AI and the future of optimization modeling

Madeleine Udell

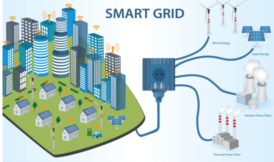
Management Science and Engineering
Stanford University

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

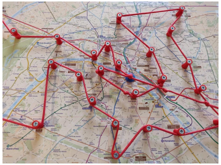
January 14, 2026

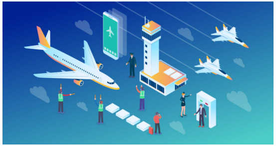
Outline

Optimization is everywhere



Energy

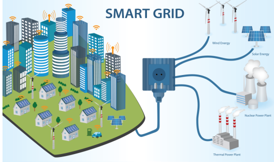

Routing


Scheduling

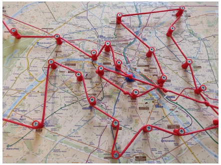

Supply chain

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

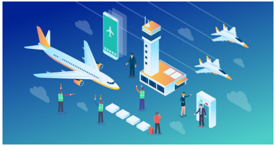
Optimization is everywhere



Energy



Routing

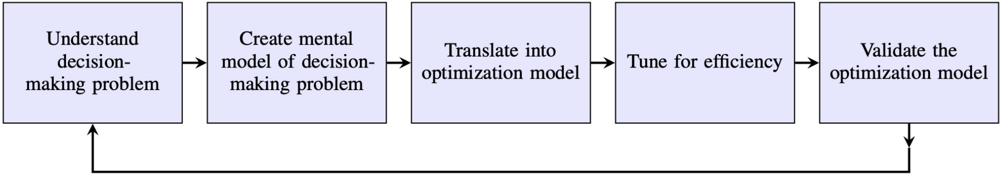


Scheduling

Supply chain

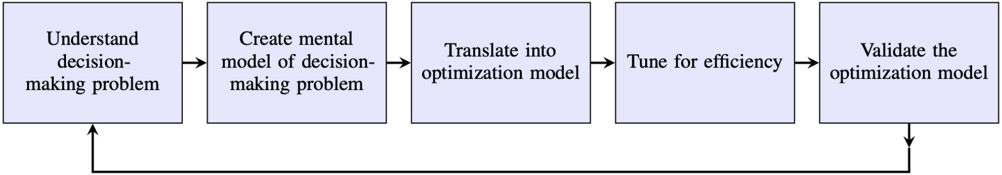
- Optimization improves efficiency throughout the economy
- $\implies$ more productivity, less waste, lower costs, lower carbon, more utility
- What limits the use of optimization?

Optimization modeling is the bottleneck



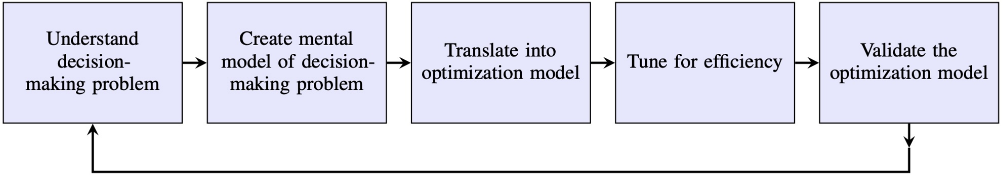
- problems can be long and complex, eg, this 64 page power systems problem

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.

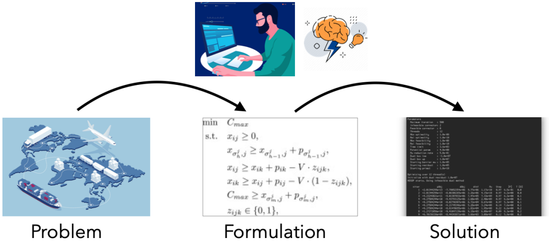
Example model: pricing

```python
# Define model
model = gp.Model('model')


# ====== Define variables ======
PriceChange = model.addVars(N, vtype=gp.GRB.CONTINUOUS, name="PriceChange")
QuantitySold = model.addVars(N, vtype=gp.GRB.CONTINUOUS, name="QuantitySold")
IndicatorFunctionPriceChange = model.addVars(N, vtype=gp.GRB.BINARY, name="IndicatorFunctionPriceChange")
Multiplier = model.addVars(N, vtype=gp.GRB.INTEGER, name="Multiplier")

# ====== Define constraints ======

# Add constraint - maximum number of articles with price changes
model.addConstr(gp.quicksum(IndicatorFunctionPriceChange[i] for i in range(N)) <= N * MaxChangeProportion, name="max_price_change_proportion")

# Add constraints to ensure quantity sold does not exceed adjusted forecast sales
for n in range(N):
    model.addConstr(QuantitySold[n] <= SalesForecast[n] * (1 - Elasticity[n] * (PriceChange[n] / InitialPrice[n])), name=f"sales_limit_{n}")

# Add non-negative price constraints
for i in range(N):
    model.addConstr(PriceChange[i] >= -InitialPrice[i], name="non_negative_price")

# Add constraints for price changes in multiples of MinimumChange
for i in range(N):
    model.addConstr(PriceChange[i] == Multiplier[i] * MinimumChange, name="price_change_multiples")

# Add constraint to limit price change occurrences
max_article_changes = MaxChangeProportion * N
model.addConstr(IndicatorFunctionPriceChange.sum() <= max_article_changes, "limit_price_changes")

# Add maximum production volume constraints for each article
for i in range(N):
    model.addConstr(QuantitySold[i] <= MaxVolume[i], name="max_volume")

# Add the constraint for forecasted quantity sold of each article adjusted for price change and elasticity of demand
for i in range(N):
    model.addConstr(QuantitySold[i] == SalesForecast[i] * (1 + Elasticity[i] * (PriceChange[i] / InitialPrice[i])), name="demand_elasticity")

# ====== Define objective ======

# Set objective
model.setObjective(gp.quicksum((InitialPrice[i] + PriceChange[i]) * SalesForecast[i] for i in range(N)), gp.GRB.MAXIMIZE)
```

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

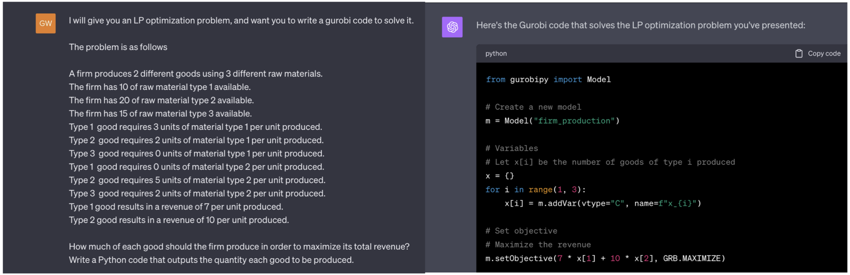
Optimization at scale

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



- (+) 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

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 ("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 (Ramamonjison 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 (Ramamonjison 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)
- 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:

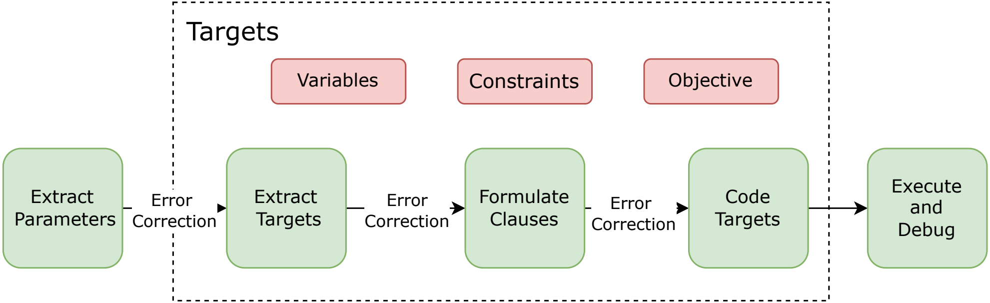
- code runs, but the result is incorrect
- some constraints are ignored
- variable doesn't match desired interpretation (eg, $\text{AbsPrice} \neq |\text{Price}|$)

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

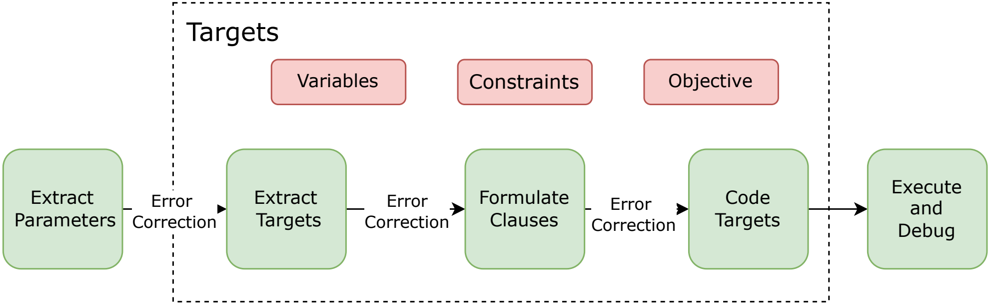


- paper: https://arxiv.org/abs/2407.19633
- code: https://github.com/teshnizi/OptiMUS

OptiMUS-0.3

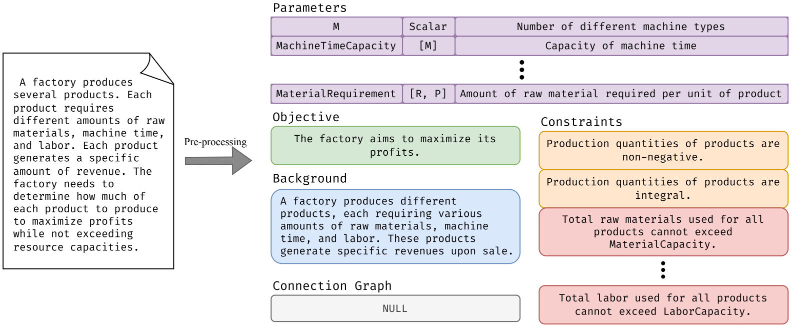
OptiMUS: Optimization Modeling Using Solvers

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



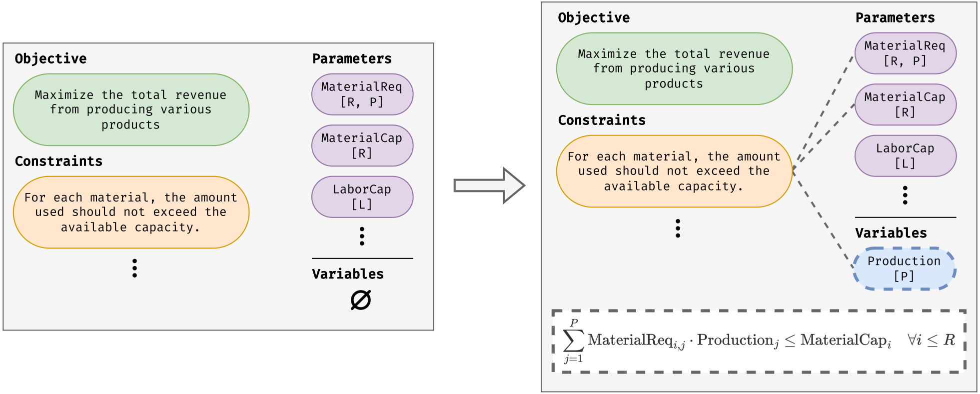
- paper: https://arxiv.org/abs/2407.19633
- code: https://github.com/teshnizi/OptiMUS
- Optimus-0.2: AhmadiTeshnizi, Gao, and Udell, ICML 2024

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.	$\sum_{i=1}^{P}$ Production_i $\times$ 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.	$\forall j$: $\sum_{i=1}^{P}$ Production_i $\times$ MaterialReq_j $\leq$ 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.	$\forall i$: Production_i ≥ 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?

GPT-4o

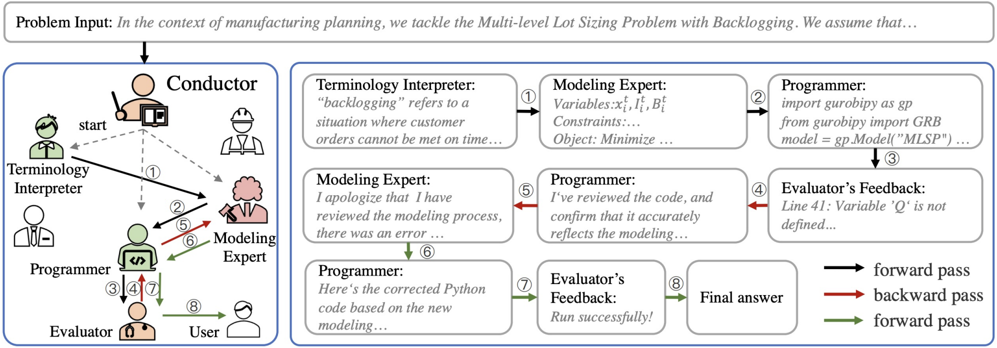
The provided parameter, MaxPriceChange, is supposed to represent the fraction of prices that can change. Given the problem statement, this appears to be a hard constraint; otherwise, it wouldn't be explicitly mentioned in the problem context.

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.**

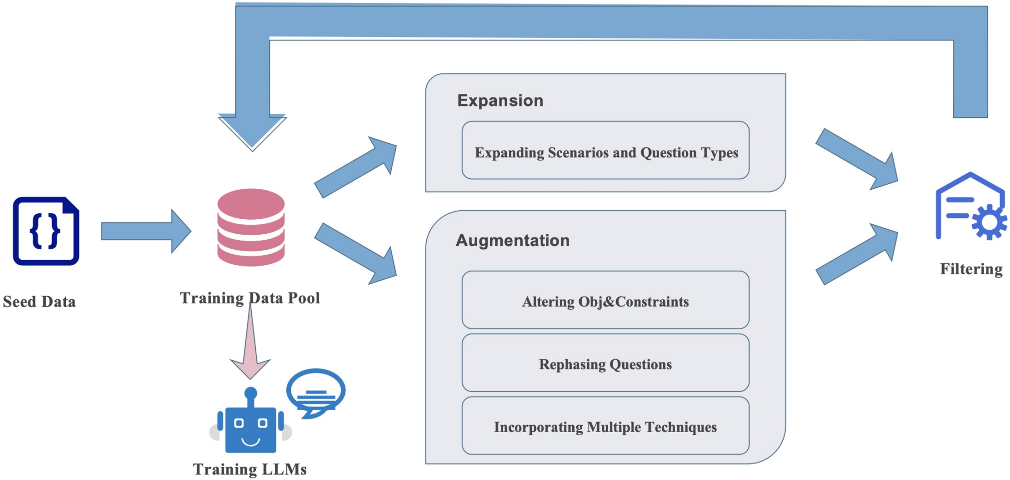
GPT4o can correct Llama's error.

Chain of Experts: an agentic system



source: Xiao et al., 2023

ORLM: pure fine-tuning



source: Tang et al., 2024

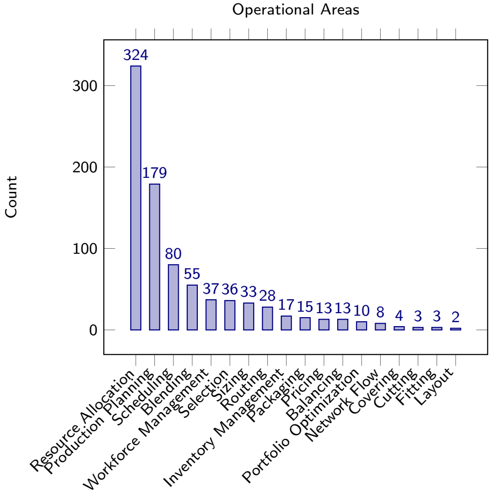
Outline

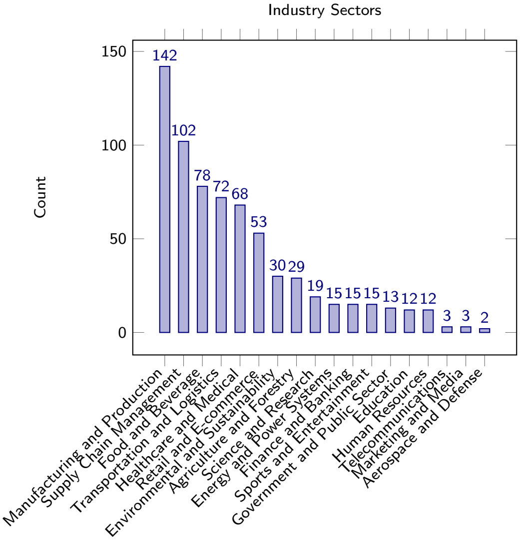
Experiments and dataset

Table: Benchmark datasets for optimization modeling

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

Diversity of problems





Performance

	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%

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?

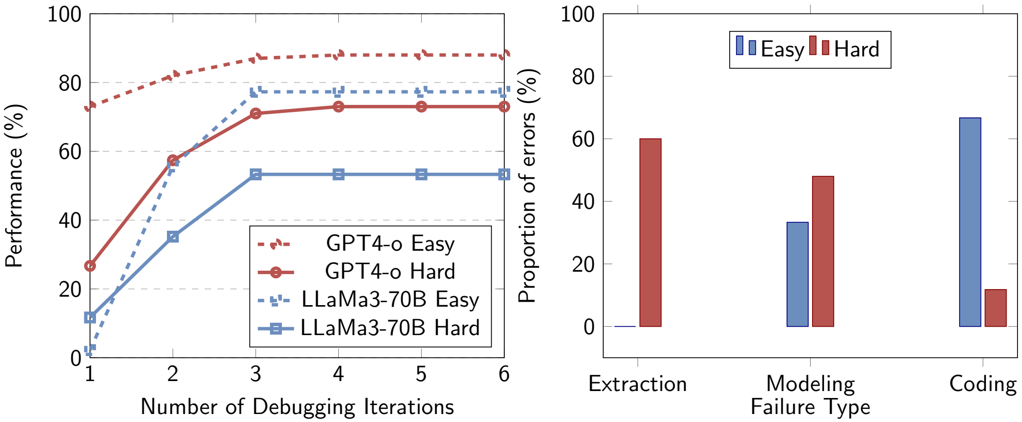


Figure: Left) Debugging improves performance. Right) For harder problems, most failures arise from clause extraction. For easier problems, most failures are due to coding.

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

	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

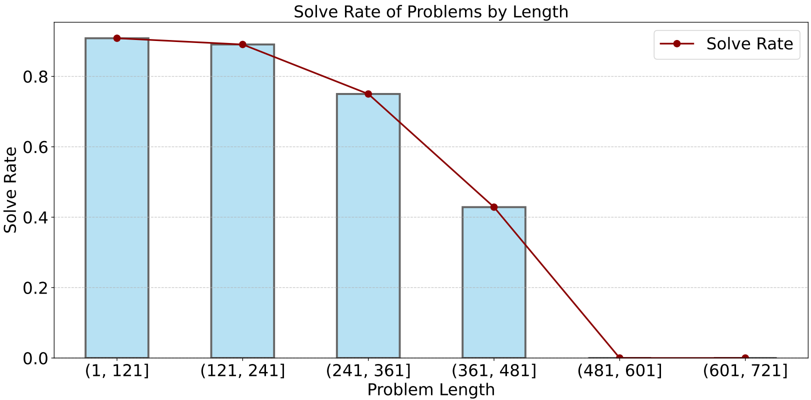


Figure: Solve rate vs. length of problem description

Outline

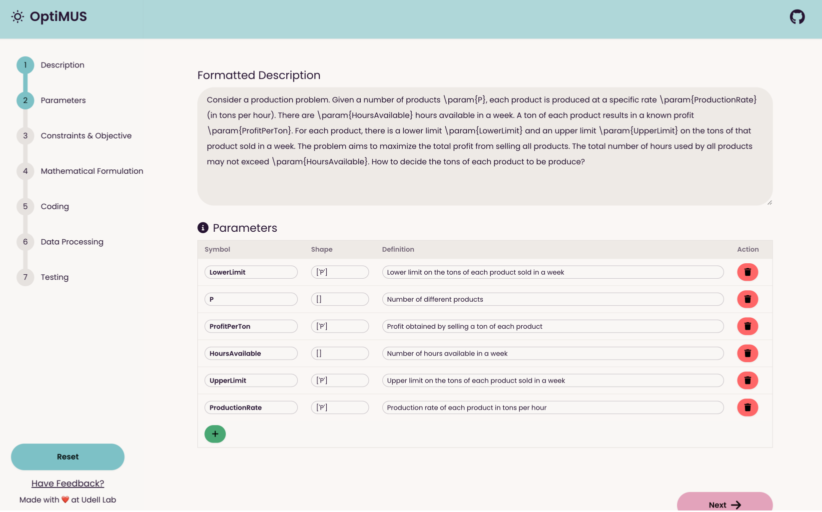
Example: production problem

A production planning problem:

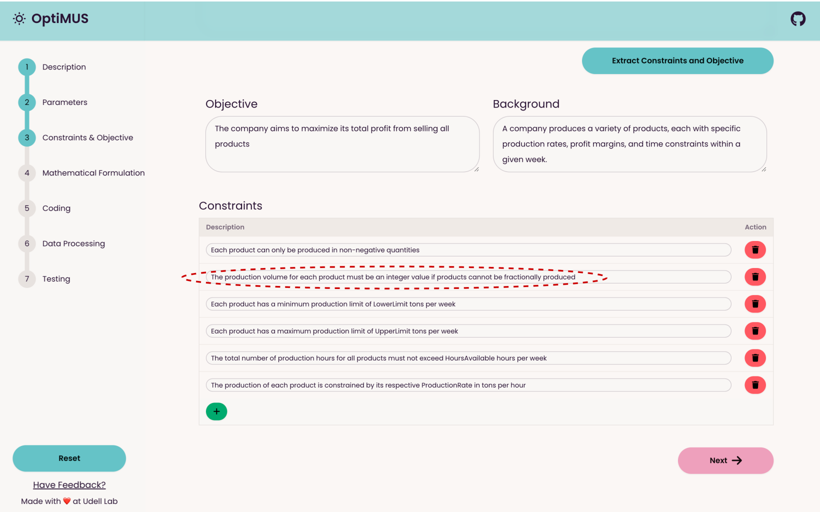
- Given a number of products, each product is produced at a specific rate (in tons per hour).
- There are a number of hours available in a week.
- A ton of each product results in a known profit.
- For each product, there is a lower limit and an upper limit on the tons of that product sold in a week.
- 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

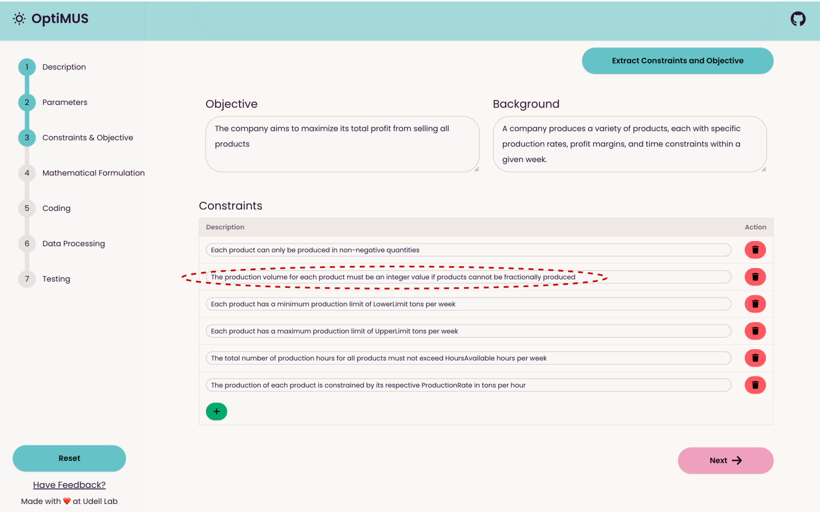


Example: hallucinated constraint



"The production volume for each product must be an integer value"

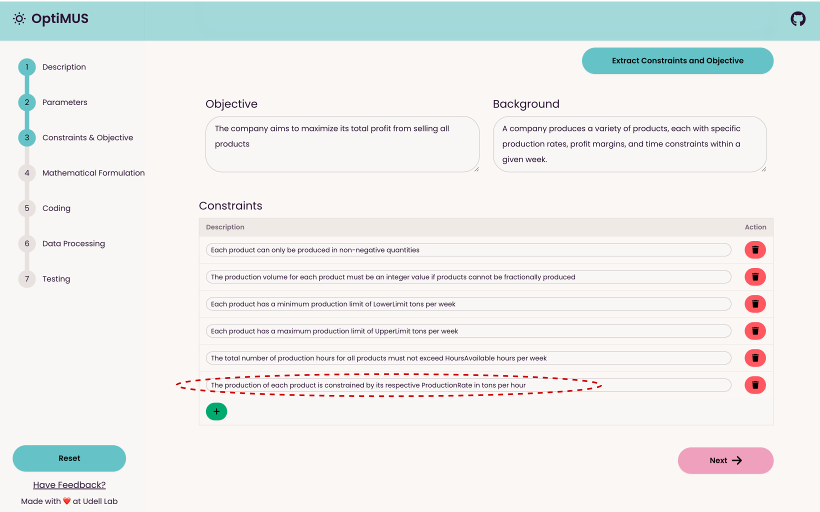
Example: hallucinated constraint



"The production volume for each product must be an integer value"

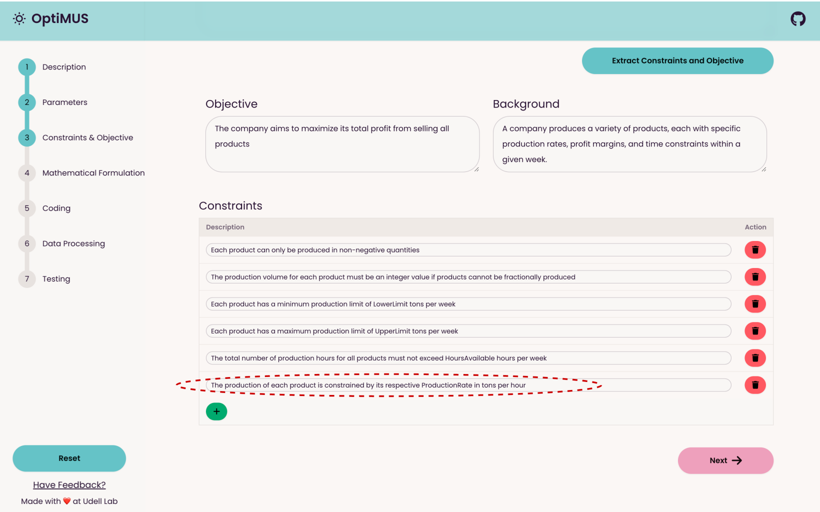
No integrality constraint is explicitly mentioned, nor is it necessary from context.

Example: spurious constraint



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

Example: spurious constraint

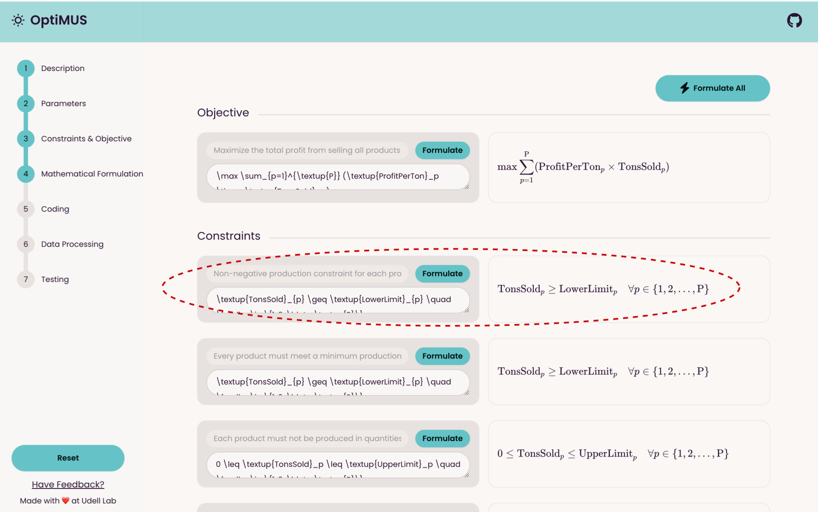


"The production for each product is constrained by its respective production in tons per hour" This is not be a constraint! It is enforced automatically as we use the

Example: incorrect coding

Constraint: "Non-negative production constraint for each product"

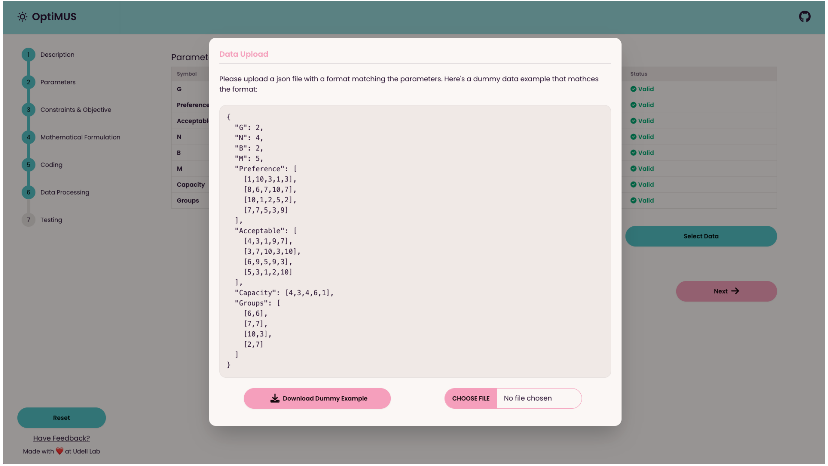
Example: incorrect coding



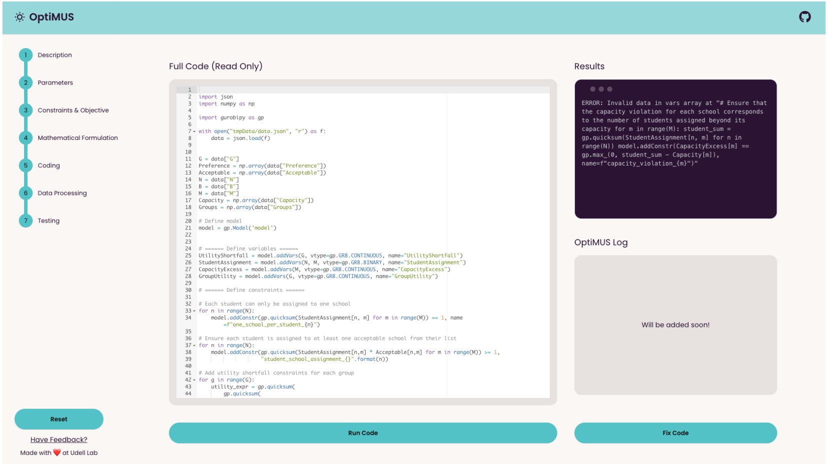
Constraint: "Non-negative production constraint for each product"

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

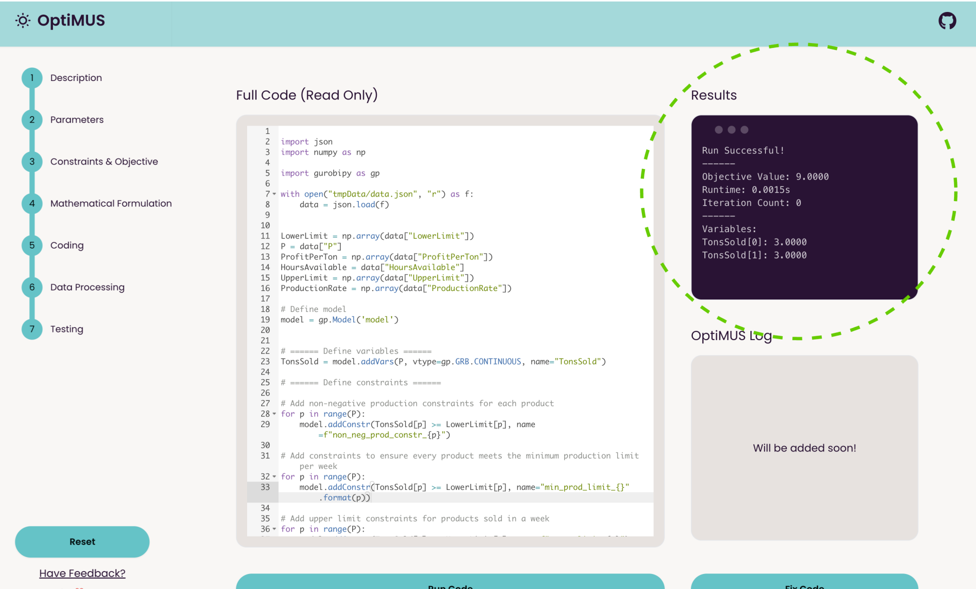
Example: data upload



Example: testing



Example: success!



Outline

Teaching LLM advanced optimization

A good optimizer exploits structure. Can an LLM?

Modeling features

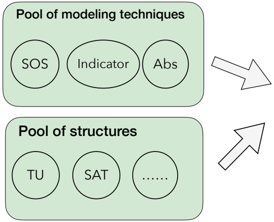
- Special Ordered Set (SOS)
- Indicator variables
- General constraints (norm, abs)
- ...

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!



I have an optimization problem and here's the description

> I have a network with a set of edges and vertices. The network has a source and a sink, and I also have information of the edge connectivity, say C_i = 1 if vertex i and j are connected

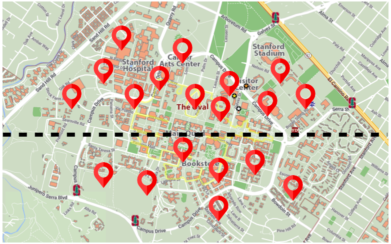
formulated as follows

$$
\begin{array}{lll}
\text{minimize} & \sum_{(i,j)\in E} u_{ij} q_{ij} & \\
\text{subject to} & q_{ij} = \max(p_j - p_i, 0), & \forall (i,j) \in E, \\
& p_d - p_s = 1, & \\
& p_i \in \{0,1\}, & \forall i \in V.
\end{array}
$$

> We know there is a type of mixed integer programming satisfying totally unimodularity, which we describe as follows
>
> Integer programming problems satisfying total unimodularity often come from the following background.
>
> 1. **Network Flow Problems**: The constraint matrices of network flow problems, such as the maximum flow problem and the minimum cost flow problem, are totally unimodular.
> 2. **Assignment Problems**: The constraint matrices for assignment problems, where tasks are to be assigned to agents
> 3. ……

> If the problem satisfies the aforementioned structure, please update the formulation accordingly

Do advanced optimization techniques help?

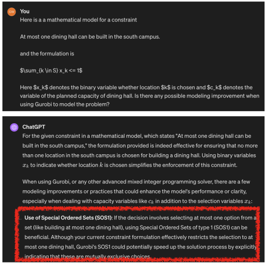


- Stanford plans to build new dining halls on campus.
- It costs Stanford $c_j = \alpha_j C$ to build a dining hall of capacity C at candidate location $j = 1, \ldots, K$.
- Each campus residence houses n_i students, $i = 1, \ldots, I$.
- No more than one hall will be built in north campus and one in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



Common MILP formulation

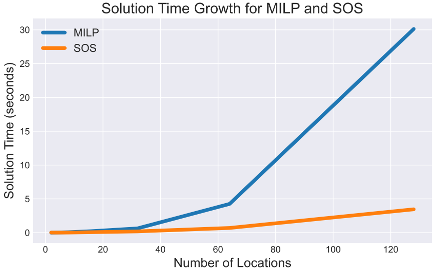
- x_k: whether location k is chosen
- $\sum_{k \in S} x_k \leq 1$

SOS formulation

- c_k: capacity at location k
- $(c_1, \ldots, c_k) \in \mathrm{SOS}_1$

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). Nl4opt 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.