Model Formulation

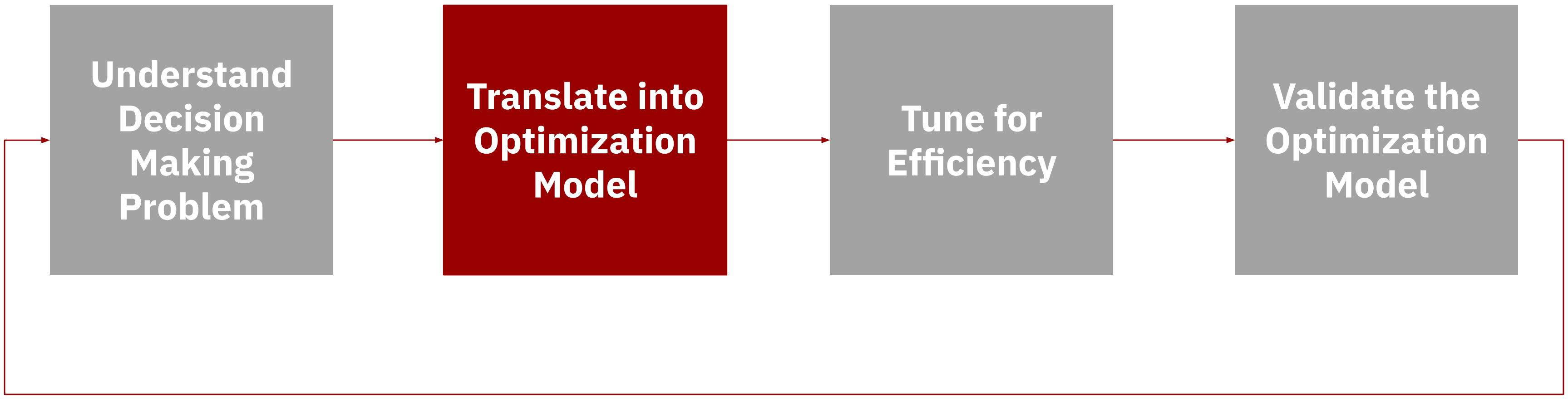
Modeling Optimization Problems via Generative AI

Connor Lawless, Stanford University

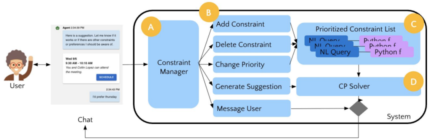
AAAI 2026 | January 20th, 2025

Auto-Formulation

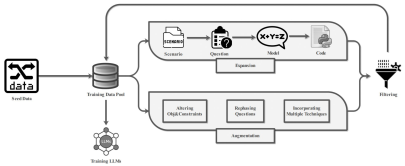
We'll start by focusing on the process of mapping a natural language description into a concrete optimization model.



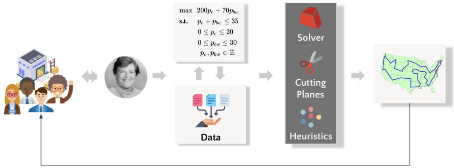
Roadmap



LLMs x CP
Why combine LLMs and Optimization Solvers?

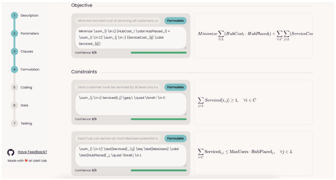


Model Fine Tuning
Can we fine-tune models for auto-formulation?

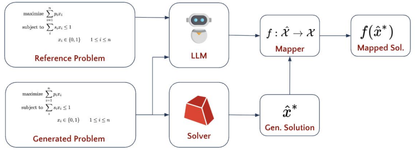


What's next?
Open research questions

Agentic Frameworks
How far can we push LLMs out of the box in formulation?



Evaluation
How do we verify formulations are correct?



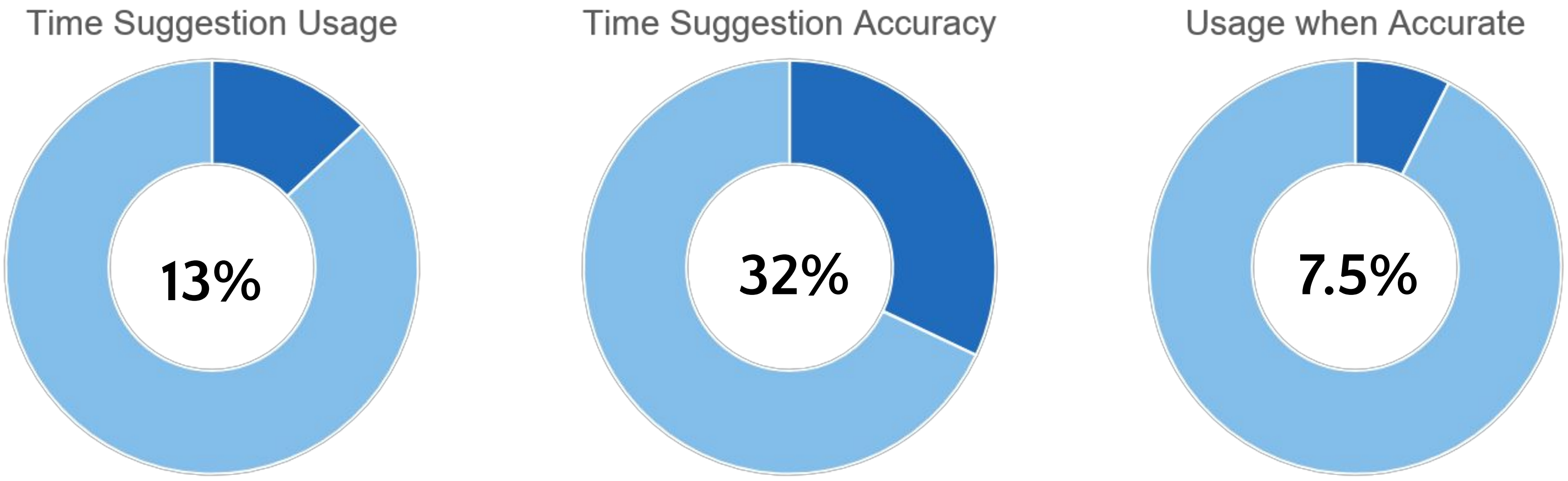
“I Want it That Way”: Leveraging LLMs and Constraint Programming for Interactive Decision Support

Connor Lawless, Jakob Schoeffer, Lindy Le, Kael Rowan, Shilad Sen, Cristina St Hill, Jina Suh, Bahar Sarrafzadeh

ACM Transactions of Intelligent and Interactive Systems

Meeting Scheduling is tough!

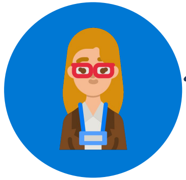
Finding a time for a meeting can be a drain, but current smart assistants (i.e. like Outlook's time suggestions) are barely used… even when they give a good suggestion!



*Statistics courtesy of Jie Fan and the associated report by Kelsey Merlo at Microsoft

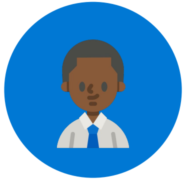
Meeting Scheduling is tough!

Users can feel frustrated by the lack of control and resort to manually scheduling.



I don't trust time suggestions because they tend to **not give me enough information or control** to do things like adhere to meeting-free Friday.

I would find it helpful to **tweak time suggestions to my need** and have all the information available that I need.



I think **insights into how to relax scheduling constraints** would be helpful in making the meeting.

*Quotes taken from previous studies on meeting scheduling run by Bahar Sarrafzadeh and the OAR team at MSR.

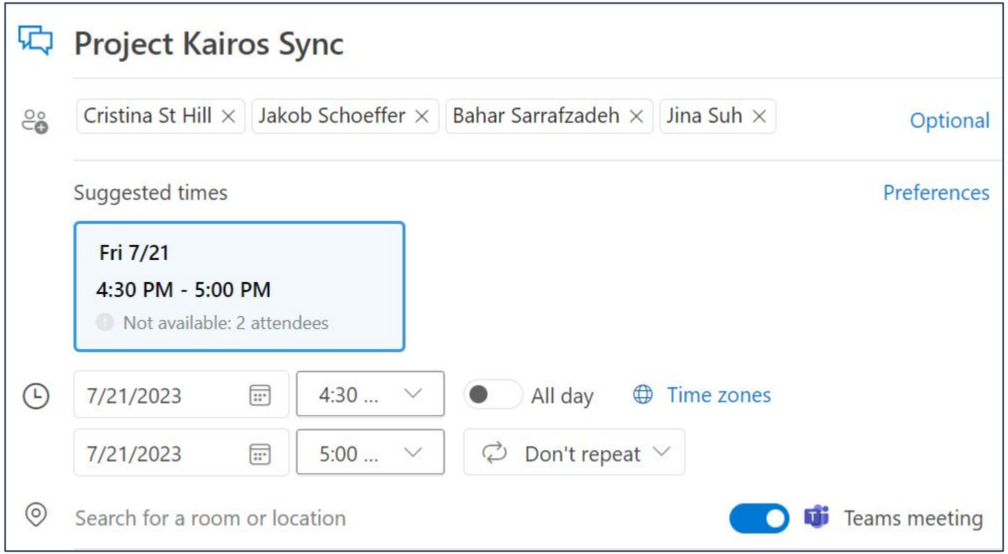
User Preferences are Diverse

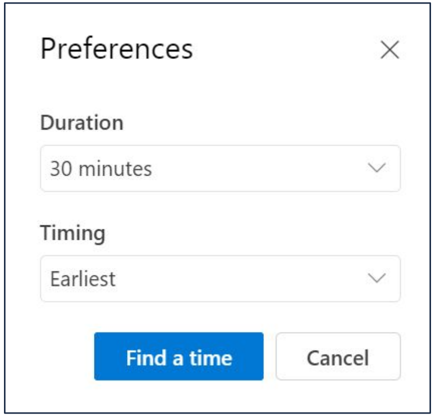
To capture the diversity of user preferences we ran a diary study (>100 participants) to collect users *in the moment* scheduling preferences and constraints .

Schedule a 30-minute meeting with [coworker and I] within the next two business days. <u>I would assume, without actually saying it in the chat</u>, that CoPilot knows that we are in the same ***time zone****, that we work normal* ***business hours****,* ***holidays*** *are observed, and not to schedule it during* ***lunch hours****. Also, I would assume that it knows that* ***we work in the same building*** *and that* ***we both work hybrid*** *(work from home sometimes), so that if we are both going to be at work at the same time, then a small conference room would be preferrable*

Current Systems

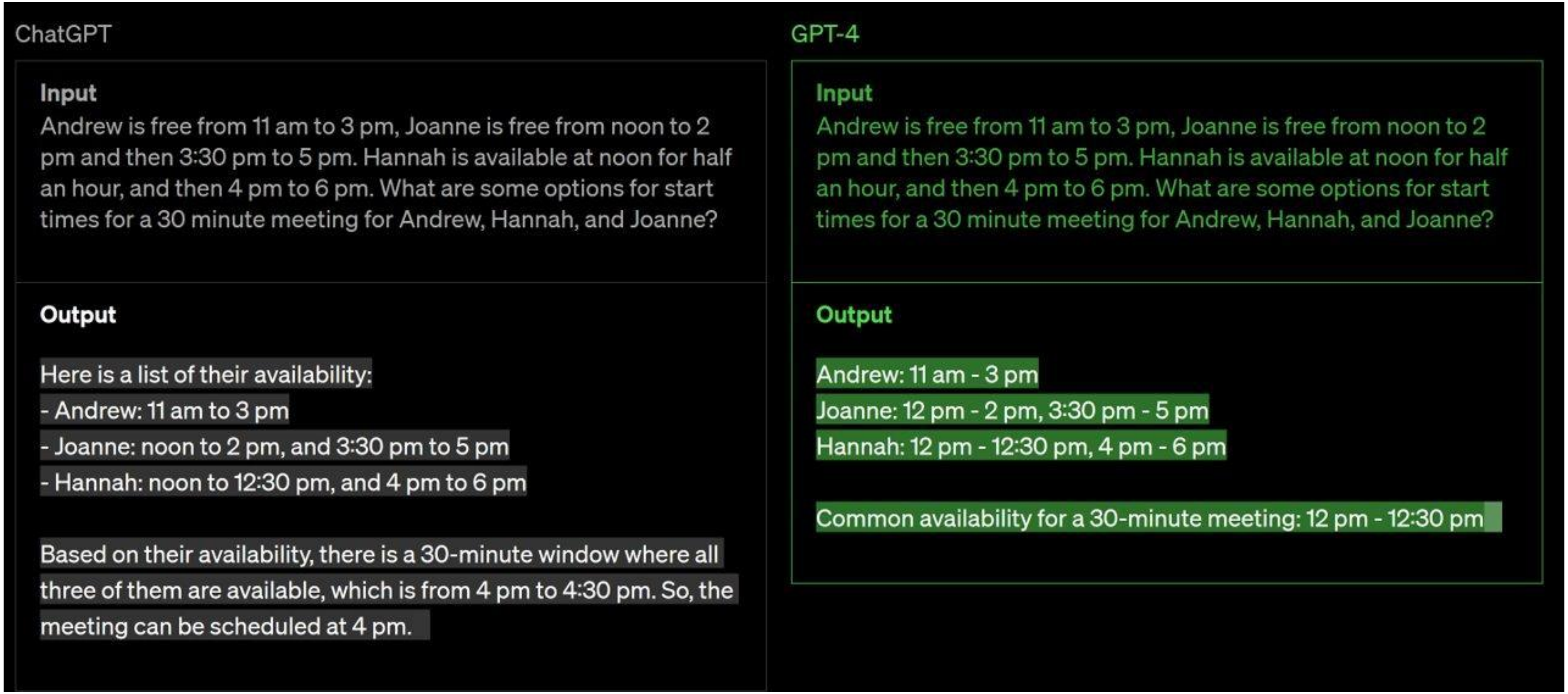
Current graphical interfaces do not have the flexibility to capture these preferences.





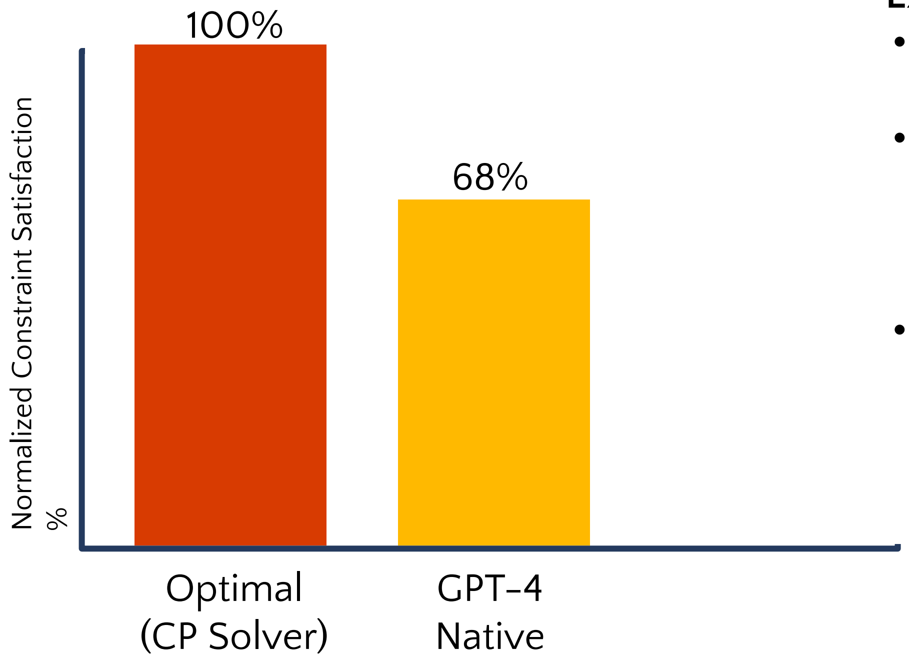
Can we leverage LLMs and OR to enable *personalized* meeting scheduling models?

GPT-4 – Problem Solved?



GPT-4 – Problem Solved? Not quite!

Initial experiments showed GPT-4 **has trouble solving constraint satisfaction problems!***



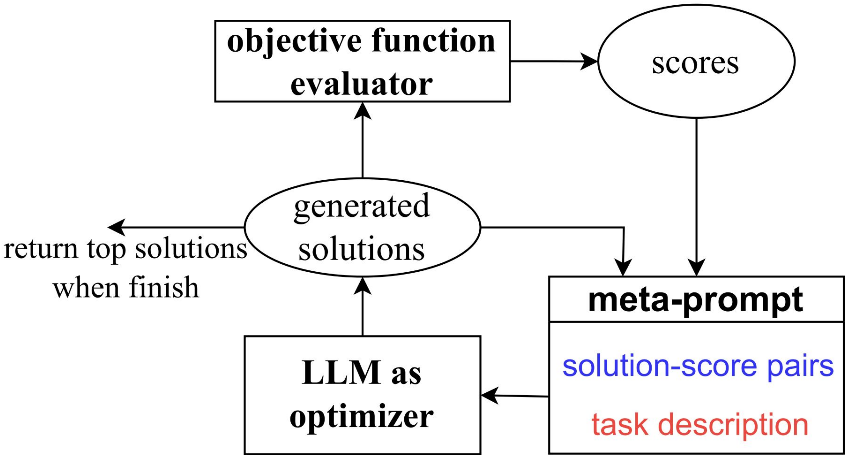
Experiment Details

- 10 synthetic scheduling scenarios with 2-8 participants
- 3 given constraints
  - Meet 11-3pm
  - No Meeting Fridays
  - Prefer to meet Tuesday or Thursday
- Evaluate based on how many constraints (including user attendance) proposed time meets

**See: 'Attention Satisfies: A Constraint Satisfaction Lens on Factual Errors of Language Models' Yuksekgonul et al. 2024*

LLMs as Optimizers

Similar Idea: Let the LLMs solve the optimization problem directly!



Now you will help me minimize a function with two input variables w, b. I have some (w, b) pairs and the function values at those points. The pairs are arranged in descending order based on their function values, where lower values are better.

input:
w=18, b=15
value:
10386334

input:
w=17, b=18
value:
9204724

Give me a new (w, b) pair that is different from all pairs above, and has a function value lower than any of the above. Do not write code. The output must end with a pair [w, b], where w and b are numerical values.

Large Language Models as Optimizers. Yang et al. *ICLR 2024*

Constraint Programming

Constraint Programming (Rossi et al., 2006) is a general optimization framework that grades a candidate solution by evaluating and weighing multiple functions.

Variables and Domain
What we can change (variable) and allowable values (domain)

$$\mathcal{X} = \{x_1, \ldots, x_n\} \quad \mathcal{D} = \{D_1, \ldots, D_n\}$$

Constraints
Functions that map from a variable setting to a score.

$$\mathcal{F} = \{f_1, \ldots, f_m\} \quad f_i : \times_{x_j \in \mathbf{x}^{f_i}} D_j \to \mathbb{R}_0^+ \cup \{\perp\}$$

Objective
The goal is to find the variable settings that maximize the score.

$$\mathcal{F}_{\mathcal{P}}(\mathbf{x}_\sigma) = \sum_{f \in \mathcal{F}, \mathbf{x}^f \subseteq \mathbf{x}_\sigma} f(\mathbf{x}_\sigma) \quad \mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} \mathcal{F}_{\mathcal{P}}(\mathbf{x})$$

Constraint Programming

Constraint Programming (Rossi et al., 2006) is a general optimization framework that grades a candidate solution by evaluating and weighing multiple functions.

Variables and Domain What we can change (variable) and allowable values (domain)	The variable is the **meeting time**, and the domain is the **set of time blocks** of correct duration in the time horizon (i.e. next 2 weeks)
Constraints Functions that map from a variable setting to a score.	Constraints could be **any user preference or meeting requirement** (i.e. user is available) and a score is an importance of the constraint.
Objective The goal is to find the variable settings that maximize the score.	Our goal is to **find the 'best' time(s).**

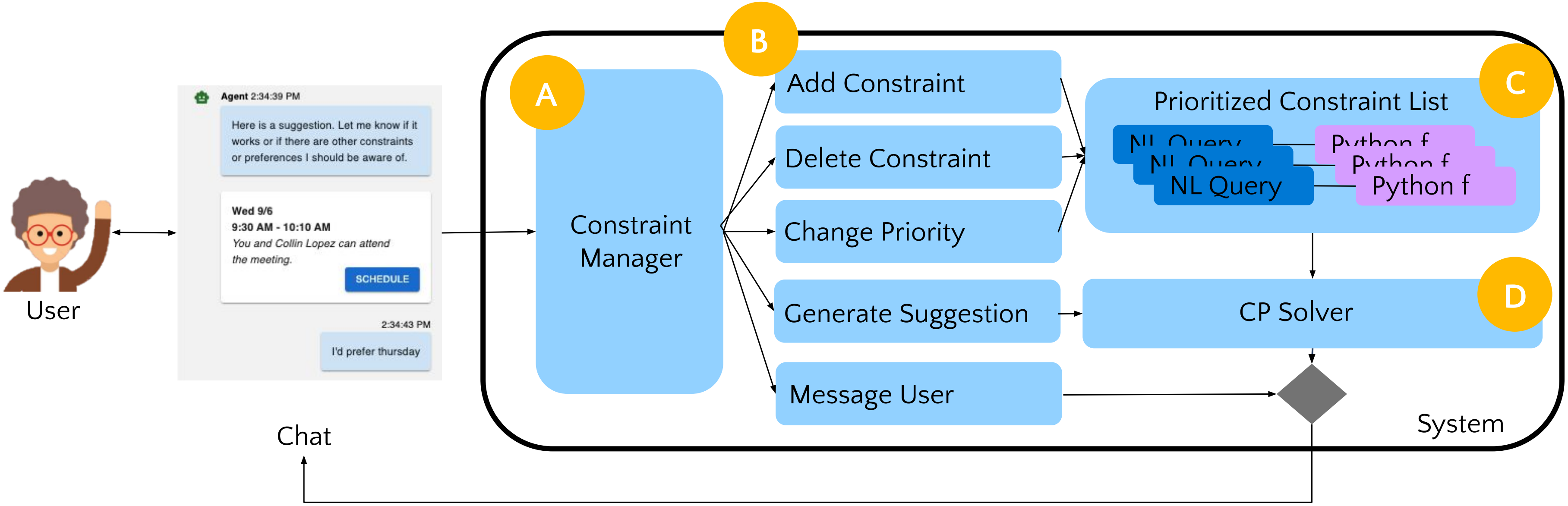
Constraint Programming

Constraint Programming (Rossi et al., 2006) is a general optimization framework that grades a candidate solution by evaluating and weighing multiple functions.

Variables and Domain What we can change (variable) and allowable values (domain)	The variable is the **meeting time**, and the domain is the **set of time blocks** of correct duration in the time horizon (i.e. next 2 weeks)
Constraints Functions that map from a variable setting to a score.	Constraints could be **any user preference or meeting requirement** (i.e. user is available) and a score is an importance of the constraint. **Not known a priori!**
Objective The goal is to find the variable settings that maximize the score.	Our goal is to **find the 'best' time(s).**

LLMs x Constraint Programming

We introduced a **hybrid LLM and optimization** system to enable non-expert users build custom constraint programming models..



LLMs for Constraint Generation

We use LLMs as a flexible tool to convert natural language constraints into code.

Documentation

```
You are a meeting scheduling assistant that is translating a user constraint to code.

You have access to the following inputs:
- organizer: a string representing the name of the meeting organizer.
- duration: an integer representing the duration of the meeting in minutes.
- candidate_time: the time to evaluate. Each candidate time has the following:
…

Your job is to write a python function called meeting_constraint that checks where or not the time
meets the condition.
```

Examples

```
Here are some examples:

User: I can only meet in the morning.
Code:
def meeting_constraint(organizer, duration, candidate_time, calendar_service):
      return candidate_time.start.hour + duration/60 < 12
…
```

Current Instance

```
The meeting you are scheduling has the following details:

organizer: Desiree Cain
attendees: Desiree Cain, Collin Lopez, Lauren Sanchez

User: Meeting before 11am
Code:
```

Output

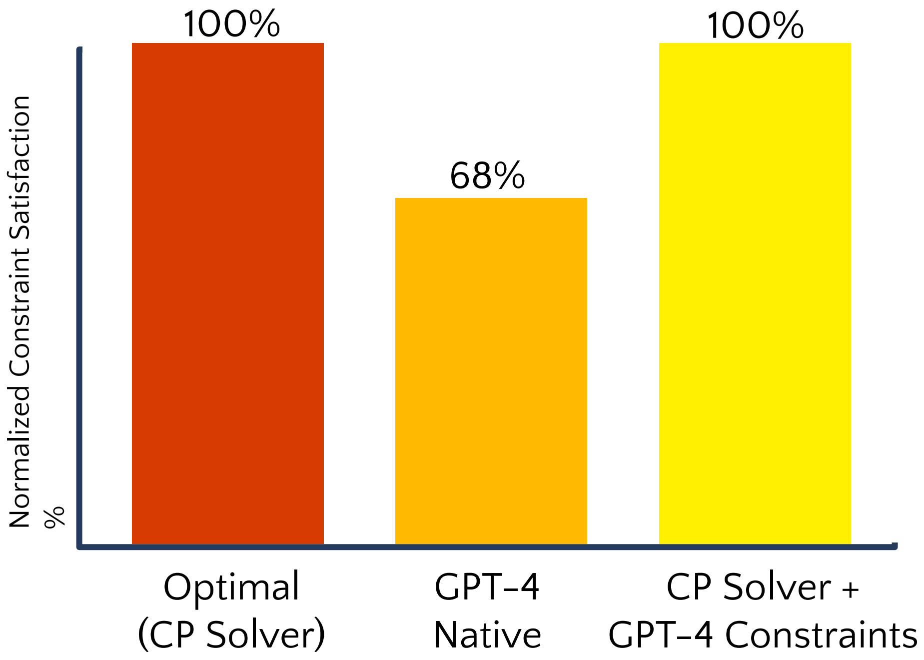
```
def meeting constraint(organizer, duration, candidate_time, calendar_service):
      return candidate_time.start.hour < 11
```

Disclaimer!

In our setting, solving the CP problem can be solved by enumeration in under a second.

Initial Hybrid Experiment Results

Initial results show that **using GPT-4 to generate python functions for natural language constraints seems to be able to achieve human-level performance!**



Experiment Details

- 10 synthetic scheduling scenarios with 2-8 participants
- 3 given constraints
  - Meet 11-3pm
  - No Meeting Fridays
  - Prefer to meet Tuesday or Thursday
- Evaluate based on how many constraints (including user attendance) proposed time meets

Quantitative Evaluation

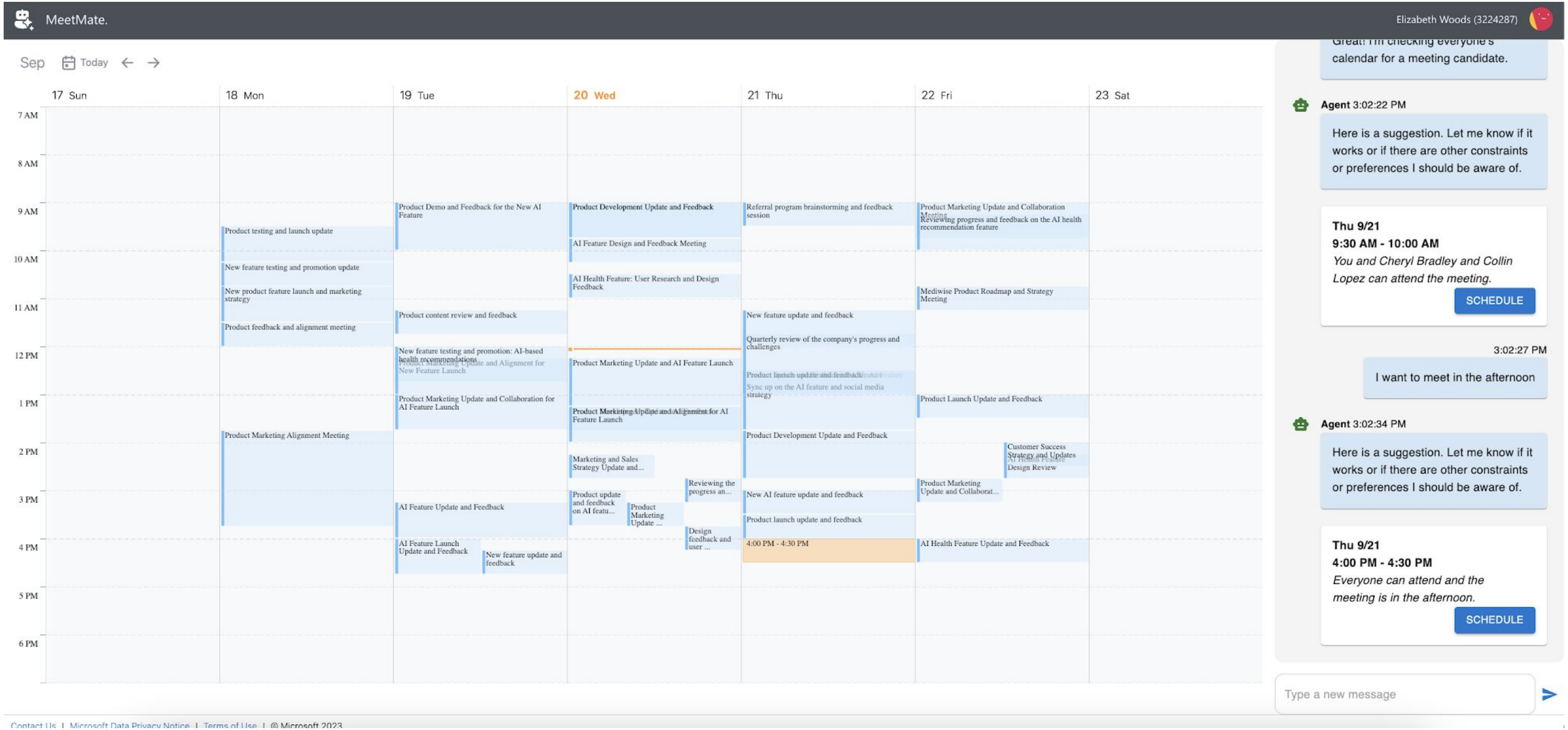
We benchmarked two LLMs on a new dataset constructed from our diary study to quantitatively **evaluate the feasibility of LLMs in constraint generation**.

Table 1. Comparison of LLM performance on information checking and code generation components on datasets generated from the results of the diary study. All numbers are reported as percentages.

LLM	Rephraser	Safeguard Accuracy	Compilation	Correctness (General)		Correctness (Example)	
				Precision	Recall	Precision	Recall
GPT-3	Y	**81.8%**	95.3%	95.5%	92.6%	94.4%	92.4%
	N	77.9%	90.7%	94.6%	86.0%	93.8%	87.2%
GPT-4	Y	79.8%	**97.2%**	**95.8%**	**94.0%**	**94.7%**	**93.8%**
	N	72.7%	93.4%	95.2%	89.6%	94.2%	90.3%

*Precision and Recall computed based on running constraint code on sample times and comparing output to ground-truth 'correct' implementations of the constraints.

Meet MeetMate



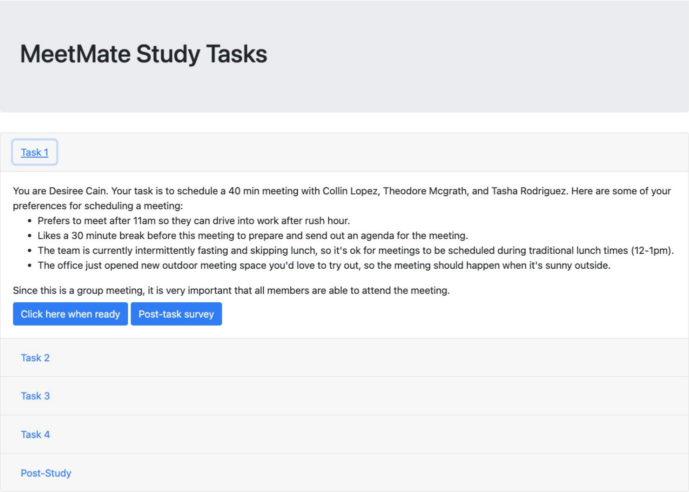
User Study

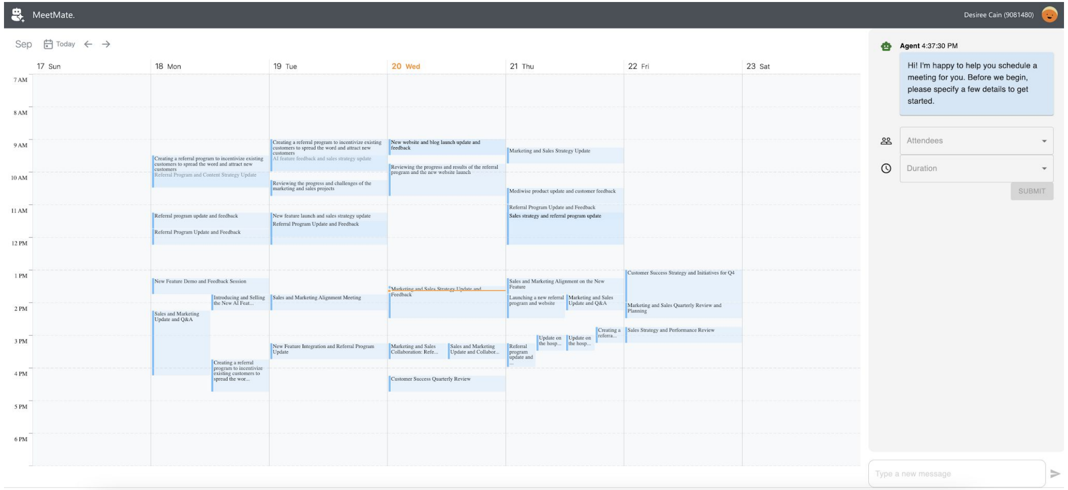
We evaluated the broader system via a user study with a prototype the system to:

(i) Characterize preference construction in situ with dynamic recommendations

(ii) Gauge usability of overall system to inform future design recommendations.

We recruited 10 participants to each do a 1 hour study session with researchers.



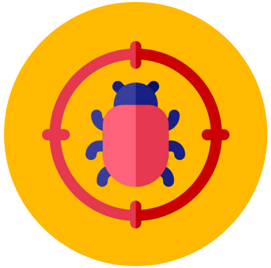


User Study: Takeaways

Interactive opt. systems are promising, but have some tough HCI challenges!

Easy to Use & Flexible
Users really enjoyed the flexibility and the responsiveness of the system to new user preferences.

Chat is Burdensome
For more complicated settings, users found it annoying to specify everything by chat.



Hard to Debug!
Since users did not understand the underlying model, it was hard to correct when things went wrong.

Constraint Programming x LLMs

There's a ton of work on auto-formulation of constraint programming models that parallels our tutorial:

Holy Grail 2.0: From Natural Language to Constraint Models
Tsouros, Verhaeghe, Kadioglu, Guns. *Preprint (2023)*

Ner4opt: Named entity recognition for optimization modelling from natural language
Dakle, Kadioglu, Uppuluri, Politi, Raghavan,Rallabandi, and Srinivasamurthy. *CPAIOR (2023), Constraints (2024)*

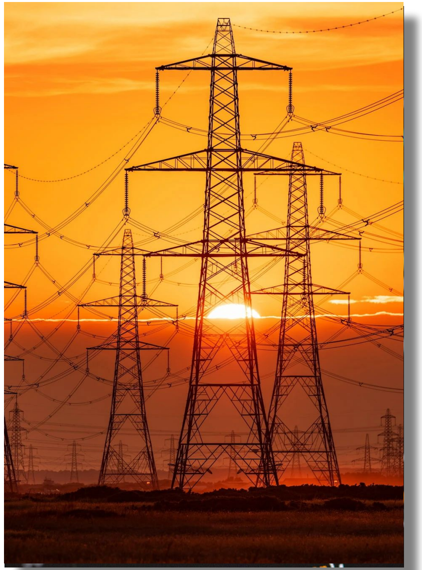
CP-bench: Evaluating large language models for constraint modelling
Michailidis, Tsouros, and Guns. *Preprint (2025)*

GALA: Global LLM Agents for Text-to-Model Translation
Cai, Kadioglu, and Dilkina. *Preprint (2025)*

CP-agent: Agentic constraint programming
Szeider. *Preprint (2025)*

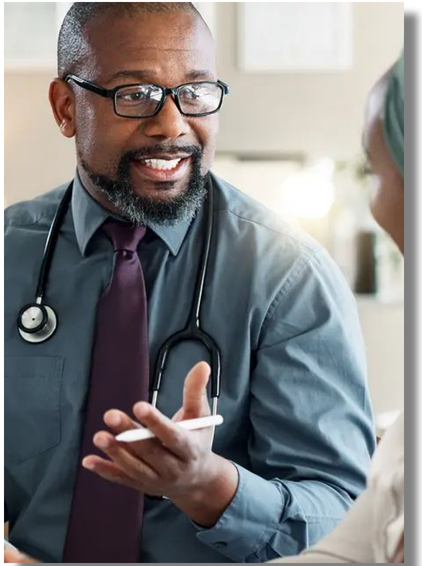
Beyond Meeting Scheduling

Iteratively eliciting modeling details to refine a solution is a feature of applied OR!



Power Systems

School Zoning



Hospital Scheduling

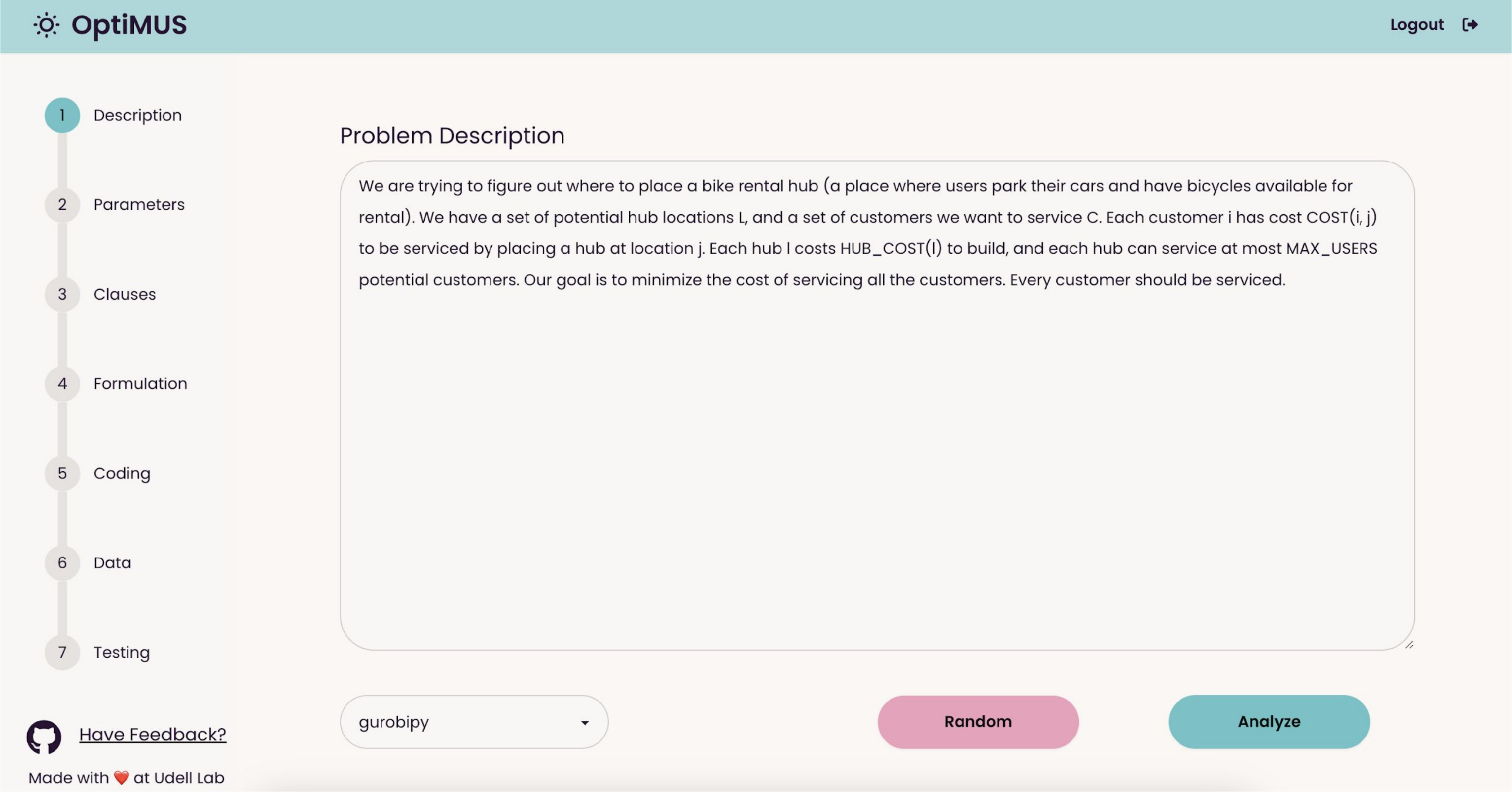
OptiMUS

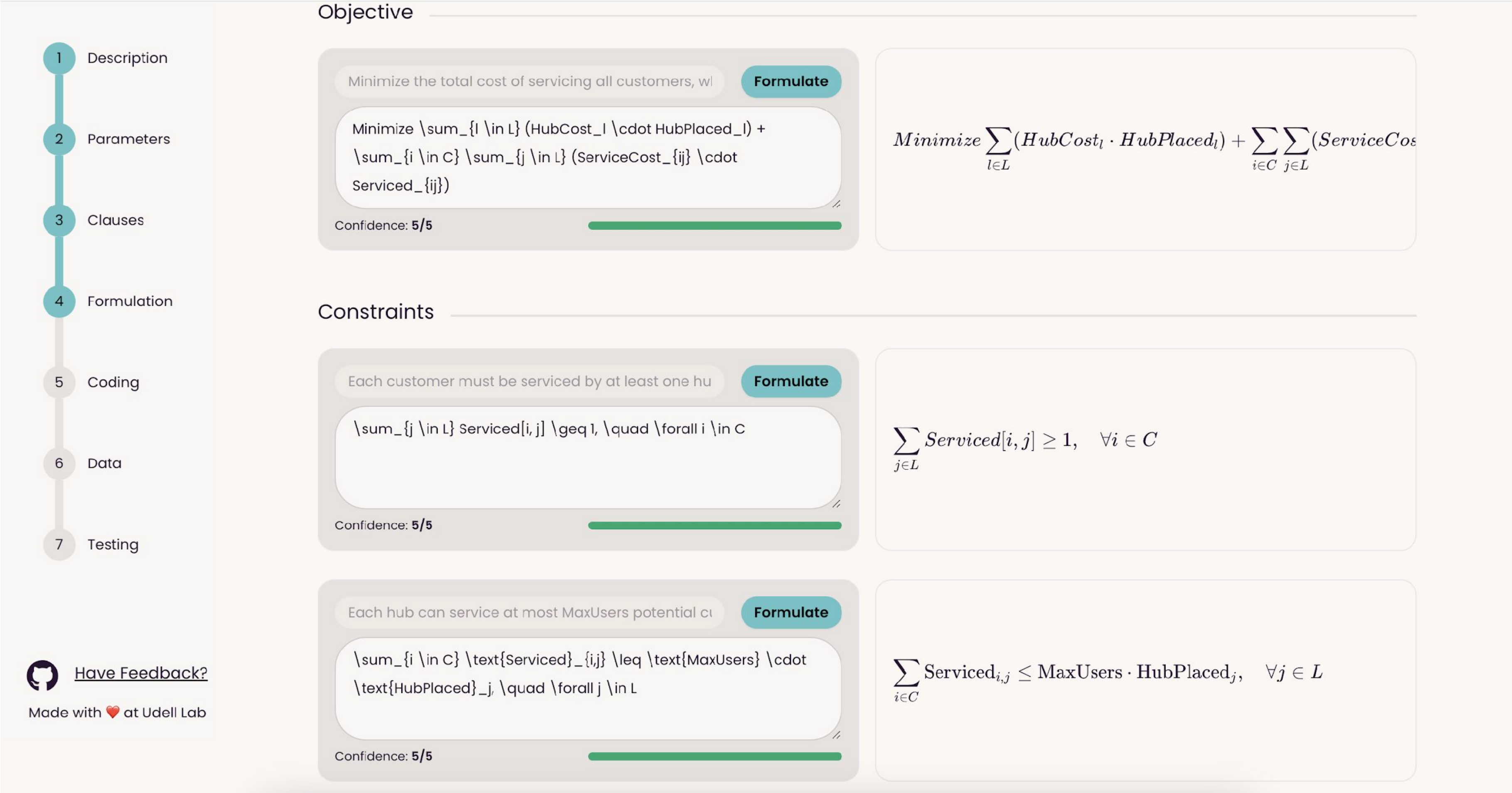
The same principles of MeetMate underpin more general systems for modelling MILP problems.

Extract Parameters → Extract Targets → Formulate In Math → Implement in Python → Execute + Debug

Targets: Variables | Constraints | Objective

OptiMUS-0.3: Using Large Language Models to Model and Solve Optimization Problems at Scale
AhmadiTeshnizi, Gao, Brunborg, Talaei, **Lawless**, and Udell. *Major Revision at Management Science*
Try it out yourself: https://optimus-solver.com/


OptiMUS
Logout
1 Description
2 Parameters
3 Clauses
4 Formulation
5 Coding
6 Data
7 Testing
Have Feedback?
Made with ❤️ at Udell Lab
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.
gurobipy
Random
Analyze


Objective
1 Description
2 Parameters
3 Clauses
4 Formulation
5 Coding
6 Data
7 Testing
Minimize the total cost of servicing all customers, wl
Formulate
Minimize \sum_{l \in L} (HubCost_l \cdot HubPlaced_l) + \sum_{i \in C} \sum_{j \in L} (ServiceCost_{ij} \cdot Serviced_{ij})
Confidence: 5/5
$Minimize \sum_{l \in L} (HubCost_l \cdot HubPlaced_l) + \sum_{i \in C} \sum_{j \in L} (ServiceCos$
Constraints
Each customer must be serviced by at least one hu
Formulate
\sum_{j \in L} Serviced[i, j] \geq 1, \quad \forall i \in C
Confidence: 5/5
$\sum_{j \in L} Serviced[i, j] \geq 1, \quad \forall i \in C$
Each hub can service at most MaxUsers potential cu
Formulate
\sum_{i \in C} \text{Serviced}_{i,j} \leq \text{MaxUsers} \cdot \text{HubPlaced}_j, \quad \forall j \in L
Confidence: 5/5
$\sum_{i \in C} \text{Serviced}_{i,j} \leq \text{MaxUsers} \cdot \text{HubPlaced}_j, \quad \forall j \in L$
Have Feedback?
Made with ❤️ at Udell Lab

1 Description

2 Parameters

3 Clauses

4 Formulation

5 Coding

6 Data

7 Testing

Have Feedback?

Made with ❤️ at Udell Lab

Objective

$$Minimize \sum_{l \in L} (HubCost_l \cdot HubPlaced_l) + \sum_{i \in C} \sum_{j \in L} (ServiceC$$

Generate Code

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

Confidence: **5/5**

Constraints

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

Generate Code

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

Confidence: **5/5**

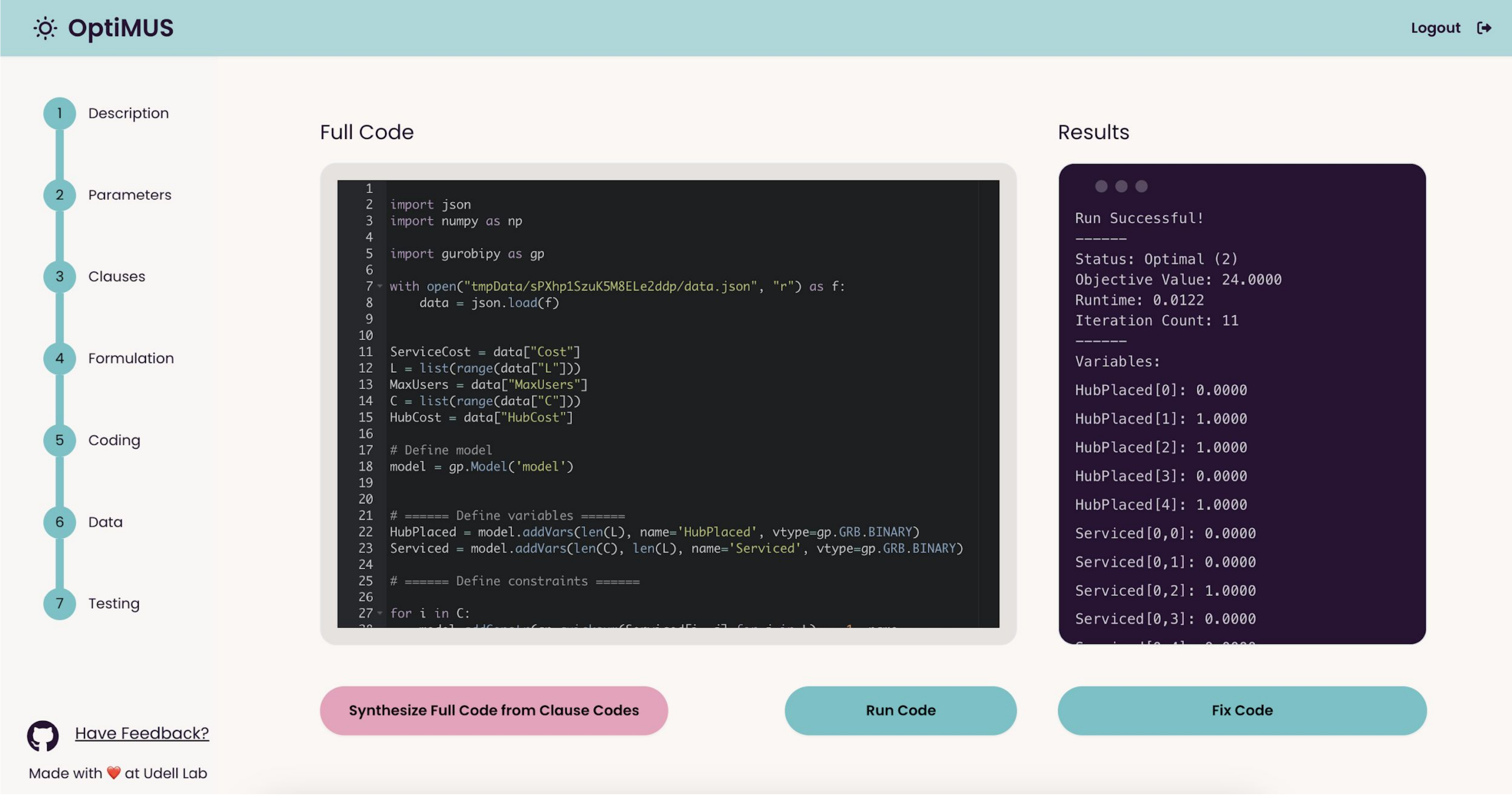
$$\sum_{i \in C} \text{Serviced}_{i,j} \leq \text{MaxUsers} \cdot \text{HubPlaced}_j, \quad \forall j \in L$$

Generate Code

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

Confidence: **5/5**

```
for i in range(len(C)):
    for j in range(len(L)):
```


OptiMUS
Logout
1 Description
2 Parameters
3 Clauses
4 Formulation
5 Coding
6 Data
7 Testing
Have Feedback?
Made with ❤️ at Udell Lab
Full Code
import json
import numpy as np
import gurobipy as gp
with open("tmpData/sPXhp1SzuK5M8ELe2ddp/data.json", "r") as f:
data = json.load(f)
ServiceCost = data["Cost"]
L = list(range(data["L"]))
MaxUsers = data["MaxUsers"]
C = list(range(data["C"]))
HubCost = data["HubCost"]
# Define model
model = gp.Model('model')
# ====== Define variables ======
HubPlaced = model.addVars(len(L), name='HubPlaced', vtype=gp.GRB.BINARY)
Serviced = model.addVars(len(C), len(L), name='Serviced', vtype=gp.GRB.BINARY)
# ====== Define constraints ======
for i in C:
Results
Run Successful!

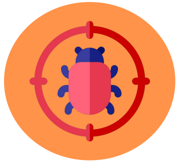
Status: Optimal (2)
Objective Value: 24.0000
Runtime: 0.0122
Iteration Count: 11

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

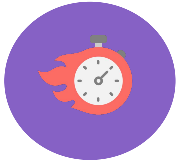
OptiMUS: Special Sauce

OptiMUS uses a number of methodological tricks to improve scalability:

Expert Crafted Error Correction Modules
Custom prompts for common errors made during modeling.



Iterative Debugging Loops
LLMs are given error messages from run code and allowed to fix.



Exploit (Simple) Optimization Structure
Check whether a given formulation has exploitable structure.

Error Correction

Expert-crafted prompts aim to correct common modelling mistakes.

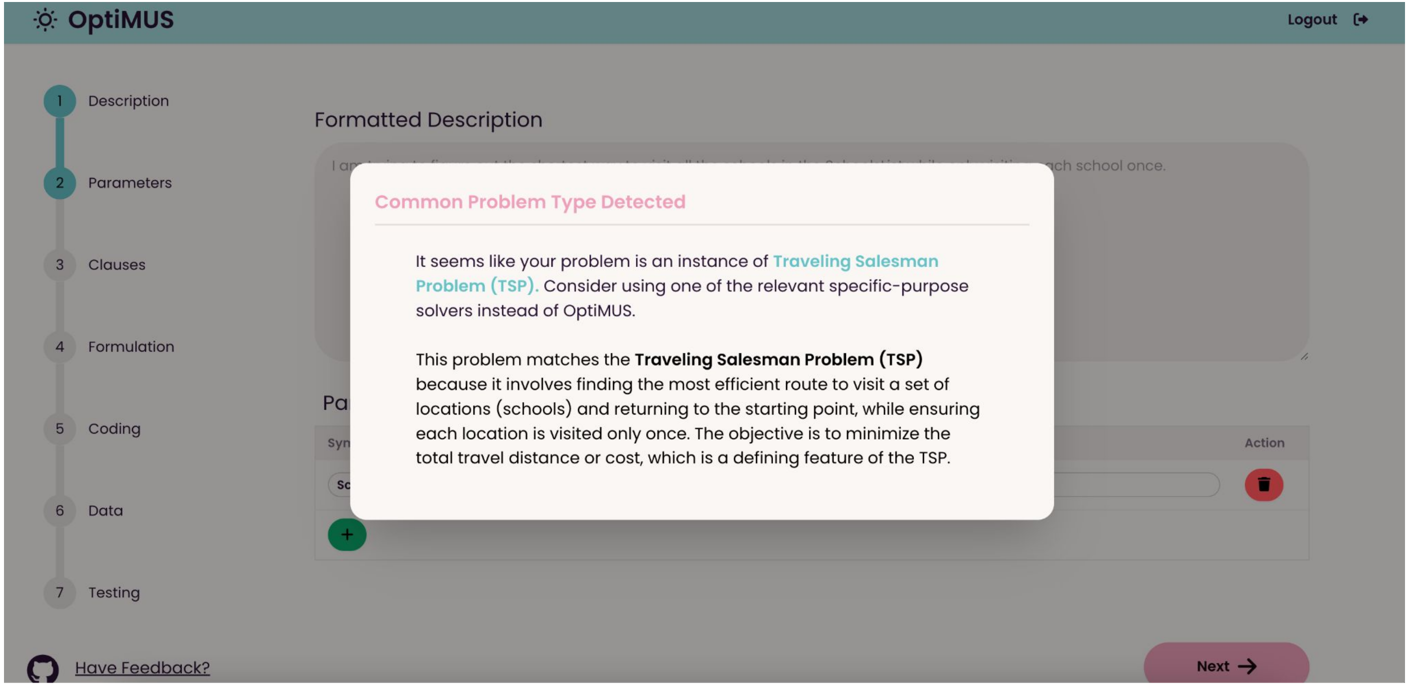
```
Is the value of DiskDriveAvailability known
or not? based on that, is it a parameter or
a variable?
```

```
The value of DiskDriveAvailability is not known
with certainty, as the description mentions that
the supply of disk drives is uncertain, with an
estimated minimum and maximum availability.
Instead of defining DiskDriveAvailability as a
parameter, we must define MinDiskDriveAvailability
and MaxDiskDriveAvailability as parameters.
```

Figure 10 **OptiMUS-0.3 can fix parameter identification errors when prompted *"Is the value of P known or not?"***

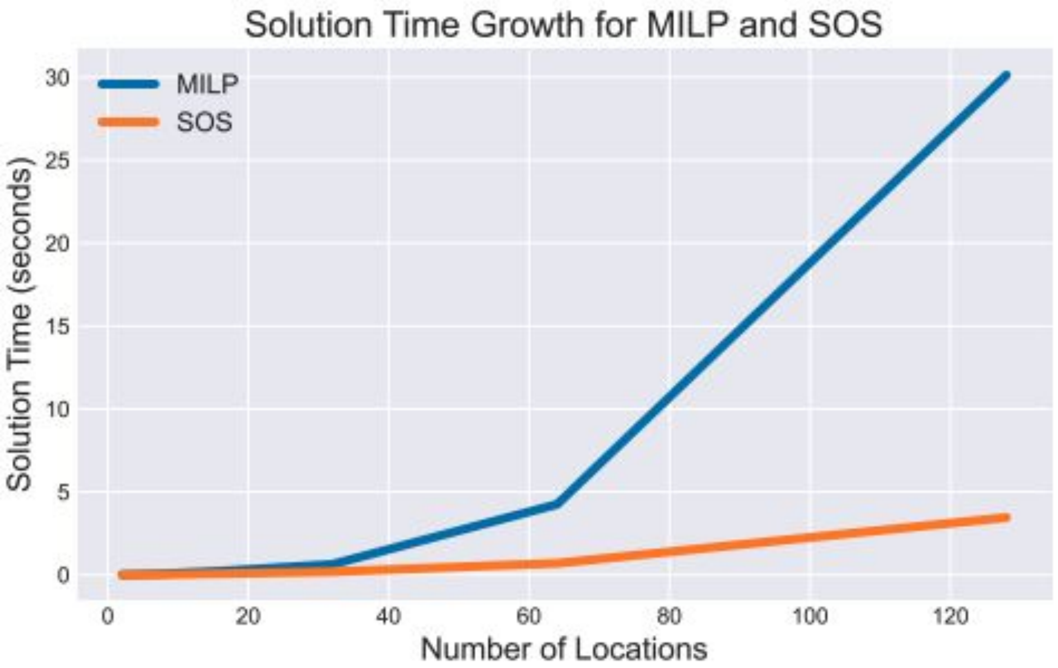
Identifying Special Problems

OptiMUS maintains a pool of well-studied problems with specialized solvers.



Case Study: SOS Constraints

Identifying SOS constraints in facility location problems can accelerate solve times.



Experiments: Datasets

A ton of work over the past few years has focused on generating good *datasets* for model formulation (i.e., with natural language descriptions + final answer).

Dataset	Description Length	Instances (#MILP)	Multi-dimensional Parameters
NL4Opt	518.0 ± 110.7	1101 (0)	×
ComplexOR	497.1 ± 247.5	37 (12)	✓
NLP4LP Easy (Ours)	507.2 ± 102.6	287 (0)	✓
NLP4LP Hard (Ours)	912.3 ± 498.2	68 (18)	✓

Note:

- Many of these problems are still extremely toy (something we would give an undergrad)!
- Not all datasets are *correct**, contain comprehensive elements (e.g., code)

***Toward a trustworthy optimization modeling agent via verifiable synthetic data generation**
Lima, Hwang, Phan, Klein, Liu, & Yeo. *arXiv Preprint*.

Experiments: Results

Takeaways: Decomposition frameworks out-perform LLMs alone (especially with cheaper models).

	LLM	NL4OPT	NLP4LP	IndustryOR
Methods based on direct prompting				
Standard	GPT-4o	47.3%	33.2%	28.0%
Standard	o1	> 95%	68.8%	44.0%
Reflexion	GPT-4o	53.0%	42.6%	–
Methods based on fine-tuning LLMs				
LLMOPT	Qwen1.5-14B	93.0%*	83.8%*	46.0%*
ORLM	Deepseek-Math	86.5%*	72.9%*	38.0%*
Methods based on agentic frameworks				
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%

Execution Accuracy

#s in the table correspond to fraction of instances that run and have the same optimal value.

Experiments: Results

Takeaways: Debugging and error correction help a lot!

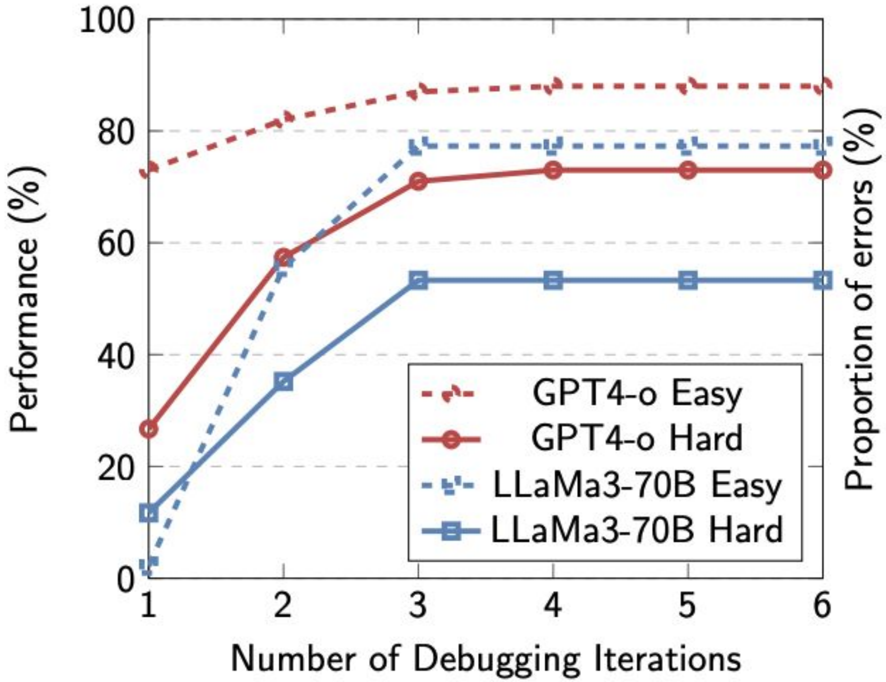
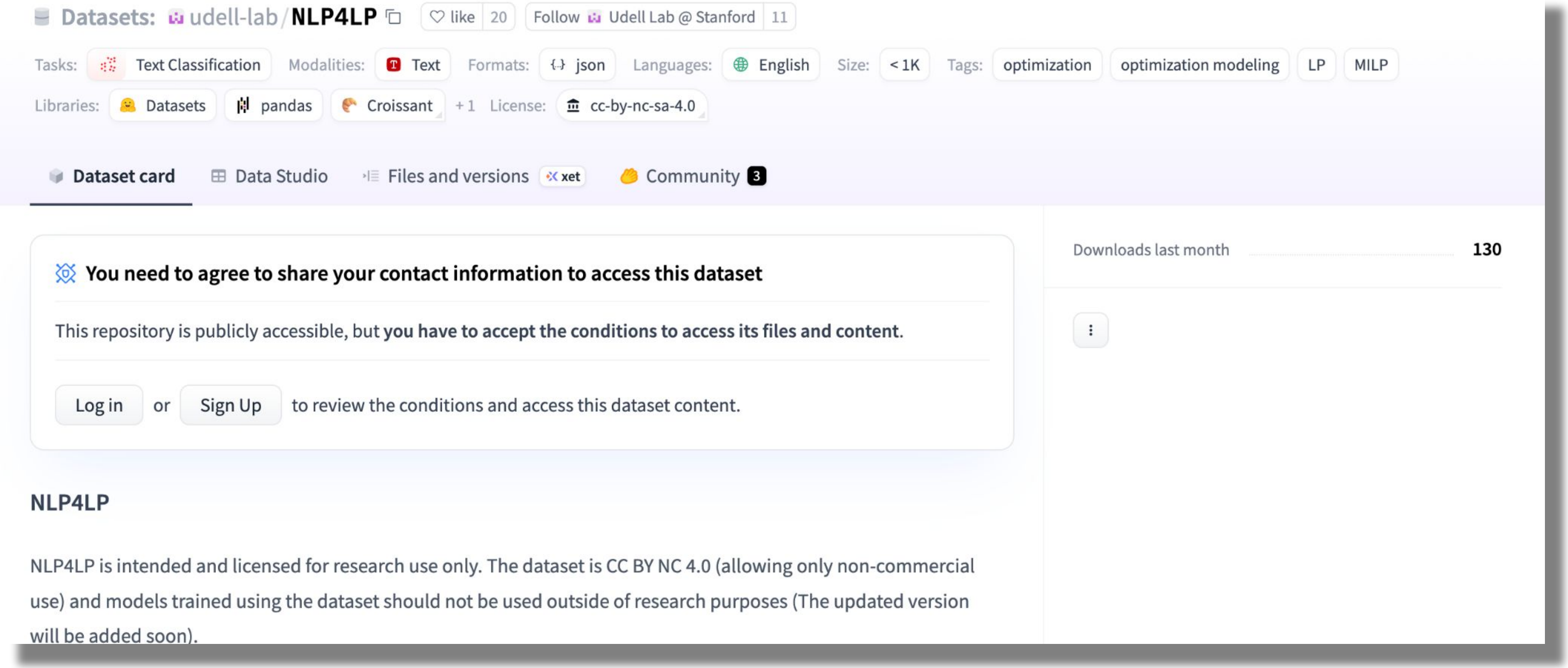


Table 3 Ablation studies on OptiMUS-0.3

	NL4OPT	NLP4LP
Importance of Different Components		
w/o Debugging	73.2%	26.7%
w/o Extraction EC	86.7%	60.5%
w/o Modeling EC	83.8%	65.7%
w/o LLM Feedback	86.6%	68.4%
OptiMUS-0.3 (GPT-4o)	**86.6%**	**73.7%**
Performance with Different LLMs		
LLaMa3.1-70B-Instruct	70.4%	31.5%
GPT-4o	**86.6%**	**73.7%**
o1	–	**80.6%**

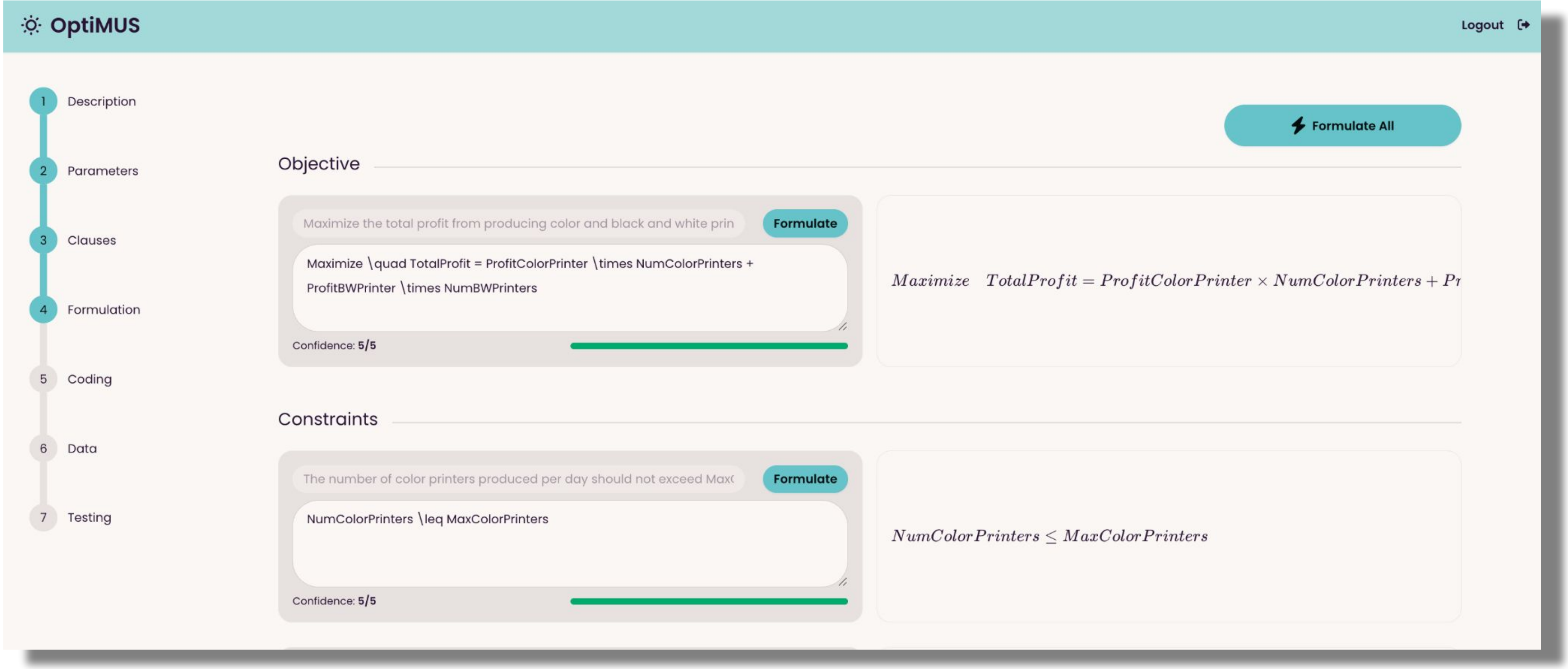
Open-Source Resources

We hope OptiMUS will serve as a framework for supporting future research on auto-formulation:



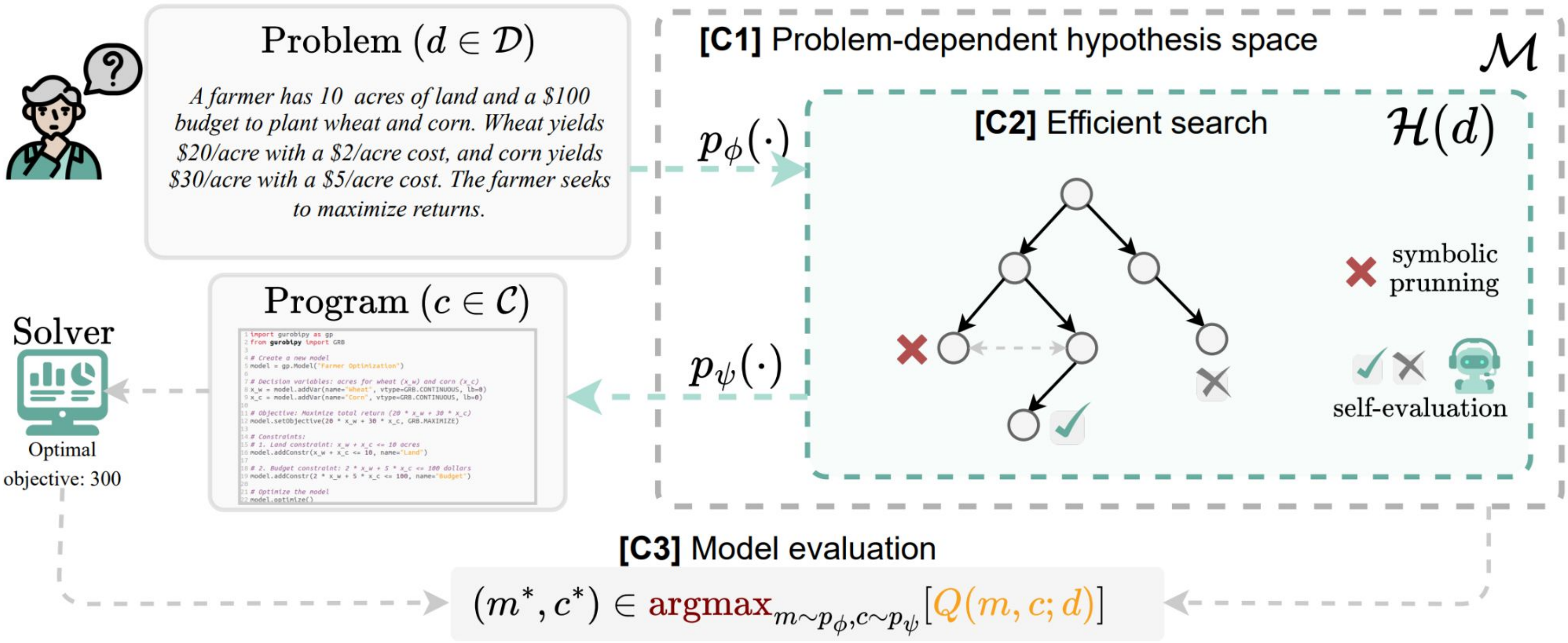
Open-Source Resources

We hope OptiMUS will serve as a framework for supporting future research on auto-formulation:



Advanced Search Strategies

Key Idea: Structured exploration of formulations via Monte-Carlo Tree Search.



Autoformulation of Mathematical Optimization Models Using LLMs. Astorga, et al. *ICML (2025)*

Can we fine-tune an LLM to improve its modeling capabilities?

Challenges with Fine-Tuning for Optimization

Fine-tuning has been an effective tool at specializing LLMs for specific tasks, but:

Insufficient data for fine-tuning
Existing optimization datasets like MIPLIB or NL4OPT are small-scale or do not have text data.

Existing test sets are homogeneous
Most benchmarking datasets focus on simpler 'textbook-style' LP questions.

Many datasets have critical quality issues!
See examples in Lima et al. (2025), or Chen et al. (2025)

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

Operations Research

Chenyu Huang, Zhengyang Tang, Shixi Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, Zizhuo Wang

Criteria for Synthetic Data

Comprehensive Coverage
Should cover different applications, modeling techniques, and difficulty.

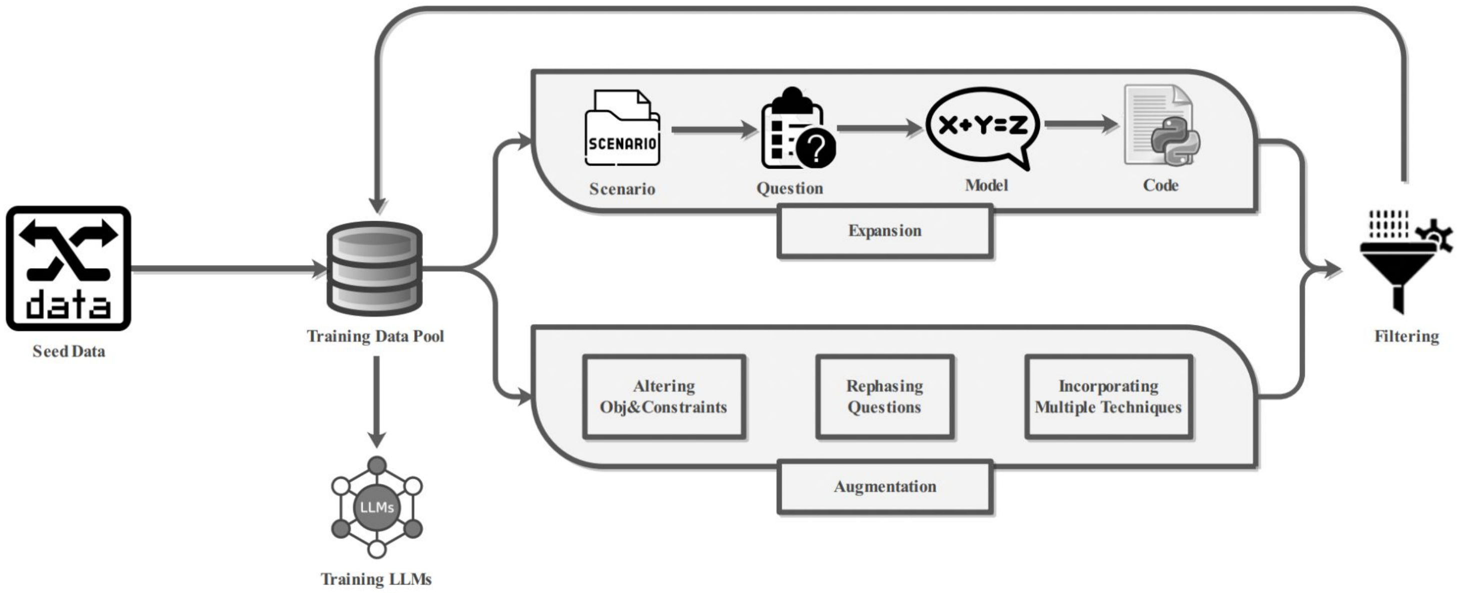
Environment Adaptability
Dataset should include dynamic changes to reflect practice.

Linguistic Diversity
Dataset should reflect variability in how to phrase a problem.

Technique Variability
There are different ways of modeling the same problem!

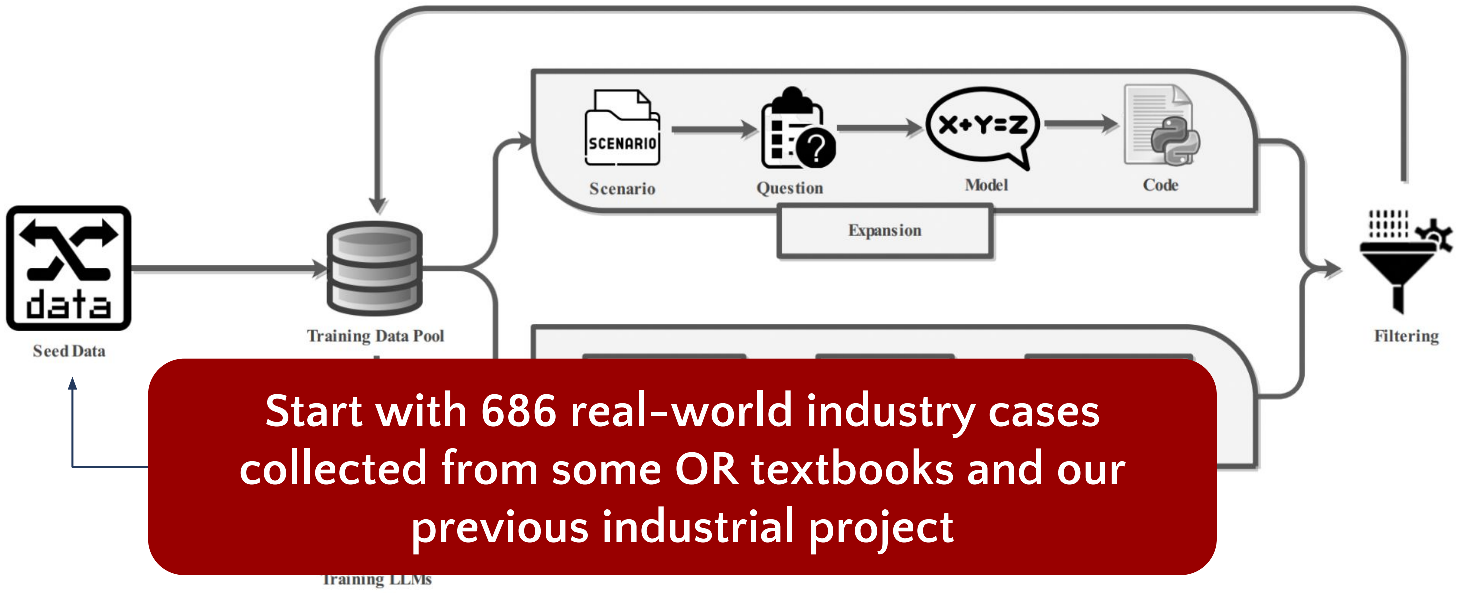
OR-Instruct: A Framework for Synthetic Data Gen.

OR-Instruct employs two key strategies (augmentation + expansion) to create a dataset for fine-tuning.



OR-Instruct: A Framework for Synthetic Data Gen.

OR-Instruct employs two key strategies (augmentation + expansion) to create a dataset for fine-tuning.

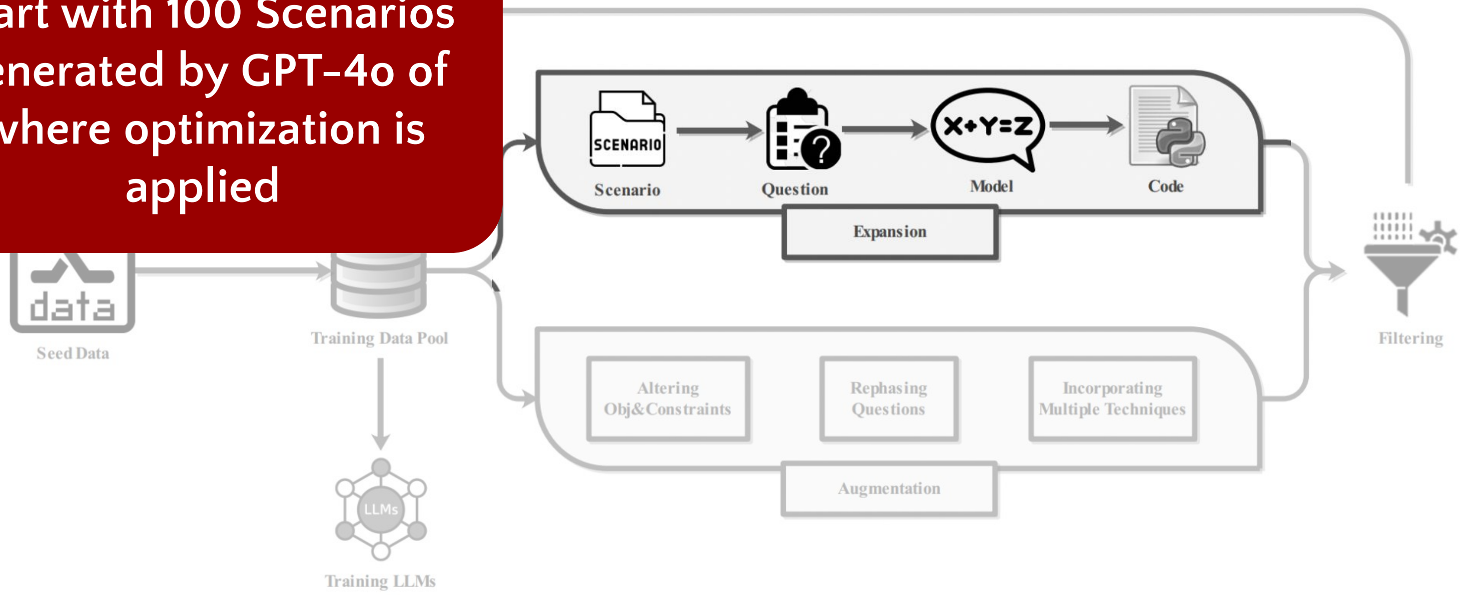


Start with 686 real-world industry cases collected from some OR textbooks and our previous industrial project

Strategy 1: Expansion

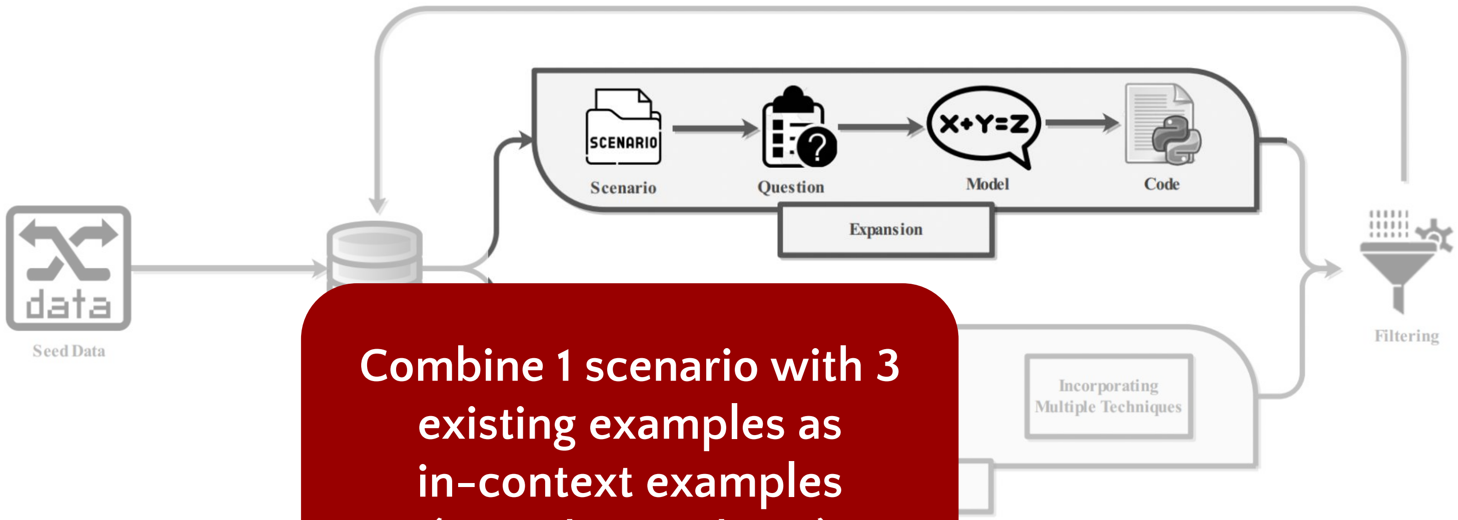
The first strategy involves creating new problems via prompting GPT-4o.

Start with 100 Scenarios generated by GPT-4o of where optimization is applied



Strategy 1: Expansion

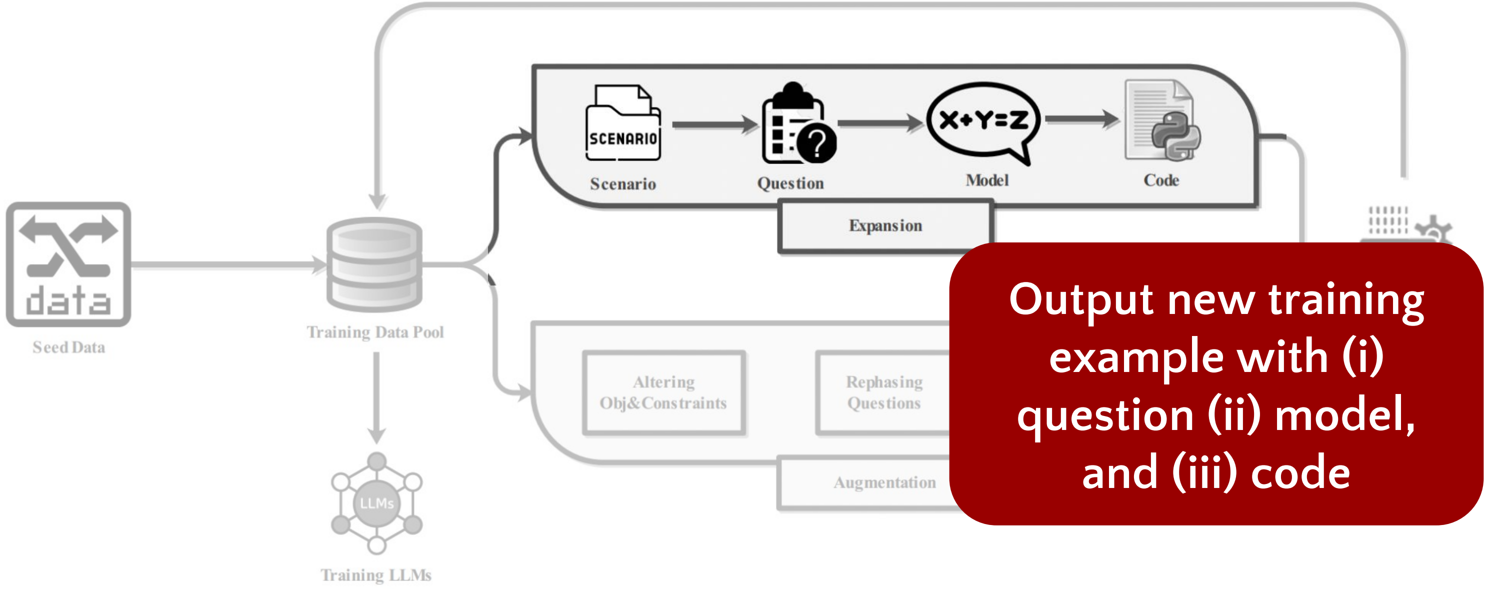
The first strategy involves creating new problems via prompting GPT-4o.



Combine 1 scenario with 3 existing examples as in-context examples (2 seed, 1 synthetic)

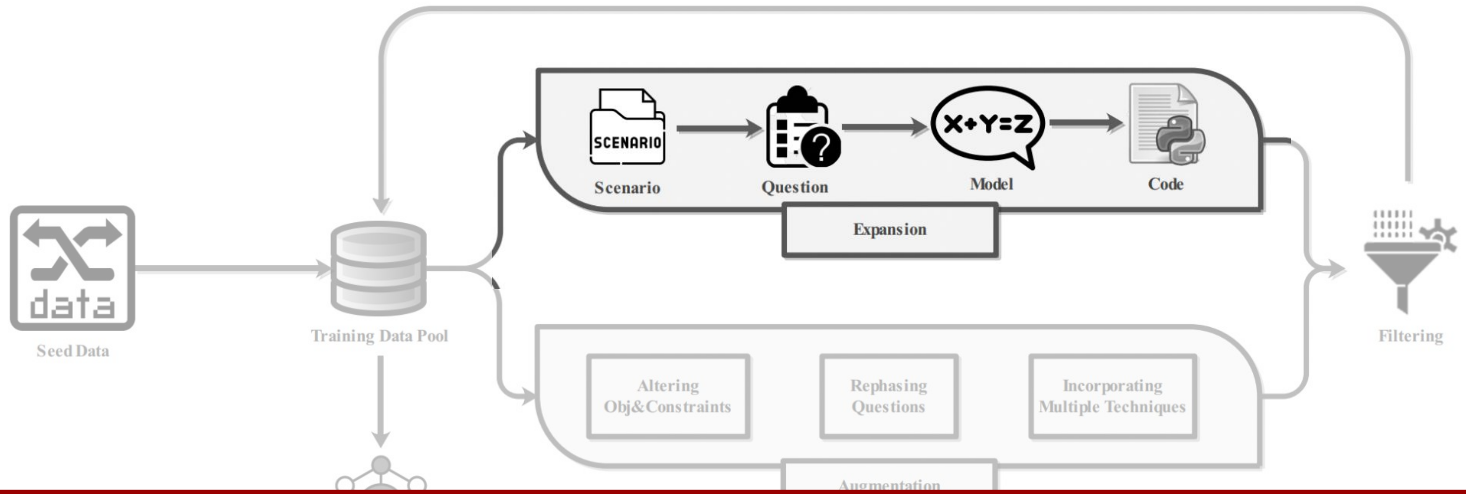
Strategy 1: Expansion

The first strategy involves creating new problems via prompting GPT-4o.



Strategy 1: Expansion

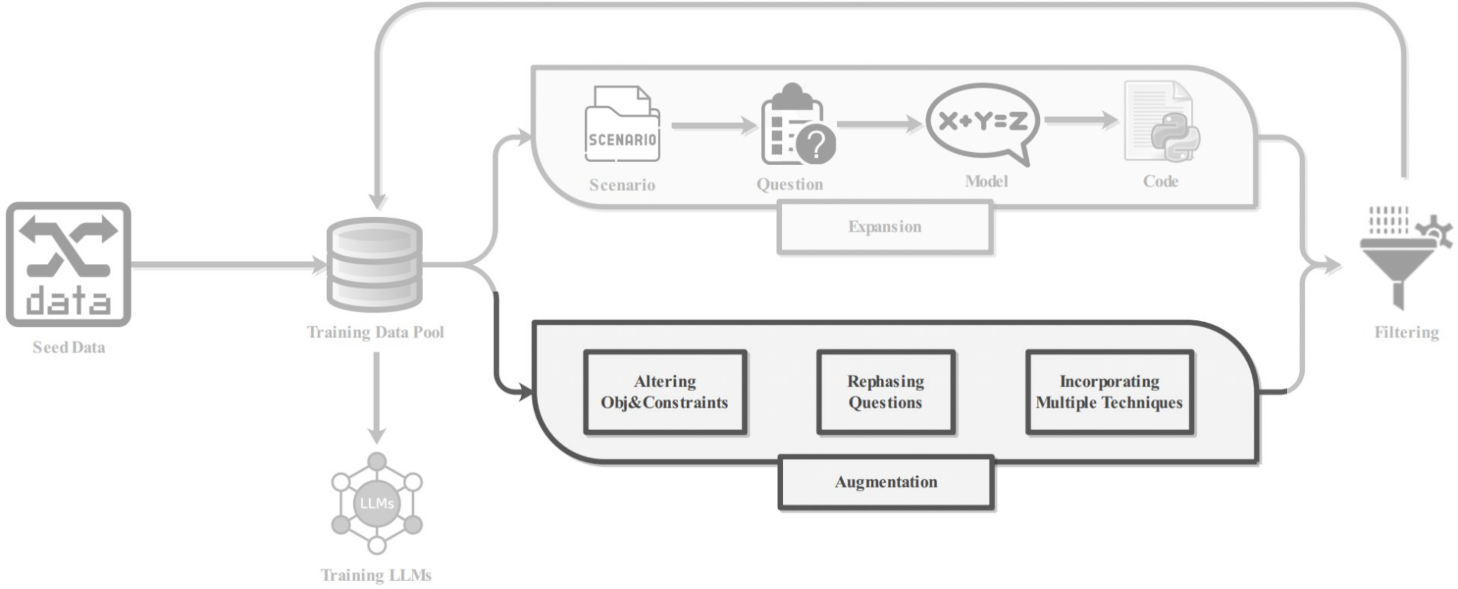
The first strategy involves creating new problems via prompting GPT-4o.



Challenge: Resulting dataset doesn't exhibit diversity in problem difficulty (skews towards easier problems)

Strategy 2: Augmentation

The second strategy involves tweaking existing problems from the seed dataset (can include hard problems!)



Altering Objective and Constraints

Prompts GPT-4o to add new constraints or alter the objective:

Altering Objectives and Constraints for Requirement 2

Original:

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

Augmented:

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

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

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

Rephrasing Question

Prompts GPT-4o to rephrase problems to promote linguistic diversity:

Rephrasing Questions for Requirement 3

Original:

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

Augmented:

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

Incorporate Multiple Modeling Techniques

Prompts GPT-4o to use different modeling techniques for the same problem:

Incorporating Multiple Modeling Techniques for Requirement 4

Original:

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

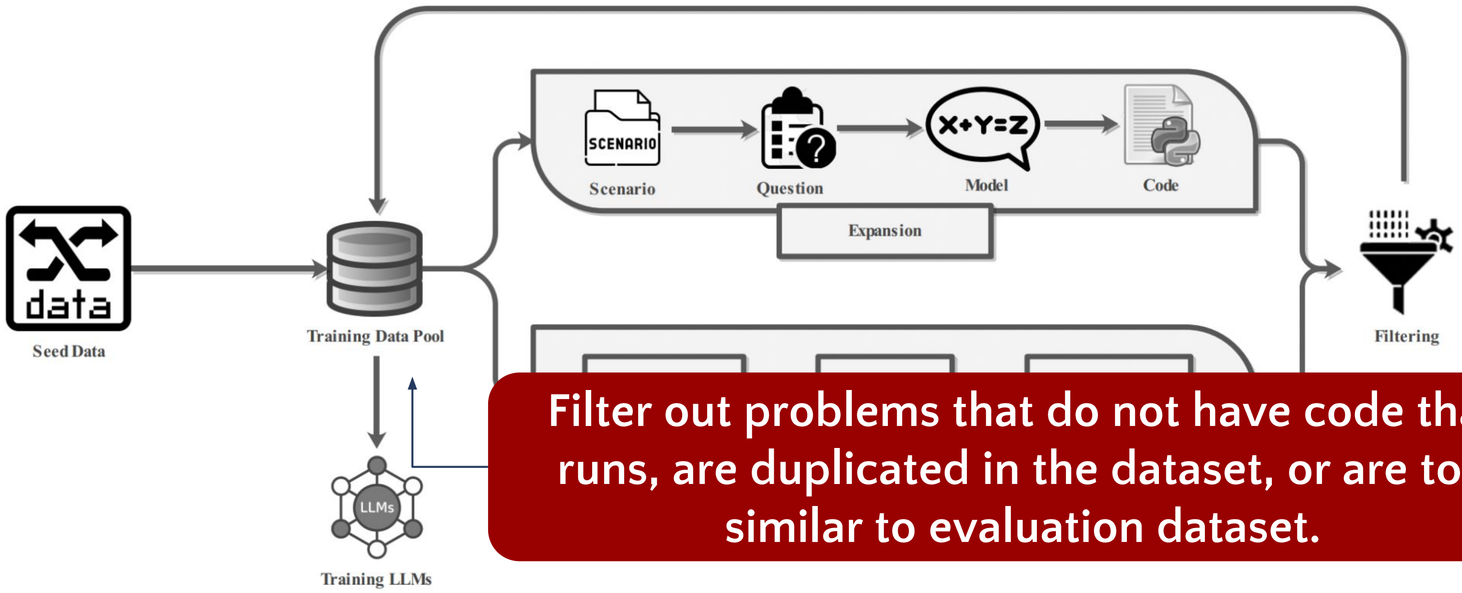
Augmented:

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

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

OR-Instruct: A Framework for Synthetic Data Gen.

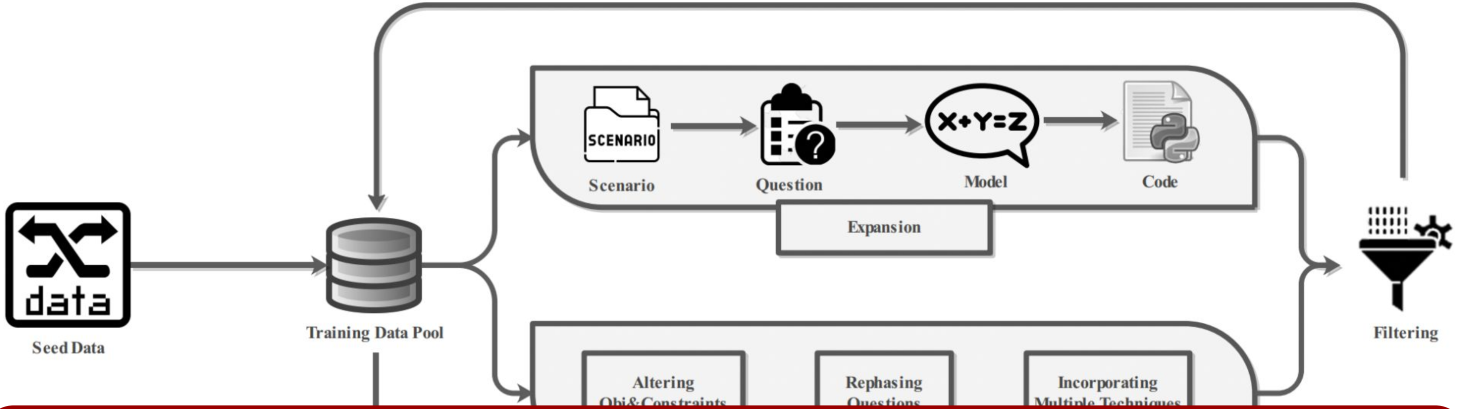
OR-Instruct employs two key strategies (augmentation + expansion) to create a dataset for fine-tuning.



Filter out problems that do not have code that runs, are duplicated in the dataset, or are too similar to evaluation dataset.

OR-Instruct: A Framework for Synthetic Data Gen.

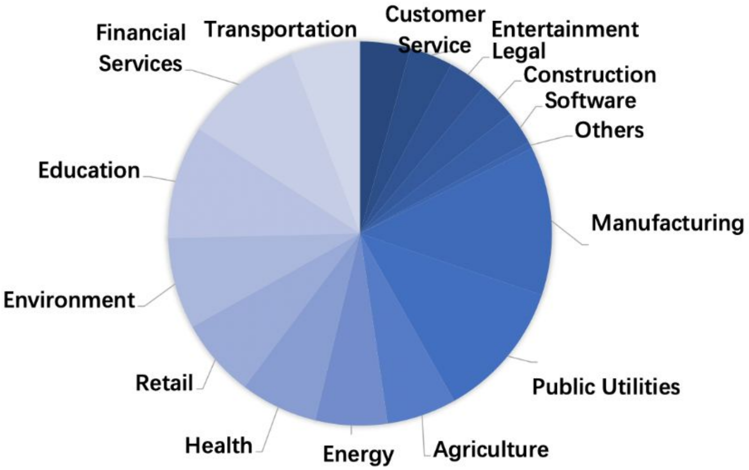
OR-Instruct employs two key strategies (augmentation + expansion) to create a dataset for fine-tuning.



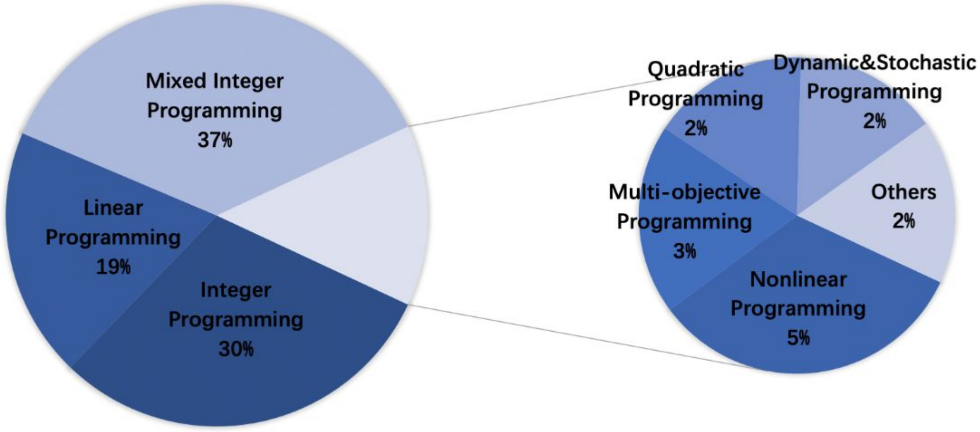
Challenge: Correctness of synthetic data is between 70-75%!

Data Generation Results

OR-Instruct run with just 686 seed cases can generate 32K+ diverse optimization problems:



(a) Distribution of industies



(b) Question type

Experiments: Fine-tuning

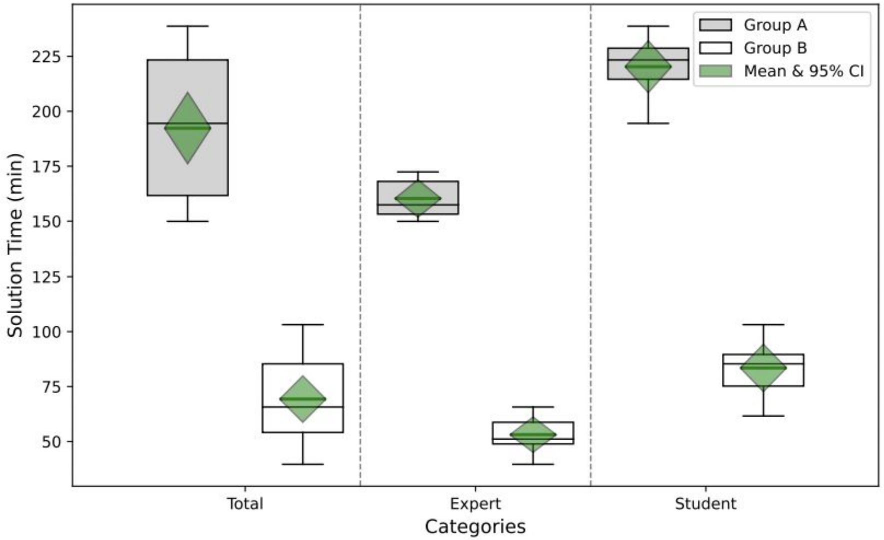
Method/Model	Size	NL4OPT	MAMO EasyLP	MAMO ComplexLP	IndustryOR	Micro Avg	Macro Avg
Methods based on PLMs							
tag-BART	140/400M	47.9%*	-	-	-	-	-
Methods based on GPT-3.5							
Standard	Unknown	42.4%*	-	-	-	-	-
Reflexion	Unknown	50.7%*	-	-	-	-	-
Chain-of-Experts	Unknown	58.9%*	-	-	-	-	-
Methods based on GPT-4							
Standard	Unknown	47.3%*	66.5%*	14.6%*	28.0%	50.2%	39.1%
Reflexion	Unknown	53.0%*	-	-	-	-	-
Chain-of-Experts	Unknown	64.2%*	-	-	-	-	-
OptiMUS	Unknown	78.8%*	-	-	-	-	-
Standard prompting based on open-source LLMs							
Llama-3.1-Instruct	405B	38.7%	35.1%	20.8%	13.0%	31.5%	26.9%
DeepSeek-V2-Chat	236B	66.5%	60.5%	32.7%	16.0%	53.1%	43.9%
Qwen2-Instruct	72B	72.6%	79.9%	29.0%	18.0%	64.4%	49.8%
DeepSeek-R1-Distill	32B	80.4%	69.1%	**45.4%**	33.0%	64.8%	56.9%
Mistral-Nemo	12B	14.6%	19.4%	3.7%	7.0%	14.6%	11.1%
ORLMs based on open-source LLMs							
ORLM-Mistral	7B	84.4%	81.4%	32.0%	27.0%	68.8%	56.2%
ORLM-Deepseek-Math	7B	**86.5%**	82.2%	37.9%	33.0%	71.2%	59.9%
ORLM-LLaMA-3	8B	85.7%	82.3%	37.4%	**38.0%**	71.4%	**60.8%**
ORLM-Qwen2.5	7B	86.1%	**85.2%**	44.1%	25%	**73.7%**	60.1%
Human Evaluation							
Senior Undergraduates	-	80.4%	84.9%	53.1%	44.0%	75.2%	65.6%
Experts	-	94.3%	90.4%	78.9%	76.0%	85.0%	88.2%

Takeaways

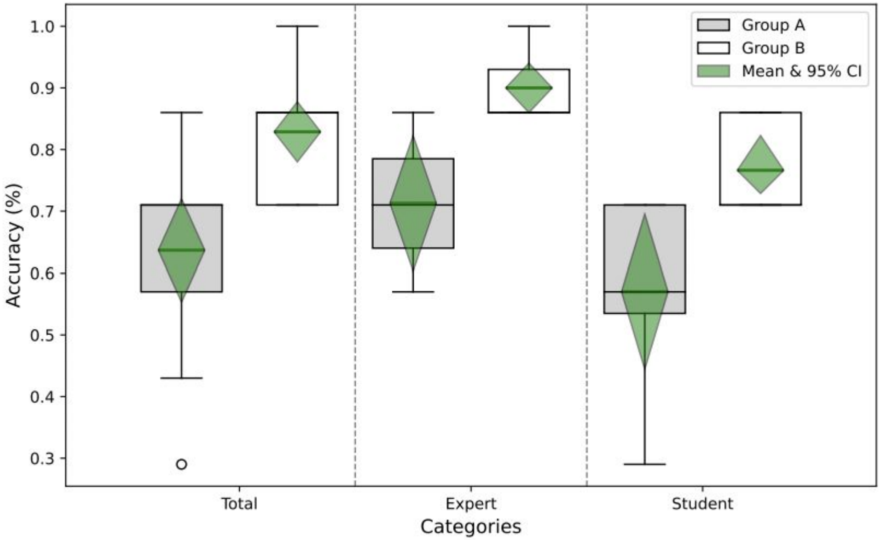
Fine-tuning can improve performance of open-source models!

Experiments: Results

Takeaways: Humans working with ORLM (group B) outperform humans alone (Group A) in terms of both of solution time and accuracy!



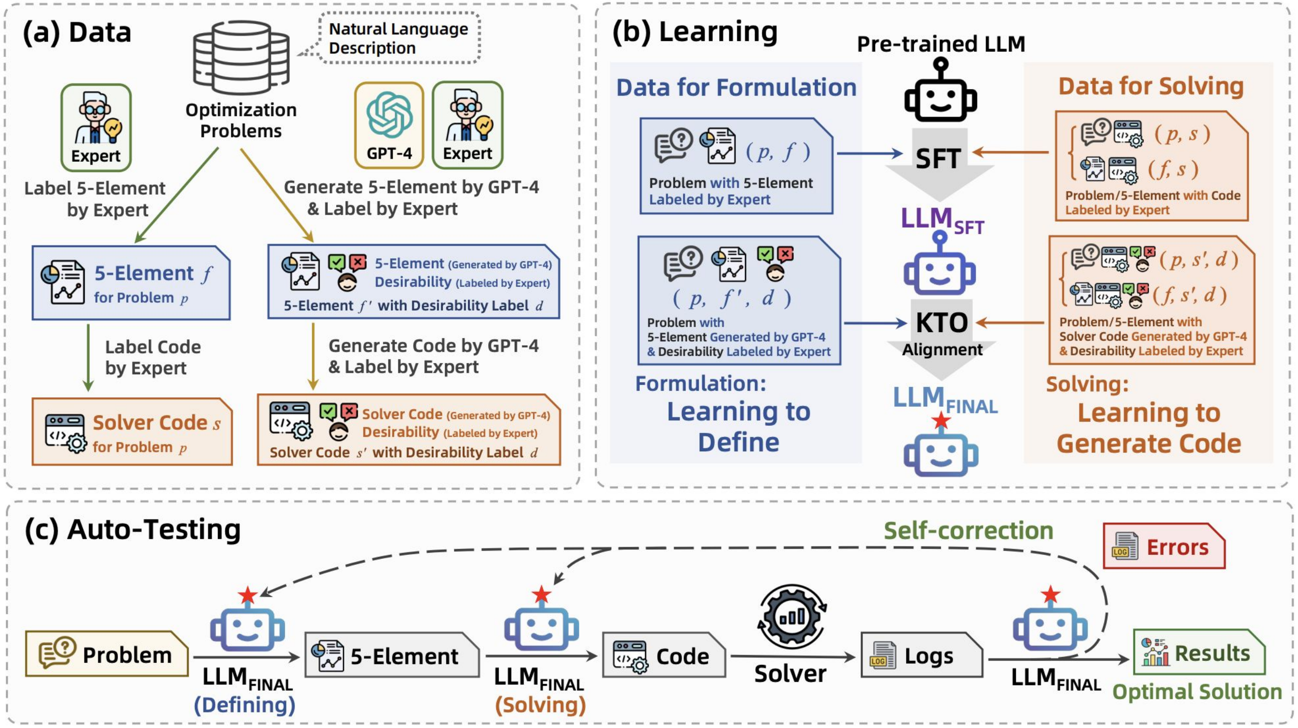
(a) Comparison of solution time



(b) Comparison of accuracy

LLMOPT

Key Idea: Combine structured data + SFT + model alignment to improve performance.

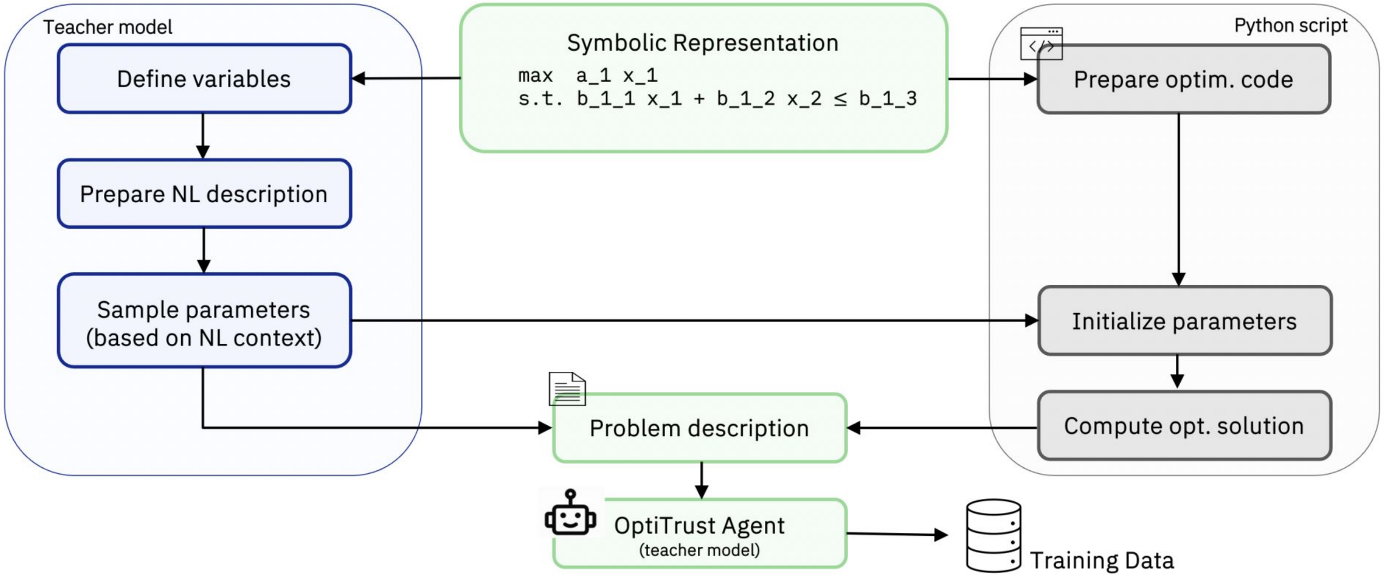


LLMOPT: Learning to define and solve optimization problems from scratch
Jian et al.. *ICLR (2025)*

How can we improve the quality of training data?

Verifiable Synthetic Data Generation

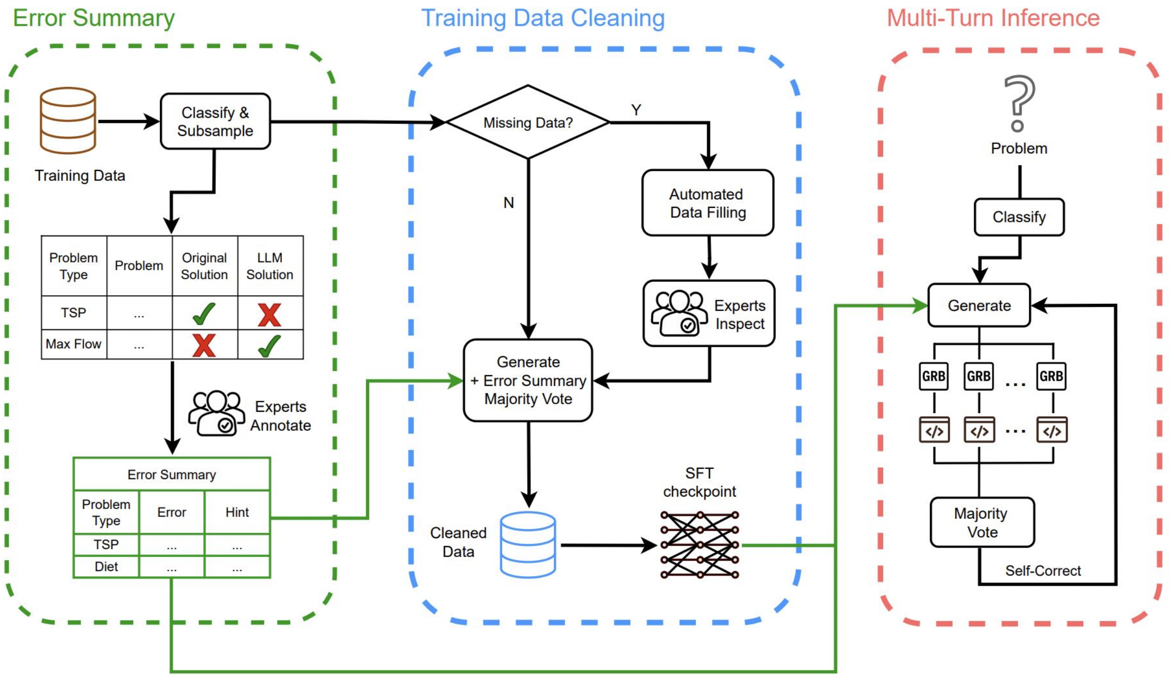
Generate natural language from an existing optimization model so we can verify whether the pipeline produced the right intermediary representations.



***Toward a trustworthy optimization modeling agent via verifiable synthetic data generation**
Lima, Hwang, Phan, Klein, Liu, & Yeo. *arXiv Preprint*.

Data Cleaning via Expert-Guided Prompts

Another approach is to use optimization experts to identify common mistakes and correct the training data directly.

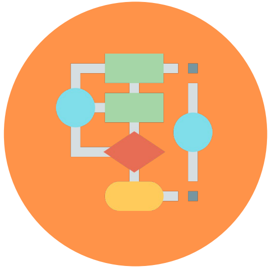


OPTIMIND: Teaching LLMs to Think like Optimization Experts
Chen et al.. *arXiv Preprint*.

Open Questions

There's been exciting progress on auto-formulation, but there's a ton more work to do!

Model Strength
Current work hasn't focused on developing *strong* MILP formulations!



Decomposition Algorithms
Can we move beyond a one-shot formulation?

More data!
ML thrives on data – help us collect more problems in natural language!

How can we check whether two MILP formulations are *equivalent?*

Formulation Equivalence

We are given two (MI)LP optimization problems:

Reference Problem

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \le b \\ & x \in \mathbb{Z}^{n-m} \times \mathbb{R}^m \end{aligned}$$

Optimal Solution: x^*

Feasible Region: $\mathcal{X}$

Generated Problem

$$\begin{aligned} \min \quad & \hat{c}^T x \\ \text{s.t.} \quad & \hat{A}x \le \hat{b} \\ & x \in \mathbb{Z}^{\hat{n}-\hat{m}} \times \mathbb{R}^{\hat{m}} \end{aligned}$$

Optimal Solution: $\hat{x}^*$

Feasible Region: $\hat{\mathcal{X}}$

Goal (inf.): Check that the two formulations solve the ***same optimization problem***

Canonical Accuracy

Take 1: Check whether the two formulations are the same element by element:

Canonical Accuracy: Do both formulations have the same constraint matrix and objective?

$$\frac{\sum_{i,j}(a_{ij}{=}{=}\hat{a}_{ij})+\sum_i(c_i{=}{=}\hat{c}_i)}{n+nm}$$

Any problems?

NL4OPT Competition: Formulating Optimization Problems from Natural Language.
Ramamonjison et al.. *NeurIPS Competition Track 2022.*

Canonical Accuracy

Take 1: Check whether the two formulations are the same element by element:

Canonical Accuracy: Do both formulations have the same constraint matrix and objective?

$$\frac{\sum_{i,j}(a_{ij}==\hat{a}_{ij})+\sum_i(c_i==\hat{c}_i)}{n+nm}$$

Problem 1:

Sensitive to Variable or Constraint Permutations!

NL4OPT Competition: Formulating Optimization Problems from Natural Language.
Ramamonjison et al.. *NeurIPS Competition Track 2022.*

Execution Accuracy

Take 2: Are the final objective values the same?

Execution Accuracy: Do both formulations have the same optimal objective value?

$$c^T x^* = \hat{c}^T \hat{x}^*$$

Any problems?

Execution Accuracy

Problem 2: Sensitive to re-scaling! Re-scaling an optimization problem leads to **semantically identical problems** but breaks metrics like execution accuracy.

You are managing a farm and need to decide how many apples or bananas to produce. You can sell apples for 2 dollars and bananas for 1 dollar. Growing an apple requires 1 unit of land, and 2 units of water. Growing a banana requires 0.8 units of land and 0.5 units of water. You have 10 units of land and 20 units of water how much should you produce of each?

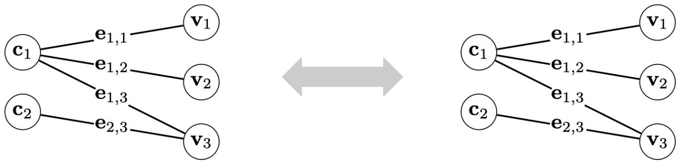
Model 1 *Total Revenue*	Model 2 *Average Revenue per unit land*
$\max \; 2a + b$	$\max \; 0.2a + 0.1b$
$a + 0.8b + s_1 = 10$	$a + 0.8b + s_1 = 10$
$2a + 0.5b + s_2 = 20$	$2a + 0.5b + s_2 = 20$
$a, b, s_1, s_2 \geq 0$	$a, b, s_1, s_2 \geq 0$
$a, b \in \mathbb{Z}$	$a, b \in \mathbb{Z}$

More broadly, the metric is **independent of the actual solution value** (e.g., you have a 50% chance of being correct for every SAT problem!)

Graph Edit Distance

Take 3: Do the formulations have an equivalent graph structure?

Graph Edit Distance: Represent both formulations as bi-partite graphs and then compute the graph edit distance between the two formulations.



Any problems?

Towards human-aligned evaluation for linear programming word problems.
Xing et al. *LREC-COLING (2024)*

Graph Edit Distance

Problem 3: Sensitive to simply strengthening or re-formulating the problem.

Model 1
Base

$$\max \sum_{i \in \mathcal{V}} x_v$$
$$x_i + x_j \leq 1 \quad \forall (i,j) \in E$$
$$x_i \in \{0,1\} \quad \forall i \in \mathcal{V}$$

Model 2
Strengthened

$$\max \sum_{i \in \mathcal{V}} x_v$$
$$x_i + x_j \leq 1 \quad \forall (i,j) \in E$$
$$\sum_{i \in k} x_i \leq 1 \quad \forall k \in \mathcal{K}$$
$$x_i \in \{0,1\} \quad \forall i \in \mathcal{V}$$

Pitfall:

- Models same problem
- **Arbitrarily large difference in number of constraints** (i.e., bad graph edit distance, canonical accuracy)
- **Similar examples for number of variables** (e.g., column generation)

EquivaMap: Leveraging LLMs for Automatic Equivalence Checking of Optimization Formulations

ICML 2025

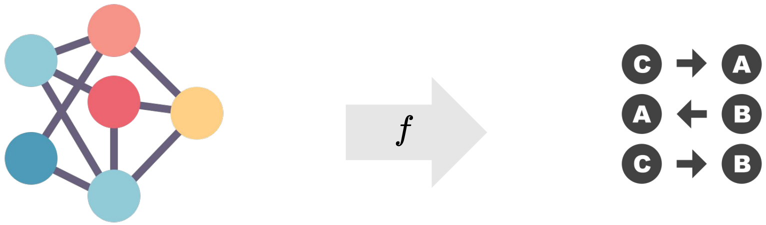
Haotian Zhai, **Connor Lawless**, Ellen Vitercik, Leqi Liu

Karp Reduction

In complexity theory, we can prove that two decision problems are equivalent if we can find a *reduction* between them.

Definition 3.4 (Karp Reduction). Two decision problems $\mathcal{P}$, $\mathcal{Q}$ are said to be equivalent if there exists a function f that maps *arbitrary instances* of $\mathcal{P}$ to $\mathcal{Q}$ such that:

- If p is a yes-instance of $\mathcal{P}$, then $f(p)$ is a yes-instance of $\mathcal{Q}$,
- If p is a no-instance of $\mathcal{P}$, then $f(p)$ is a no-instance of $\mathcal{Q}$, and
- f can be computed in polynomial time.



Decision Problem P

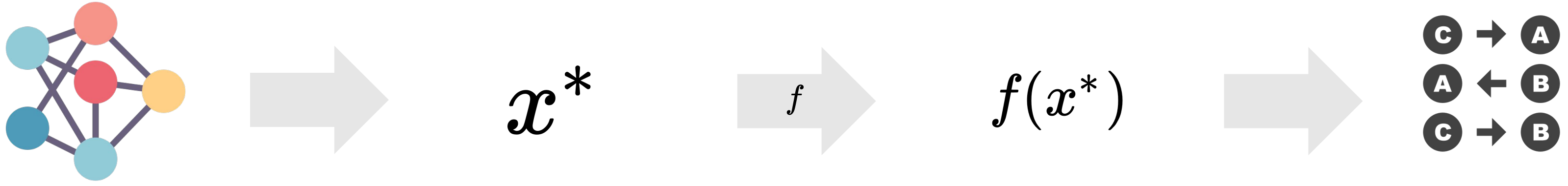
Decision Problem Q

Quasi -Karp Equivalence

Inspired by Karp Reductions we introduce a formal criterion to check whether two MILP formulations are equivalent:

Definition 3.5 (Quasi-Karp Equivalence). Suppose α and α' are two optimization problems over $\mathbb{R}^d$ and $\mathbb{R}^{d'}$, respectively. We say α' is *Quasi-Karp equivalent* to α if there exists an algorithm $\mathcal{A}(\alpha, \alpha')$ that produces a mapping $f : \mathbb{R}^{d'} \to \mathbb{R}^d$ such that:

- If x^* is an optimal solution to α', then $f(x^*)$ is an optimal solution to α,
- f can be computed in polynomial time, and
- $\mathcal{A}(\alpha, \alpha')$ runs in polynomial time for all α, α'.



Optimization Problem alpha’

Optimal Solution

Mapped Optimal Solution

Optimal for Problem alpha?

Quasi -Karp Equivalence

Consider a simple example where *f* is a linear function:

Example: *You are managing a farm and need to decide how many apples or bananas to produce. You can sell apples for 2 dollars and bananas for 1 dollar. Growing an apple requires 1 unit of land, and 2 units of water. Growing a banana requires 0.8 units of land and 0.5 units of water. You have 10 units of land and 20 units of water how much should you produce of each?*

Model 1
Total Land

$$
\begin{aligned}
\max \quad & 2a + b \\
& a + 0.8b + s_1 = 10 \\
& 2a + 0.5b + s_2 = 20 \\
& a, b, s_1, s_2 \geq 0 \\
& a, b \in \mathbb{Z}
\end{aligned}
$$

Model 2
Fraction of Land

$$
\begin{aligned}
\max \quad & 20x + 12.5y \\
& x + y \leq 1 \\
& 20x + 6.25y \leq 20 \\
& 0 \leq x, y \leq 1
\end{aligned}
$$

$$
\begin{aligned}
x^* &= 1 \\
y^* &= 0
\end{aligned}
$$

Mapping from 2 to 1

$$
\begin{aligned}
a &= 10x \\
b &= 12.5y \\
s_1 &= 10 - 10x - 10y \\
s_2 &= 20 - 20x - 6.25y
\end{aligned}
$$

$$
\begin{aligned}
a &= 10 \\
b &= 0 \\
s_1 &= 0 \\
s_2 &= 0
\end{aligned}
$$

Quasi -Karp Equivalence

Consider a simple example where *f* is a linear function:

Example: *You are managing a farm and need to decide how many apples or bananas to produce. You can sell apples for 2 dollars and bananas for 1 dollar. Growing an apple requires 1 unit of land, and 2 units of water. Growing a banana requires 0.8 units of land and 0.5 units of water. You have 10 units of land and 20 units of water how much should you produce of each?*

Model 1
Total Land

$$
\begin{aligned}
\max \quad & 2a + b \\
& a + 0.8b + s_1 = 10 \\
& 2a + 0.5b + s_2 = 20 \\
& a, b, s_1, s_2 \geq 0 \\
& a, b \in \mathbb{Z}
\end{aligned}
$$

Model 2
Fraction of Land

$$
\begin{aligned}
\max \quad & 20x + 12.5y \\
& x + y \leq 1 \\
& 20x + 6.25y \leq 20 \\
& 0 \leq x, y \leq 1
\end{aligned}
$$

Feasible: Yes
Optimality Gap: 0

x
$y^* = 0$

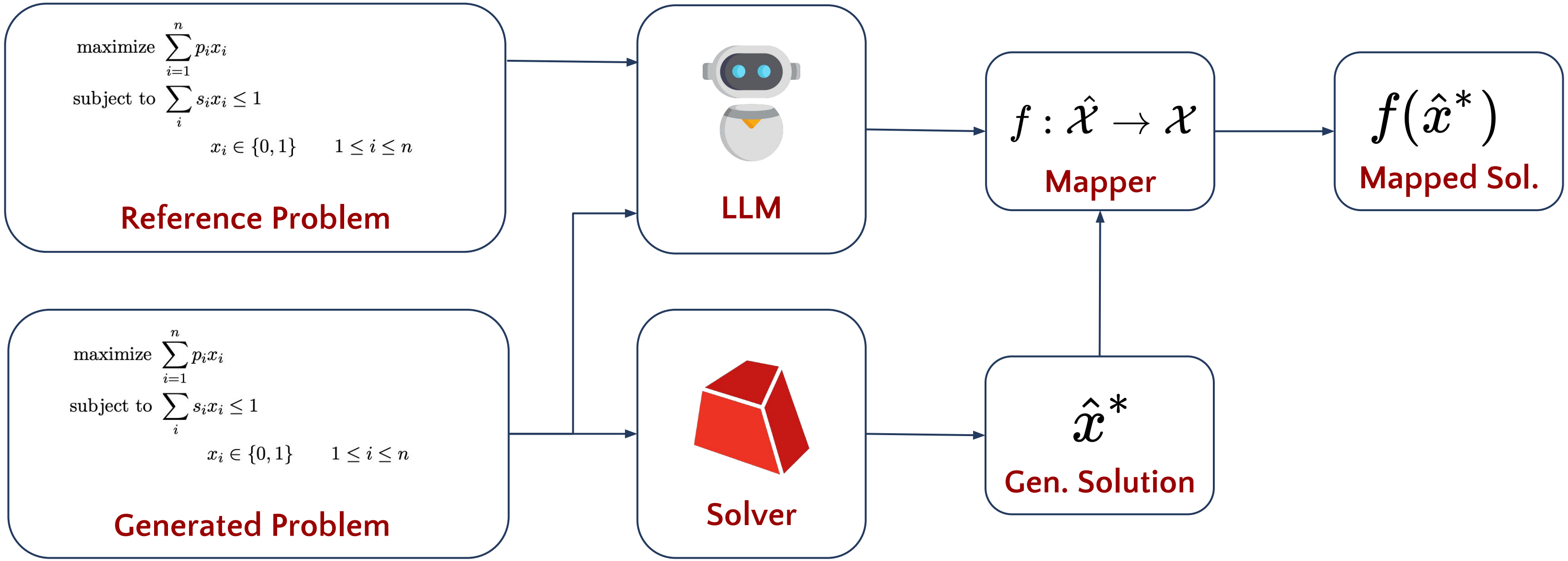
Mapping from 2 to 1

$$
\begin{aligned}
a &= 10x \\
b &= 12.5y \\
s_1 &= 10 - 10x - 10y \\
s_2 &= 20 - 20x - 6.25y
\end{aligned}
$$

$$
\begin{aligned}
a &= 10 \\
b &= 0 \\
s_1 &= 0 \\
s_2 &= 0
\end{aligned}
$$

EquivaMap

Key Idea: Use a LLM to generate the mapping function $f : \hat{\mathcal{X}} \to \mathcal{X}$



Why is this reasonable?

Huh? We're using a LLM to check if an LLM can formulate an optimization problem?

Simpler Problem
Mapping between variables is much easier than checking equivalence! Rich natural language information available to help.

Flexible Output Format
While ideally we would want some closed form mapping, for mapping an optimal solution we can also generate this mapping in a 'code' space

Verifiable
If we find a mapping, and the mapped solution is optimal and feasible we have verification! We don't need to depend on the LLM being correct *every time*.

Evaluation

Transformation Name	How It Is Transformed	Example (Before/After)	Equivalent?	Size
Substitute Objective Functions	Replace objective function $\min c^\top x$ with an auxiliary variable z, adding new constraint $z = c^\top x$	**Before:** $\min c^\top x$ **After:** $\min z,\ \text{s.t. } z = c^\top x$	Yes	92LP + 140MILP
Add Slack Variables	Transform constraint $g(\mathbf{x}) \leq b$ into $g(\mathbf{x}) + s = b,\ s \geq 0$	**Before:** $x + 2y \leq 5$ **After:** $x + 2y + s = 5,\ s \geq 0$	Yes	59LP + 134MILP
Replace by Base-10 Representation	Express an integer variable N in its decimal expansion	**Before:** $x \leq 10^6$ **After:** $x = \sum_{i=0}^{6} d_i \cdot 10^i,\ 0 \leq d_i \leq 9,\ d_i \in \mathbb{Z}$	Yes	44LP + 123MILP
Add Valid Inequalities	Include cutting planes or valid linear combinations that do not exclude any integer feasible solution	**Before:** $\{x + 2y \leq 3,\ x \leq 1.5\}$ **After:** $\{x + 2y \leq 3,\ x \leq 1.5,\ 2x + 2y \leq 4.5\}$	Yes	92LP + 142MILP
Rescaling	Change units/scales for variables or objectives (e.g., hours to minutes)	**Before:** x (hours) **After:** $60x'$ (minutes)	Yes	60LP + 133MILP
Replace by Linear Combinations	Decompose a variable x into $x = x^+ - x^-$ with $x^+, x^- \geq 0$	**Before:** x **After:** $x^+ - x^-$	Yes	77LP + 115MILP
Random Order	Substitute the original instance with a completely unrelated, randomly chosen instance	**Before:** $\min z,\ \text{s.t. } z = c^\top x$ **After:** $\max y,\ \text{s.t. } y = 3$	No	87LP + 142MILP
Loose Constraints	Delete certain constraints that are tight at the optimum, altering the feasible set	**Before:** $x + 2y \leq 3$ (binding) **After:** remove $x + 2y \leq 3$	No	53LP + 120MILP
Feasibility	Turn both the original and a randomly chosen instance into feasibility problems (replace objectives with 0)	**Before:** $\min 0,\ \text{s.t. } z = c^\top x$ **After:** $\max 0,\ \text{s.t. } y = 3$	No	87LP + 142MILP

We introduce a new dataset with a set of predefined equivalent and nonequivalent formulations.

EquivaMap Results

Takeaway: EquivaMap correctly verifies formulation equivalence across settings where existing heuristics break down

Table 2: Accuracy of equivalence-checking methods on formulations obtained from equivalent and non-equivalent transformations.

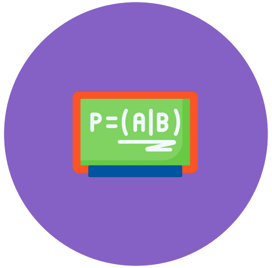
	Canonical Acc.	Execution Acc.	WL-test	naive-LLM	EquivaMap
Equivalent Transformations					
Worst Case	0%	0%	0%	3.3%	**100%**
Substitute Objective Functions	0%	**100%**	0%	91.2%	**100%**
Add Slack Variables	0%	**100%**	0%	36.1%	**100%**
Replace by Base-10 Representation	0%	**100%**	0%	53.1%	**100%**
Add Valid Inequalities	0%	**100%**	0%	3.3%	**100%**
Rescaling	0%	0%	0%	69.9%	**100%**
Replace by Linear Combinations	0%	**100%**	0%	24.4%	**100%**
Non-Equivalent Transformations					
Worst Case	**100%**	0%	**100%**	93.6%	**100%**
Random Order	**100%**	**100%**	**100%**	98.7%	**100%**
Loose Constraints	**100%**	**100%**	**100%**	93.6%	**100%**
Feasibility	**100%**	0%	**100%**	**100%**	**100%**

Open Questions

EquivaMap works great for simple transformations, but there’s a ton more work to do!

Constraint Verification
Some constraints aren’t tight at optimality? How can we verify they’re still implemented correctly?

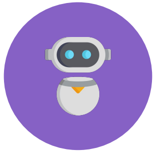
Verification without Labels
What if we don’t have a ground-truth ‘correct’ formulation? Can we *reliably* verify models based on NL?



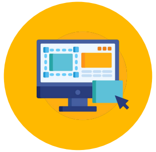
Automated Complexity Proofs
Can we push the same algorithmic ideas to help do automated proofs?

Takeaways

Modeling is a barrier to access optimization tools!
Domain experts often do not have expertise to model problems.



LLMs (with the right framework) can model optimization problems!
LLMs can bridge expertise gaps in modeling CP and MILP problems.



More work to do!
LLMs still struggle to model complex problems and can be difficult to trust…
open-source tools like OptiMUS and ORLM can promote future research!

Thanks! Questions?