± 0.010
SFT	0.884	0.351	0.540 ± 0.013	**0.764**	0.676	$\mathbf{0.661 \pm 0.014}$	0.801	0.285	0.216 ± 0.009

Table 12: Comprehensive metrics for performance on prediction case (one variable).

Model/Method	NLP4LP			Mamo Easy			Mamo Complex		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
GPT-4o Mini	0.423	0.247	0.418 ± 0.049	0.330	0.298	0.426 ± 0.043	0.451	0.257	0.482 ± 0.044
GPT-4o	0.670	**0.480**	0.435 ± 0.036	0.527	0.585	0.446 ± 0.057	0.754	0.410	0.513 ± 0.052
Gemini 2.5 Pro	0.330	0.065	0.312 ± 0.068	0.242	0.268	0.315 ± 0.068	0.024	0.035	0.411 ± 0.045
Claude Haiku 4.5	0.800	0.401	0.407 ± 0.047	0.684	0.422	0.416 ± 0.044	0.851	0.504	$\mathbf{0.529 \pm 0.054}$
Claude Sonnet 4.6	0.751	0.317	0.364 ± 0.037	0.716	0.493	0.468 ± 0.056	0.854	**0.694**	0.498 ± 0.055
Claude Opus 4.6	0.853	0.391	0.415 ± 0.041	0.669	**0.690**	0.452 ± 0.057	**0.872**	0.680	0.520 ± 0.060
LLaMA 3.1 8B Inst.	0.571	0.000	0.355 ± 0.034	0.465	0.000	0.389 ± 0.041	0.501	0.000	0.374 ± 0.049
Ministral 8B	0.692	0.000	0.388 ± 0.041	0.564	0.011	0.417 ± 0.057	0.564	0.000	0.436 ± 0.052
LLaMA 3.3 70B Inst.	0.139	0.102	0.385 ± 0.039	0.237	0.317	0.442 ± 0.051	0.159	0.094	0.454 ± 0.052
Mistral Large	0.290	0.121	0.443 ± 0.047	0.075	0.057	0.438 ± 0.054	0.200	0.058	0.470 ± 0.052
Five-element	0.864	0.267	0.453 ± 0.017	0.682	0.100	0.421 ± 0.011	0.729	0.200	0.288 ± 0.034
CoE	0.715	0.108	$\mathbf{0.592 \pm 0.018}$	0.589	0.231	0.522 ± 0.012	0.726	0.057	0.374 ± 0.011
SFT	**0.869**	0.325	0.449 ± 0.017	**0.732**	0.417	$\mathbf{0.576 \pm 0.007}$	0.795	0.285	0.124 ± 0.015

Table 13: Comprehensive metrics for performance on prediction case (one variable) with irrelevant attributes.

Model/Method	NLP4LP			Mamo Easy			Mamo Complex		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
GPT-4o Mini	0.334	0.283	0.568 ± 0.056	0.323	0.298	0.481 ± 0.055	0.461	0.242	0.548 ± 0.061
GPT-4o	0.776	0.596	0.600 ± 0.050	0.531	0.538	0.503 ± 0.062	0.711	0.447	0.574 ± 0.047
Gemini 2.5 Pro	0.410	0.199	0.480 ± 0.039	0.245	0.184	0.395 ± 0.063	0.137	0.000	0.426 ± 0.041
Claude Haiku 4.5	0.879	**0.630**	0.640 ± 0.039	0.632	0.394	0.454 ± 0.040	0.907	0.467	0.557 ± 0.056
Claude Sonnet 4.6	0.842	0.328	0.489 ± 0.051	0.622	0.493	0.465 ± 0.046	0.853	0.598	0.528 ± 0.046
Claude Opus 4.6	0.852	0.518	0.493 ± 0.049	0.680	**0.633**	0.460 ± 0.054	**0.917**	**0.754**	$\mathbf{0.593 \pm 0.053}$
LLaMA 3.1 8B Inst.	0.795	0.000	0.498 ± 0.049	0.551	0.000	0.404 ± 0.034	0.596	0.000	0.420 ± 0.042
Ministral 8B	0.829	0.000	0.546 ± 0.059	0.559	0.000	0.470 ± 0.043	0.670	0.000	0.494 ± 0.058
LLaMA 3.3 70B Inst.	0.429	0.311	0.493 ± 0.060	0.389	0.399	0.492 ± 0.050	0.404	0.220	0.522 ± 0.058
Mistral Large	0.308	0.324	0.598 ± 0.060	0.123	0.090	0.490 ± 0.062	0.341	0.296	0.546 ± 0.049
Five-element	**0.890**	0.200	$\mathbf{0.745 \pm 0.011}$	0.677	0.100	0.418 ± 0.012	0.796	0.233	0.267 ± 0.018
CoE	0.826	0.298	0.736 ± 0.009	0.691	0.034	$\mathbf{0.518 \pm 0.008}$	0.768	0.033	0.358 ± 0.005
SFT	0.879	0.314	0.610 ± 0.034	**0.730**	0.339	0.321 ± 0.014	0.780	0.285	0.160 ± 0.018

Table 14: Comprehensive metrics for performance on prediction case (multiple variables).

Model/Method	NLP4LP			Mamo Easy			Mamo Complex		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
GPT-4o Mini	0.397	0.264	0.426 ± 0.035	0.250	0.181	0.415 ± 0.050	0.292	0.140	0.472 ± 0.050
GPT-4o	0.743	0.569	0.453 ± 0.037	0.492	0.510	0.419 ± 0.059	0.711	0.471	0.485 ± 0.052
Gemini 2.5 Pro	0.300	0.206	0.333 ± 0.055	0.192	0.130	0.298 ± 0.074	0.090	0.034	0.395 ± 0.058
Claude Haiku 4.5	0.778	0.599	0.428 ± 0.048	0.610	0.345	0.409 ± 0.045	0.825	0.389	0.503 ± 0.045
Claude Sonnet 4.6	0.803	0.253	0.331 ± 0.069	0.683	0.429	0.444 ± 0.052	0.773	**0.605**	0.473 ± 0.048
Claude Opus 4.6	**0.908**	**0.656**	0.432 ± 0.047	0.660	**0.558**	0.441 ± 0.045	**0.864**	0.636	$\mathbf{0.518 \pm 0.058}$
LLaMA 3.1 8B Inst.	0.583	0.000	0.384 ± 0.034	0.427	0.000	0.337 ± 0.037	0.457	0.000	0.354 ± 0.045
Ministral 8B	0.676	0.000	0.392 ± 0.033	0.493	0.011	0.395 ± 0.053	0.591	0.000	0.439 ± 0.042
LLaMA 3.3 70B Inst.	0.319	0.244	0.366 ± 0.040	0.273	0.230	0.394 ± 0.036	0.263	0.139	0.435 ± 0.046
Mistral Large	0.353	0.270	0.410 ± 0.046	0.117	0.111	0.407 ± 0.039	0.285	0.190	0.478 ± 0.045
Five-element	0.894	0.300	0.507 ± 0.015	0.681	0.100	0.329 ± 0.010	0.768	0.233	0.269 ± 0.019
CoE	0.722	0.309	$\mathbf{0.743 \pm 0.003}$	0.576	0.090	$\mathbf{0.471 \pm 0.033}$	0.719	0.094	0.294 ± 0.011
SFT	0.904	0.158	0.439 ± 0.022	**0.712**	0.164	0.286 ± 0.013	0.778	0.032	0.141 ± 0.019

Table 15: Comprehensive metrics for performance on prediction case (multiple variables) with irrelevant attributes.

Case	NLP4LP			Mamo Easy			Mamo Complex		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
NL	0.5488	0.7231	0.4190	0.7933	0.7517	0.6199	0.8062	0.7066	0.6224
NL + Data	0.8342	0.7080	0.6033	0.7268	0.6463	0.5453	0.8319	0.7241	0.6148
NL + Data + Irr	0.7640	0.5451	0.3848	0.7143	0.5813	0.4926	0.8550	0.7068	0.5743
Derivation	0.7589	0.5547	0.4595	0.6676	0.3369	0.3955	0.8043	0.5772	0.5443
Derivation + Irr	0.6921	0.3794	0.3519	0.6362	0.3118	0.3698	0.7848	0.5151	0.4881
Prediction (1-Var)	0.8021	0.5076	0.5361	0.7157	0.5865	0.4940	0.8392	0.6213	0.5524
Prediction (1-Var) + Irr	0.7768	0.3283	0.4112	0.6944	0.5157	0.4468	0.8524	0.5625	0.5057
Prediction (Multi-Var)	0.8162	0.5632	0.5409	0.7121	0.5492	0.4736	0.8539	0.5708	0.5667
Prediction (Multi-Var) + Irr	0.7988	0.4144	0.4100	0.6892	0.3748	0.4151	0.8013	0.3794	0.5081

Table 16: Evaluation results for Claude Opus 4.6 on the full benchmark.

Case	NLP4LP			Mamo Easy			Mamo Complex		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
NL	0.5172	0.0129	0.4554	0.7853	0.0484	0.6422	0.6145	0.1378	0.5933
NL + Data	0.7512	0.4018	0.6506	0.6280	0.4861	0.5301	0.6822	0.2486	0.6205
NL + Data + Irr	0.7097	0.3552	0.4253	0.5781	0.4292	0.4368	0.6216	0.2468	0.5376
Derivation	0.6503	0.2272	0.4502	0.5233	0.2200	0.4094	0.5393	0.1769	0.4886
Derivation + Irr	0.5797	0.1991	0.3636	0.4748	0.1738	0.3671	0.5504	0.1390	0.4419
Prediction (1-Var)	0.7091	0.2103	0.5197	0.6053	0.3059	0.4634	0.6577	0.1895	0.5148
Prediction (1-Var) + Irr	0.6776	0.1870	0.3941	0.5707	0.3303	0.4126	0.5850	0.1720	0.4752
Prediction (Multi-Var)	0.7128	0.3186	0.6082	0.5802	0.2610	0.4498	0.6852	0.1890	0.5138
Prediction (Multi-Var) + Irr	0.6671	0.2798	0.4201	0.5582	0.2586	0.4033	0.6206	0.1657	0.4595

Table 17: Evaluation results for GPT-4o Mini on the full benchmark.

Case	NLP4LP			Mamo Easy			Mamo Complex		
	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES	EF1	EBC	LLM-ES
NL	0.6900	0.2745	0.5193	0.6711	0.1124	0.6403	0.6371	0.2787	0.6095
NL + Data	0.4554	0.2995	0.6033	0.5043	0.3895	0.5202	0.6037	0.2272	0.5914
NL + Data + Irr	0.4710	0.2009	0.4004	0.4215	0.3046	0.4354	0.5017	0.1382	0.4881
Derivation	0.4192	0.1636	0.4498	0.3829	0.1930	0.4097	0.5778	0.1262	0.4990
Derivation + Irr	0.3707	0.0847	0.3699	0.3103	0.1161	0.3628	0.4487	0.0749	0.4305
Prediction (1-Var)	0.4522	0.1412	0.5041	0.5204	0.3221	0.4656	0.5822	0.1203	0.5043
Prediction (1-Var) + Irr	0.4067	0.0984	0.3929	0.3973	0.1974	0.4079	0.4091	0.0739	0.4243
Prediction (Multi-Var)	0.4629	0.2351	0.5591	0.5128	0.2345	0.4514	0.4703	0.0608	0.4895
Prediction (Multi-Var) + Irr	0.3385	0.1451	0.4145	0.3767	0.1190	0.3953	0.3629	0.0283	0.4186

Table 18: Evaluation results for LLaMA 3.3 70B Instruct on the full benchmark.