Grammar-Aware Literate Generative Mathematical Programming with Compiler-in-the-Loop

Roberto Rossi, Steven D. Prestwich

UNIVERSITY OF EDINBURGH
Business School

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

GENERATIVE AI LABORATORY

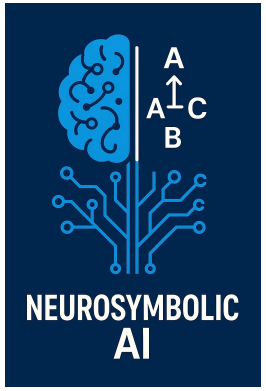
Insight

Artificial Intelligence

Reasoning

Learning

Raphael, detail of Plato and Aristotle, *School of Athens*, 1509-1511, fresco (Stanza della Segnatura, Palazzi Pontifici, Vatican)



”Neurosymbolic AI" combines:

- the **pattern-recognition & natural language capabilities** of neural networks; and
- the **structured reasoning and logic** of symbolic systems

The Future Is Neuro-Symbolic: Where Has It Been, and Where Is It Going?

Vaishak Belle[1], **Gary Marcus**[2]

[1]University of Edinburgh
vaishak@ed.ac.uk
[2]New York University (Emeritus)

Abstract

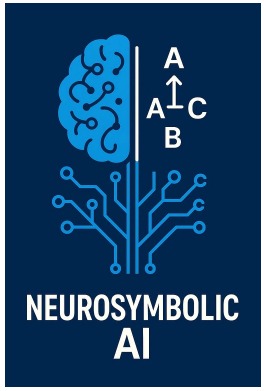
This report explores the evolution and current state of neuro-symbolic artificial intelligence, an approach that integrates neural network capabilities with symbolic reasoning. We trace the historical context from early AI aspirations to modern implementations and successes, highlighting key paradigms, and other logical and semantical considerations. We argue against the “scaling is all you need” hypothesis, and point to persistent challenges in reliable symbolic reasoning with deep and large models. We conclude by suggesting that despite numerous implementation choices and the “broad church” nature of neuro-symbolic AI, these approaches offer the most promising path towards AI systems that combine pattern recognition with robust reasoning, particularly for applications requiring structured knowledge, explainability, and trustworthiness.

Introduction

In its original inception, the field of artificial intelligence (AI) was deeply concerned with how human cognition could be automated on a computer (Turing 1950). John McCarthy, for example, argued for the development of so-called *common-sense programs* that could reason and problem solve, using the mathematical apparatus of logic for formalising the application domain. The idea was arguably

To deal with possibly false facts, and uncertainty in general, *probabilistic logics* emerged (Bacchus 1990; Halpern 2003), that allowed for a mixture of probabilistic and logical assertions. However, because they inherited the expressive power of (often first-order) logic but then further extended it for probabilistic knowledge, they too suffered from scalability issues. And they largely glossed over the point of where the probabilities came from. Presumably, these need to be drawn from data, but integrating the learning of statistical information and ensuring its consistency with prior logical knowledge is a non-trivial matter (Valiant 1999).

When limited to graphs, however, Bayesian (and later causal) networks offered a reasonable compromise between expert knowledge together with probabilistic and statistical information (Pearl 1998), and were amenable to certain types of learning from data (Koller and Friedman 2009). Building on this success, the field of statistical relational learning (SRL) aimed to unify logical and probabilistic frameworks by controlling the expressiveness (Raedt et al. 2016), such as by investigating finite-domain relational extensions to Bayesian networks (Koller and Pfeffer 1997; Getoor et al. 2001). Be that as it may, a purely expert-driven paradigm for dealing with high-dimensional data is unlikely to succeed – unless they outsource data-heavy computation to neural networks.



”Neurosymbolic AI" combines:

- the **pattern-recognition & natural language capabilities** of neural networks; and
- the **structured reasoning and logic** of symbolic systems

Leveraging LLMs. Exploiting and augmenting LLMs for symbolic reasoning tasks constitutes an entirely new class of approaches. When used as blackbox helper functions, systems like Logic-LM (Pan et al. 2023) employ LLMs to translate natural language into logical formulas, after which symbolic executors compute solutions directly. This serves as an effective countermeasure against LLM confabulations, particularly for mathematical tasks and puzzles. A pre-ChatGPT solution making a similar argument can be found in (Dries et al. 2017), where combinatorial problems formulated in natural language are converted to symbolic constructs solved using logic programming. Notably, Wolfram Alpha implemented a comparable pipeline just as ChatGPT was released (Wolfram 2023). These symbolic executor pipelines demonstrate considerable power; they can correctly handle modal reasoning problems such as theory of mind (Tang and Belle 2024). Similarly, (Athalye et al. 2024) build symbolic world models from language models. Importantly, such symbolic guardrails may be essential, as LLMs consistently struggle with reasoning and planning tasks (Valmeekam et al. 2022).

IJCAI
International Joint Conferences on Artificial Intelligence Organization

Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI-25)
Survey Track

A Survey of Optimization Modeling Meets LLMs: Progress and Future Directions

Ziyang Xiao[1], **Jingrong Xie**[1], **Lilin Xu**[1], **Shisi Guan**[1], **Jingyan Zhu**[1], **Xiongwei Han**[2], **Xiaojin Fu**[2], **WingYin Yu**[2], **Han Wu**[2], **Wei Shi**[2], **Qingcan Kang**[2], **Jiahui Duan**[2], **Tao Zhong**[2], **Mingxuan Yuan**[2], **Jia Zeng**[2], **Yuan Wang**[3], **Gang Chen**[1] and **Dongxiang Zhang**[1,4*]

[1]The State Key Laboratory of Blockchain and Data Security, Zhejiang University

[2]Huawei Noah's Ark Lab

[3]School of Business, Singapore University of Social Sciences

[4]Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security

{xiaoziyang, zhangdongxiang, cg}@zju.edu.com, {hanxiongwei, rocket.yuwingyin}@huawei.com

https://www.ijcai.org/proceedings/2025/1192.pdf

Optimisation Modelling meets LLM

Input: problem description

Power generation units are grouped into three distinct types, with different characteristics for each type (**power output**, **cost per megawatt hour**, **startup cost**, etc.). A unit can be on or off, with a startup cost associated with **transitioning from off to on**, and power output that can fall anywhere between **a specified minimum and maximum value** when the unit is on. A **5-hour time horizon** is with an expected total power demand for each hour. The model decides which units to turn on, and when, in order to **satisfy demand for each time period**. The model also captures a **reserve requirement**, where the selected power plants must be capable of increasing their output, while still respecting their maximum output, in order to cope with the situation where **actual demand exceeds predicted demand**.

Output: modeling result

Parameters: $P_i^{min}, P_i^{max}, c_i, c_i^{SU}, D_i, R_t$

Variables: $u_{i,t}, \in \{0,1\}, x_{i,t} \geq 0, y_{i,t} \in \{0,1\}$

Constraints:

Power Limits for On Units: $P_i^{min} u_{i,t} \leq x_{i,t} \leq P_i^{max} u_{i,t}, \quad \forall i \in I, \forall t \in T$

Demand Satisfaction: $\sum_{i \in I} x_{i,t} \geq D_t, \quad \forall t \in T$

Reserve Requirement: $\sum_{i \in I} (P_i^{max} - x_{i,t}) \geq R_t$

Startup Definition: $y_{i,t} \geq u_{i,t} - u_{i,t-1}, \quad \forall i \in I, \forall t \in T$

Objective: *minimize* $\sum_{t=1}^{T} \sum_{i \in I} (c_i x_{i,t} + c_i^{SU} y_{i,t})$

Figure 1: An example of an optimization modeling task. The orange text in the problem description implies domain-specific terminology, and the green text denotes implicit constraints.

The dynamic lot sizing problem

A company plans its inventory over several periods (e.g. weeks or months). For each period, it knows exactly how much demand it will face. The company can place orders at the beginning of any period, but each order comes with a fixed cost, regardless of its size. Additionally, if the company orders more than it needs for the current period, it must hold the excess inventory, which incurs a per-unit holding cost. The challenge is to decide: in which periods to place orders; and how much to order in each of those periods. The goal is to meet all demand on time (no shortages allowed) while minimising the total cost, which includes both ordering and holding costs.

PyOPL mathematical programming model

```
// Uncapacitated Lot Sizing (Wagner-Whitin setting)
// Decision: choose order timing and size to minimize ordering + holding cost

// Number of periods (must be declared before use in range and elsewhere)
int N = ...;

range T = 1..N;                    // periods (1-based)

// Parameters
param float demand[T] = ...;       // known demand per period
param float K = ...;               // fixed cost per order
param float h = ...;               // per-unit holding cost per period

// Decision variables
dvar float+ x[T];        // x[t]: units ordered at period t (>=0)
dvar boolean y[T];       // y[t]: 1 if order is placed at period t, 0 otherwise
dvar float+ s[T];        // s[t]: inventory held after satisfying demand in period t (>=0)

// Order upper bound to tighten the big-M constraint, M >= total demand
param float M = sum(t in T) demand[t];

// Objective: minimize total cost (ordering + holding)
minimize total_cost:
    sum(t in T) (K * y[t]) + sum(t in T) (h * s[t]);

subject to {
    // inv_balance_1: Initial period inventory balance (initial inventory is zero)
    inv_balance_1: s[1] == x[1] - demand[1];

    // inv_balance_t: Inventory balance for periods >= 2
    forall(t in 2..N)
        inv_balance_t: s[t] == s[t-1] + x[t] - demand[t];

    // order_link: Only allow ordering if y[t]=1 (big-M linking for x[t])
    forall(t in T)
        order_link: x[t] <= M * y[t];

    // nonneg_inventory: No shortages allowed; inventory remains nonnegative
    forall(t in T)
        nonneg_inventory: s[t] >= 0;
}
```

Figure 1: A GenMP task — translating the "dynamic lot sizing" problem description into a mathematical programme.

2.1 Problem Definition

Optimization modeling transforms a problem description in natural language $\mathcal{P}$ into a model $\mathcal{M}$. Mathematically, an optimization model is defined by an objective and a set of constraints, as shown in Equation 1.

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, ..., m \\ & h_j(\mathbf{x}) = 0, \quad j = 1, ..., p \end{aligned} \tag{1}$$

Here, $\mathbf{x}$ is the vector of decision variables, $f(\mathbf{x})$ denotes the objective function, $g_i(\mathbf{x})$ and $h_j(\mathbf{x})$ represent the inequality and equality constraints respectively.

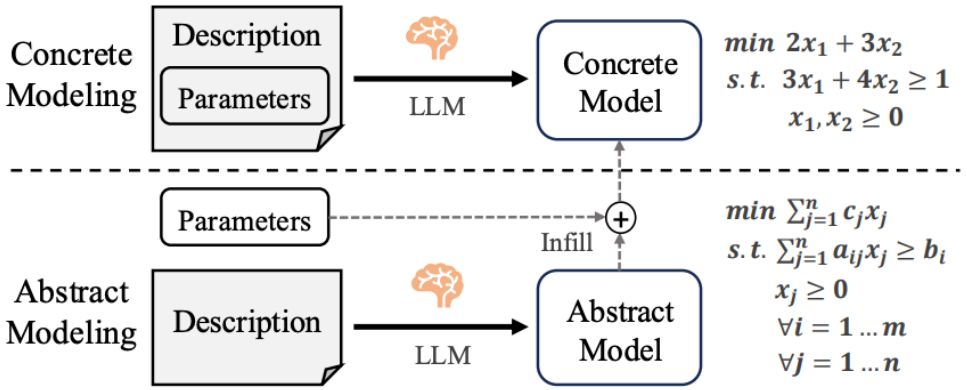
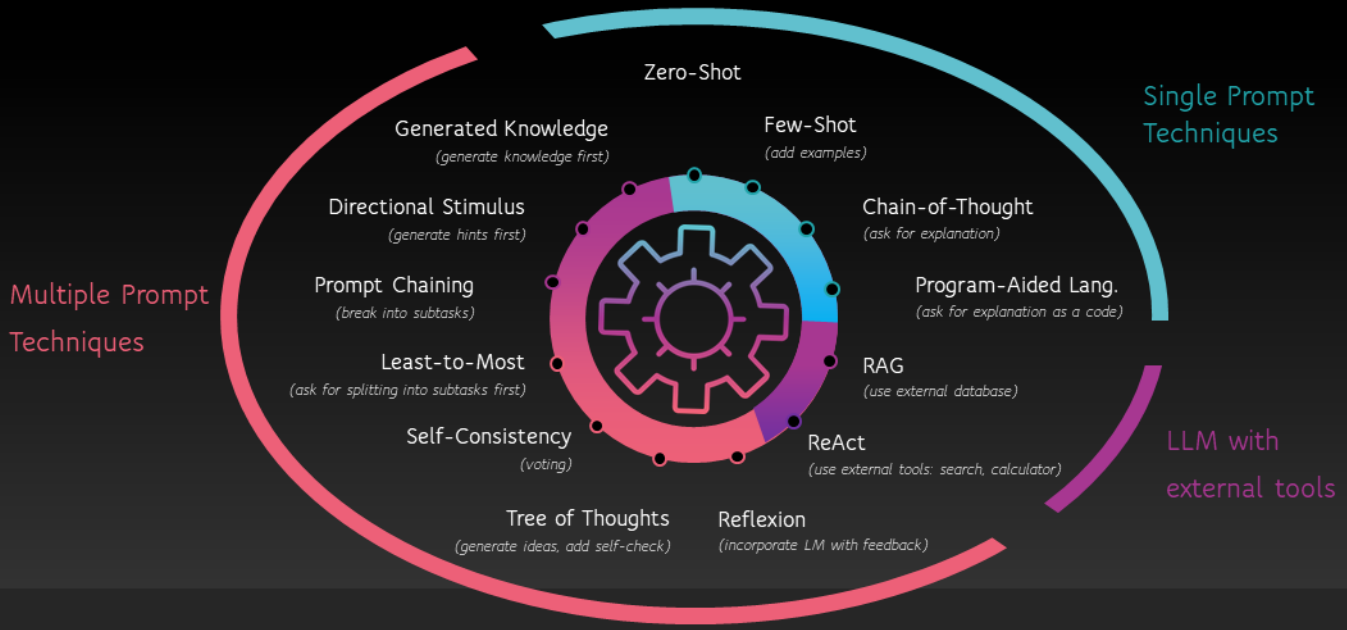


Figure 2: Comparison between concrete and abstract models. The right part illustrates a linear programming formulation example.

Prompt Engineering Techniques



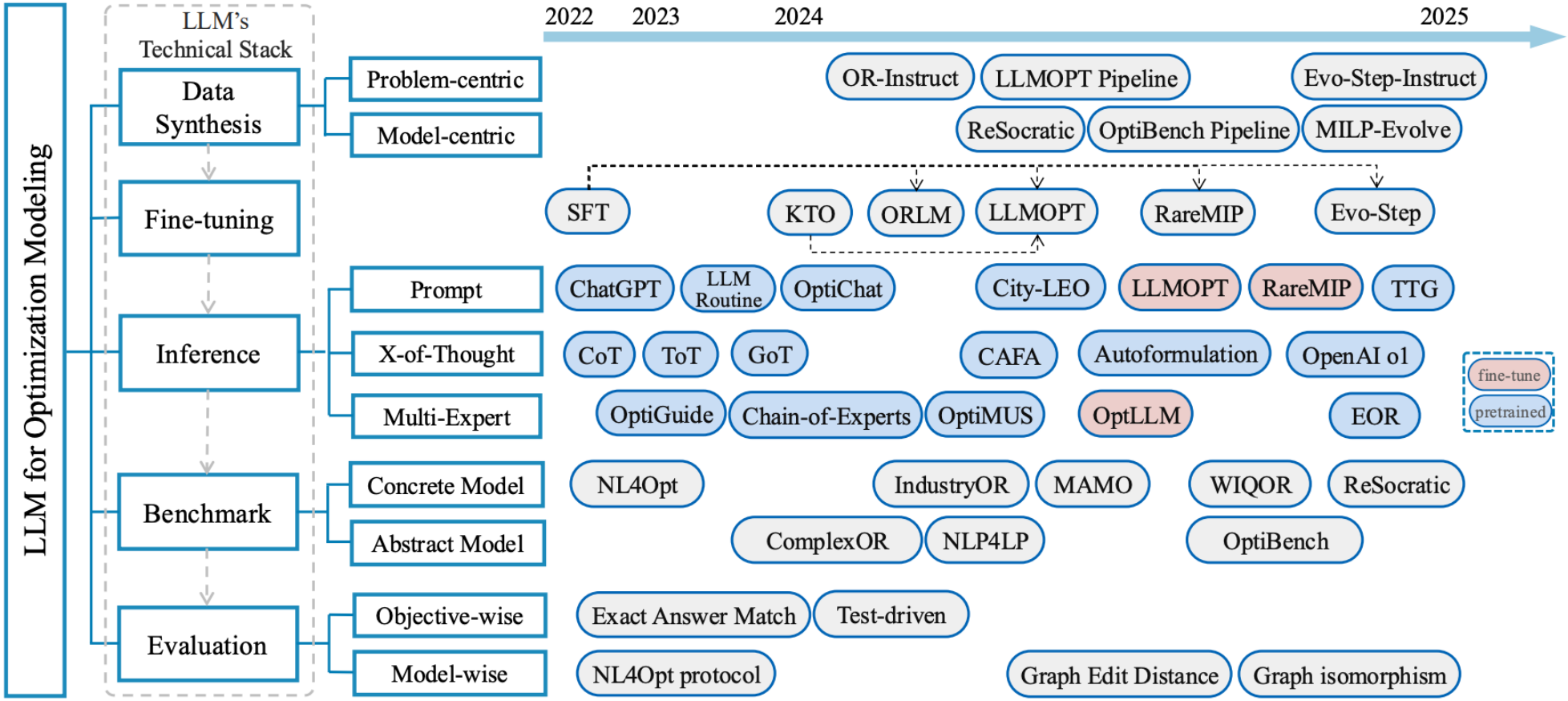


Figure 3: Left: Taxonomy of LLMs-based optimization modeling, organized according to the LLMs' technical stack. Right: Representative works for each category are presented in chronological order. The dashed arrows indicates where later works build upon techniques proposed in earlier studies.

Mathematics > Optimization and Control

[Submitted on 20 May 2026]

Large Language Models for Operations Research: A Comprehensive Survey

Xianchao Xiu, Jianhao Li, Jun Fan, Wanquan Liu

Operations Research (OR) serves as a core decision-support methodology for complex systems, with significant applications across mathematics, management science, and computer science. Traditional approaches heavily rely on expert knowledge and often struggle to efficiently solve large-scale and multi-constraint problems. The rapid advancement of Large Language Models (LLMs) in recent years has offered a novel research paradigm to address these challenges. This paper presents a systematic survey of Large Language Models for Operations Research (LLM4OR). We begin by introducing the definition of OR problems and the fundamental principles of LLMs. We then focus on analyzing the roles of LLMs in OR, specifically covering such as model formulation, algorithm design, and solution verification. In addition, we discuss practical applications in representative scenarios and summarize benchmark datasets in this field. Finally, we outline the key challenges and provide perspectives on future research directions. A collection of related literature is available at this https URL.

Subjects: **Optimization and Control (math.OC)**
Cite as: arXiv:2605.20849 **[math.OC]**
(or arXiv:2605.20849v1 **[math.OC]** for this version)
https://doi.org/10.48550/arXiv.2605.20849

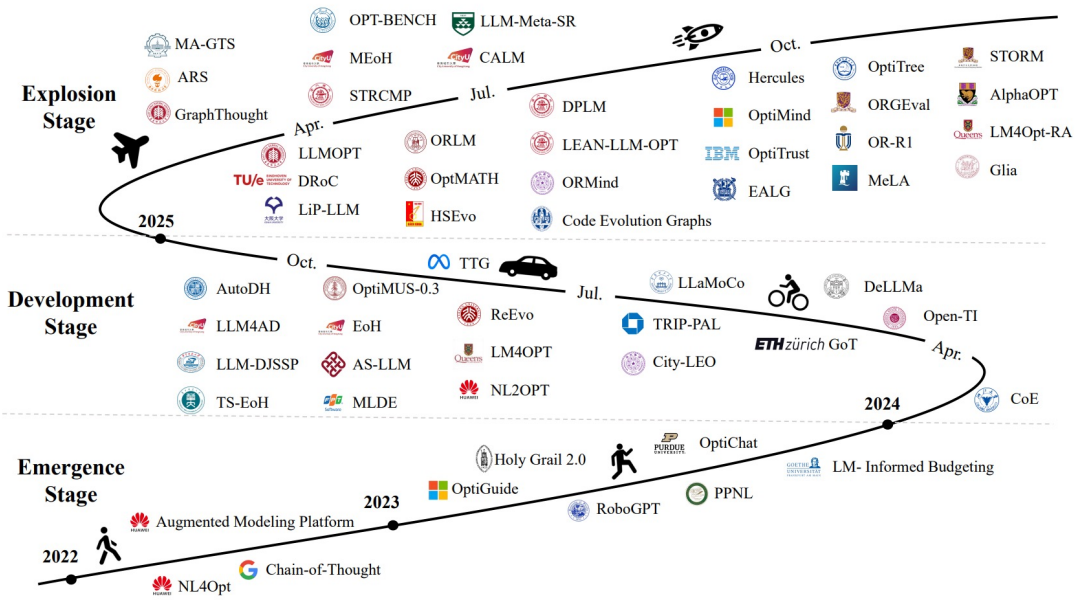


Fig. 1 Evolution timeline of LLM4OR (2022-2025).

Benchmarks

Dataset	Size	Complexity	Error Rate
NL4Opt	289	5.59	$\geq 26.4\%$
IndustryOR	100	**14.06**	$\geq$ **54.0%**
EasyLP	652	7.12	$\geq 8.13\%$
ComplexLP	211	**13.35**	$\geq 23.7\%$
ReSocratic	605	7.45	$\geq 16.0\%$
NLP4LP	269	5.58	$\geq 21.7\%$
ComplexOR	37	5.98	$\geq 24.3\%$

Table 1: Quality statistics of optimization modeling benchmarks.

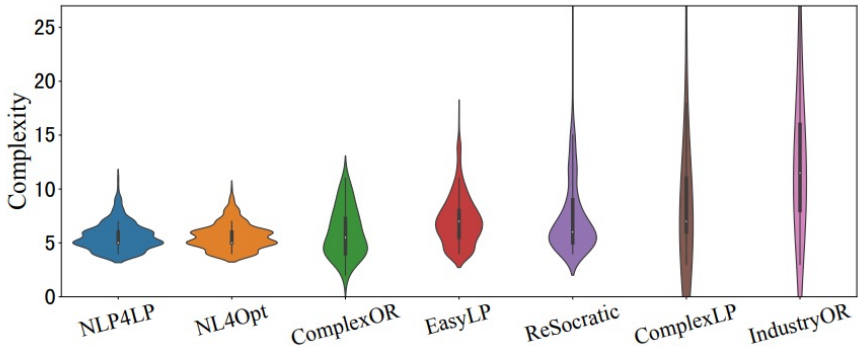


Figure 5: Statistics of complexity distribution for each benchmark visualized using a violin plot. X-axis shows different benchmarks, and y-axis shows the complexity indicator.

Datasets: CardinalOperations / IndustryOR

like 21 | Follow Cardinal Operations 29

Modalities: Text | Formats: json | Languages: English | Size: < 1K | ArXiv: arxiv:2405.17743 | Libraries: Datasets | pandas | Croissant + 1

License: cc-by-nc-4.0

Dataset card | Data Studio | Files and versions xet | Community 2

Dataset Viewer

Auto-converted to Parquet | API | Embed | Duplicate | Data Studio

Split (1)
test · 100 rows

Search this dataset

en_question string · lengths 377 5.99k	en_answer string · lengths 1 18	difficulty string · classes 3 values	id int64 1 100
A factory produces two types of food, I and II, and currently has 50 skilled workers. It is known…	219816.0	Medium	1
A fighter jet is a crucial combat tool, but to make it effective, enough trained pilots are…	125	Medium	2
A company specializing in foldable tables needs to create an optimal production and human resources…	10349920.0	Hard	3
A farmer needs to decide how many cows, sheep, and			

Downloads last month 195

Use this dataset

Size of downloaded dataset files: 115 kB

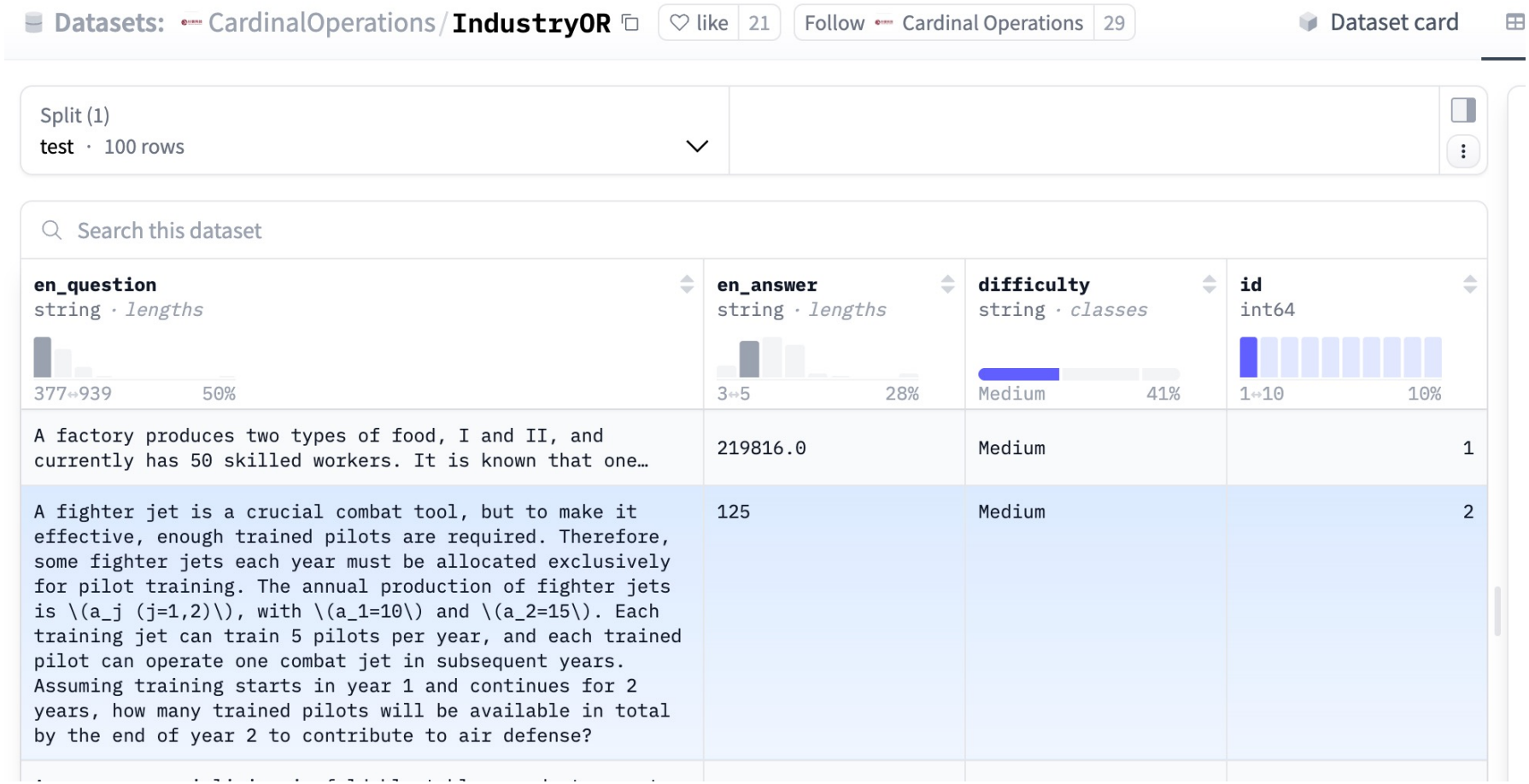
Size of the auto-converted Parquet files: 60.1 kB

Number of rows: 100

Paper for CardinalOperations/IndustryOR

ORLM: Training Large Language Models for Opti…

Paper • 2405.17743 • Published May 28, 2024 • 3


Datasets: CardinalOperations / IndustryOR
like 21
Follow Cardinal Operations 29
Dataset card
Split (1)
test · 100 rows
Search this dataset
en_question
string · lengths
377↔939 50%
en_answer
string · lengths
3↔5 28%
difficulty
string · classes
Medium 41%
id
int64
1↔10 10%
A factory produces two types of food, I and II, and currently has 50 skilled workers. It is known that one…
219816.0
Medium
1
A fighter jet is a crucial combat tool, but to make it effective, enough trained pilots are required. Therefore, some fighter jets each year must be allocated exclusively for pilot training. The annual production of fighter jets is \(a_j (j=1,2)\), with \(a_1=10\) and \(a_2=15\). Each training jet can train 5 pilots per year, and each trained pilot can operate one combat jet in subsequent years. Assuming training starts in year 1 and continues for 2 years, how many trained pilots will be available in total by the end of year 2 to contribute to air defense?
125
Medium
2

A sample problem from the ReSocratic benchmark


```
{
    "en_question": "You want to sell a kind of item in order
to maximize your profit. Market research tells you that if you
set the price at $1.50, you will be able to sell 5000 items,
and for every 10 cents you lower the price below $1.50 you
will be able to sell another 1000 items. You can only reduce
the price by a multiple of 10 cents. Suppose that your fixed
costs ( “start-up costs” ) total $2000, and the per item cost
of production ( “marginal cost” ) is $0.50. Find the price to
set per item and the number of items sold in order to maximize
profit, and also determine the maximum profit you can get.",
    "index": 0,
    "type": "linear-notable",
    "en_complete_answer": {
      "The maximum profit you can get": "3600.0"
    },
    "en_answer": "3600.0",
    "en_objective_key": "The maximum profit you can get"
  }
```

Methods	NL4Opt	IndustryOR	EasyLP	ComplexLP	NLP4LP	ReSocratic	ComplexOR
Standard	61.2%	38.1%	70.3%	57.7%	73.6%	48.4%	42.9%
CoT	62.2%	40.5%	49.5%	42.3%	74.7%	43.6%	39.2%
Chain-of-Experts	66.7%	31.2%	**94.4%**	50.6%	**87.4%**	**71.2%**	**57.1%**
CAFA	68.1%	41.1%	71.2%	44.5%	50.0%	40.1%	46.4%
ORLM-LLaMA-3 8B	**73.8%**	**42.9%**	90.4%	**59.5%**	76.4%	61.8%	50.0%

Table 2: Performance comparison of existing fully open-source methods on cleaned benchmarks in a unified setting (use GPT-4o for training-free methods and use accuracy as metric). All results are reproduced using our standardized evaluation method.

Computer Science > Programming Languages

[Submitted on 25 Jan 2026]

Grammar-Aware Literate Generative Mathematical Programming with Compiler-in-the-Loop

Roberto Rossi, Steven D. Prestwich

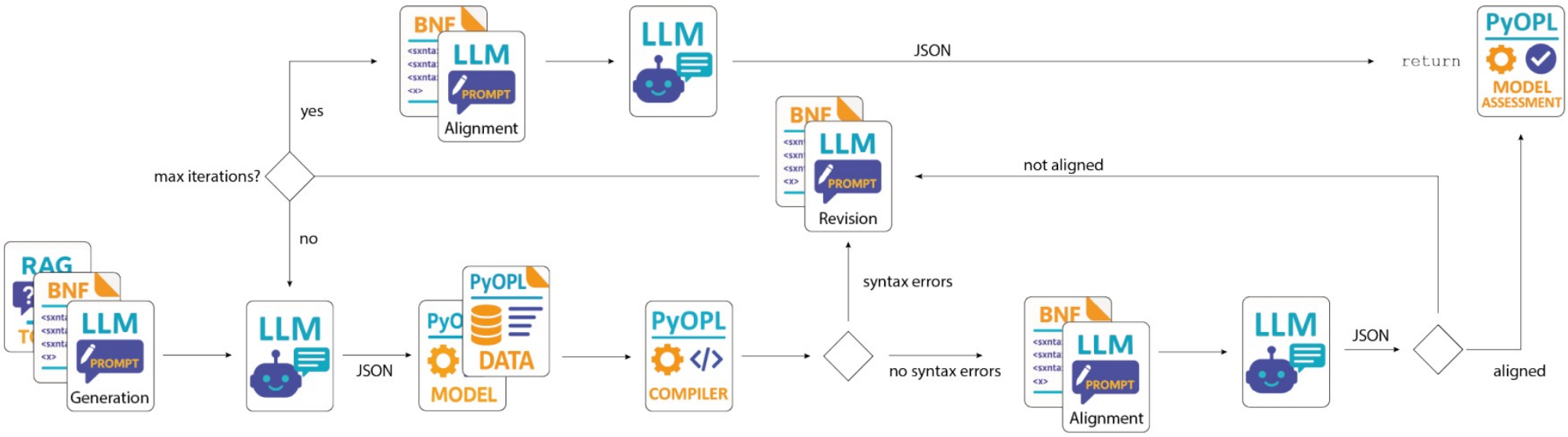
This work investigates generative mathematical programming through the lens of Algebraic Modelling Languages (AMLs) and compiler-guided model synthesis. By leveraging PyOPL, an OPL-like AML compiler that provides detailed syntax diagnostics, we introduce SyntAGM, an end-to-end system that translates natural language problem descriptions into PyOPL models via a generate--compile--assess--revise loop. SyntAGM is grammar-aware thanks to in-context exposure to the PyOPL BNF grammar, and benefits from few-shot retrieval of literate PyOPL model exemplars. To obtain a valid PyOPL model that matches the problem description, SyntAGM mobilises compiler feedback and an LLM-based alignment judge. In a comparative study against established prompting baselines SyntAGM achieves competitive accuracy with superior token, cost, and latency profiles.

Comments: 18 pages, 10 figures
Subjects: **Programming Languages (cs.PL)**; Artificial Intelligence (cs.AI)
Cite as: arXiv:2601.17670 **[cs.PL]**
(or arXiv:2601.17670v1 **[cs.PL]** for this version)
https://doi.org/10.48550/arXiv.2601.17670

Submission history

From: Roberto Rossi [view email]
[v1] Sun, 25 Jan 2026 03:19:49 UTC (5,329 KB)

Syntax-Aware Generative Modelling (SyntAGM)



F.1 Generation

```
<role>
You are an expert in mathematical optimisation and PyOPL.
</role>

<task>
Think step by step to produce a syntactically valid PyOPL
     model (.mod) and matching data (.dat) that faithfully
      implement the problem.
First, reason in a private scratchpad to identify sets,
     parameters, decision variables, objective, and
     constraints.
Ensure indices, domains (binary/integer/float), and data
     are correct and consistent with the problem
     description.
Choose correct domains (binary/integer/float) from context
     . Add clear labels and explanatory comments.
Label the objective and each constraint.
Add concise comments explaining variables, parameters, and
      constraints, aligned to the problem (literate style)
     .
If any data are missing, create a small, plausible mock
     instance consistent with the model.
</task>

<grammar_reference>
--- BEGIN PYOPL SYNTAX IMPLEMENTATION ---
{{GRAMMAR_IMPLEMENTATION}}
--- END PYOPL SYNTAX IMPLEMENTATION ---
</grammar_reference>

{{FEW_SHOT_EXAMPLES_SECTION}}

<problem_description>
{{PROMPT}}
</problem_description>

<output_requirements>
- Output ONLY the final JSON with the model and data; do
     not include your scratchpad in the output.
- Return ONLY a JSON object with keys "model" and "data".
     Values are single strings; escape quotes and
     backslashes; encode newlines as \n. No extra keys.
- You MAY wrap the JSON in a ```json fence containing only
      the JSON.
</output_requirements>

<json_schema>
{ "type": "object", "additionalProperties": false, "
     required": ["model","data"],
  "properties": { "model":{"type":"string"}, "data":{"type
       ":"string"} } }
</json_schema>

<example_output>
{ "model": "// minimal example\\nfloat a;\\nfloat b;\\
     ndvar float x;\\nminimize z: a*x;\\nsubject to {\\n
     c1: b*x >= 0;\\n}\\n", "data": "a = 10;\\n b = 5;" }
</example_output>
```

F.2 Revision

```
<role>
You are an expert in mathematical optimisation and PyOPL.
</role>

<task>
Revise the model/data to resolve the specified issues
     while preserving the intended formulation.
Change only what is necessary; keep syntax valid.
Label the objective and each constraint.
Add concise comments explaining variables, parameters, and
      constraints, aligned to the problem (literate style)
     .
Use the PyOPL reference strictly for syntax.
</task>

<revision_guidelines>
{{REVISION_GUIDELINE}}
- Make the minimal set of changes necessary to correct
     syntax/semantic errors.
- Preserve the original modeling structure when possible.
- Ensure the objective, constraints, indices, and variable
      domains reflect the problem description.
- Keep syntax strictly valid.
- Return complete model and data strings; do not return
     diffs.
</revision_guidelines>

<grammar_reference>
--- BEGIN PYOPL SYNTAX IMPLEMENTATION ---
{{GRAMMAR_IMPLEMENTATION}}
--- END PYOPL SYNTAX IMPLEMENTATION ---
</grammar_reference>

{{FEW_SHOT_EXAMPLES_SECTION}}

<problem_description>
{{PROMPT}}
</problem_description>

<previous_attempt>
<model>
{{MODEL_CODE}}
</model>

<data>
{{DATA_CODE}}
</data>
</previous_attempt>

<compiler_errors>
{{COMPILER_ERRORS}}
</compiler_errors>

<alignment_assessment>
{{ASSESSMENT}}
</alignment_assessment>

<output_requirements>
- Return ONLY a JSON object with keys "model" and "data".
     Values are single strings; escape quotes/backslashes;
      encode newlines as \n. No extra keys.
- You MAY wrap the JSON in a ```json fence containing only
      the JSON.
</output_requirements>

<json_schema>
{ "type":"object", "additionalProperties": false, "
     required":["model","data"],
  "properties": { "model":{"type":"string"}, "data":{"type
       ":"string"} } }
</json_schema>

<example_output>
{
  "model": "// minimal example\\nfloat a;\\nfloat b;\\
       ndvar float x;\\nminimize z: a*x;\\nsubject to {\\n
        c1: b*x >= 0;\\n}\\n",
  "data": "a = 10;\\n b = 5;"
}
</example_output>
```

F.3 Alignment

```
<role>
You are an expert in mathematical optimization and PyOPL.
</role>

<task>
Judge if model/data fully align with the problem (
     objective, constraints, variables, indices, and data
     consistency).
Be specific and critical.
</task>

<grammar_reference>
--- BEGIN PYOPL SYNTAX IMPLEMENTATION ---
{{GRAMMAR_IMPLEMENTATION}}
--- END PYOPL SYNTAX IMPLEMENTATION ---
</grammar_reference>

<inputs>
<problem_description>
{{PROMPT}}
</problem_description>

<model>
{{MODEL_CODE}}
</model>

<data>
{{DATA_CODE}}
</data>
</inputs>

<assessment_focus>
- Objective and constraints reflect the prompt intent.
- Decision variables have correct domains and indices.
- Data is consistent with sets/parameters used by the
     model.
- Signs, units, and indexing are correct; no missing links
     .
- Any critical omissions or extraneous constraints.
- Most impactful improvements if misaligned.
</assessment_focus>

<output_requirements>
- Return ONLY a JSON object with keys "aligned" (boolean)
     and "assessment" (string, 3--6 sentences). No extra
     keys.
- You MAY wrap the JSON in a ```json fence containing only
      the JSON.
</output_requirements>

<json_schema>
{ "type":"object", "additionalProperties": false, "
     required":["aligned","assessment"],
  "properties": { "aligned":{"type":"boolean"}, "
       assessment":{"type":"string"} } }
</json_schema>
```

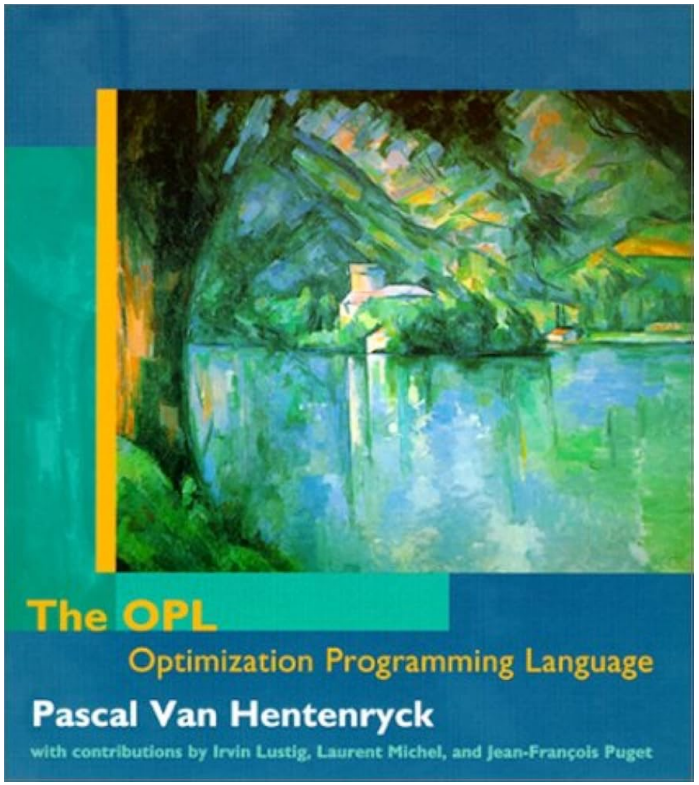
PyOPL – Python Optimisation Programming Language

knapsack.mod

```
range Items = 1..5;

float weight[Items] = [2, 3, 4, 5, 5];
float value[Items] = [2, 3, 4, 5, 5];

float C = 10;

dvar boolean x[Items];

maximize
  sum(i in Items) value[i] * x[i];
subject to {
  cap: (sum(i in Items) weight[i] * x[i]) <= C;
}
```

knapsack.dat

```
weight = [2, 3, 4, 5, 5];
value = [2, 3, 4, 5, 5];
```



ibm.com/products/ilog-cplex-optimization-studio

ILOG CPLEX Optimization Studio | Use cases | Licensing options | Resources

Mathematical programming	Constraint scheduling	OPL modeling	Development environment

Model complex optimization problems using OPL (Optimization Programming Language), a declarative language with native support for sets, arrays, tuples and both linear and non-linear constraints. It also supports scheduling-specific constructs such as intervals (activities) and Cumul functions (resources). This simplify the modelling process compared to general-purpose programming languages.

Generative AI & Optimization in 2026: Bob, CPLEX, and the Comeback of OR (Operations Research)

Alex Fleischer
Data and AI Technical Sales - Quantum Ambassador

March 18, 2026

https://www.linkedin.com/pulse/generative-ai-optimization-2026-bob-cplex-comeback-alex-fleischer-xfkge/

Mark Howells · 2nd
Joint appoint: L'boro Uni & Imperial
3h • Edited •

+ Follow ···

🚀 **What if optimisation modelling were as simple as writing the maths?** 📐

==========

Today we are introducing **MOSOX** to everyone who builds, teaches or uses optimisation models. 🎉

=== using AI industry standard, RUST ===

Whether you work in:

⚡ Energy; 🚚 Logistics; 🏭 Manufacturing; 💰 Finance; 🌾 Agriculture; 💧 Water; 🏙️ Urban planning; 📡 Telecommunications; 📈 Economics; 🔬 Research; 🎓 Teaching

==========

Many optimisation models are built with powerful frameworks such as Linopy, Pyomo and JuMP. These tools are flexible, but they also add programming layers between the equations and the solver.

🤔 We asked a simpler question:

==========

If your problem is an LP or MIP, why not keep the model as close to the mathematics as possible?

✨ MOSOX compiles readable maths-like (AMPL/GMPL MathProg) models directly into standard solver matrices.

It can work with HiGHS and other solvers.

📈 In **OSeMOSYS** benchmarks, MOSOX:

⚡ Compiled models up to 6.5× faster than GLPK's glpsol

💾 Used less peak memory on the largest case

This can mean:

⚡ Faster iteration

💰 Lower computing costs

🌍 Lower energy use for the same work

CAMBRIDGE UNIVERSITY PRESS

Engage How To Submit Browse Events Communities About

Computer Science

MOSOX: a fast model-to-matrix compiler for linear and mixed-integer optimisation

12 July 2026, Version 1

Working Paper | Climate Change And Sustainability

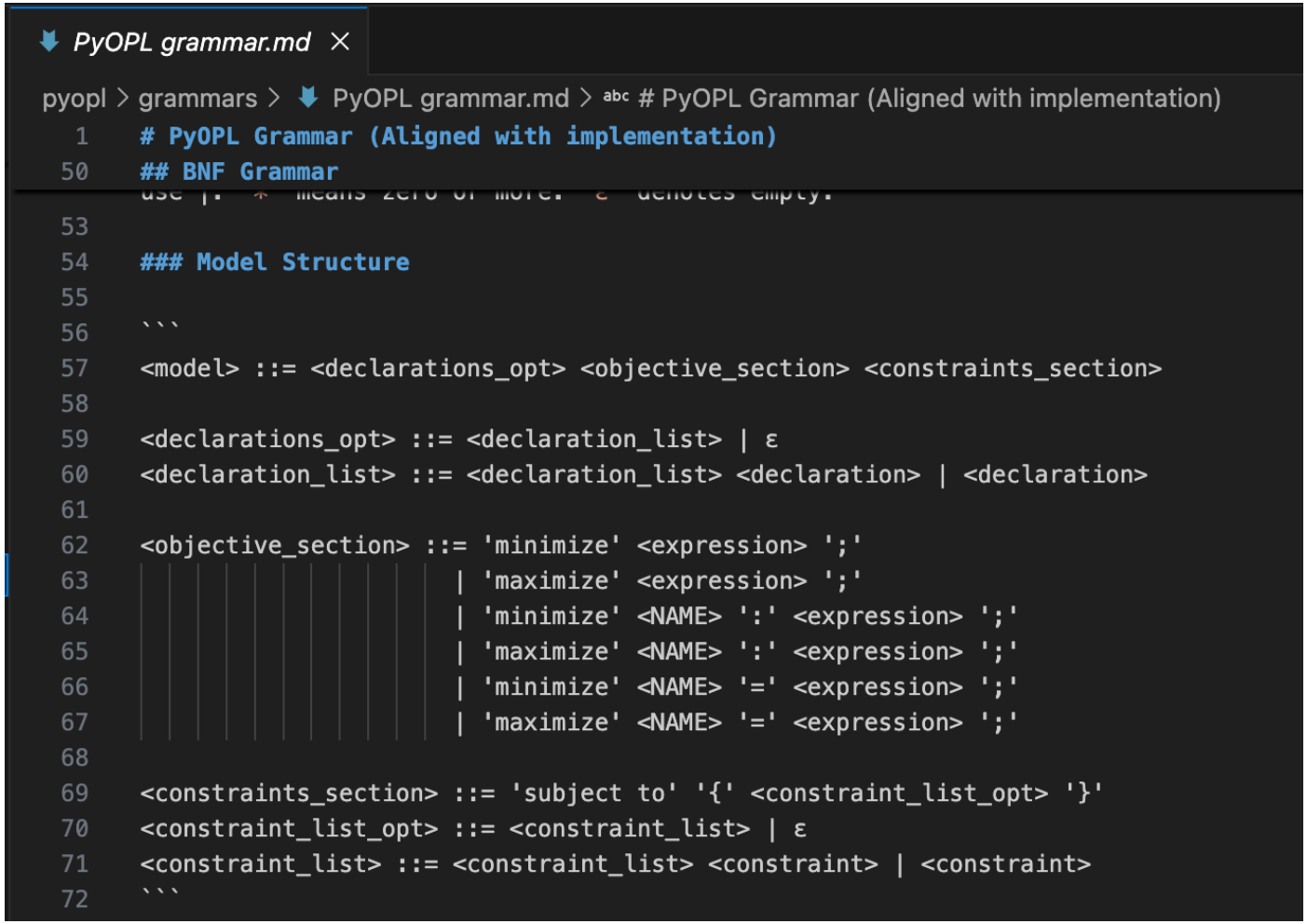
Chris Arderne, Lara Dixon, Vedran Kapor, Iain Staffell, Nathan Johnson, Mohamed Bassam Ben-Ticha, Maelle Baronnet, Claire Nicolas, Nolwazi Khumalo, Daniel Russo, Larissa Nogueira, Simon Benmarraze, Adam Hawkes, Steve Pye, Tom Alfstad, Mario Tot, Leigh Martindale, Jairo Quirós-Tortós, Leonhard Hofbauer, Pietro Lubello, Naomi Tan, Fernando Plazas-Niño, Ariane Millot, Francesco Gardumi, Mark Howells

Show author details

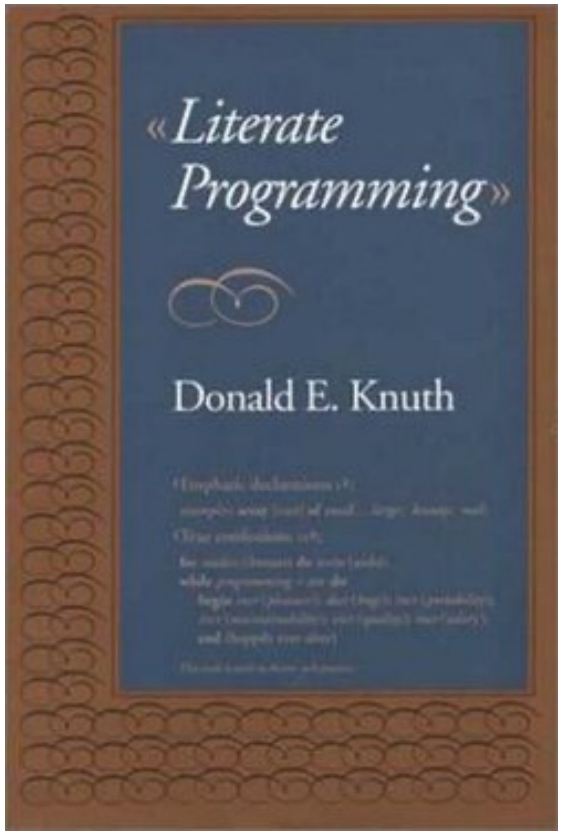
Abstract

Linear and mixed-integer optimisation models are widely used across economics and engineering to study resource allocation, infrastructure planning, and energy-system transitions. Algebraic modelling languages such as GNU MathProg (GMPL), AMPL, and GAMS let researchers write these models close to their mathematical form, keeping them transparent and reviewable even without extensive coding experience. However, as models grow in scale, translating algebraic formulations into solver-ready sparse matrices becomes a major computational bottleneck. This paper introduces MOSOX, a Rust-based command-line tool and library that compiles a targeted subset of GMPL model and data files into sparse matrices, covering the constructs required by the OSeMOSYS energy-system model family. It expands sets, parameters, variables, objectives, and constraints into matrices, exports them in standard MPS format, and can solve models directly via the HiGHS solver. On OSeMOSYS benchmarks, MOSOX compiles matrices up to 6.5 times faster than GLPK's glpsol while also reducing peak memory use on the largest tested model. By combining fast, low-memory compilation with the readability of GMPL and solver-independent output, MOSOX - developed within the Climate Compatible Growth Program - supports reproducible, auditable, and automatable optimisation workflows for large-scale energy-system modelling.

Grammar Aware



Literate...



Literate Modelling

Literate Model

```
/* Candidate hiring optimization problem
 - choose subset from F,G,H,I,J
 - minimize total salary with budget 40k and max 4 hires
 - skill and experience requirements
 - exclusivity: at most one of G and J
*/

int N = 5;                  // number of candidates
range Cand = 1..N;          // 1=F, 2=G, 3=H, 4=I, 5=J

/* Parameters (indexed by Cand) */
param int salary[Cand];      // salary[c] of candidate c
param int skill[Cand];       // skill level of candidate c
param int exp[Cand];         // project management experience of candidate c

/* Decision variables */
dvar boolean hire[Cand];      // hire[c] = 1 if candidate c is hired

/* Objective: minimize total salary of hired candidates */
minimize
  TotalSalary: sum(c in Cand) salary[c] * hire[c];

subject to {

  // Budget constraint: total salary <= 40,000
  Budget: sum(c in Cand) salary[c] * hire[c] <= 40000;

  // Maximum number of hires
  MaxHires: sum(c in Cand) hire[c] <= 4;

  // Skill requirement: total skill >= 8
  SkillSum: sum(c in Cand) skill[c] * hire[c] >= 8;

  // Experience requirement: total exp >= 8
  ExpSum: sum(c in Cand) exp[c] * hire[c] >= 8;

  // Exclusivity: at most one of G and J can be hired
  ExcludeGJ: hire[2] + hire[5] <= 1;
}
```

Non-literate Model

```
// Candidate hiring optimization

dvar boolean HireF;
dvar boolean HireG;
dvar boolean HireH;
dvar boolean HireI;
dvar boolean HireJ;

// Salary constants
int SalaryF = 12000;
int SalaryG = 15000;
int SalaryH = 18000;
int SalaryI = 5000;
int SalaryJ = 10000;

// Data-driven budgets
int Budget;
int MaxHires;

minimize TotalCost: SalaryF*HireF + SalaryG*HireG +
       SalaryH*HireH + SalaryI*HireI + SalaryJ*HireJ;

subject to {
  BudgetCon: SalaryF*HireF + SalaryG*HireG +
             SalaryH*HireH + SalaryI*HireI + SalaryJ*HireJ <= Budget;
  SkillCon: 2*HireF + 3*HireG + 4*HireH + 1*HireI + 2*HireJ >= 8;
  PMCon: 1*HireF + 2*HireG + 2*HireH + 5*HireI + 4*HireJ >= 8;
  GJCon: HireG + HireJ <= 1;
  HireCap: HireF + HireG + HireH + HireI + HireJ <= MaxHires;
}
```

Figure 9: A comparison illustrating the impact of literate-style comments on model readability.

Retrieval Augmented Generation (Few-shot prompting)

	assignment	crew_pairing	crew_scheduling	graph_coloring	inventory_routing	jobshop	knapsack	knapsackp	lot_sizing	plant_location	production	set_covering	set_partitioning	transportation	tsp	vehicle_routing	warehouse_location	workforce_planning	p_dispersion	on_off_outsourcing	stochastic_production	stochastic_scheduling
Logical NOT ($1-y$)				✓		✓										✓			✓	✓		
Logical AND ($z=x\ y$)																			✓			
Logical OR																				✓		
Bipartite matching	✓																					
At-least-one coverage constraints		✓										✓										
Pattern/column selection variables		✓										✓	✓									
Exactly-one coverage constraints	✓		✓										✓		✓	✓						
Resource capacity constraints			✓			✓	✓	✓		✓	✓					✓	✓	✓		✓		✓
Demand satisfaction equalities										✓				✓			✓					
Demand coverage inequalities		✓									✓	✓						✓		✓	✓	✓
Supply satisfaction equalities														✓								
Subtour elimination (SE)															✓							
Capacity-based SE																✓						
Setup Costs/Batch Sizes									✓	✓							✓			✓		
Conditional Expression (big-M)				✓		✓			✓	✓					✓	✓	✓		✓	✓		
Disjunctive Rules				✓		✓																
Separation via disjunctive big-M																						
Precedence Constraint						✓																
Min-Max Objective				✓		✓																
Max-Min Objective																			✓			
Inventory Balance					✓				✓									✓				✓
Initial state constraints					✓				✓									✓				✓
Inventory with Backlogs																						✓
Stock capacity limits					✓																	
Activity Start/End																				✓		

Why use SyntAGM?

Method	**ReSocratic**					
	Accur. (95% CI)	CE %	RE %	WA %	Avg. P/C Tkns	Avg. L (s)
Standard	22.7 (18.6–27.4)	60.9	12.2	3.99	7.21k/2.78k	82.7
Chain-of-Thought	45.5 (40.4–50.7)	0.85	47.2	6.27	6.39k/2.39k	129
Tree-of-Thoughts	33.0 (28.3–38.1)	13.9	47.8	5.12	10.3k/5.13k	234
Reflexion	61.5 (56.3–66.4)	3.99	28.4	5.98	12.8k/5.47k	203
Chain-of-Experts	66.1 (61.0–70.9)	4.84	21.9	7.12	65.6k/16.4k	651
SyntAGM	74.9 (70.1–79.2)	5.41	11.1	8.55	12.0k/5.53k	159

Table 4: ReSocratic (gpt-oss-20b)

Why use SyntAGM?

Method	**ChallengeOR**					
	Accuracy	CE	RE	WA	Avg. Cost	Avg. L (s)
Standard	52.5%	35.0%	5.00%	7.5%	$0.026181	13.0
Reflexion	77.5%	2.5%	1.25%	18.7%	$0.091497	51.5
Chain-of-Experts	78.7%	5.00%	3.75%	12.5%	$0.367764	144
SyntAGM	78.7%	1.25%	6.25%	13.7%	$0.069069	35.5

Table 6: ChallengeOR (GPT-5.3-codex)

Why use SyntAGM?

Method	**ChallengeOR**					
	Accuracy	CE	RE	WA	Avg. Cost	Avg. L (s)
Standard	31.2%	47.5%	3.75%	17.5%	$0.014719	6.90
Reflexion	38.7%	35.0%	17.5%	8.75%	$0.082784	42.2
Chain-of-Experts	51.2%	27.5%	11.2%	10.0%	$0.344987	141
SyntAGM	60.0%	18.7%	3.75%	17.5%	$0.077193	32.2

Table 8: ChallengeOR (GPT-5.4-mini)

Ablation study

Removed	ChallengeOR						
	Accuracy	CE rate	RE rate	WA	Avg. Cost	Avg. L (s)	Avg. I
BNF	60.0%	12.5%	6.25%	21.2%	$0.039884	36.4	3.36
RAG	40.0%	30.0%	20.0%	8.00%	$0.055800	30.1	3.90
Alignment	58.7%	10.0%	10.0%	21.25%	$0.031938	14.3	1.75
Diag. (Partial)	53.7%	32.5%	1.25%	12.5%	$0.074340	37.7	4.15
Diag. (none)	46.2%	37.5%	5.00%	11.2%	$0.077084	37.7	4.11
Baseline	60.0%	18.7%	3.75%	17.5%	$0.077193	32.2	3.95

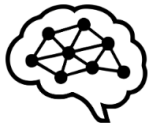
Table 15: Leave-one-out ablation study on SyntAGM components (GPT-5.4-mini)

Ablation study

Retained	ChallengeOR						
	Accuracy	CE	RE	WA	Avg. Cost	Avg. L (s)	Avg. I
None	10.0%	88.7%	0.00%	1.25%	$0.023154	23.5	4.62
BNF	37.5%	27.5%	16.2%	18.7%	$0.033723	17.3	2.49
RAG	42.5%	46.2%	2.50%	8.75%	$0.029195	23.4	2.96
Alignment	13.7%	83.7%	0.00%	2.50%	$0.021952	24.0	4.64
Diag. (Full)	31.2%	50.0%	7.50%	11.2%	$0.019838	24.0	3.81
Diag. (Partial)	11.2%	87.5%	0.00%	1.25%	$0.022825	24.0	4.58
Baseline	60.0%	18.7%	3.75%	17.5%	$0.077193	32.2	3.95

Table 9: Hard-ablation study on SyntAGM components (GPT-5.4-mini); Avg. I: average iterations

Rhetor



Rhetor is an Integrated Modelling Environment for modelling and solving mixed-integer linear programming problems. Rhetor supports a subset of the Optimisation Programming Language (OPL) syntax and it is built on `pyopl`, a Python library for parsing and solving OPL-like mathematical programming models using Gurobi or Scipy (Highs). Rhetor integrates GenAI features to support and automate the modelling process.

Installation & start

Install:

```
pip install rhetor
```

Start Rhetor:

```
pyopl
```

To enable GenAI features, set at least one of the following environment variables before starting Rhetor:

- `OPENAI_API_KEY` — for OpenAI models
- `GEMINI_API_KEY` — for Google Gemini

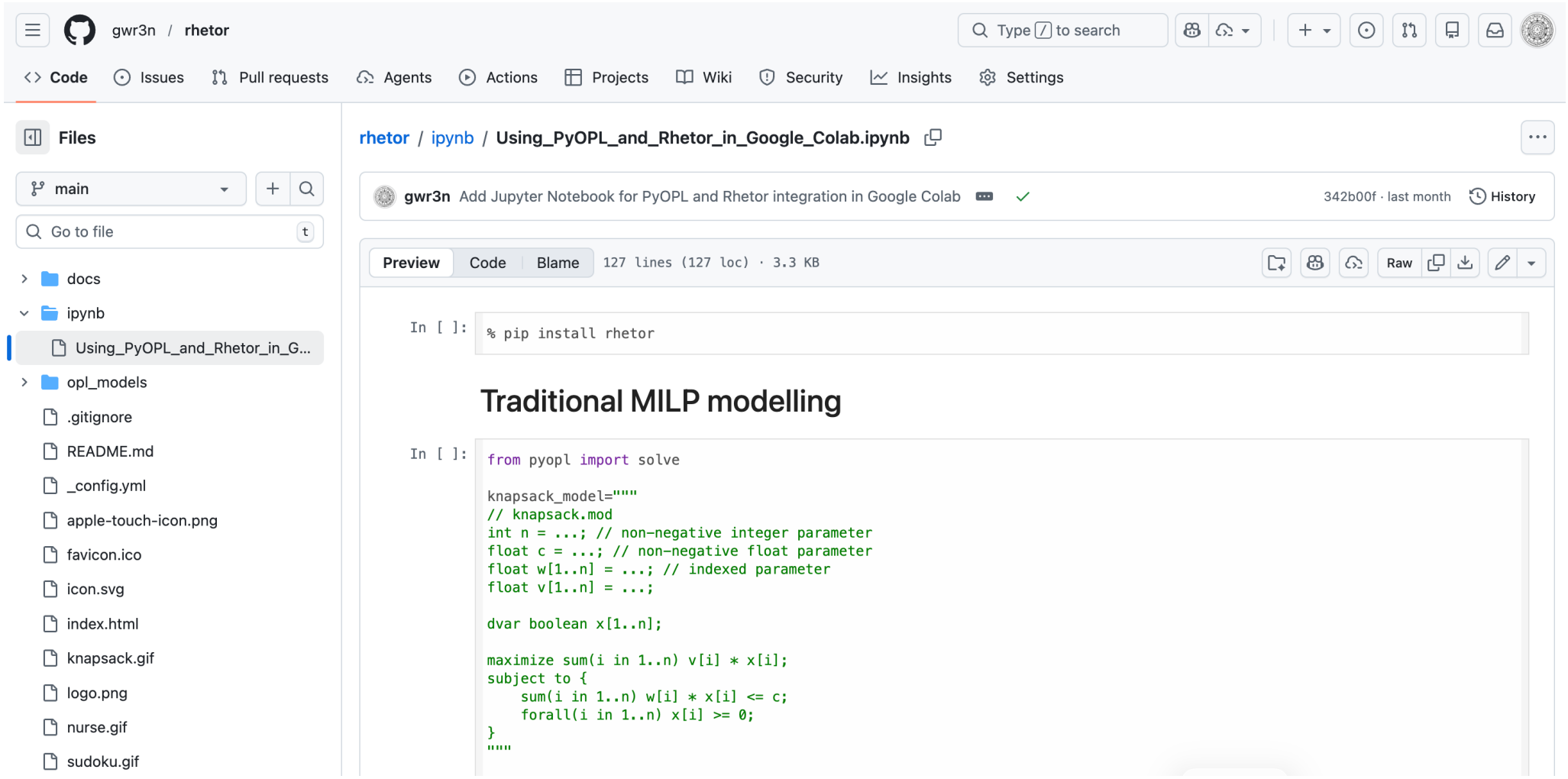
or install and run `ollama` locally.

Example (macOS / bash):

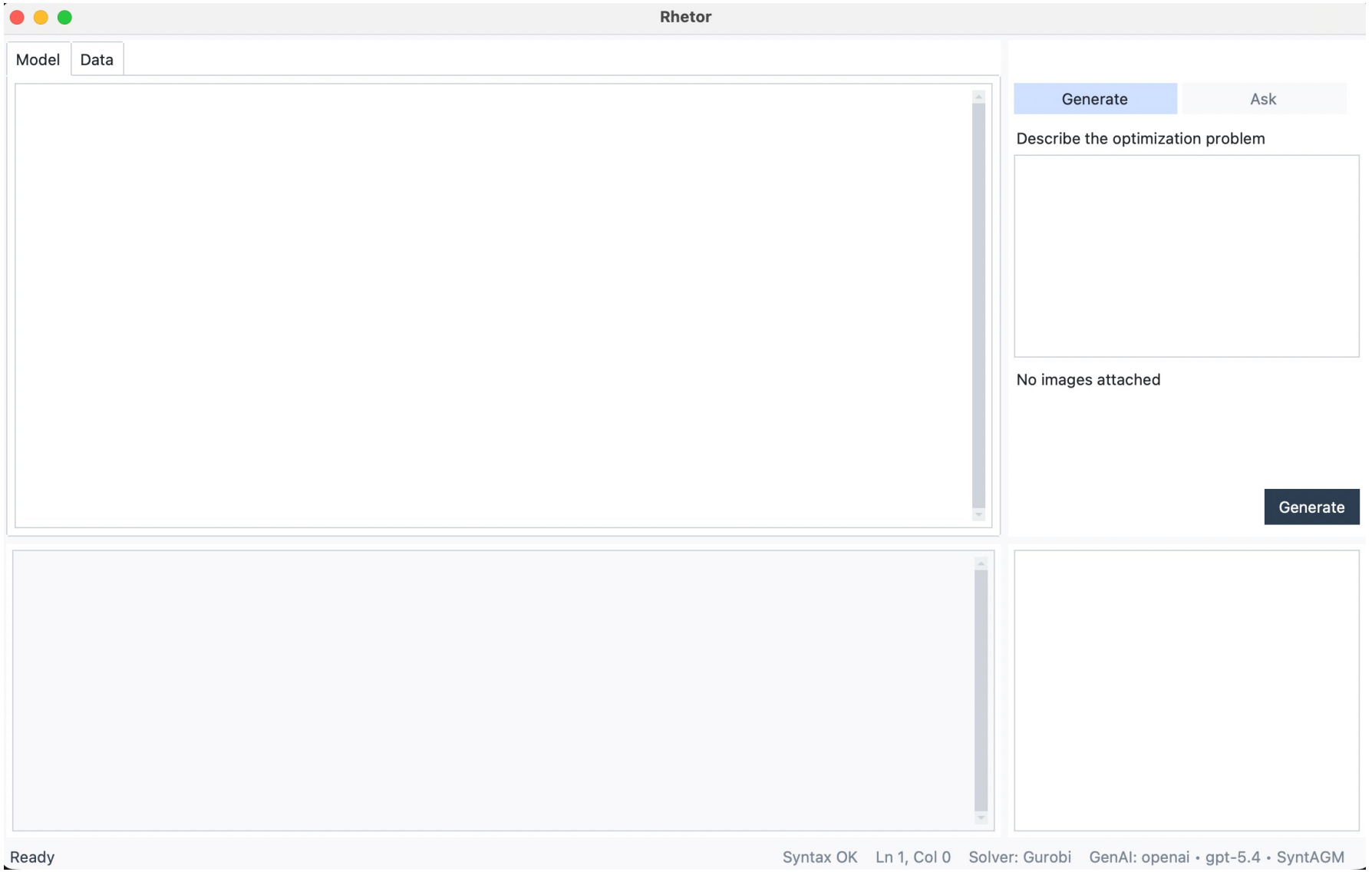
```
export OPENAI_API_KEY="sk-..."
```

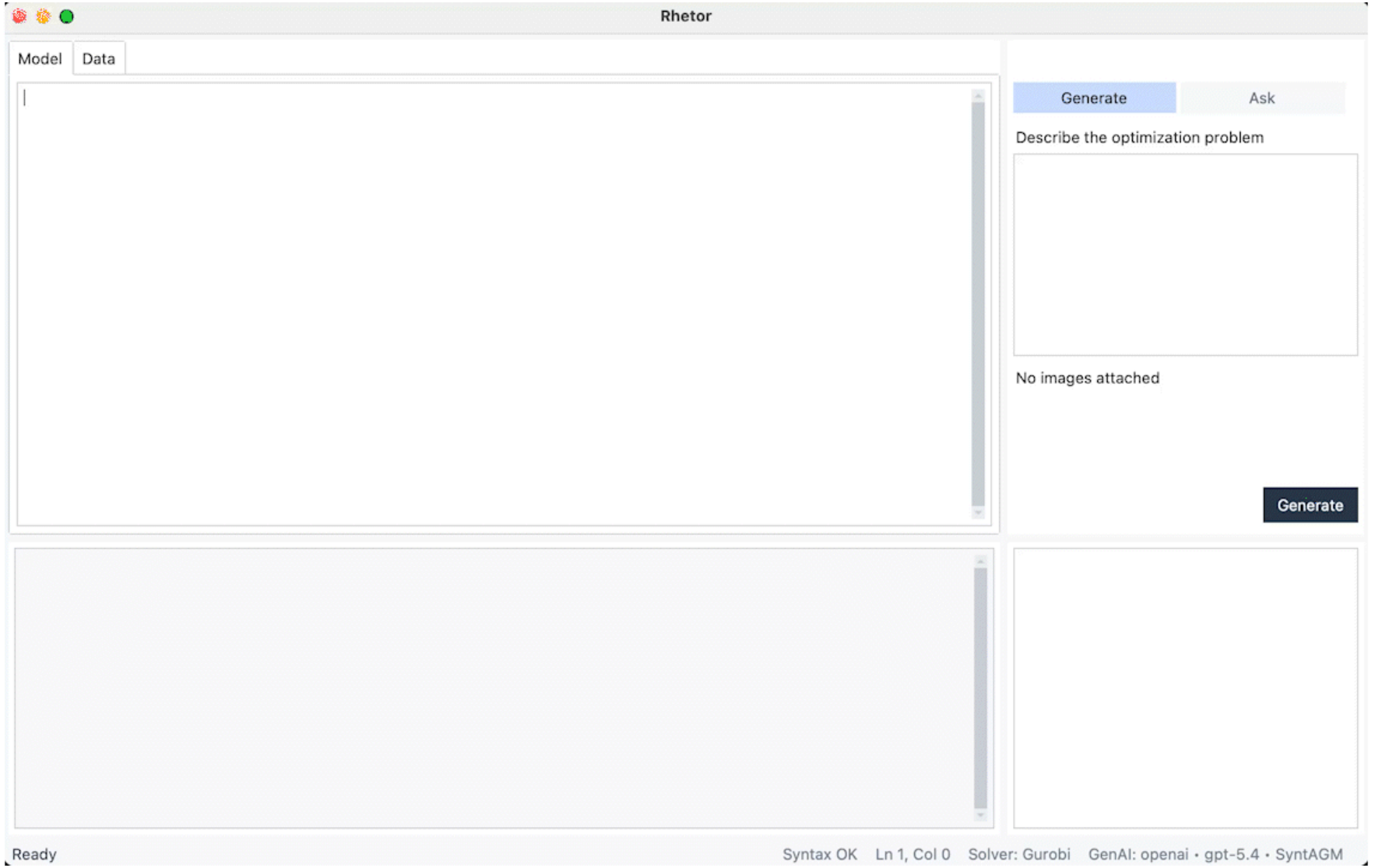
Alternatively, you can use `rhetor` as illustrated in the following Jupyter notebook.

https://gwr3n.github.io/rhetor/

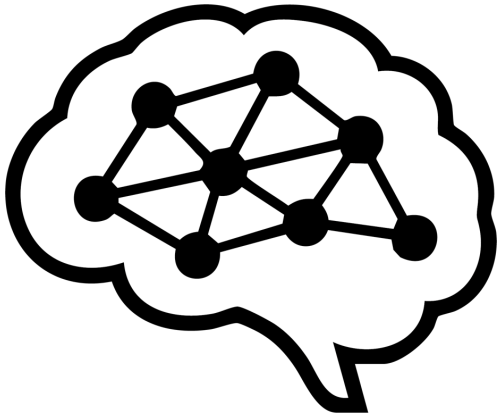


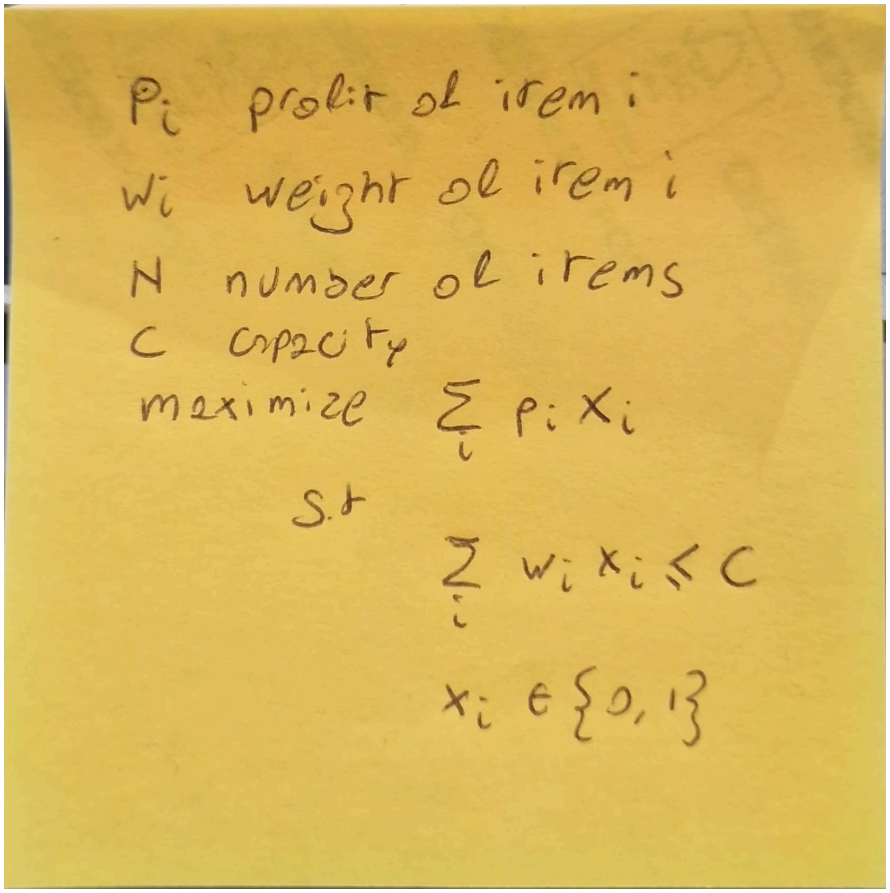
https://github.com/gwr3n/rhetor/blob/main/ipynb/Using_PyOPL_and_Rhetor_in_Google_Colab.ipynb


Rhetor
Model
Data
Generate
Ask
Describe the optimization problem
No images attached
Generate
Ready
Syntax OK
Ln 1, Col 0
Solver: Gurobi
GenAI: openai • gpt-5.4 • SyntAGM

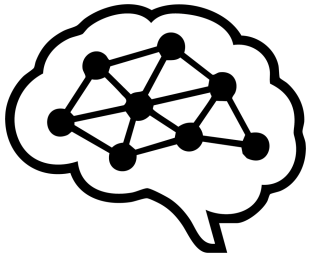

Rhetor
Model
Data
Generate
Ask
Describe the optimization problem
No images attached
Generate
Ready
Syntax OK
Ln 1, Col 0
Solver: Gurobi
GenAI: openai • gpt-5.4 • SyntAGM

Conversational Modelling





“Back of the envelope modelling”



c_{ij} distance between i and j

$$\text{minimize} \quad \sum_i \sum_j x_{ij} c_{ij}$$

s.t.

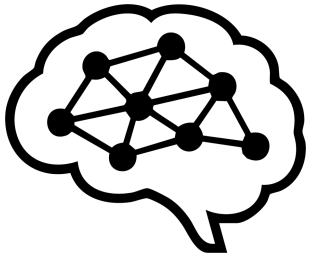
$$\sum_i x_{ij} = 1 \quad \forall j$$

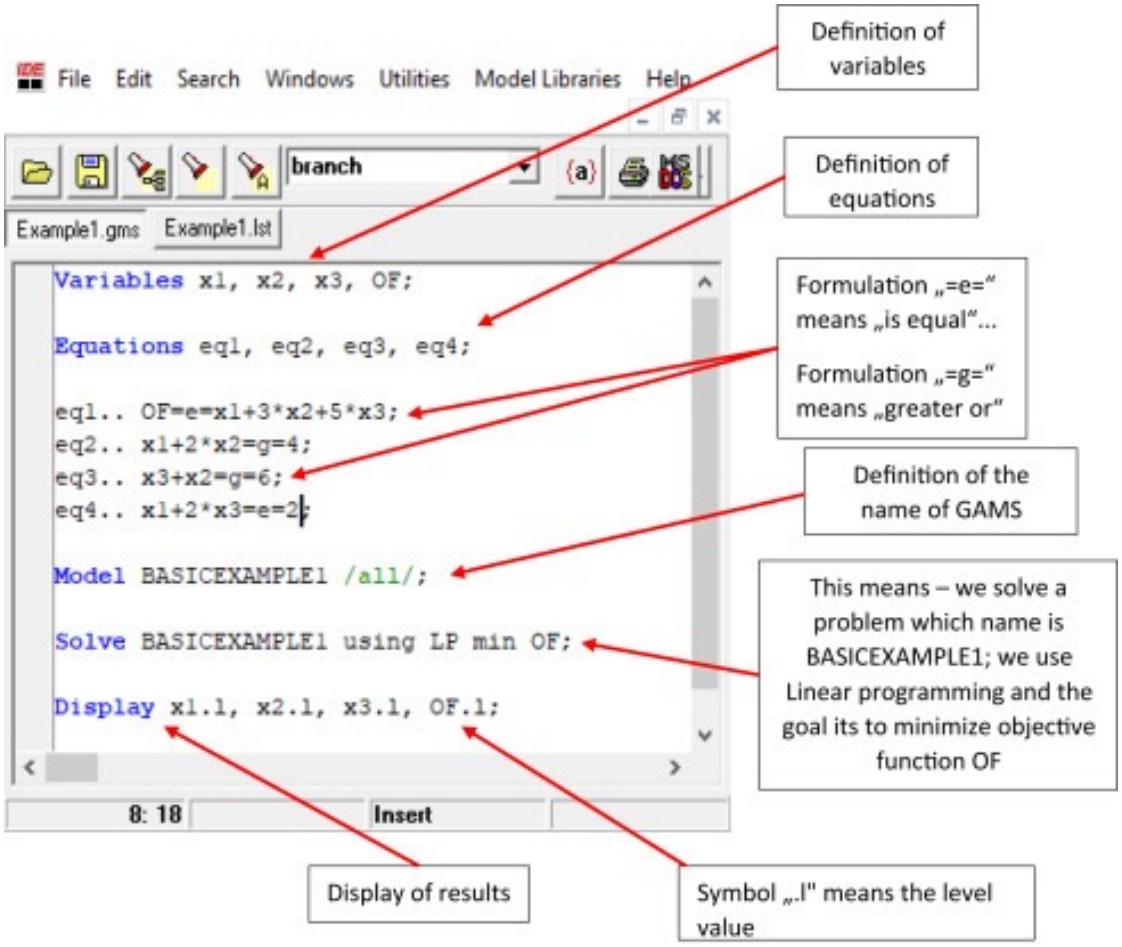
$$\sum_j x_{ij} = 1 \quad \forall i$$

subtour elimination constraints

$$x_{ij} \in \{0, 1\}$$

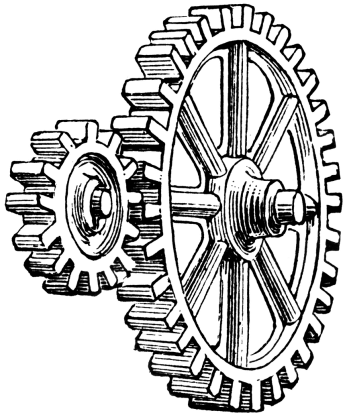
“Hybrid/conceptual modelling”





Example of GAMS code

"I was the future once..."



Ćalasan, M., Kecojević, K., Lukačević, O., & Ali, Z. M. (2021). Testing of influence of SVC and energy storage device's location on power system using GAMS. In Uncertainties in Modern Power Systems (pp. 297–342). Elsevier. https://doi.org/10.1016/b978-0-12-820491-7.00010-4

Ask your burning questions!

Did your apoxection?

Ask me anything (except for the meaning of life)

Is there a snack break soon?

Did you hallucinate that last slide?

Write a haiku about this entire presentation.

Can you explain this again, but for for a 5-year-old?

SUBMIT