

# 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 Brunborg, and Shayan Talaei

October 21, 2024

# Outline

Challenge

Architecture

Results

Teaching LLMs optimization

# Optimization is everywhere



Energy



Routing



Scheduling



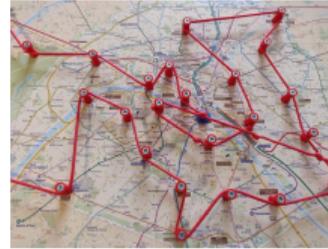
Supply chain

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

# Optimization is everywhere



Energy



Routing



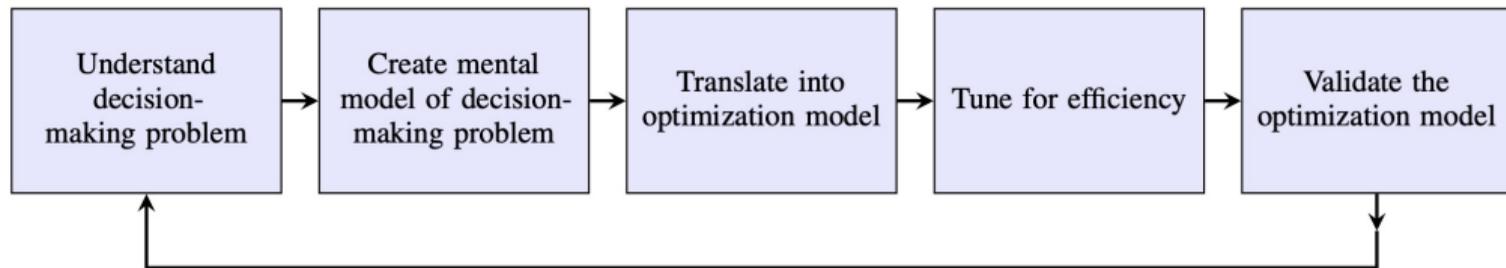
Scheduling



Supply chain

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

## Optimization modeling is the bottleneck



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

- ▶ 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

## Example problem: production

A production planning problem:

- ▶ The factory produces several products, each at a specific rate (in tons per hour).
- ▶ There are a number of work 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.
- ▶ Each product must be produced in increments of 5 tons.

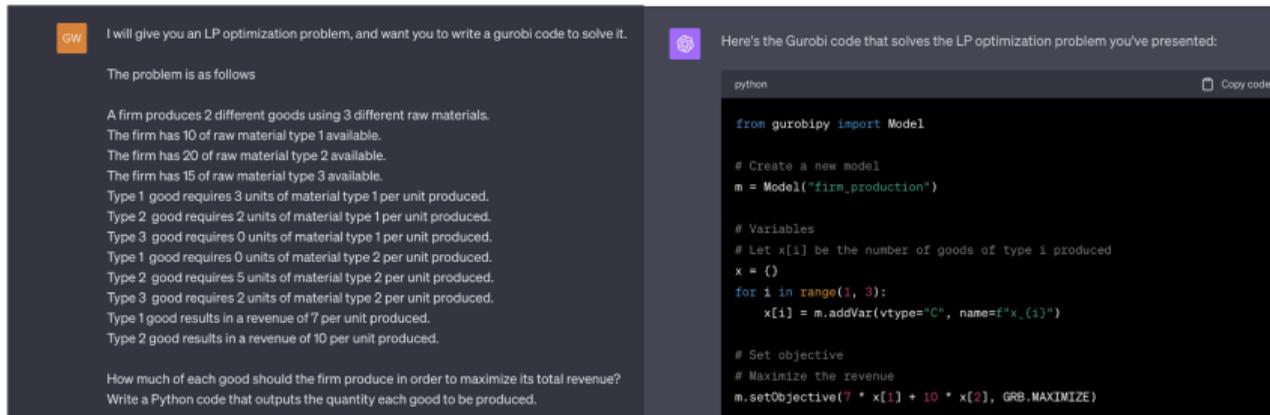
Decide how many tons of each product should be produced.

## Example model: production

```
19 # Define model
20 model = gp.Model('model')
21
22
23 # ===== Define variables =====
24 TonsProduced = model.addVar(name='TonsProduced', vtype=gp.GRB.FLOAT)
25 IntegerProduced = model.addVar(name='IntegerProduced', vtype=gp.GRB
    .INTEGER)
26
27 # ===== Define constraints =====
28
29 model.addConstr(gp.quicksum(x / ProductionRate[i] for i, x in enumerate
    ([TonsProduced])) <= HoursAvailable, name="hours_limit_constraint")
30
31 for i in range(P):
32     model.addConstr(TonsProduced[i] >= LowerLimit[i], name
        =f"min_production_limit_{i}")
33
34 for i in range(P):
35     model.addConstr(TonsProduced[i] <= UpperLimit[i], name
        =f"upper_limit_product_{i}")
36
37 for i in range(P):
38     model.addConstr(TonsProduced[i] == Increment[i] * IntegerProduced[i]
        , name=f"IncrementConstraint_{i}")
39
40 for i in range(P):
41     model.addConstr(TonsProduced[i] >= 0, name
        =f"non_negativity_TonsProduced_{i}")
42
43 # ===== Define objective =====
44
```

# Optimization modeling using LLMs

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



The screenshot shows a chat window with a dark background. On the left, a user prompt (indicated by an orange 'GW' icon) asks for Gurobi code to solve an LP problem. The prompt describes a firm producing two goods from three raw materials, lists material availability, production requirements, and revenue per unit. On the right, the GPT response (indicated by a purple icon) provides the Gurobi code. The code defines a model named 'firm\_production', sets variables x[1] and x[2], and sets the objective function to maximize revenue (7\*x[1] + 10\*x[2]).

```
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 Copy code  
  
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="x_{}".format(i))  
  
# Set objective  
# Maximize the revenue  
m.setObjective(7 * x[1] + 10 * x[2], GRB.MAXIMIZE)
```

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

# Outline

Challenge

Architecture

Results

Teaching LLMs optimization

## How to use LLMs for optimization?

- ▶ ask the LLM for a solution
  - ▶ NL4opt competition (Ramamonjison 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 (Ramamonjison et al, 2022, 2023)
  - ▶ LLMs as optimizers (Yang et al., 2023)
- ▶ ask the LLM to write a solver
  - ▶ FunSearch (Romera-Paredes et al., 2024)

## How to use LLMs for optimization?

- ▶ ask the LLM for a solution
  - ▶ NL4opt competition (Ramamonjison et al, 2022, 2023)
  - ▶ LLMs as optimizers (Yang et al., 2023)
- ▶ ask the LLM to write a solver
  - ▶ FunSearch (Romera-Paredes et al., 2024)
- ▶ ask the LLM to call a solver
  - ▶ OptiMUS: structured pipeline with error correction
  - ▶ chain of experts: agentic system (Xiao et al., 2023)
  - ▶ ORLM: semisynthetic data + finetuning (Tang et al., 2024)

## How to use LLMs for optimization?

- ▶ ask the LLM for a solution
  - ▶ NL4opt competition (Ramamonjison et al, 2022, 2023)
  - ▶ LLMs as optimizers (Yang et al., 2023)
- ▶ ask the LLM to write a solver
  - ▶ FunSearch (Romera-Paredes et al., 2024)
- ▶ ask the LLM to call a solver
  - ▶ OptiMUS: structured pipeline with error correction
  - ▶ chain of experts: agentic system (Xiao et al., 2023)
  - ▶ ORLM: semisynthetic data + finetuning (Tang et al., 2024)
- ▶ chatbots for optimization
  - ▶ 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:

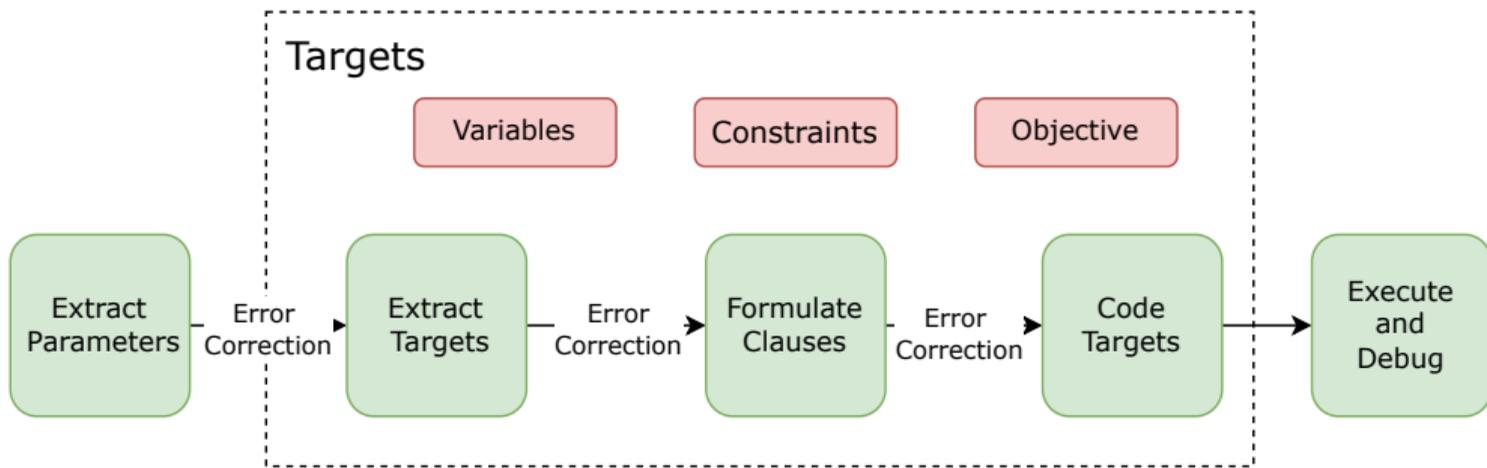
- ▶ code runs, but the result is incorrect
- ▶ some constraints are ignored
- ▶ variable doesn't match desired interpretation (eg,  $\text{AbsPrice} \neq |\text{Price}|$  )

exploit the structure of optimization to reduce errors!

## OptiMUS-0.3

OptiMUS: **O**ptimization **M**odeling **U**sing **S**olvers

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



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

## 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.

# Outline

Challenge

Architecture

Results

Teaching LLMs optimization

## Experiments and dataset

Table: Benchmark datasets for optimization modeling

Dataset	Description Length	Instances (#MILP)	Multi-dimensional Parameters
NL4Opt	$518.0 \pm 110.7$	1101 (0)	×
ComplexOR	$497.1 \pm 247.5$	37 (12)	✓
NLP4LP Easy (Ours)	$507.2 \pm 102.6$	287 (0)	✓
NLP4LP Hard (Ours)	$912.3 \pm 498.2$	67 (18)	✓

## Performance

	Easy	Hard
Standard	47.3%	33.2%
Reflexion	53.0%	42.6%
CoE	64.2%	49.2%
OptiMUS-0.2	78.8%	68%
<b>OptiMUS-0.3</b>	<b>88.6%</b>	<b>73.7%</b>

- ▶ beats state-of-the-art across different datasets by a large margin.

## Performance

	Easy	Hard
Standard	47.3%	33.2%
Reflexion	53.0%	42.6%
CoE	64.2%	49.2%
OptiMUS-0.2	78.8%	68%
<b>OptiMUS-0.3</b>	<b>88.6%</b>	<b>73.7%</b>

- ▶ beats state-of-the-art across different datasets by a large margin.
- ▶ reliable? not yet. . .

## Longer problems are still more challenging

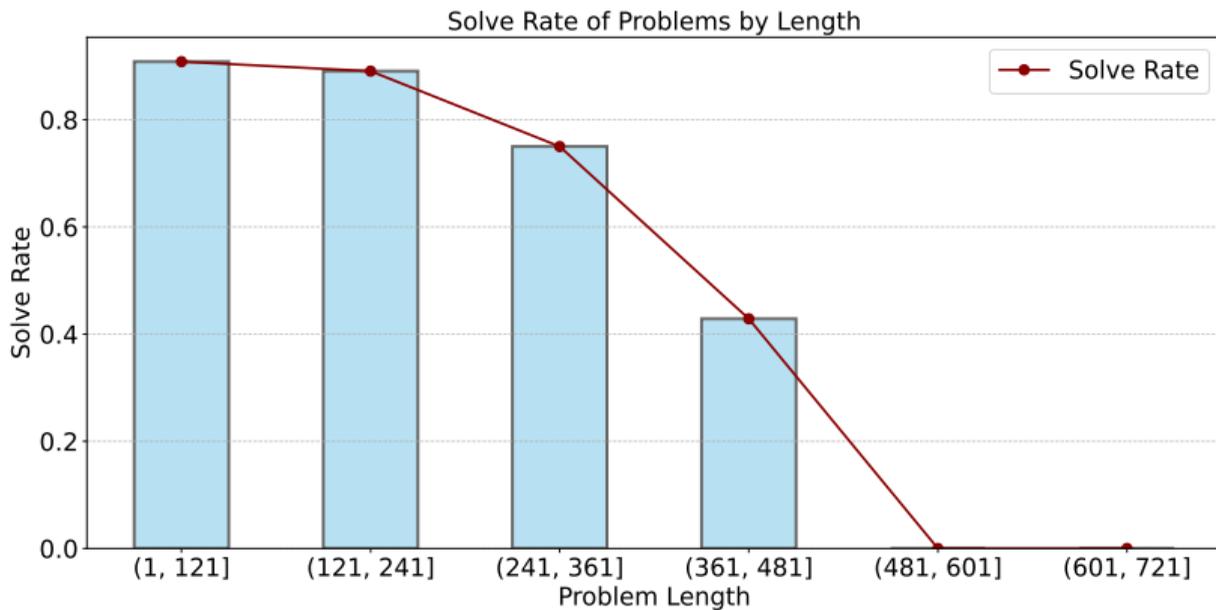


Figure: Solve rate vs. length of problem description

# Outline

Challenge

Architecture

Results

Teaching LLMs optimization

## Teaching LLM advanced optimization

A good optimizer exploits structure. Can an LLM?

### Modeling features

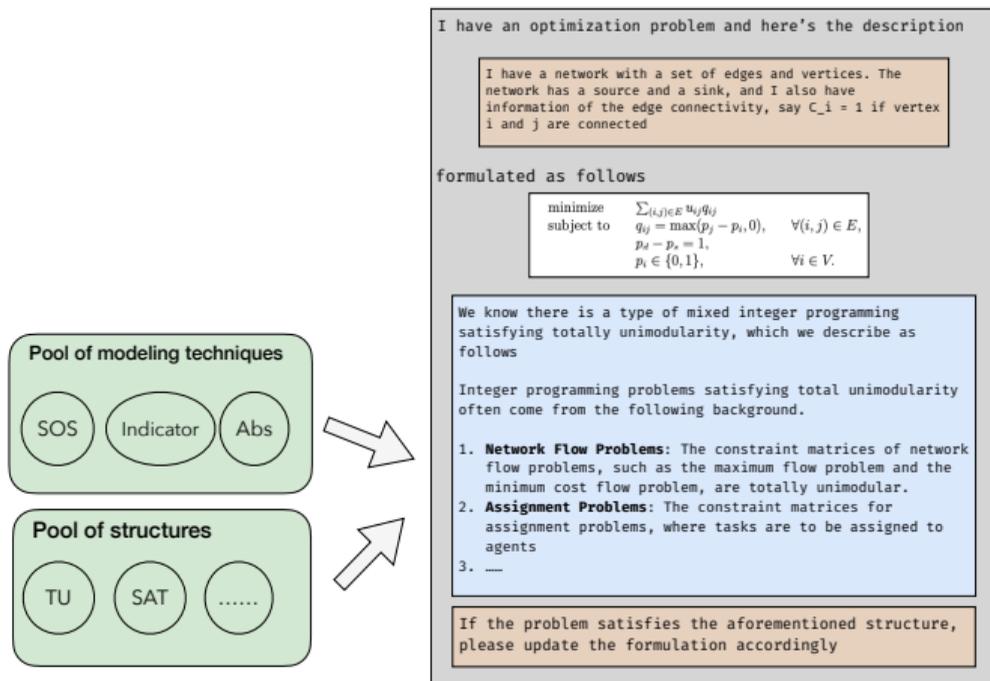
- ▶ Special Ordered Set (SOS)
- ▶ Indicator variables
- ▶ General constraints (norm, abs)
- ▶ ...

### 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?

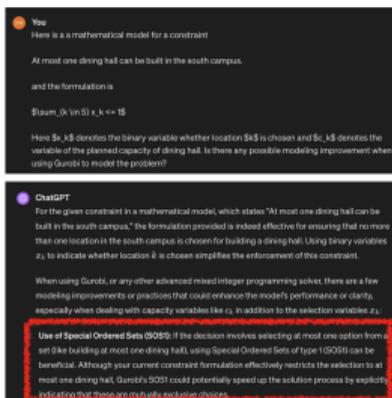


- ▶ Stanford plans to build new dining halls on campus.
- ▶ It costs Stanford  $c_j = \alpha_j C$  to build a dining hall of capacity  $C$  at candidate location  $j = 1, \dots, K$ .
- ▶ Each campus residence houses  $n_i$  students,  $i = 1, \dots, l$ .
- ▶ No more than one hall will be built in north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



**Yes**  
Here is 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  $S$ ,  $x_k$  denotes the binary variable whether location  $k$  is chosen and  $S$ ,  $x_k$  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!

## 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."

## Future directions

machine learning thrives on data!

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

## Future directions

machine learning thrives on data!

- ▶ 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

machine learning thrives on data!

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

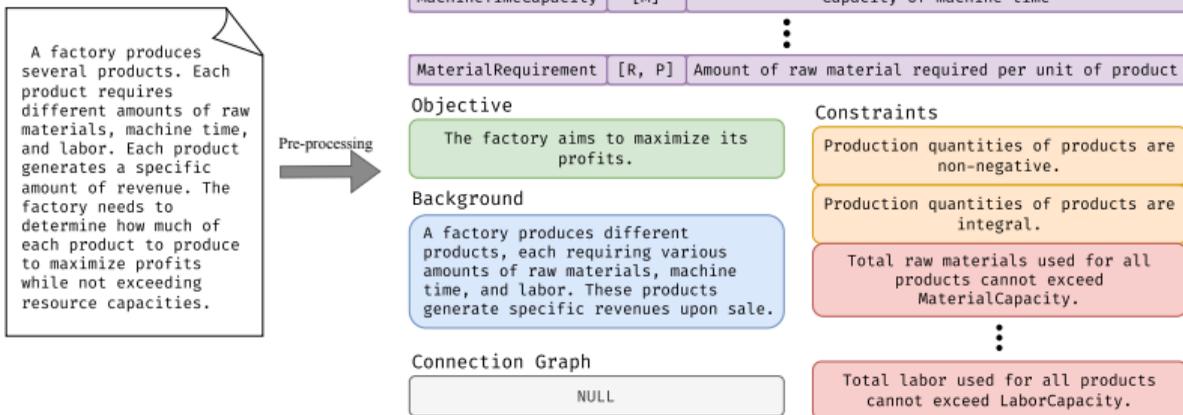
can a natural-language specification ever be unambiguous?

- ▶ 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!

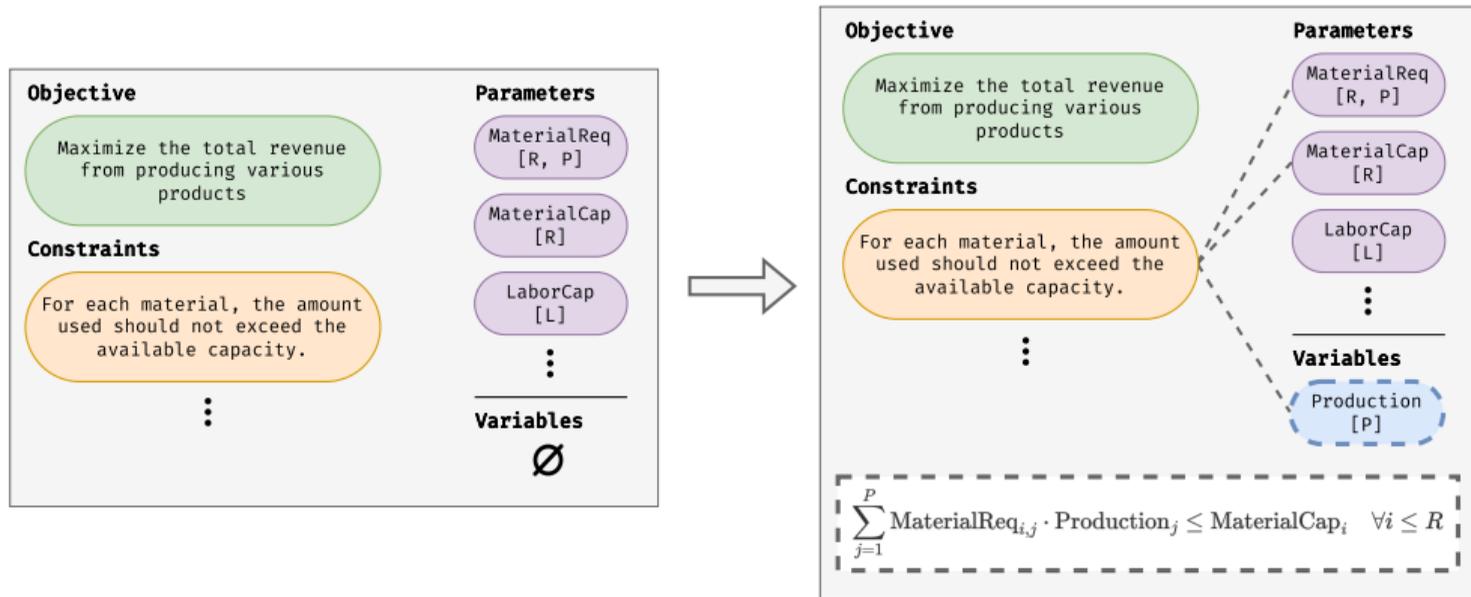
- ▶ enable dynamic, adaptive, editable models
- ▶ choose backend solver, heuristic, or custom solver as requirements change

# 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  $\Rightarrow$  **OptiMUS scales!**

# 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.  $\forall j: \sum_{i=1}^P \text{Production}_i \times \text{MaterialReq}_{ij} \leq \text{MaterialCapacity}_j$

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

Llama-3-70B

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 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:

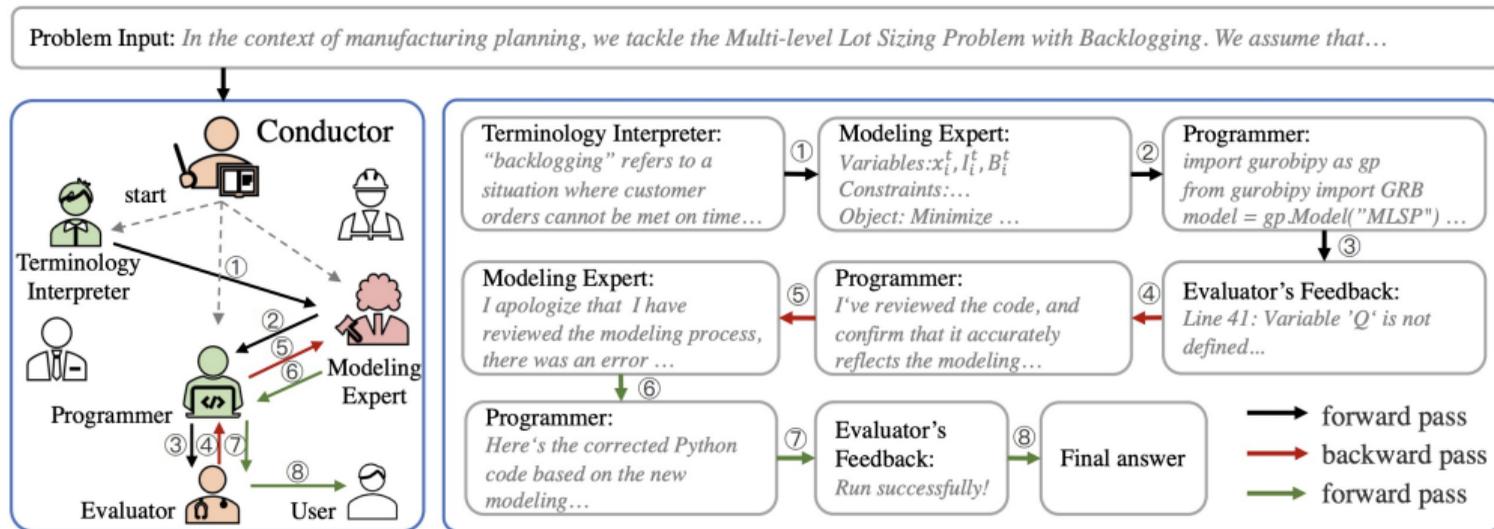
- The description clearly mentions that no more than a certain fraction of prices can be changed.
- The parameter MaxPriceChange is defined explicitly for this purpose.
- While implementing this constraint does add some complexity to the model, omitting it might lead to solutions that violate the business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

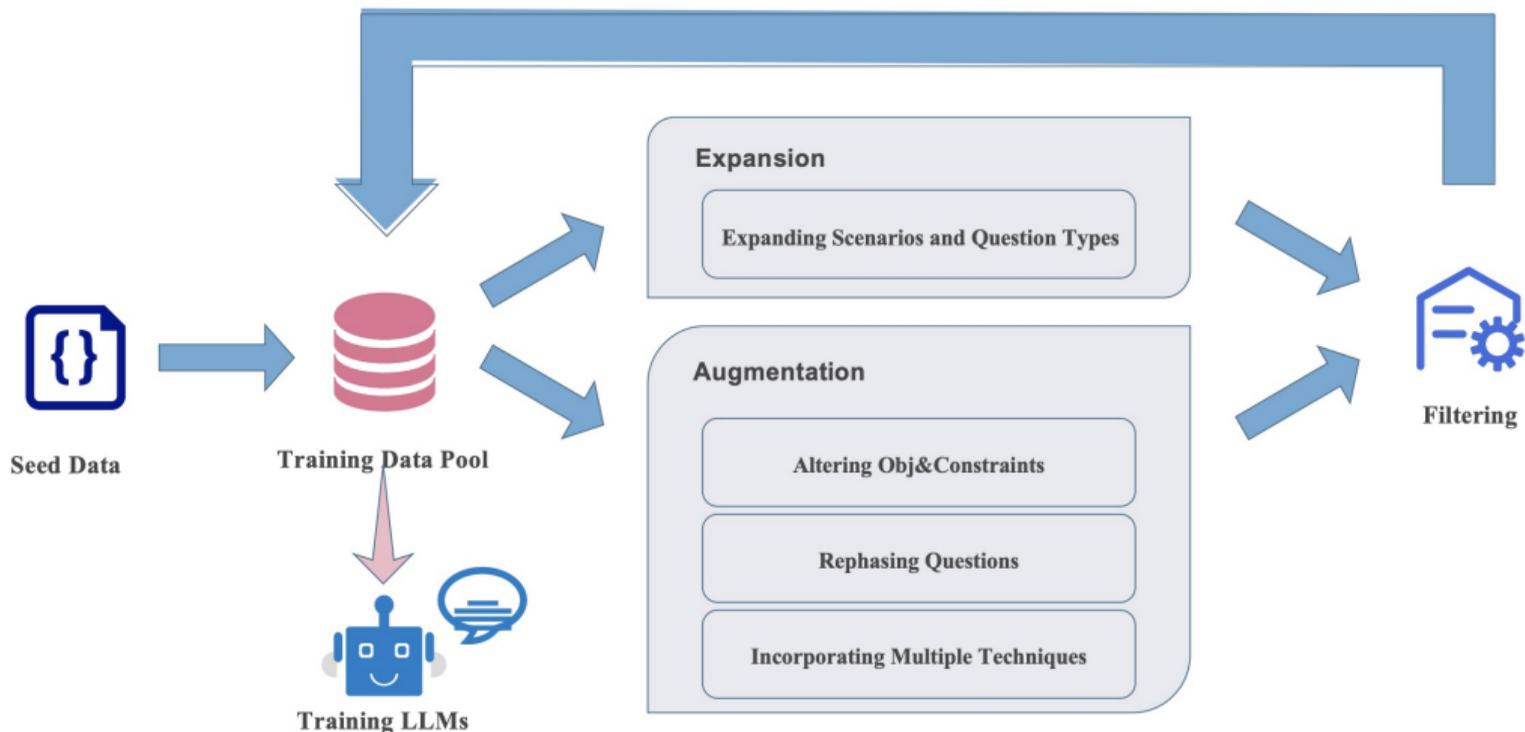
GPT4o can correct Llama's error.

# Chain of Experts: an agentic system



source: Xiao et al., 2023

# ORLM: pure fine-tuning



source: Tang et al., 2024

## Ablation study

	Easy	Hard
<b>Importance of Different Components</b>		
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	88.6%	68.4%
<b>OptiMUS-0.3</b>	<b>88.6%</b>	<b>73.7%</b>
<b>Performance with Different LLMs</b>		
LLaMa3-70B	77.3%	53.3%
<b>GPT4-o</b>	<b>88.6%</b>	<b>73.7%</b>

- ▶ 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?

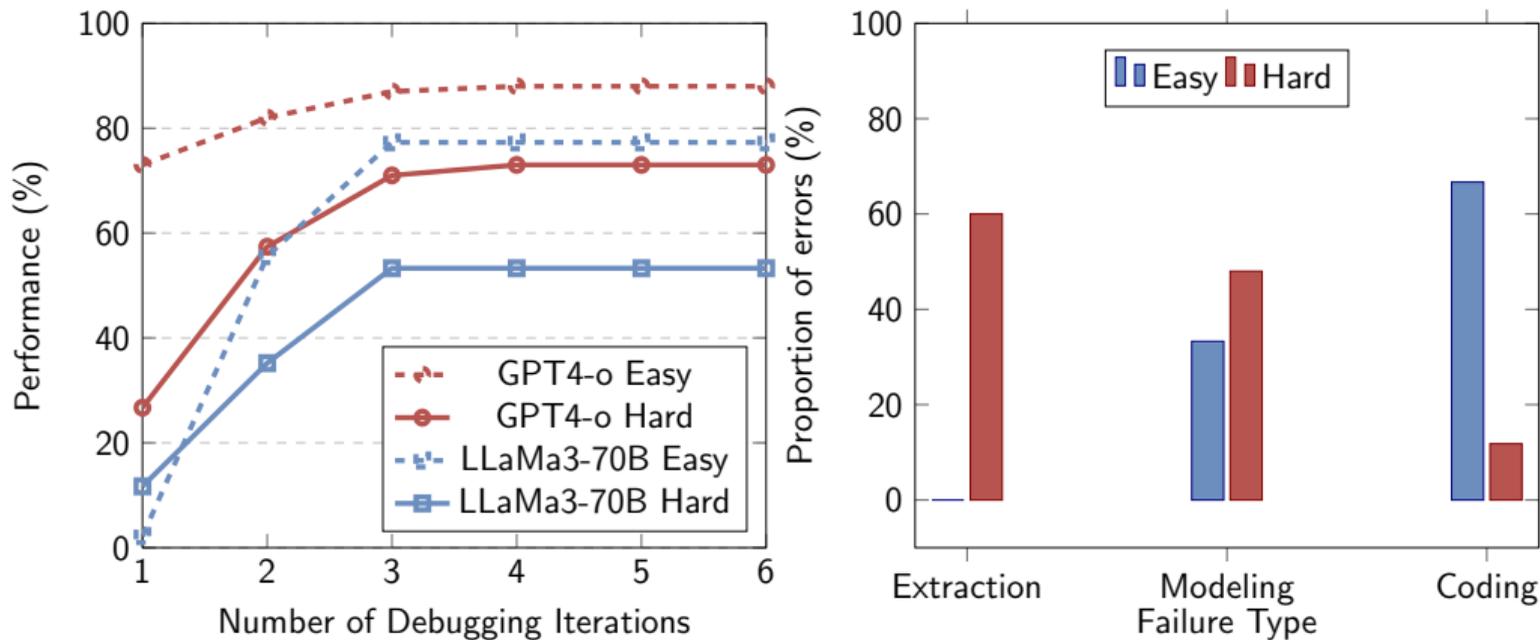


Figure: Left) Debugging improves performance. Right) For harder problems, most failures arise from clause extraction. For easier problems, most failures are due to coding.

## 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		
	Not Modified	Modified
Right	219	7
Wrong	9	41

Modeling		
	Not Modified	Modified
Right	231	2
Wrong	4	22

## Longer problems are still more challenging

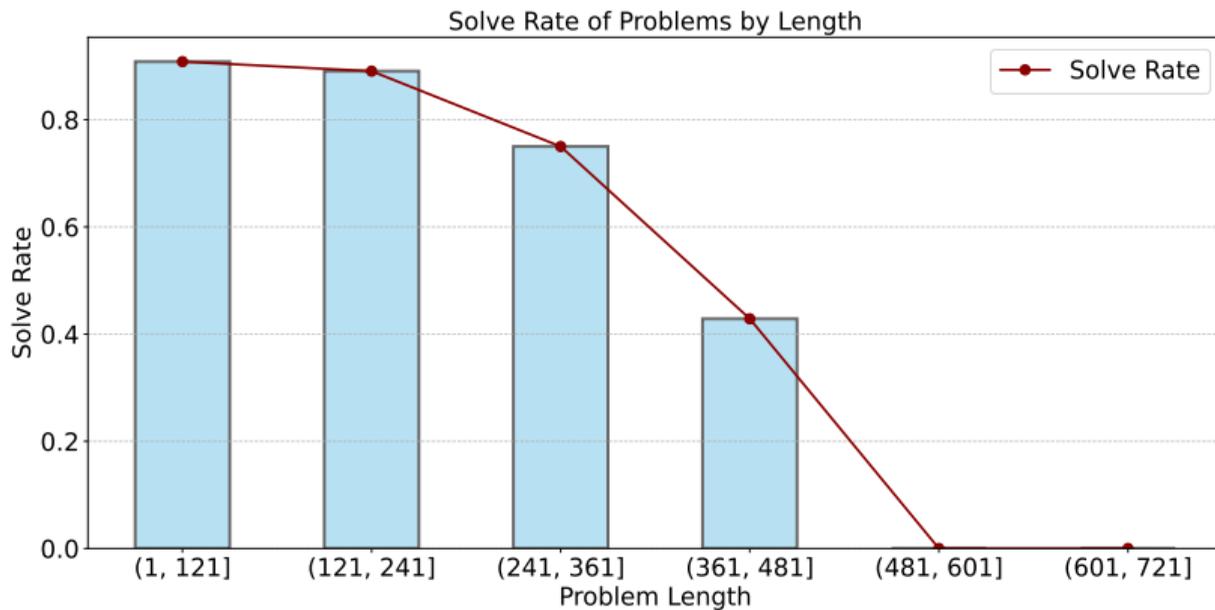


Figure: Solve rate vs. length of problem description

## 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/>
- 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., Barekatin, 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>
- 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>

## References (cont.)

- 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.