

OptiMUS-0.3: Using LLMs to model and solve optimization problems at scale

Journal version of ICML'24 paper

Ali AhmadiTeshnizi, Wenzhi Gao, Herman Brunborg, Shayan
Taleai, Connor Lawless, Madeleine Udell

[Stanford CS/MS&E 331](#)

Automating the modeling bottleneck

Integer programming powers decision-making in operations

- E.g., power system scheduling, medical resource allocation, ...

Expertise barrier [Gurobi '23]:

- 81% of Gurobi users hold advanced degrees
- 49% have formal training in operations research

Small firms, municipalities, NGOs lack modeling expertise

- Leads to missed opportunities in efficiency

Goal: automate modeling to democratize optimization

Challenges

- **Long problem descriptions**
 - Real specs can span dozens of pages → more modeling errors
- **Large problem data**
 - Industrial problems involve massive data tables
- **Hallucination**
 - LLMs invent constraints or API calls
 - Hard to detect: code may run but model logic is wrong
- **Poor model quality**
 - Solve time depends on formulation structure
 - LLMs rarely exploit modeling tricks used by experts

Dataset

355 problems: 287 easy LPs, 68 hard LP/MILPs

- Easy: short text, scalar params
- Hard: long, multi-dimensional

Each instance includes text, LaTeX, code, and solution

Covers domains like scheduling, routing, energy, and retail

Guarded release to prevent leakage

Components of an integer program

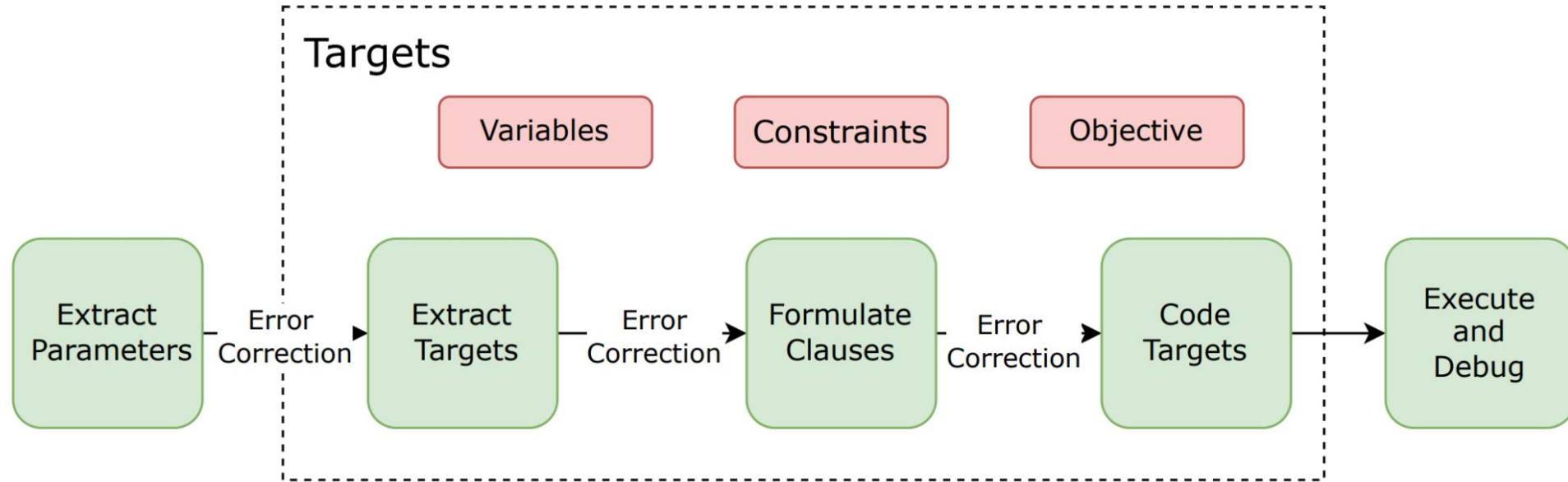
maximize $\mathbf{c} \cdot \mathbf{z}$

subject to $A\mathbf{z} \leq \mathbf{b}$

Some variables must be integral

- *Parameters:* $\mathbf{c}, A, \mathbf{b}$
- *Clauses:* Objective, constraints
- *Variables:* \mathbf{x}

OptiMUS pipeline



- LLMs at every stage
- Human + solver feedback:
 - Guide iterative LLM corrections and debugging for reliability

1 Description

2 Parameters

3 Clauses

4 Formulation

5 Coding

6 Data

7 Testing

Problem Description

We are trying to figure out where to place a bike rental hub (a place where users park their cars and have bicycles available for rental). We have a set of potential hub locations L , and a set of customers we want to service C . Each customer i has cost $COST(i, j)$ to be serviced by placing a hub at location j . Each hub l costs $HUB_COST(l)$ to build, and each hub can service at most MAX_USERS potential customers. Our goal is to minimize the cost of servicing all the customers. Every customer should be serviced.

[Have Feedback?](#)

Made with ❤ at Udell Lab

gurobipy



Random

Analyze

Objective

Minimize the total cost of servicing all customers, w

Formulate

Minimize $\sum_{l \in L} (\text{HubCost}_l \cdot \text{HubPlaced}_l) +$
 $\sum_{i \in C} \sum_{j \in L} (\text{ServiceCost}_{ij} \cdot \text{Serviced}_{ij})$

Confidence: 5/5

$$\text{Minimize } \sum_{l \in L} (\text{HubCost}_l \cdot \text{HubPlaced}_l) + \sum_{i \in C} \sum_{j \in L} (\text{ServiceCost}_{ij} \cdot \text{Serviced}_{ij})$$

Constraints

Each customer must be serviced by at least one hu

Formulate

$\sum_{j \in L} \text{Serviced}[i, j] \geq 1, \quad \text{forall } i \in C$

Confidence: 5/5

$$\sum_{j \in L} \text{Serviced}[i, j] \geq 1, \quad \forall i \in C$$

Each hub can service at most MaxUsers potential c

Formulate

$\sum_{i \in C} \text{Serviced}_{ij} \leq \text{MaxUsers} \cdot \text{HubPlaced}_j, \quad \text{forall } j \in L$

Confidence: 5/5

$$\sum_{i \in C} \text{Serviced}_{ij} \leq \text{MaxUsers} \cdot \text{HubPlaced}_j, \quad \forall j \in L$$

 [Have Feedback?](#)

Made with ❤ at Udell Lab

Objective

$$\text{Minimize } \sum_{l \in L} (\text{HubCost}_l \cdot \text{HubPlaced}_l) + \sum_{i \in C} \sum_{j \in L} (\text{ServiceCost}_{i,j} \cdot \text{Served}_{i,j})$$

Generate Code

```
1 model.setObjective(gp.quicksum(HubCost[l] * HubPlaced[l] for l
  in L) + gp.quicksum(ServiceCost[i, j] * Served[i, j] for
  i in C for j in L), gp.GRB.MINIMIZE)
```

Confidence: 5/5

Constraints

$$\sum_{j \in L} \text{Served}_{i,j} \geq 1, \quad \forall i \in C$$

Generate Code

```
1 for i in C:
2     model.addConstr(gp.quicksum(Served[i, j] for j in L) >= 1
  , name=f"customer_served_{i}")
```

Confidence: 5/5

$$\sum_{i \in C} \text{Served}_{i,j} \leq \text{MaxUsers} \cdot \text{HubPlaced}_j, \quad \forall j \in L$$

Generate Code

```
1 for j in range(len(L)):
2     model.addConstr(gp.quicksum(Served[i, j] for i in range
  (len(C))) <= MaxUsers * HubPlaced[j], name
  =f"hub_service_capacity_{j}")
```

Confidence: 5/5

1 Description

2 Parameters

3 Clauses

4 Formulation

5 Coding

6 Data

7 Testing



[Have Feedback?](#)

Made with ❤ at Udell Lab

- 1 Description
- 2 Parameters
- 3 Clauses
- 4 Formulation
- 5 Coding
- 6 Data
- 7 Testing

Full Code

```

1
2 import json
3 import numpy as np
4
5 import gurobipy as gp
6
7 with open("tmpData/sPXhp1SzuK5M8ELe2ddp/data.json", "r") as f:
8     data = json.load(f)
9
10
11 ServiceCost = data["Cost"]
12 L = list(range(data["L"]))
13 MaxUsers = data["MaxUsers"]
14 C = list(range(data["C"]))
15 HubCost = data["HubCost"]
16
17 # Define model
18 model = gp.Model('model')
19
20
21 # ===== Define variables =====
22 HubPlaced = model.addVars(len(L), name='HubPlaced', vtype=gp.GRB.BINARY)
23 Serviced = model.addVars(len(C), len(L), name='Serviced', vtype=gp.GRB.BINARY)
24
25 # ===== Define constraints =====
26
27 for i in C:

```

Results

```

● ● ●
Run Successful!
-----
Status: Optimal (2)
Objective Value: 24.0000
Runtime: 0.0122
Iteration Count: 11
-----
Variables:
HubPlaced[0]: 0.0000
HubPlaced[1]: 1.0000
HubPlaced[2]: 1.0000
HubPlaced[3]: 0.0000
HubPlaced[4]: 1.0000
Serviced[0,0]: 0.0000
Serviced[0,1]: 0.0000
Serviced[0,2]: 1.0000
Serviced[0,3]: 0.0000

```

Synthesize Full Code from Clause Codes

Run Code

Fix Code



[Have Feedback?](#)

Made with ❤ at Udell Lab

Error correction

- **Goal:** Mitigate hallucinations
 - Typical errors: wrong parameters, redundant constraints, invalid code
- **Two correction layers:**
 - *Reflective prompts:* LLM self-checks and revises outputs
 - *Confidence-based feedback:* uncertain results flagged for user review
- Reflective prompting process:
 - Analyzed errors at every modeling stage
 - Designed targeted reflective prompts for each error type
- Substantially lowers modeling error rates

Are units the same for both sides of this constraint?

$$(p_a + x_a) \cdot d_a \cdot (1 + e_a \cdot \frac{x_a}{p_a}) \leq m_a, \forall a \in A?$$

... Left-hand side (LHS):

- $(p_a + x_a)$ represents the new price for article a , which is in euros (€).
- d_a represents the sales forecast (demand) for article a for the next twelve months at the current price, which is in units of the article.
- $(1 + e_a \cdot \frac{x_a}{p_a})$ is a unitless factor ...

Therefore, the unit of the left-hand side is: **euros (€) × units of the article**

Right-hand side (RHS):

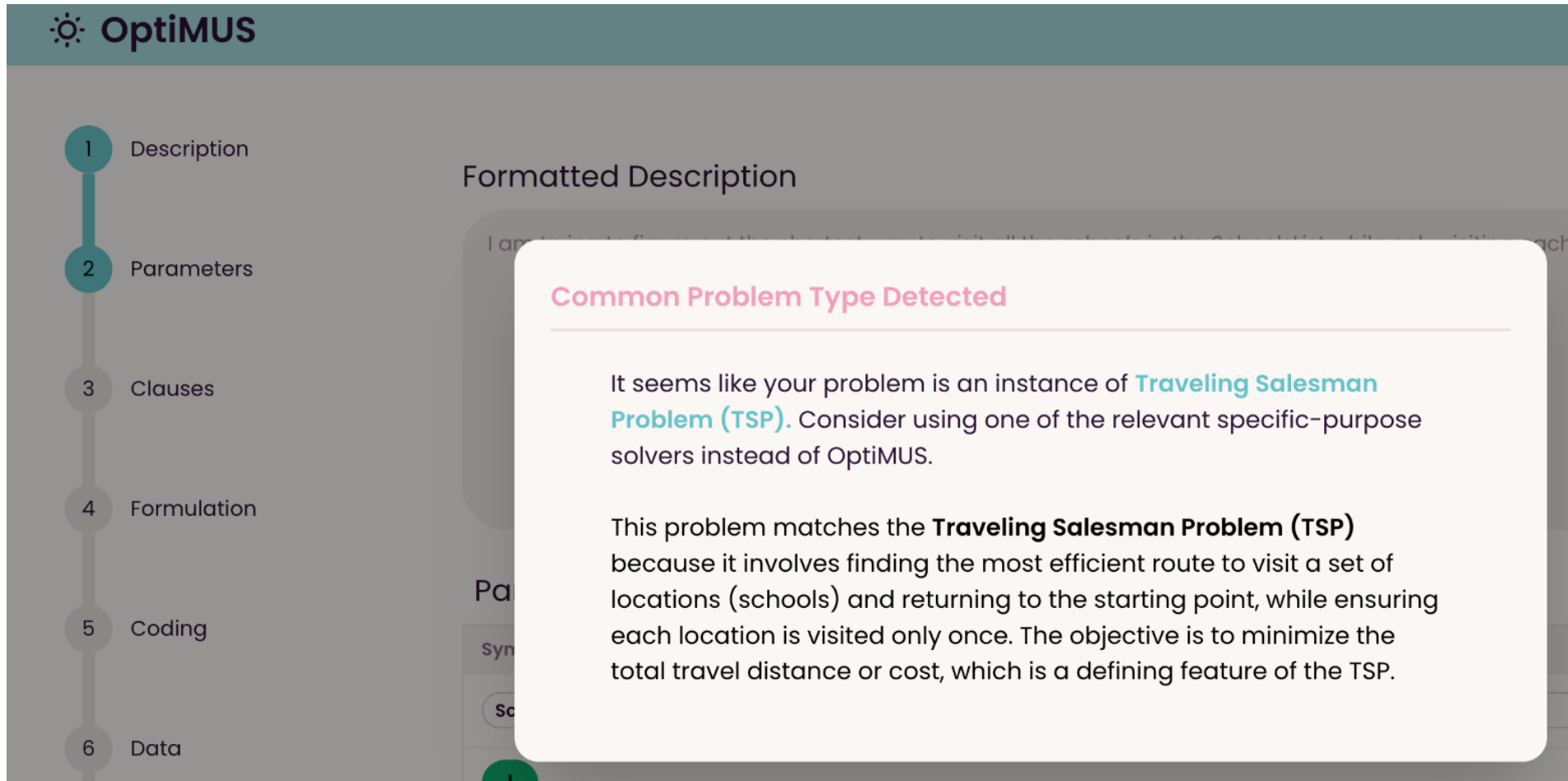
- m_a represents the maximum production volume for article a , which is in units of the article (e.g., number of shirts, pants, etc.).

The unit of the right-hand side is: **units of the article**

... this inconsistency suggests an error in the formulation of Constraint 5. To correct this, we should ... here is the corrected constraint:

$$d_a \cdot (1 + e_a \cdot \frac{x_a}{p_a}) \leq m_a, \forall a \in A$$

Identifying special problems



Structure detection agent

- **Goal:** Identify and exploit special structures
 - Enhances solver performance and simplifies formulations
- Common structures:
 - Special Ordered Sets (SOS)
 - Indicator and semi-continuous variables
 - Piecewise-linear constraints
- Appear in ~10% of NLP4LP problems
- **Method:**
 - Iterates through known structures
 - LLM decides whether structure applies, then reformulates

	LLM	NL4OPT	NLP4LP	IndustryOR
<i>Methods based on direct prompting</i>				
Standard	GPT-4o	47.3%	33.2%	28.0%
Standard	o1	> 95%	68.8%	44.0%
Reflexion	GPT-4o	53.0%	42.6%	–
<i>Methods based on fine-tuning LLMs</i>				
LLMOPT	Qwen1.5-14B	93.0%*	83.8%*	46.0%*
ORLM	Deepseek-Math	86.5%*	72.9%*	38.0%*
<i>Methods based on agentic frameworks</i>				
CoE	GPT-4o	64.2%	49.2%	–
OptiMUS-0.2	GPT-4o	78.8%	68.0%	–
OptiMUS-0.3	GPT-4o	86.6%	73.7%	37.0%
OptiMUS-0.3	o1	–	80.6%	46.0%

Takeaways:

- Decomposition frameworks out-perform LLMs alone
 - Especially with cheaper models
- Fine-tuning adds a performance increase
 - But OptiMUS is competitive without fine-tuning