

AI and the future of optimization modeling

Madeleine Udell

Management Science and Engineering
Stanford University

Joint work with
Ali AhmadiTeshnizi, Wenzhi Gao, Connor Lawless, Herman Bruborg, and
Shayan Talaei (Stanford)

January 14, 2026

Outline

Challenge

Architecture

Results

Interactive optimization modeling

Teaching LLMs optimization

Safety, ethics, and future directions

Optimization is everywhere



Energy



Routing



Scheduling



Supply chain

- ▶ Optimization improves efficiency throughout the economy
- ▶ ⇒ more productivity, less waste, lower costs, lower carbon, more utility

Optimization is everywhere



Energy



Routing



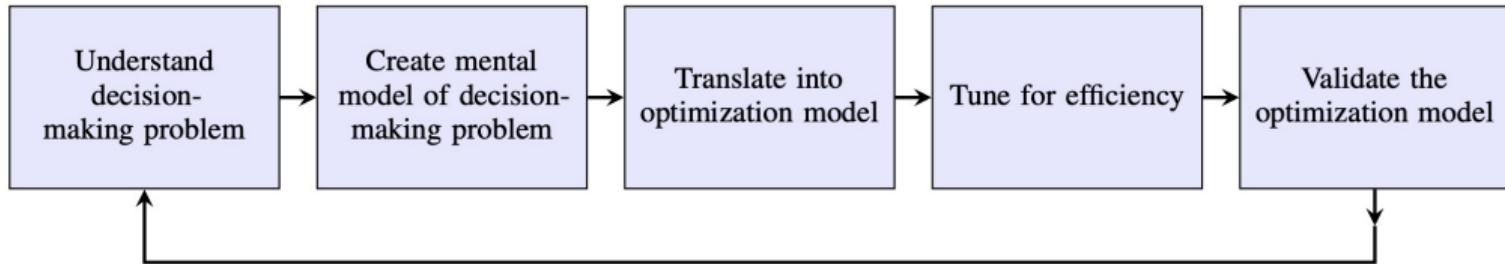
Scheduling



Supply chain

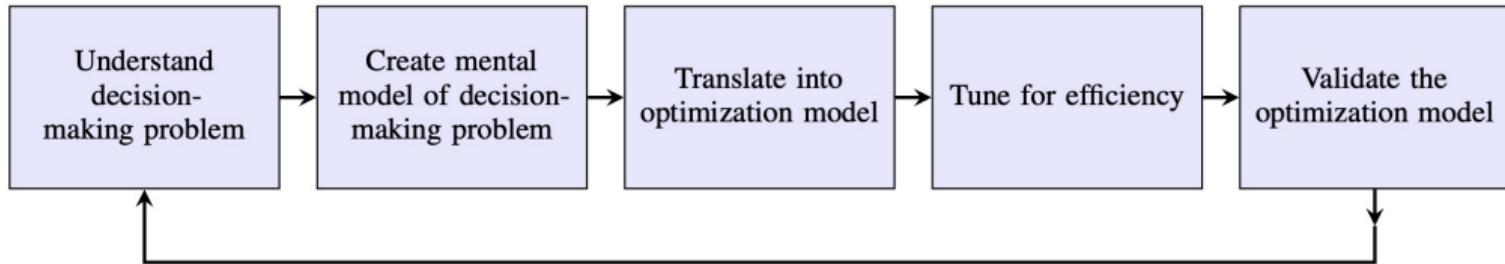
- ▶ Optimization improves efficiency throughout the economy
- ▶ ⇒ more productivity, less waste, lower costs, lower carbon, more utility
- ▶ What limits the use of optimization?

Optimization modeling is the bottleneck



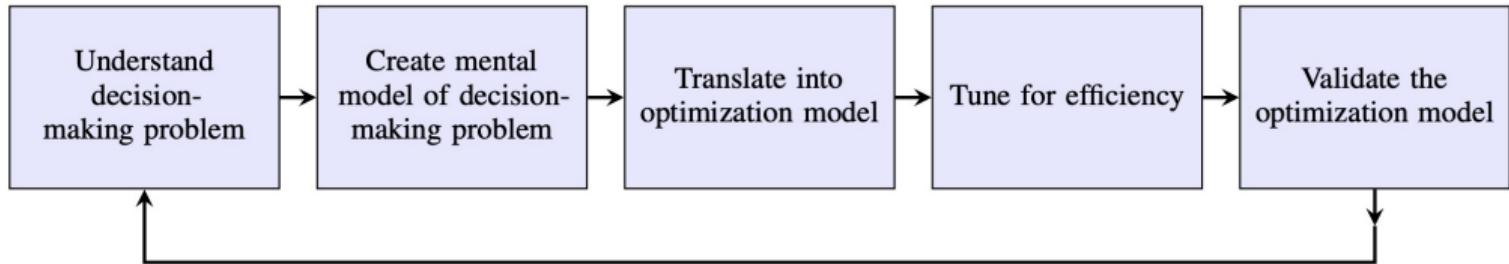
- ▶ problems can be long and complex, eg, this 64 page power systems problem

Optimization modeling is the bottleneck



- ▶ problems can be long and complex, eg, this 64 page power systems problem
- ▶ require expert knowledge to model: among Gurobi's commercial solver users, 81% have advanced degrees, 49% in operations research

Optimization modeling is the bottleneck



- ▶ problems can be long and complex, eg, this 64 page power systems problem
- ▶ require expert knowledge to model: among Gurobi's commercial solver users, 81% have advanced degrees, 49% in operations research
- ▶ why is modeling difficult?
 - ▶ what formulations will be slow or fast to solve?
 - ▶ what backend solver will suit the problem requirements?
 - ▶ what approximations are warranted?
 - ▶ linear/quadratic, discrete/continuous, uncertain predictions, ...

source: Wasserkrug et al., 2024; Gurobi Optimization, 2023

Example problem: pricing

- ▶ A global fashion brand sells articles of clothing in several markets.
- ▶ We have an estimate of how price changes affect sales for each article, assuming constant price elasticity.
- ▶ Price changes must be in multiples of 1 euro.
- ▶ Initial prices for each article are given, and no more than 20% of the prices can change.
- ▶ Each article has a maximum production volume; we cannot sell more than that.
- ▶ Given the sales forecast per article for the next twelve months and past elasticities, the goal is to choose new prices for each article to maximize expected revenue.

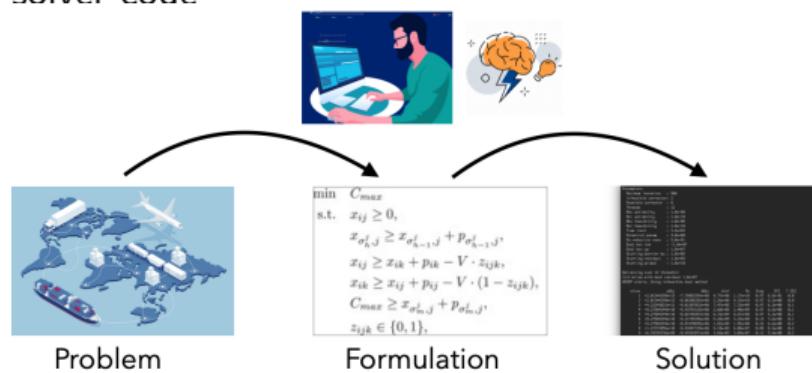
Example model: pricing

```
19 # Define model
20 model = gp.Model('model')
21
22
23 # ===== Define variables =====
24 PriceChange = model.addVars(N, vtype=gp.GRB.CONTINUOUS, name="PriceChange")
25 QuantitySold = model.addVars(N, vtype=gp.GRB.CONTINUOUS, name="QuantitySold")
26 IndicatorFunctionPriceChange = model.addVars(N, vtype=gp.GRB.BINARY, name="IndicatorFunctionPriceChange")
27 Multiplier = model.addVars(N, vtype=gp.GRB.INTEGER, name="Multiplier")
28
29 # ===== Define constraints =====
30
31 # Add constraint - maximum number of articles with price changes
32 model.addConstr(gp.quicksum(IndicatorFunctionPriceChange[i] for i in range(N)) <= N * MaxChangeProportion, name="max_price_change_proportion")
33
34 # Add constraints to ensure quantity sold does not exceed adjusted forecast sales
35 for n in range(N):
36     model.addConstr(QuantitySold[n] <= SalesForecast[n] * (1 - Elasticity[n] * (PriceChange[n] / InitialPrice[n])), name=f"sales_limit_{n}")
37
38 # Add non-negative price constraints
39 for i in range(N):
40     model.addConstr(PriceChange[i] >= -InitialPrice[i], name="non_negative_price")
41
42 # Add constraints for price changes in multiples of MinimumChange
43 for i in range(N):
44     model.addConstr(PriceChange[i] == Multiplier[i] * MinimumChange, name="price_change_multiples")
45
46 # Add constraint to limit price change occurrences
47 max_article_changes = MaxChangeProportion * N
48 model.addConstr(IndicatorFunctionPriceChange.sum() <= max_article_changes, "limit_price_changes")
49
50 # Add maximum production volume constraints for each article
51 for i in range(N):
52     model.addConstr(QuantitySold[i] <= MaxVolume[i], name="max_volume")
53
54 # Add the constraint for forecasted quantity sold of each article adjusted for price change and elasticity of demand
55 for i in range(N):
56     model.addConstr(QuantitySold[i] == SalesForecast[i] * (1 + Elasticity[i] * (PriceChange[i] / InitialPrice[i])), name="demand_elasticity")
57
58 # ===== Define objective =====
59
60 # Set objective
61 model.setObjective(gp.quicksum((InitialPrice[i] + PriceChange[i]) * SalesForecast[i] for i in range(N)), gp.GRB.MAXIMIZE)
62
```

Challenge of modeling an optimization problem

solving a real-life problem = **modeling** + **solving**

- ▶ solvers are extremely reliable
- ▶ but modeling requires expert knowledge
 - ▶ understanding business logic
 - ▶ mathematical modeling
 - ▶ implementing solver code



How to make optimization more accessible? Answer: automate it!

Optimization at scale

What makes an optimization problem “large”?

- ▶ many variables and constraints
- ▶ large data matrices

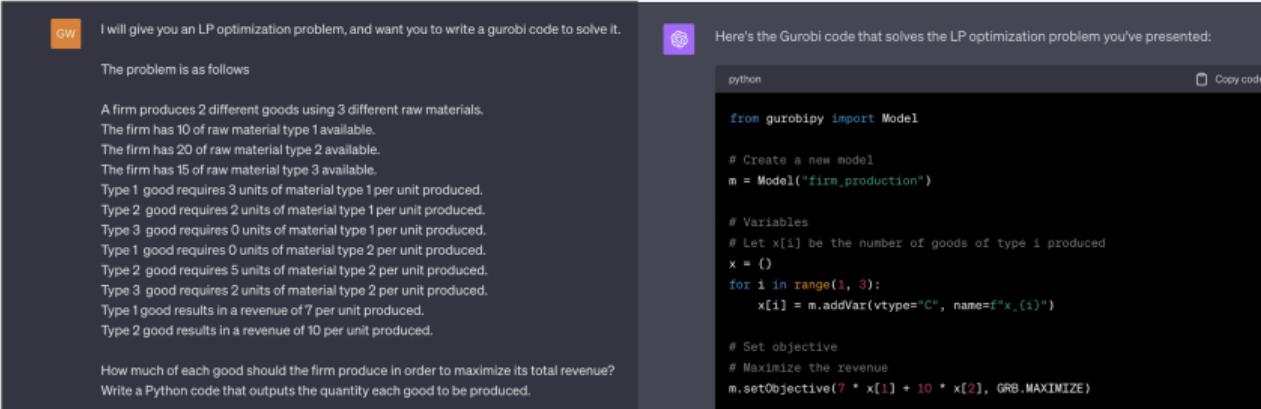
Optimization at scale

What makes an optimization problem “large”?

- ▶ many variables and constraints
- ▶ large data matrices
- ▶ long problem description
- ▶ many stakeholders
- ▶ complex business logic
- ▶ changing requirements
- ▶ solution requires patching together multiple techniques
- ▶ ...

Optimization modeling using LLMs

Attempt 1: just ask ChatGPT to write solver (e.g., gurobipy) code



gw I will give you an LP optimization problem, and want you to write a gurobi code to solve it.

The problem is as follows

A firm produces 2 different goods using 3 different raw materials. The firm has 10 of raw material type 1 available. The firm has 20 of raw material type 2 available. The firm has 15 of raw material type 3 available. Type 1 good requires 3 units of material type 1 per unit produced. Type 2 good requires 2 units of material type 1 per unit produced. Type 3 good requires 0 units of material type 1 per unit produced. Type 1 good requires 0 units of material type 2 per unit produced. Type 2 good requires 5 units of material type 2 per unit produced. Type 3 good requires 2 units of material type 2 per unit produced. Type 1 good results in a revenue of 7 per unit produced. Type 2 good results in a revenue of 10 per unit produced.

How much of each good should the firm produce in order to maximize its total revenue? Write a Python code that outputs the quantity each good to be produced.

Here's the Gurobi code that solves the LP optimization problem you've presented:

```
python
```

```
from gurobipy import Model

# Create a new model
m = Model("firm_production")

# Variables
# Let x[i] be the number of goods of type i produced
x = {}
for i in range(1, 3):
    x[i] = m.addVar(vtype="C", name=f"x_{i}")

# Set objective
# Maximize the revenue
m.setObjective(7 * x[1] + 10 * x[2], GRB.MAXIMIZE)
```

Copy code

- ▶ (+) easy
- ▶ (−) unreliable
- ▶ (−) doesn't scale to large problems

LLMs for everything?

unreliability is a problem for LLMs in many domains.

- ▶ code
- ▶ information retrieval and summarization
- ▶ mathematics

LLMs for everything?

unreliability is a problem for LLMs in many domains.

- ▶ code
- ▶ information retrieval and summarization
- ▶ mathematics

key attribute: can the output be reliably checked by

- ▶ traditional code?
- ▶ an LLM?
- ▶ a human?

exploit the unique attributes of optimization to reduce errors!

Outline

Challenge

Architecture

Results

Interactive optimization modeling

Teaching LLMs optimization

Safety, ethics, and future directions

How to use LLMs for optimization?

- ▶ ask the LLM for a solution
 - ▶ NL4opt competition (Ramamujison et al, 2022, 2023)
 - ▶ LLMs as optimizers (Yang et al., 2023)

How to use LLMs for optimization?

- ▶ ask the LLM for a solution
 - ▶ NL4opt competition (Ramamujison et al, 2022, 2023)
 - ▶ LLMs as optimizers (Yang et al., 2023)
- ▶ ask the LLM to write a solver (“Code generation”)
 - ▶ FunSearch (Romera-Paredes et al., 2024)
 - ▶ Evolution frameworks: AlphaEvolve, OpenEvolve, ShinkaEvolve, DeepEvolve, ...

How to use LLMs for optimization?

- ▶ ask the LLM for a solution
 - ▶ NL4opt competition (Ramamujison et al, 2022, 2023)
 - ▶ LLMs as optimizers (Yang et al., 2023)
- ▶ ask the LLM to write a solver (“Code generation”)
 - ▶ FunSearch (Romera-Paredes et al., 2024)
 - ▶ Evolution frameworks: AlphaEvolve, OpenEvolve, ShinkaEvolve, DeepEvolve, ...
- ▶ ask the LLM to call a solver (“Tool use”)
 - ▶ OptiMUS: structured pipeline with error correction
 - ▶ chain of experts: agentic system (Xiao et al., 2023)
 - ▶ ORLM: semisynthetic data + finetuning (Tang et al., 2024)
 - ▶ LLMOPT: structured pipeline + finetuning (Jiang et al., 2024)

How to use LLMs for optimization?

- ▶ ask the LLM for a solution
 - ▶ NL4opt competition (Ramamujison et al, 2022, 2023)
 - ▶ LLMs as optimizers (Yang et al., 2023)
- ▶ ask the LLM to write a solver (“Code generation”)
 - ▶ FunSearch (Romera-Paredes et al., 2024)
 - ▶ Evolution frameworks: AlphaEvolve, OpenEvolve, ShinkaEvolve, ...
- ▶ ask the LLM to call a solver (“Tool use”)
 - ▶ OptiMUS: structured pipeline with error correction
 - ▶ chain of experts: agentic system (Xiao et al., 2023)
 - ▶ ORLM: semisynthetic data + finetuning (Tang et al., 2024)
 - ▶ LLMOPT: structured pipeline + finetuning (Jiang et al., 2024)
- ▶ chatbots for optimization (“Copilot”)
 - ▶ fixing infeasibility (Chen, Constante-Flores, & Li, 2023)
 - ▶ what-if analysis (Li et al., 2023)
 - ▶ modeling chatbot (Alibaba Cloud, 2022)

Challenges and solutions

challenges:

Challenges and solutions

challenges:

- ▶ long problem descriptions

Challenges and solutions

challenges:

- ▶ long problem descriptions
- ▶ big data matrices

Challenges and solutions

challenges:

- ▶ long problem descriptions
- ▶ big data matrices
- ▶ hallucination

Challenges and solutions

challenges:

- ▶ long problem descriptions
- ▶ big data matrices
- ▶ hallucination
- ▶ bad models

Challenges and solutions

challenges:

- ▶ long problem descriptions
- ▶ big data matrices
- ▶ hallucination
- ▶ bad models

solutions:

Challenges and solutions

challenges:

- ▶ long problem descriptions
- ▶ big data matrices
- ▶ hallucination
- ▶ bad models

solutions:

- ▶ 1) decomposition 2) bigger context windows

Challenges and solutions

challenges:

- ▶ long problem descriptions
- ▶ big data matrices
- ▶ hallucination
- ▶ bad models

solutions:

- ▶ 1) decomposition 2) bigger context windows
- ▶ use a solver

Challenges and solutions

challenges:

- ▶ long problem descriptions
- ▶ big data matrices
- ▶ hallucination
- ▶ bad models

solutions:

- ▶ 1) decomposition 2) bigger context windows
- ▶ use a solver
- ▶ reflection and testing

Challenges and solutions

challenges:

- ▶ long problem descriptions
- ▶ big data matrices
- ▶ hallucination
- ▶ bad models

solutions:

- ▶ 1) decomposition 2) bigger context windows
- ▶ use a solver
- ▶ reflection and testing
- ▶ education

Challenges and solutions

challenges:

- ▶ long problem descriptions
- ▶ big data matrices
- ▶ hallucination
- ▶ bad models

solutions:

- ▶ 1) decomposition 2) bigger context windows
- ▶ use a solver
- ▶ reflection and testing
- ▶ education

LLMs can introduce subtle errors:

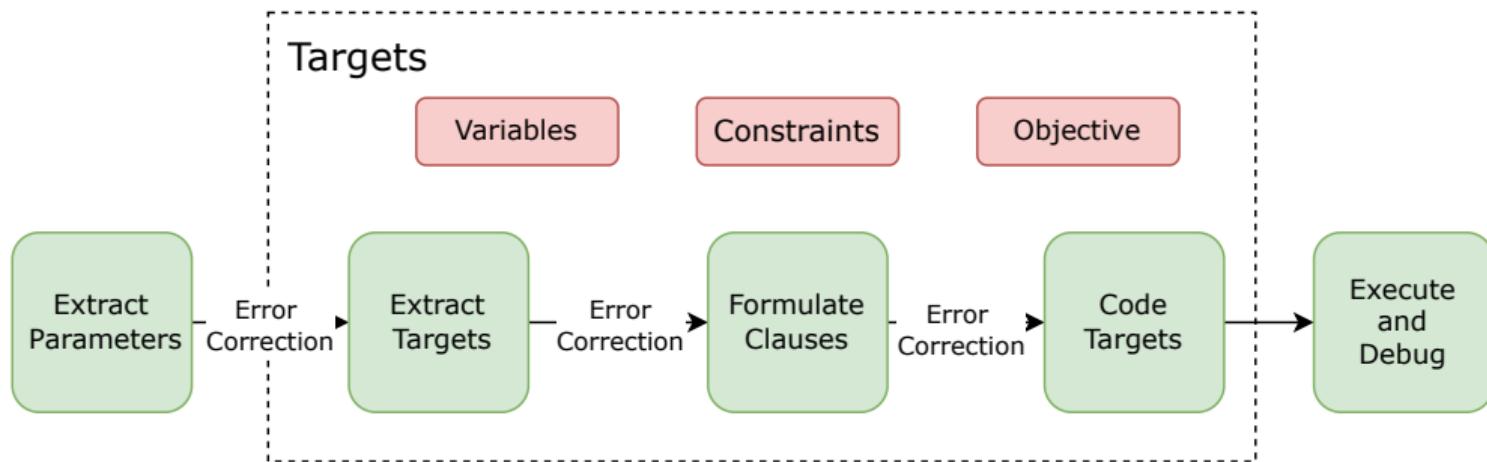
- ▶ code runs, but the result is incorrect
- ▶ some constraints are ignored
- ▶ variable doesn't match desired interpretation (eg, $\text{AbsPrice} \neq |\text{Price}|$)

exploit the structure of optimization to reduce errors!

OptiMUS-0.3

OptiMUS: Optimization Modeling Using Solvers

- ▶ a project to automate optimization modeling
- ▶ a suite of opensource tools for building an optimization copilot

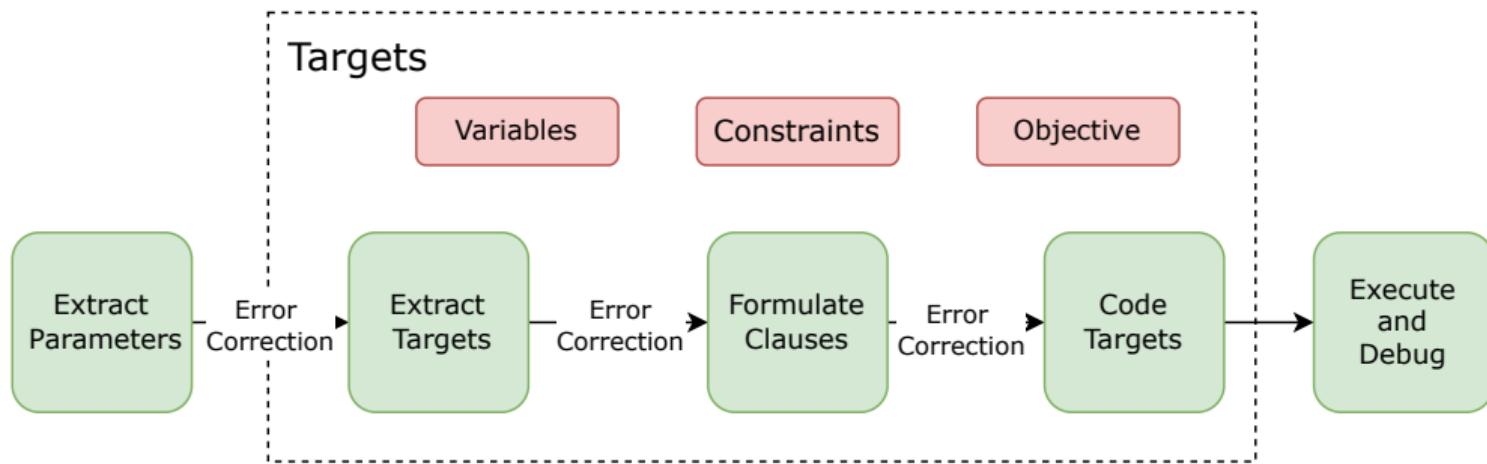


- ▶ paper: <https://arxiv.org/abs/2407.19633>
- ▶ code: <https://github.com/teshnizi/OptiMUS>

OptiMUS-0.3

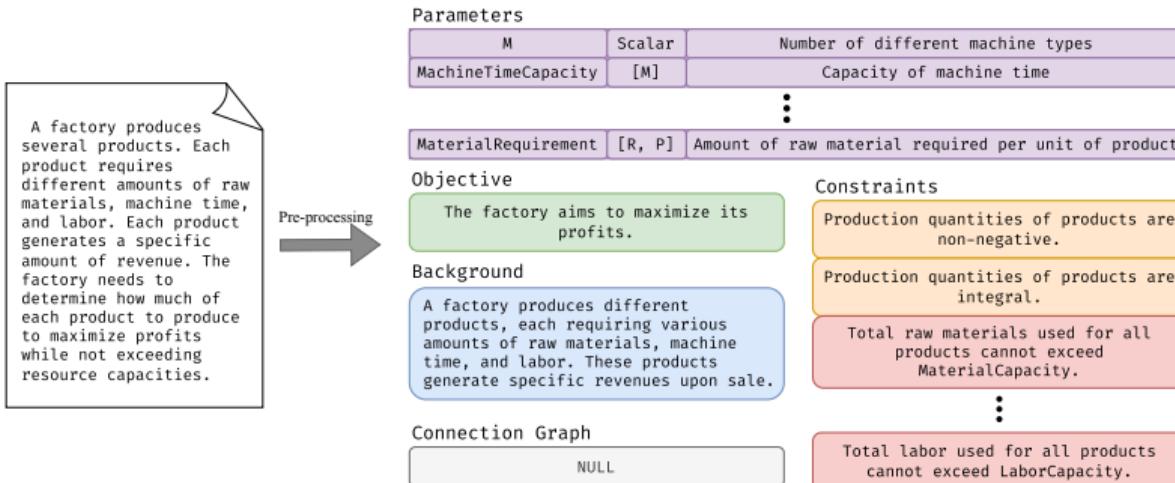
OptiMUS: Optimization Modeling Using Solvers

- ▶ a project to automate optimization modeling
- ▶ a suite of opensource tools for building an optimization copilot



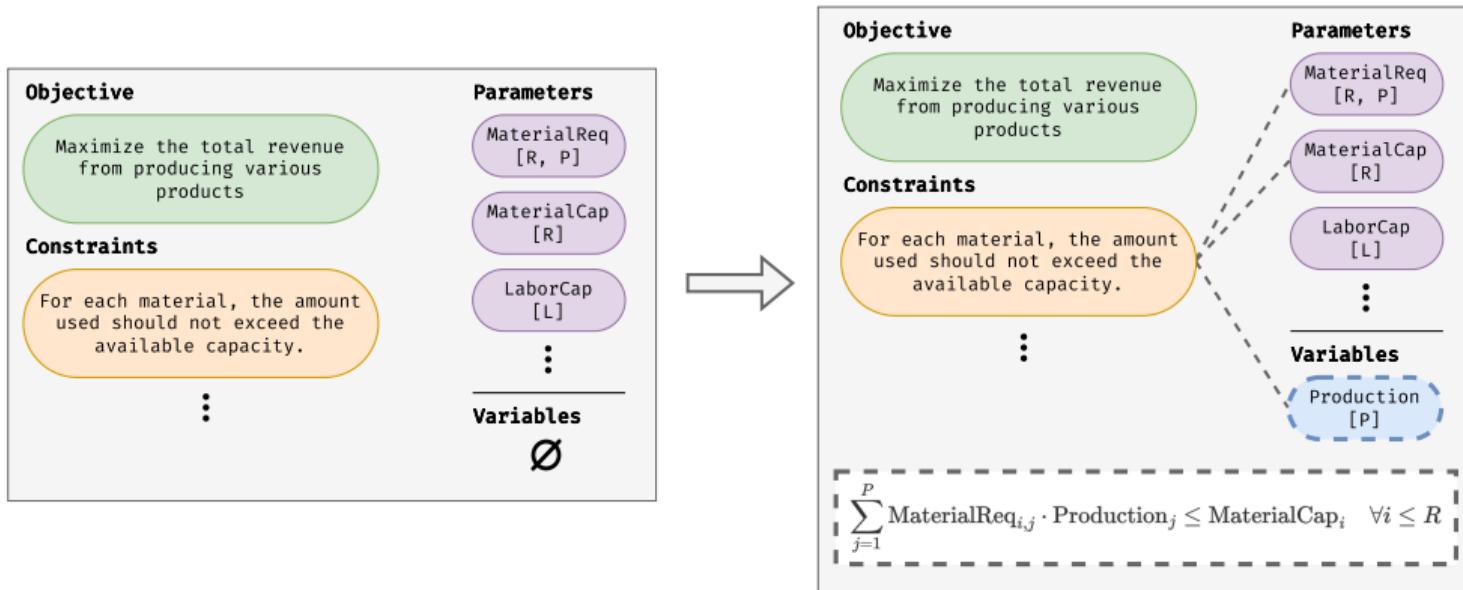
- ▶ paper: <https://arxiv.org/abs/2407.19633>
- ▶ code: <https://github.com/teshnizi/OptiMUS>
- ▶ Optimus-0.2: AhmadiTeshnizi, Gao, and Udell, ICML 2024

OptiMUS: segment optimization problem



OptiMUS preprocessor extracts parameters, constraints, objective, and background information on the problem.

OptiMUS: formulating a single constraint



- ▶ OptiMUS identifies relevant variables and parameters for each constraint
- ▶ LLM only needs to parse and understand the relevant context for modeling, coding, and debugging \Rightarrow needs much less context

OptiMUS: completed formulation

Parameters

MaterialReq	[R, P]	Raw material required per unit of product
M	Scalar	Number of different machine types

⋮

MachineTimeCap	[M]	Capacity of machine time
----------------	-----	--------------------------

Variables

Production	[P]	Continuous	Amount of each product produced
------------	-----	------------	---------------------------------

Background

A factory produces different products, each requiring various amounts of raw materials, machine time, and labor. These products generate specific revenues upon sale.

Objective

The factory aims to maximize its profits.

$$\sum_{i=1}^P \text{Production}_i \times \text{Price}_i$$

```
m.setObjective(sum(Production[i] * Price[i] for i in range(P)), GRB.MAXIMIZE)
```

Constraints

ID:1 Total raw materials used for all products cannot exceed MaterialCapacity. $\forall j: \sum_{i=1}^P \text{Production}_i \times \text{MaterialReq}_{i,j} \leq \text{MaterialCapacity}_{j}$

```
for j in range(J):  
    m.addConstr(sum(Production[i] * MaterialReq[i][j] for i in range(P)) <= MaterialCapacity[j], name=f"MaterialCapacity_{j}")
```

⋮

ID:k Production quantities of products are non-negative. $\forall i: \text{Production}_i \geq 0$

```
for i in range(P):  
    m.addConstr(Production[i] >= 0, name=f"Non-negativity_{i}")
```

Error correction

OptiMUS uses self-reflection to identify and fix common errors.

Does constraint *Each system uses an integer number of disk drives*. need to be explicitly modeled in the mathematical formulation?

This statement seems to imply that we need to ensure that the number of disk drives used by each system is an integer. However, this is already implicit in the variable definition, as we are dealing with a discrete number of disk drives.

Therefore, I don't think this statement needs to be explicitly modeled as a constraint in the (MI)LP formulation.

LLM reflects on its confidence

Consider this potential constraint: No more than MaxPriceChange fraction of prices can change

Is it actually a constraint? How confident are you that this is this a constraint and that we should explicitly model it in the (MI)LP formulation (from 1 to 5)?

The parameter MaxPriceChange is defined as the "maximum fraction of prices that can be changed." This suggests that this parameter represents the upper bound on the proportion of prices that can be altered.

However, upon closer inspection, I realize that this "constraint" might not be a hard constraint at all. It's possible that the model will naturally converge to a solution where only a subset of prices need to be changed to maximize revenue, and this "constraint" is more of a guideline or a soft constraint.

Furthermore, if we were to explicitly model this as a constraint, it would require introducing binary variables to indicate whether a price is changed or not, and then counting the number of binary variables that are 1 to ensure that it doesn't exceed the MaxPriceChange fraction. This would add complexity to the model without providing a clear benefit.

Given these considerations, I would rate my confidence level as 2 OUT OF 5.

Llama-3-70B

Llama isn't sure whether the statement is a hard constraint or not.

Errors can be corrected with a more powerful LLM

Your colleague is suggesting that the following constraint should be added to the list:

No more than MaxPriceChange fraction of prices can change

Here is its explanation:

The parameter MaxPriceChange ... Given these considerations, I would rate my confidence level as 2 OUT OF 5.

Do you want to keep this constraint?

The provided parameter, MaxPriceChange, is supposed to represent the fraction of prices that can change. Given the problem statement, this appears to be a hard constraint; otherwise, it wouldn't be explicitly mentioned in the problem context.

Here are the key points that align with this being a hard constraint:

- The description clearly mentions that no more than a certain fraction of prices can be changed.
- The parameter MaxPriceChange is defined explicitly for this purpose.
- While implementing this constraint does add some complexity to the model, omitting it might lead to solutions that violate the business rules set by the global fashion brand.

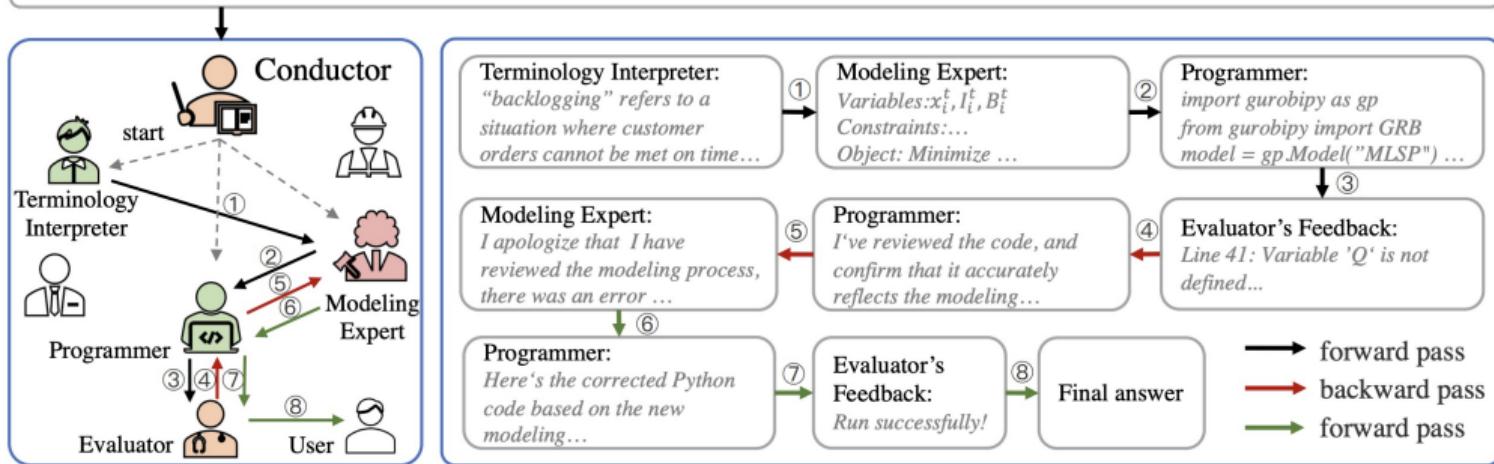
If we do not enforce this constraint, we risk creating a solution that might not be feasible or acceptable to the stakeholders. Therefore, the constraint **should indeed be added to the model**.

GPT-4o

GPT4o can correct Llama's error.

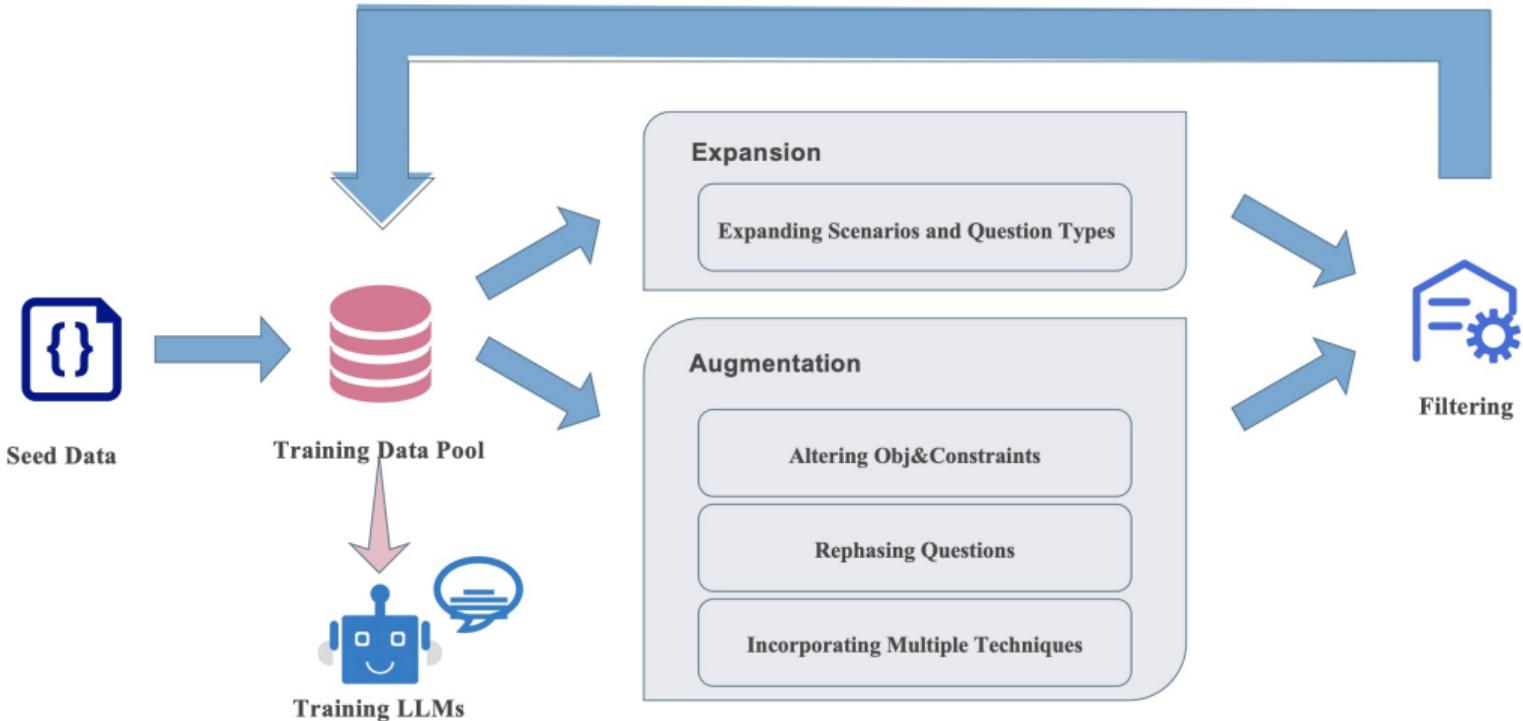
Chain of Experts: an agentic system

Problem Input: In the context of manufacturing planning, we tackle the Multi-level Lot Sizing Problem with Backlogging. We assume that...



source: Xiao et al., 2023

ORLM: pure fine-tuning



source: Tang et al., 2024

Madeleine Udell, Stanford. *OptiMUS*

Outline

Challenge

Architecture

Results

Interactive optimization modeling

Teaching LLMs optimization

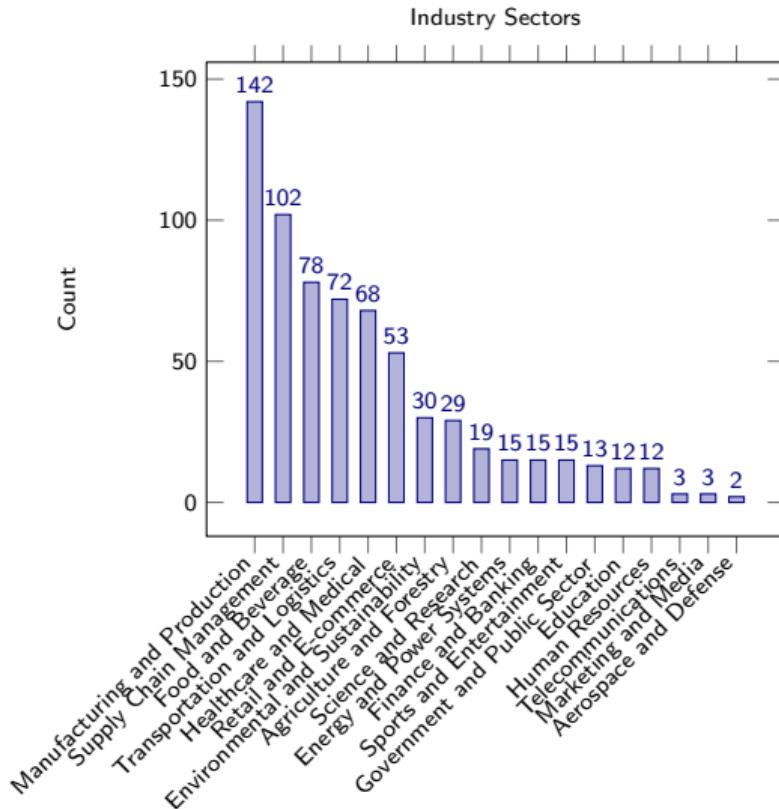
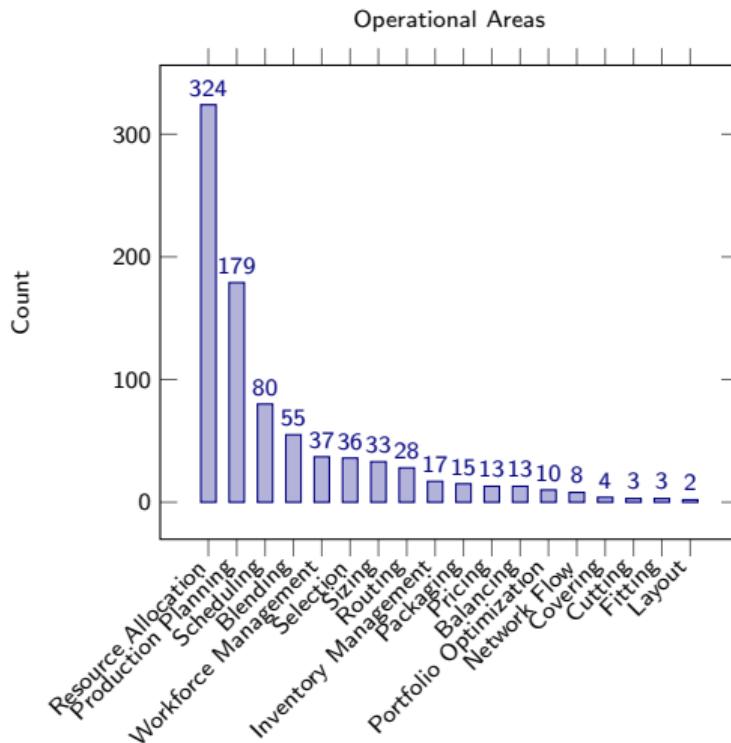
Safety, ethics, and future directions

Experiments and dataset

Table: Benchmark datasets for optimization modeling

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	67 (18)	✓

Diversity of problems



Performance

	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%

Ablation study

	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%

- ▶ easy problems need just a bit of debugging
- ▶ harder problems require error correction and LLM feedback
- ▶ harder problems benefit from more powerful LLM

Ablation study: does debugging help?

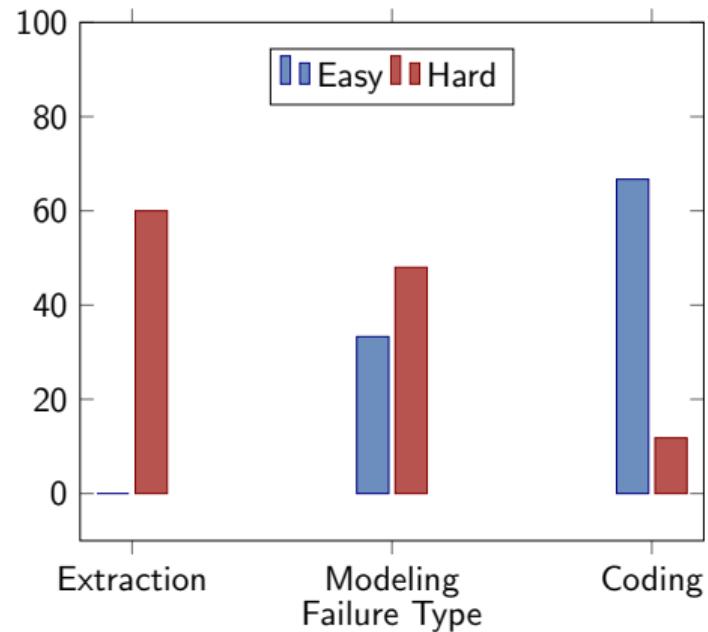
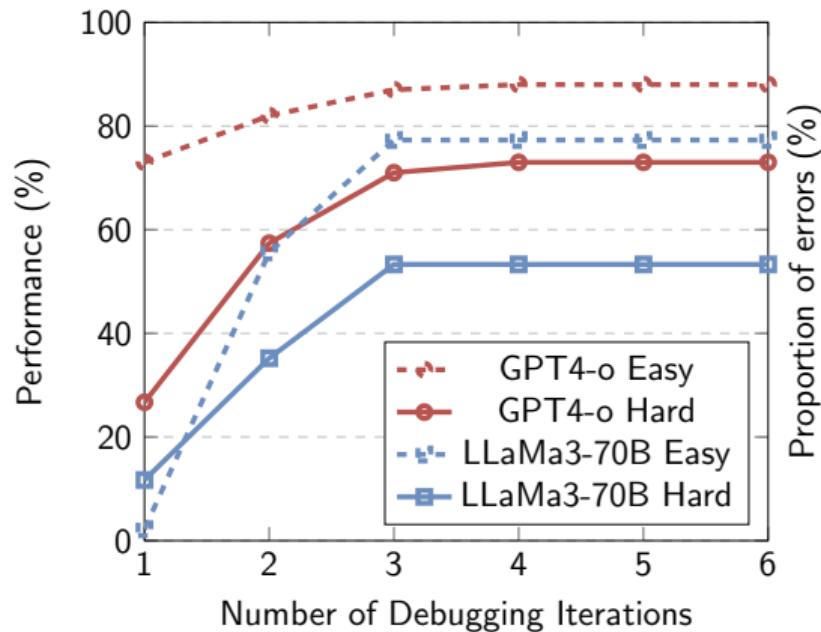


Figure: Left) Debugging improves performance. Right) For harder problems, most failures arise from clause extraction. For easier problems, most failures are due to coding.

Error correction finds and corrects most errors

Table: Error correction methods can find and fix a large fraction of errors in constraint extraction (left) and constraint modeling (right), without modifying most correct items. (Perfect performance is diagonal.)

Extraction		Modeling			
	Not Modified	Modified			
Right	219	7	Right	231	2
Wrong	9	41	Wrong	4	22

Longer problems are still more challenging

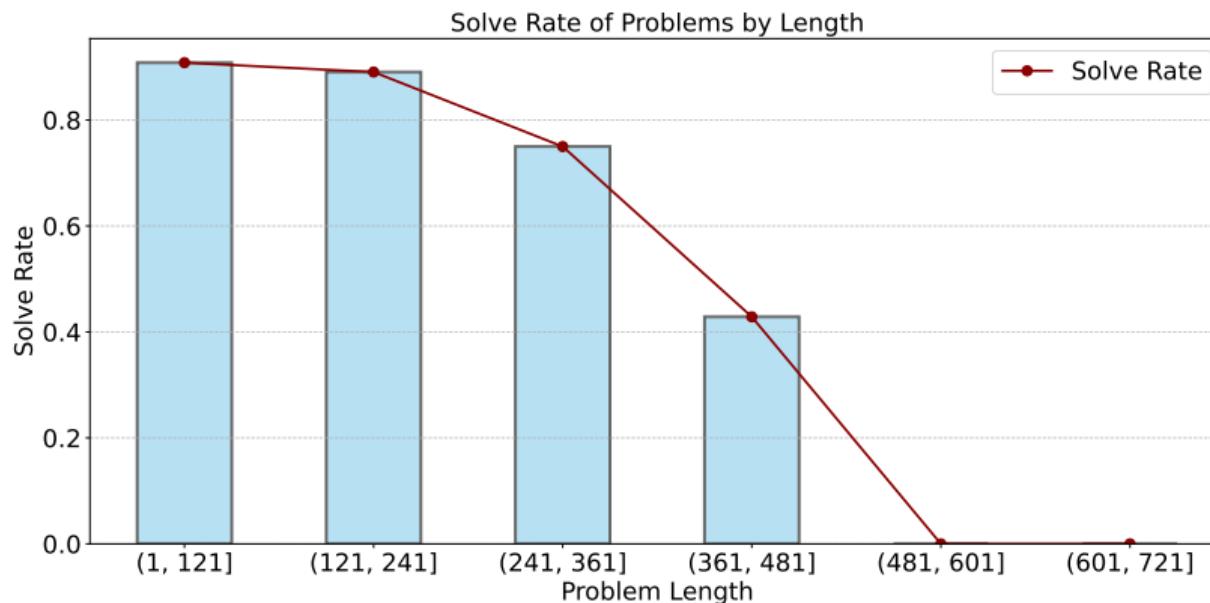


Figure: Solve rate vs. length of problem description

Outline

Challenge

Architecture

Results

Interactive optimization modeling

Teaching LLMs optimization

Safety, ethics, and future directions

Example: production problem

A production planning problem:

- ▶ Given a number of products, each product is produced at a specific rate (in tons per hour).
- ▶ There are a number of hours available in a week.
- ▶ A ton of each product results in a known profit.
- ▶ For each product, there is a lower limit and an upper limit on the tons of that product sold in a week.
- ▶ The problem aims to maximize the total profit from selling all products.
- ▶ The total number of hours used by all products may not exceed the hours available.
- ▶ How to decide the tons of each product to be produced?

Let's try using the OptiMUS WebApp to solve!

Example: identify parameters

 OptimUS



1 Description
2 Parameters
3 Constraints & Objective
4 Mathematical Formulation
5 Coding
6 Data Processing
7 Testing

Formatted Description

Consider a production problem. Given a number of products `\param{P}`, each product is produced at a specific rate `\param{ProductionRate}` (in tons per hour). There are `\param{HoursAvailable}` hours available in a week. A ton of each product results in a known profit `\param{ProfitPerTon}`. For each product, there is a lower limit `\param{LowerLimit}` and an upper limit `\param{UpperLimit}` on the tons of that product sold in a week. The problem aims to maximize the total profit from selling all products. The total number of hours used by all products may not exceed `\param{HoursAvailable}`. How to decide the tons of each product to be produced?

Parameters

Symbol	Shape	Definition	Action
<code>LowerLimit</code>	<code>[T]</code>	Lower limit on the tons of each product sold in a week	
<code>P</code>	<code>[]</code>	Number of different products	
<code>ProfitPerTon</code>	<code>[T]</code>	Profit obtained by selling a ton of each product	
<code>HoursAvailable</code>	<code>[]</code>	Number of hours available in a week	
<code>UpperLimit</code>	<code>[T]</code>	Upper limit on the tons of each product sold in a week	
<code>ProductionRate</code>	<code>[T]</code>	Production rate of each product in tons per hour	



Reset

[Have Feedback?](#)

Made with ❤ at Udell Lab

Next →

Example: hallucinated constraint

The screenshot shows the OptiMUS interface with a sidebar on the left and various sections on the right.

Left Sidebar (Vertical Progress Bar):

- 1 Description
- 2 Parameters
- 3 Constraints & Objective
- 4 Mathematical Formulation
- 5 Coding
- 6 Data Processing
- 7 Testing

Top Right Buttons:

- Extract Constraints and Objective
- ?

Objective Section:

The company aims to maximize its total profit from selling all products

Background Section:

A company produces a variety of products, each with specific production rates, profit margins, and time constraints within a given week.

Constraints Section:

Description	Action
Each product can only be produced in non-negative quantities	Remove
The production volume for each product must be an integer value if products cannot be fractionally produced	Remove
Each product has a minimum production limit of LowerLimit tons per week	Remove
Each product has a maximum production limit of UpperLimit tons per week	Remove
The total number of production hours for all products must not exceed HoursAvailable hours per week	Remove
The production of each product is constrained by its respective ProductionRate in tons per hour	Remove

Bottom Buttons:

- Reset
- Have Feedback?
- Made with ❤ at Udell Lab
- Next →

“The production volume for each product must be an integer value”

Example: hallucinated constraint

The screenshot shows the OptimUS interface with a sidebar on the left and a main content area on the right.

Left Sidebar (Vertical Progress Bar):

- 1 Description
- 2 Parameters
- 3 Constraints & Objective
- 4 Mathematical Formulation
- 5 Coding
- 6 Data Processing
- 7 Testing

Top Right Buttons:

- Extract Constraints and Objective

Content Area:

Objective: The company aims to maximize its total profit from selling all products

Background: A company produces a variety of products, each with specific production rates, profit margins, and time constraints within a given week.

Constraints:

Description	Action
Each product can only be produced in non-negative quantities	Remove
The production volume for each product must be an integer value if products cannot be fractionally produced	Remove
Each product has a minimum production limit of LowerLimit tons per week	Remove
Each product has a maximum production limit of UpperLimit tons per week	Remove
The total number of production hours for all products must not exceed HoursAvailable hours per week	Remove
The production of each product is constrained by its respective ProductionRate in tons per hour	Remove

Bottom Buttons:

- Reset
- Have Feedback? [Feedback](#)
- Made with ❤ at Udell Lab
- Next →

“The production volume for each product must be an integer value”

No integrality constraint is explicitly mentioned, nor is it necessary from context.

Example: spurious constraint

The screenshot shows the OptiMUS interface with a vertical workflow on the left and a main content area on the right.

Workflow:

- Description
- Parameters
- Constraints & Objective
- Mathematical Formulation
- Coding
- Data Processing
- Testing

Main Content Area:

Extract Constraints and Objective

Objective:
The company aims to maximize its total profit from selling all products

Background:
A company produces a variety of products, each with specific production rates, profit margins, and time constraints within a given week.

Constraints:

Description	Action
Each product can only be produced in non-negative quantities	<input checked="" type="checkbox"/>
The production volume for each product must be an integer value if products cannot be fractionally produced	<input checked="" type="checkbox"/>
Each product has a minimum production limit of LowerLimit tons per week	<input checked="" type="checkbox"/>
Each product has a maximum production limit of UpperLimit tons per week	<input checked="" type="checkbox"/>
The total number of production hours for all products must not exceed HoursAvailable hours per week	<input checked="" type="checkbox"/>
The production of each product is constrained by its respective ProductionRate in tons per hour	<input checked="" type="checkbox"/>

Buttons:

- Reset
- Next →

[Have Feedback?](#)
Made with ❤ at Udell Lab

“The production for each product is constrained by its respective production in tons per hour”

Example: spurious constraint

The screenshot shows the OptiMUS interface with a vertical workflow on the left and various sections on the right.

Workflow:

- Description
- Parameters
- Constraints & Objective
- Mathematical Formulation
- Coding
- Data Processing
- Testing

Extract Constraints and Objective button is located at the top right.

Objective:

The company aims to maximize its total profit from selling all products

Background:

A company produces a variety of products, each with specific production rates, profit margins, and time constraints within a given week.

Constraints:

Description	Action
Each product can only be produced in non-negative quantities	<input checked="" type="checkbox"/>
The production volume for each product must be an integer value if products cannot be fractionally produced	<input checked="" type="checkbox"/>
Each product has a minimum production limit of LowerLimit tons per week	<input checked="" type="checkbox"/>
Each product has a maximum production limit of UpperLimit tons per week	<input checked="" type="checkbox"/>
The total number of production hours for all products must not exceed HoursAvailable hours per week	<input checked="" type="checkbox"/>
The production of each product is constrained by its respective ProductionRate in tons per hour	<input checked="" type="checkbox"/>

Buttons:

- Reset
- Next →

[Have Feedback?](#)

Made with ❤ at Udell Lab

“The production for each product is constrained by its respective production in tons per hour” This is not be a constraint! It is enforced automatically as we use the

Example: incorrect coding

OPTIMUS

1 Description

2 Parameters

3 Constraints & Objective

4 Mathematical Formulation

5 Coding

6 Data Processing

7 Testing

Objective

Maximize the total profit from selling all products

Formulate

$$\max \sum_{p=1}^P (\text{ProfitPerTon}_p \times \text{TonsSold}_p)$$

Constraints

Non-negative production constraint for each product

Formulate

$$\text{TonsSold}_p \geq \text{LowerLimit}_p \quad \forall p \in \{1, 2, \dots, P\}$$

Every product must meet a minimum production

Formulate

$$\text{TonsSold}_p \geq \text{LowerLimit}_p \quad \forall p \in \{1, 2, \dots, P\}$$

Each product must not be produced in quantities

Formulate

$$0 \leq \text{TonsSold}_p \leq \text{UpperLimit}_p \quad \forall p \in \{1, 2, \dots, P\}$$

Reset

Have Feedback?

Made with ❤ at Udell Lab

Constraint: “Non-negative production constraint for each product”

Example: incorrect coding

The screenshot shows the OptimUS software interface. The left sidebar has a vertical list of steps: 1. Description, 2. Parameters, 3. Constraints & Objective, 4. Mathematical Formulation, 5. Coding, 6. Data Processing, and 7. Testing. Step 4 is currently selected. The main area is divided into 'Objective' and 'Constraints' sections. The 'Objective' section contains a box with the text 'Maximize the total profit from selling all products' and a 'Formulate' button. Below it is the mathematical formulation:
$$\max \sum_{p=1}^P (\text{ProfitPerTon}_p \times \text{TonsSold}_p)$$
. The 'Constraints' section contains a box with the text 'Non-negative production constraint for each product' and a 'Formulate' button. Below it is the mathematical formulation:
$$\text{TonsSold}_p \geq \text{LowerLimit}_p \quad \forall p \in \{1, 2, \dots, P\}$$
. This constraint is circled with a red dashed line. The 'Constraints' section also contains two more boxes, each with a 'Formulate' button and a mathematical formulation:
$$\text{TonsSold}_p \geq \text{LowerLimit}_p \quad \forall p \in \{1, 2, \dots, P\}$$
 and
$$0 \leq \text{TonsSold}_p \leq \text{UpperLimit}_p \quad \forall p \in \{1, 2, \dots, P\}$$
. At the bottom left, there are 'Reset' and 'Have Feedback?' buttons, and a footer note 'Made with ❤ at Udell Lab'.

Constraint: “Non-negative production constraint for each product”

Formulation: $\text{TonsSold}_p \geq \text{LowerLimit}_p$

Example: data upload

The screenshot shows the OptimUS web application interface. On the left, a vertical navigation bar lists steps 1 through 7: Description, Parameters, Constraints & Objective, Mathematical Formulation, Coding, Data Processing, and Testing. Step 1 is currently selected. To the right of the navigation bar is a sidebar with sections for Parameters, Symbols, Preferences, Acceptability, and Capacity. Below these sections is a 'Groups' section. The main content area is titled 'Data Upload' and contains a text box with sample JSON data. The JSON data is as follows:

```
{  
  "G": 2,  
  "N": 4,  
  "B": 2,  
  "M": 5,  
  "Preference": [  
    [1, 10, 3, 1, 3],  
    [8, 6, 7, 10, 7],  
    [10, 1, 2, 5, 2],  
    [7, 7, 5, 3, 9]  
  ],  
  "Acceptable": [  
    [4, 3, 1, 9, 7],  
    [3, 7, 10, 3, 10],  
    [6, 9, 5, 9, 3],  
    [5, 3, 1, 2, 18]  
  ],  
  "Capacity": [4, 3, 4, 6, 1],  
  "Groups": [  
    [6, 6],  
    [7, 7],  
    [10, 3],  
    [2, 7]  
  ]  
}
```

Below the JSON text box are two buttons: 'Download Dummy Example' and 'CHOOSE FILE'. A file input field shows 'No file chosen'. To the right of the file input is a 'Select Data' button. Further right is a 'Next →' button. On the far right, a 'Status' table lists 12 items, all of which are marked as 'Valid' with green checkmarks. At the bottom left of the main content area is a 'Reset' button.

Example: testing

OptIMUS

1

- 1 Description
- 2 Parameters
- 3 Constraints & Objective
- 4 Mathematical Formulation
- 5 Coding
- 6 Data Processing
- 7 Testing

Full Code (Read Only)

```

1  # Import JSON
2  import json
3  import numpy as np
4
5  import gurobipy as gp
6
7  # with open('tmpData/auto.json', 'r') as f:
8  #     data = json.load(f)
9
10 # data = {
11 #     'Preference': np.array(data['Preference']),
12 #     'Acceptable': np.array(data['Acceptable']),
13 #     'N': data['N'],
14 #     'B': data['B'],
15 #     'T': data['T'],
16 #     'Capacity': np.array(data['Capacity']),
17 #     'Groups': np.array(data['Groups']),
18 # }
19
20 # Define model
21 model = gp.Model('model')
22
23
24 # ----- Define variables -----
25 UtilityShortfall = model.addVar(G, vtype=gp.GRB.CONTINUOUS, name="UtilityShortfall")
26 StudentAssignment = model.addVar(N, M, vtype=gp.GRB.BINARY, name="StudentAssignment")
27 CapacityExcess = model.addVar(N, vtype=gp.GRB.CONTINUOUS, name="CapacityExcess")
28 GroupUtility = model.addVar(G, vtype=gp.GRB.CONTINUOUS, name="GroupUtility")
29
30
31 # ----- Define constraints -----
32
33 # Each student can only be assigned to one school
34 for n in range(N):
35     model.addConstr(gp.quicksum(StudentAssignment[n, m] for m in range(M)) == 1, name="one_school_per_student_[n]")
36
37 # Ensure each student is assigned to at least one acceptable school from their list
38 for n in range(N):
39     model.addConstr(gp.quicksum(StudentAssignment[n, m] * Acceptable[n, m] for m in range(M)) >= 1, name="student_school_assignment_[n]_formulation")
40
41 # Add utility shortfall constraints for each group
42 for g in range(G):
43     utility_zgp = gp.quicksum(
44         gp.quicksum(

```

Results

...

ERROR: Invalid data in vars array at "# Ensure that the capacity violation for each school corresponds to the number of students assigned beyond its capacity for a in range(M): student_sum = gp.quicksum(StudentAssignment[i, a] for n in range(N)) model.addConstr((CapacityExcess[i] == gp.max([0, student_sum - Capacity[i]]), name="capacity_violation_(m)"))"

OptIMUS Log

Will be added soon!

Reset

Have Feedback?

Made with  at Udel Lab

Run Code

Fix Code

Example: success!

The image shows the OptimUS interface with a success status. The left sidebar lists steps 1 through 7: Description, Parameters, Constraints & Objective, Mathematical Formulation, Coding, Data Processing, and Testing. The 'Coding' step is currently selected. The main area displays the 'Full Code (Read Only)' for a Gurobi model, which reads data from 'tmpData/data.json', defines variables for 'TonsSold', adds constraints for non-negative production and minimum weekly production, and adds an upper limit constraint. The 'Results' section shows a successful run with an objective value of 9.0000, runtime of 0.0015s, and iteration count of 0. The 'Variables' section shows 'TonsSold[0]: 3.0000' and 'TonsSold[1]: 3.0000'. The 'Optimus Log' section is empty and noted as 'Will be added soon!'. A 'Run Code' button is at the bottom.

Optimus

1 Description

2 Parameters

3 Constraints & Objective

4 Mathematical Formulation

5 Coding

6 Data Processing

7 Testing

Full Code (Read Only)

```
1 import json
2 import numpy as np
3 import gurobipy as gp
4
5 with open("tmpData/data.json", "r") as f:
6     data = json.load(f)
7
8     LowerLimit = np.array(data["LowerLimit"])
9     P = data["P"]
10    ProfitPerTon = np.array(data["ProfitPerTon"])
11    HoursAvailable = data["HoursAvailable"]
12    UpperLimit = np.array(data["UpperLimit"])
13    ProductionRate = np.array(data["ProductionRate"])
14
15    # Define model
16    model = gp.Model('model')
17
18    # ===== Define variables =====
19    TonsSold = model.addVars(P, vtype=gp.GRB.CONTINUOUS, name="TonsSold")
20
21    # ===== Define constraints =====
22
23    # Add non-negative production constraints for each product
24    for p in range(P):
25        model.addConstr(TonsSold[p] >= LowerLimit[p], name=f"non_neg_prod_constr_{p}")
26
27    # Add constraints to ensure every product meets the minimum production limit
28    # per week
29    for p in range(P):
30        model.addConstr(TonsSold[p] >= LowerLimit[p], name=f"min_prod_limit_{p}")
31
32    # Add upper limit constraints for products sold in a week
33    for p in range(P):
34        model.addConstr(TonsSold[p] <= UpperLimit[p], name=f"upper_limit_{p}")
35
36    # Add upper limit constraints for products sold in a week
37    for p in range(P):
```

Results

Run Successful!

Objective Value: 9.0000

Runtime: 0.0015s

Iteration Count: 0

Variables:

TonsSold[0]: 3.0000

TonsSold[1]: 3.0000

Optimus Log

Will be added soon!

Reset

Have Feedback?

Run Code

Fix Code

Outline

Challenge

Architecture

Results

Interactive optimization modeling

Teaching LLMs optimization

Safety, ethics, and future directions

Teaching LLM advanced optimization

A good optimizer exploits structure. Can an LLM?

Modeling features

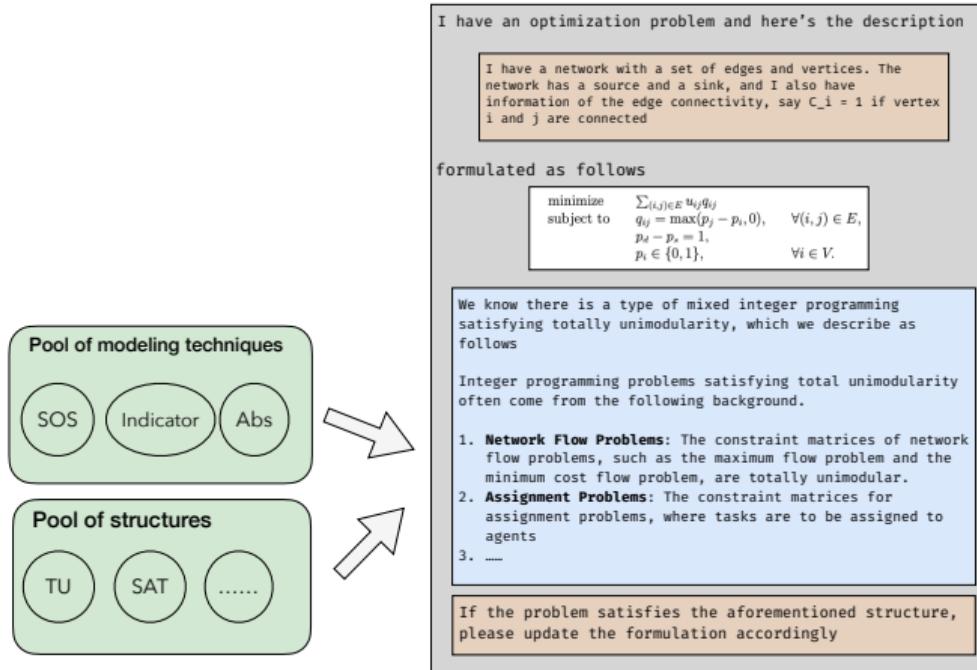
- ▶ Special Ordered Set (SOS)
- ▶ Indicator variables
- ▶ General constraints (norm, abs)
- ▶ ...

Structures

- ▶ Total-unimodularity (network)
- ▶ SAT problem
- ▶ Constraint programming
- ▶ ...

OptiMUS prompts the LLM to identify and deploy each relevant structure/technique.

Illustration: LLM, consider total unimodularity!



Do advanced optimization techniques help?

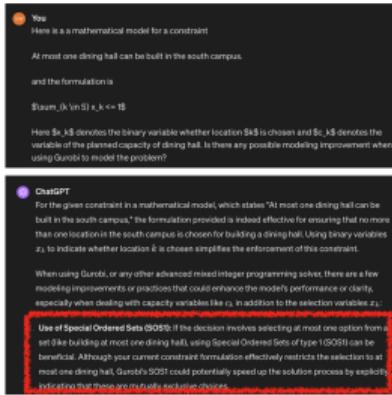


- ▶ Stanford plans to build new dining halls on campus.
- ▶ It costs Stanford $c_j = \alpha_j C$ to build a dining hall of capacity C at candidate location $j = 1, \dots, K$.
- ▶ Each campus residence houses n_i students, $i = 1, \dots, I$.
- ▶ No more than one hall will be built in north campus and one in south campus.

Goal: minimize **distance from students to food + building cost**

MILP formulation

Constraint: at most one dining hall can be built on south campus
Variable c_k : capacity at location k (0 if not built at location k)



You
Here is a a mathematical model for a constraint
At most one dining hall can be built in the south campus.
and the formulation is
 $\sum_{k \in S} x_k \leq 1$
Here x_k denotes the binary variable whether location k is chosen and c_k denotes the variable of the planned capacity of dining hall. Is there any possible modeling improvement when using Gurobi to model the problem?

ChatGPT
For the given constraint in a mathematical model, which states "At most one dining hall can be built in the south campus," the formulation provided is indeed effective for ensuring that no more than one location in the south campus is chosen for building a dining hall. Using binary variables x_k to indicate whether location k is chosen simplifies the enforcement of this constraint.
When using Gurobi, or any other advanced mixed integer programming solver, there are a few modeling improvements or practices that could enhance the model's performance or clarity, especially when dealing with capacity variables like c_k in addition to the selection variables x_k .
Use of Special Ordered Sets (SOS): If the decision involves selecting at most one option from a set (like building at most one dining hall), using Special Ordered Sets of type 1 (SOS1) can be beneficial. Although your current constraint formulation effectively restricts the selection to at most one dining hall, Gurobi's SOS1 could potentially speed up the solution process by explicitly indicating that these are mutually exclusive choices.



Common MILP formulation

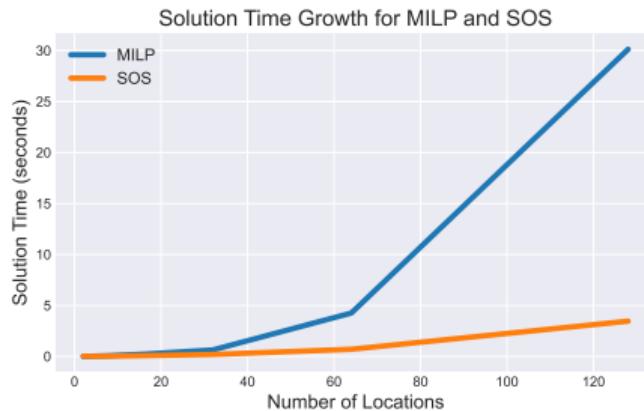
- ▶ x_k : whether location k is chosen
- ▶ $\sum_{k \in S} x_k \leq 1$

SOS formulation

- ▶ c_k : capacity at location k
- ▶ $(c_1, \dots, c_k) \in \text{SOS}_1$

Solution time

- ▶ OptiMUS generates two codes: one standard (MILP) and the other after considering the special ordered set (SOS) technique.
- ▶ We test the performance of the model as the number of candidate locations grows.



Prompting LLMs to consider advanced techniques produces scalable models!

Outline

Challenge

Architecture

Results

Interactive optimization modeling

Teaching LLMs optimization

Safety, ethics, and future directions

Optimization and AI safety



It . . . seems perfectly possible to have a superintelligence whose sole goal is something completely arbitrary, such as to manufacture as many paperclips as possible, and who would resist with all its might any attempt to alter this goal. For better or worse, artificial intellects need not share our human motivational tendencies.

— Nick Bostrom, 2003. "Ethical Issues in Advanced Artificial Intelligence."

AI and the future of work

AI can reduce inequality if it enables lower-ranked workers to perform more valuable work... Because so many of the routine tasks that workers previously performed have already been automated, a large fraction of current jobs require non-routine problemsolving and decision-making tasks. Empowering workers to perform these tasks more effectively, and to accomplish even more sophisticated decision-making tasks, will require providing workers with better information and decision-support tools.

— Daron Acemoglu, David Autor, and Simon Johnson, 2023. "Can we have pro-worker AI?"



Future directions.

Submit a problem:

machine learning thrives on data — help us!

- ▶ better automated optimization modeling will require larger, more complex, more realistic (natural language) problems



Future directions.

Submit a problem:

machine learning thrives on data — help us!

- ▶ better automated optimization modeling will require larger, more complex, more realistic (natural language) problems

can a natural-language specification ever be unambiguous?

- ▶ pin down: query user to clarify goals
- ▶ quantify: assist with finding or assembling problem data
- ▶ build trust: enable non-expert oversight of optimization model with visualizations, simple checks on synthetic data, constraint learning, ...
- ▶ identify fragility: suggest scenarios that might break optimization model, and robust formulations that reduce fragility



Future directions.

Submit a problem:

machine learning thrives on data — help us!

- ▶ better automated optimization modeling will require larger, more complex, more realistic (natural language) problems

can a natural-language specification ever be unambiguous?

- ▶ pin down: query user to clarify goals
- ▶ quantify: assist with finding or assembling problem data
- ▶ build trust: enable non-expert oversight of optimization model with visualizations, simple checks on synthetic data, constraint learning, ...
- ▶ identify fragility: suggest scenarios that might break optimization model, and robust formulations that reduce fragility

real-world problems are constantly changing! need dynamic, editable models.

References

Alibaba Cloud. (2022). Alibaba cloud mindopt copilot. <https://opt.alibabacloud.com/chat>

Chen, H., Constante-Flores, G. E., & Li, C. (2023). Diagnosing infeasible optimization problems using large language models. *arXiv preprint arXiv:2308.12923*.

Gurobi Optimization. (2023). 2023 state of mathematical optimization report.
<https://www.gurobi.com/resources/report-state-of-mathematical-optimization-2023/>

Jiang, C., Shu, X., Qian, H., Lu, X., Zhou, J., Zhou, A., & Yu, Y. (2024). Llmopt: Learning to define and solve general optimization problems from scratch. *arXiv preprint arXiv:2410.13213*.

Li, B., Mellou, K., Zhang, B., Pathuri, J., & Menache, I. (2023). Large language models for supply chain optimization. *arXiv preprint arXiv:2307.03875*.

Ramamonjison et al, . (2022). Augmenting operations research with auto-formulation of optimization models from problem descriptions. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 29–62. <https://aclanthology.org/2022.emnlp-industry.4>

Ramamonjison et al, . (2023). NI4opt competition: Formulating optimization problems based on their natural language descriptions. <https://arxiv.org/abs/2303.08233>

Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J., Ellenberg, J. S., Wang, P., Fawzi, O., et al. (2024). Mathematical discoveries from program search with large language models. *Nature*, 625(7995), 468–475.

Tang, Z., Huang, C., Zheng, X., Hu, S., Wang, Z., Ge, D., & Wang, B. (2024). ORLM: Training large language models for optimization modeling. <https://arxiv.org/abs/2405.17743>

References (cont.)

Wasserkrug, S., Boussioux, L., den Hertog, D., Mirzazadeh, F., Birbil, I., Kurtz, J., & Maragno, D. (2024). From large language models and optimization to decision optimization copilot: A research manifesto. <https://arxiv.org/abs/2402.16269>

Xiao, Z., Zhang, D., Wu, Y., Xu, L., Wang, Y. J., Han, X., Fu, X., Zhong, T., Zeng, J., Song, M., et al. (2023). Chain-of-experts: When LLMs meet complex operations research problems. *The Twelfth International Conference on Learning Representations*.

Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., & Chen, X. (2023). Large language models as optimizers.