Gala: Global LLM Agents for Text-to-Model Translation

Amazon Quantum Solutions Lab, AWS Professional Services, 2025

Serdar Kadıoğlu
[1] Dept. of Computer Science, Brown University
[2] AI Center of Excellence, Fidelity Investments
skadio.github.io

Learning & Reasoning

Data Science: ML/DL/NLP/LLMs/etc.

Focuses on **machine learning using historical data** to identify patterns and make predictions. Excels at pattern recognition, classification, and forecasting.

System 1 - Predictive Models

- Learning from historical data patterns
- Probabilistic predictions and insights
- Ideal for unstructured problems
- Applications include recommendation systems, image recognition, and natural language processing

Decision Science: OR/MIP/CP/SAT/LS/etc.

Focuses on **combinatorial satisfaction and optimization** using logical and mathematical models. Provides provable optimality and explicit reasoning.

System 2 - Prescriptive Models

- Mathematical and logical formulations
- Provably optimal for deterministic environments
- Perfect for structured problems
- Applications include verification, planning, scheduling, routing, and resource allocation

The Evolution of AI Paradigms: From Classical AI to Modern and Generative AI [AAAI YouTube]

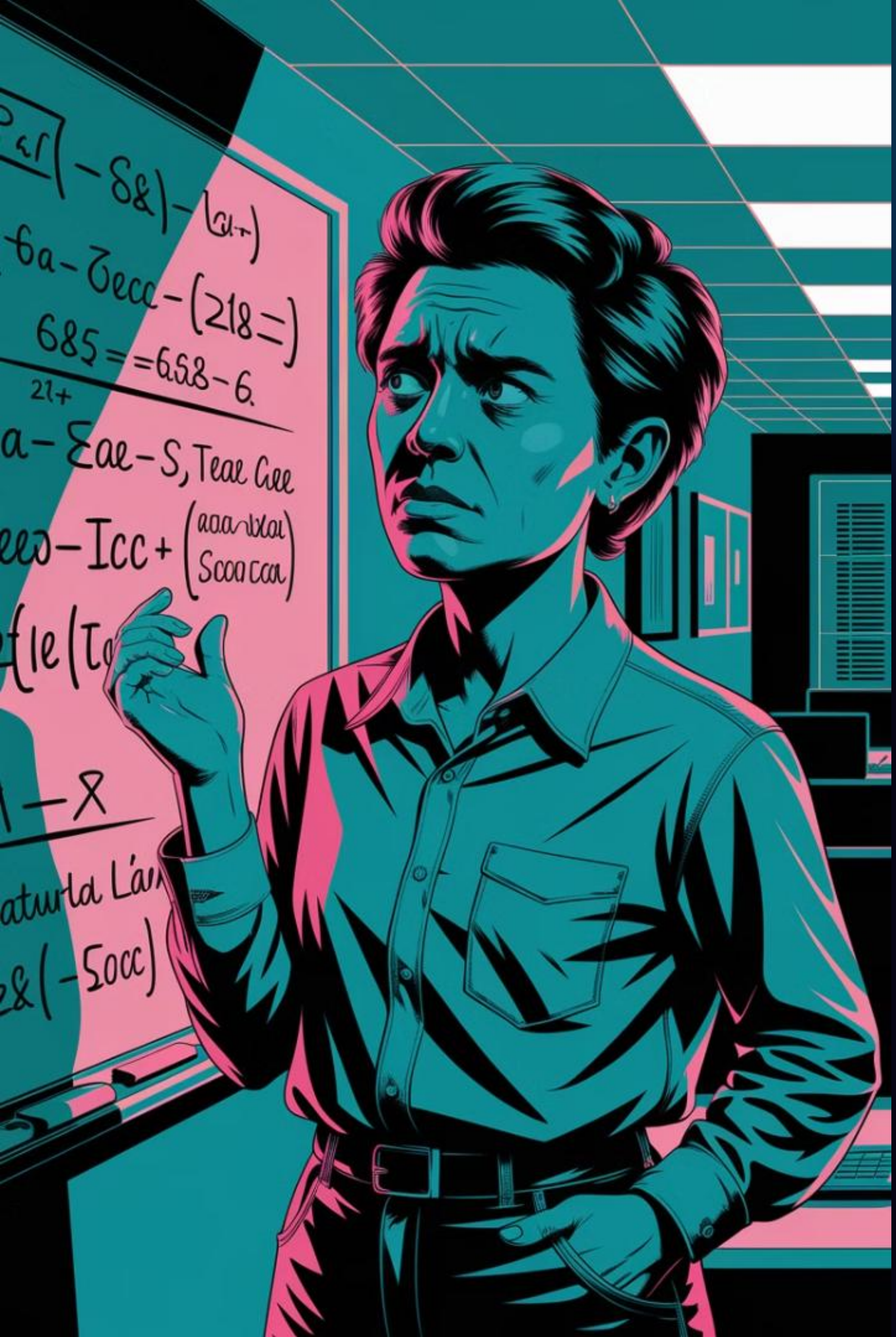
Integration with Optimization Technology

Existing ML-OR Integration

- Algorithm configuration procedures
- Variable and constraint selection
- Branching strategies
- Cut selection
- Node selection
- Tree-search configuration

Emerging NLP-OR Integration

- Named entity recognition for optimization
- Natural language interfaces for solvers
- Automated model formulation
- Explanation generation
- Interactive modeling assistants
- Domain-specific optimization co-pilots



The De-Facto Model-and-Run Strategy

1

Problem Description

Users **describe optimization problems** in natural language, which contains ambiguous references to variables, constraints, and objectives that must be precisely identified.

2

Model Formulation

Experts must **manually transform problem descriptions** into formal mathematical models, a process that requires specialized knowledge and is prone to errors.

3

Solution Finding

Once properly modeled, optimization solvers can find optimal solutions, but the **modeling barrier** remains a significant obstacle to wider adoption of optimization technology.

Decision Making in the Era of Large-Language Models

1. **Reasoning: Optimization**

- Optimization technology and constraint solving techniques are powerful and have many applications.
- The cognitive barrier of translating problem descriptions into formal constraint models persists.

2. **Learning: Large-Language Models**

- LLMs have found success in many fields recently.
- However, they still face challenges in generating constraint models from free-form natural language text.

Our Vision: Modeling Co-Pilots

A paradigm shift integrating **automated modeling assistants** capable of translating natural language into formal optimization formulations.

Natural Language

Problem descriptions in free-form text

LLM Co-Pilot

Automated translation and formulation

Formal Models

Executable constraint model code

Solution | Interactivity | Feedback Loop

Verified results

Our Contributions

Holy Grail 2.0

Blueprint for **optimization modelling co-pilots** with feedback loop and user interactivity.
Tsouros et. al., 2023

Ner4Opt

A principled approach to **extracting components of optimization models** such as the objective, variables, and constraints from free-form natural language text.
Kadioglu et. al, Constraints'24

Text2Zinc

A **unified cross-domain dataset** curated to work with **LLM co-pilots** and **leaderboard** to evaluate strategies to generate **MiniZinc models** from free-from natural language text. *Singirikonda et. al., AAAI'25*

Gala: Global LLM Agents

A **global agentic approach** with multiple specialized LLM agents decompose the modeling task by global constraint type.
Cai et. al. NeurIPS'25

Definition: Combinatorial Problems

CSPs: Constraint Satisfaction Problem

A Constraint Satisfaction Problem is defined as a triple:

$$CSP = (X, D, C)$$

- **X**: Finite set of decision variables
- **D**: Domains assigning each variable admissible values
- **C**: Constraints mapping assignments to truth values

A **feasible solution** assigns all variables such that all constraints evaluate to *true*.

COPs: Constrained Optimization Problems

Optimization augment CSPs with an objective function:

$$COP = (X, D, C, O)$$

- **O:** Objective assigns cost/value to assignments.

An **optimal solution** minimizes or maximizes O while respecting all constraints.

Definition: Modeling Co-Pilots

Let **I = (T, P, O, D)** represent the input specification where:

- **T:** Natural language problem description
- **P:** Set of input parameters with definitions, symbols, and shapes
- **O:** Set of output variables with specifications
- **D:** Metadata containing problem properties

Given **input I** and data **instance d**, learn function:

$$f : (I, d) \rightarrow M$$

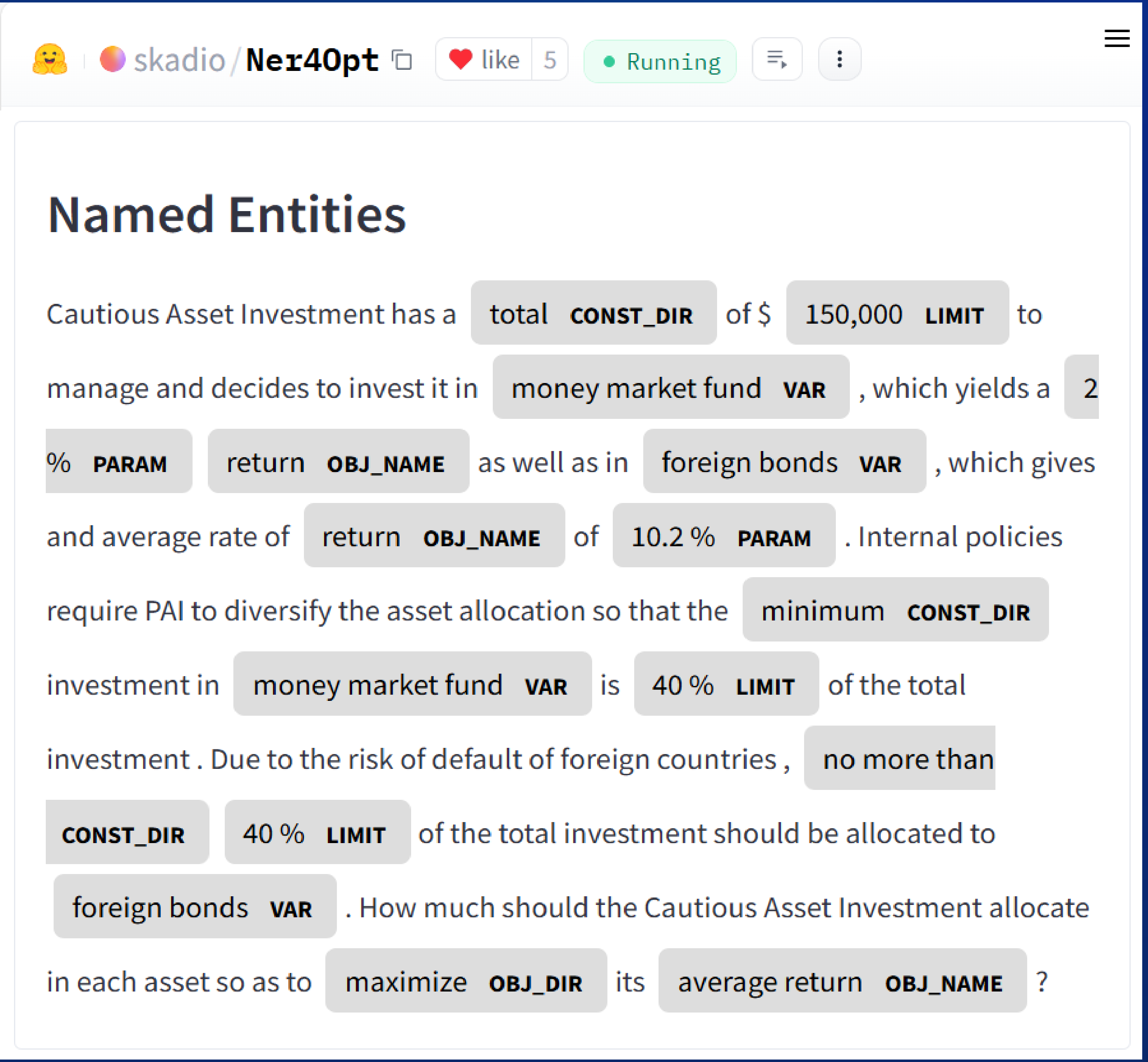
where **M** represents the space of valid constraint models that correctly implement the specifications.

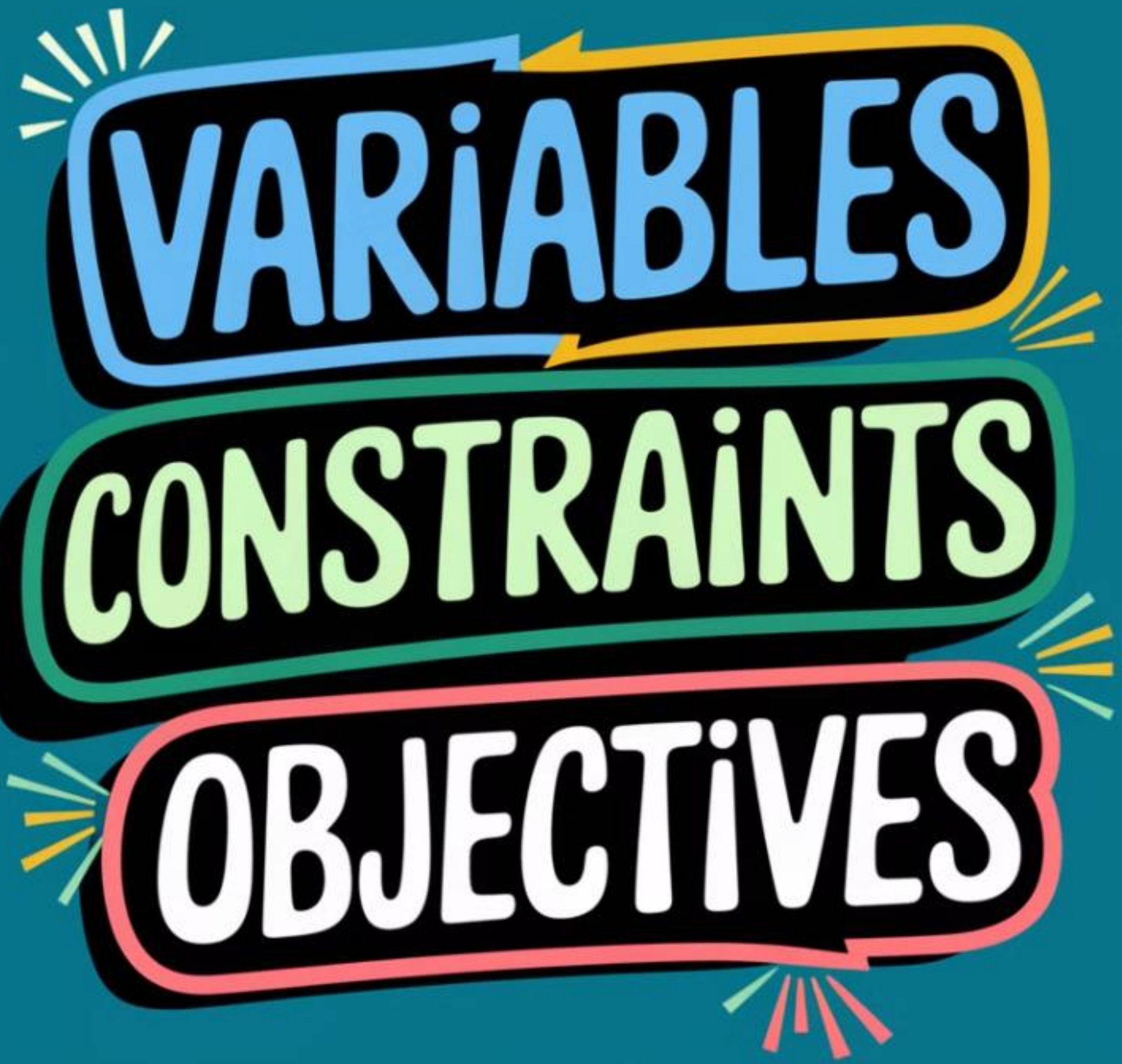
Our Contributions

Ner4Opt

A principled approach to **extracting components of optimization models** such as the objective, variables, and constraints from free-form natural language text.

Kadıoğlu et. al. Ner4Opt [Constraints'24] ACP YouTube





Introducing Ner4Opt

Named Entity Recognition

Ner4Opt extends traditional named entity recognition to identify optimization-specific components like **variables, parameters, constraints, limits, and objectives** from natural language text.

Optimization Context

Unlike standard NER which focuses on people, places, and organizations, **Ner4Opt** targets elements needed for mathematical optimization models across **diverse application domains**.

Modeling Assistance

By automatically extracting these entities, Ner4Opt helps bridge the gap between problem descriptions and formal optimization models, making **optimization technology more accessible**.

Unique Challenges of LLM Co-Pilots

1 Domain-Agnostic Generalization

Optimization technology applies to diverse domains, requiring Ner4Opt solutions to generalize across applications rather than being domain-specific.

2 Low Data Regime

The specialized nature of optimization makes it difficult and expensive to obtain large annotated datasets, necessitating to perform well with limited training data.

3 Multi-Sentence Dependency

Optimization problems typically span multiple sentences with high levels of ambiguity, requiring models to capture relationships across longer text spans.

4 Counter-intuitive Linguistics

Unlike traditional NER where entities share grammatical properties, optimization entities vary widely in linguistic characteristics while belonging to the same class.

5 Aleatoric Uncertainty

Inherent ambiguity in entity boundaries and classifications creates challenges even for human annotators, placing an upper bound on achievable performance.

6 Linguistic Variability

Optimization problems exhibit significant variability in linguistic patterns, problem structures, and application domains, making entity recognition more challenging.

Unique Challenges of LLM Co-Pilots

A doctor can prescribe two types of medication for high glucose levels , a [diabetic pill **VAR**] and a [diabetic shot **VAR**] . Per dose , [diabetic pill **VAR**] delivers [1 **PARAM**] unit of glucose reducing medicine and [2 **PARAM**] units of [blood pressure reducing medicine **OBJ_NAME**] . Per dose , a [diabetic shot **VAR**] delivers [2 **PARAM**] units of glucose reducing medicine and [3 **PARAM**] units of [blood pressure reducing medicine **OBJ_NAME**] . In addition , [diabetic pills **VAR**] provide [0.4 **PARAM**] units of stress and the [diabetic shot **VAR**] provides [0.9 **PARAM**] units of stress . [At most **CONST_DIR**] [20 **LIMIT**] units of stress can be applied over a week and the doctor must deliver [at least **CONST_DIR**] [30 **LIMIT**] units of glucose reducing medicine . How many doses of each should be delivered to [maximize **OBJ_DIR**] the [amount of blood pressure reducing medicine **OBJ_NAME**] delivaered to the patient ?

1 Domain-Agnostic Generalization

Inherent ambiguity in entity boundaries and classifications creates challenges even for human annotators, placing an upper bound on achievable performance.

2 Low Data Regime

Optimization problems exhibit significant variability in linguistic patterns, problem structures, and application domains, making entity recognition more challenging.

Technical Approaches to Ner4Opt

Classical NLP

Feature engineering with Conditional Random Fields (CRF) leverages grammatical, morphological, and syntactic info.

Custom features like gazetteers and automata capture **optimization specific patterns**.

Modern Language Models

Transformer-based approaches like RoBERTa and XLM-RB generate contextual embeddings that capture semantic relationships.

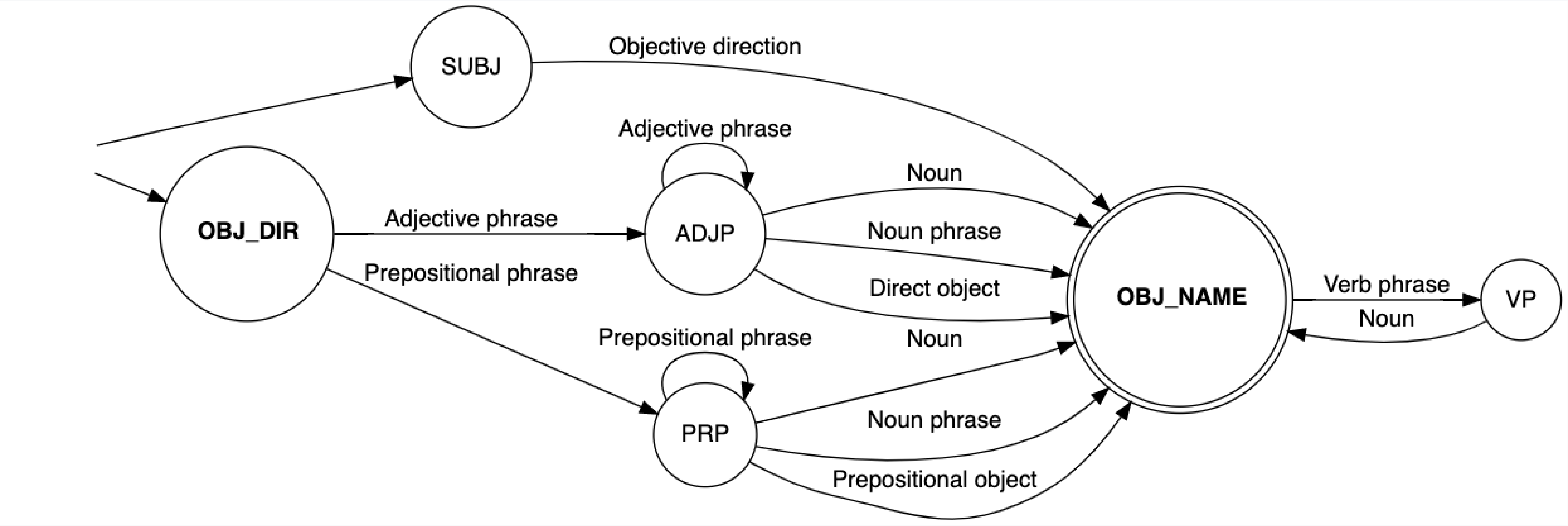
These models are **fine-tuned on optimization corpora** to improve domain-specific understanding.

Hybrid Solutions

Combination of classical feature engineering with modern language models yields the best performance.

Data augmentation techniques address challenges like long-range dependencies and disambiguation between variables and objectives.

Classical+: Feature Engineering for Optimization



profit SUBJ to be maximized OBJ_DIR

maximize OBJ_DIR the total monthly ADJP profit NOUN

Modern+: Training on Optimization Corpora

Text Extraction

Extracting textual data from PDF versions of **optimization textbooks** to create a domain-specific corpus.

Masked Language Modeling

Continued pre-training via masked language modeling by randomly masking 15% of words and training the model to predict them.

Token Replacement Strategy

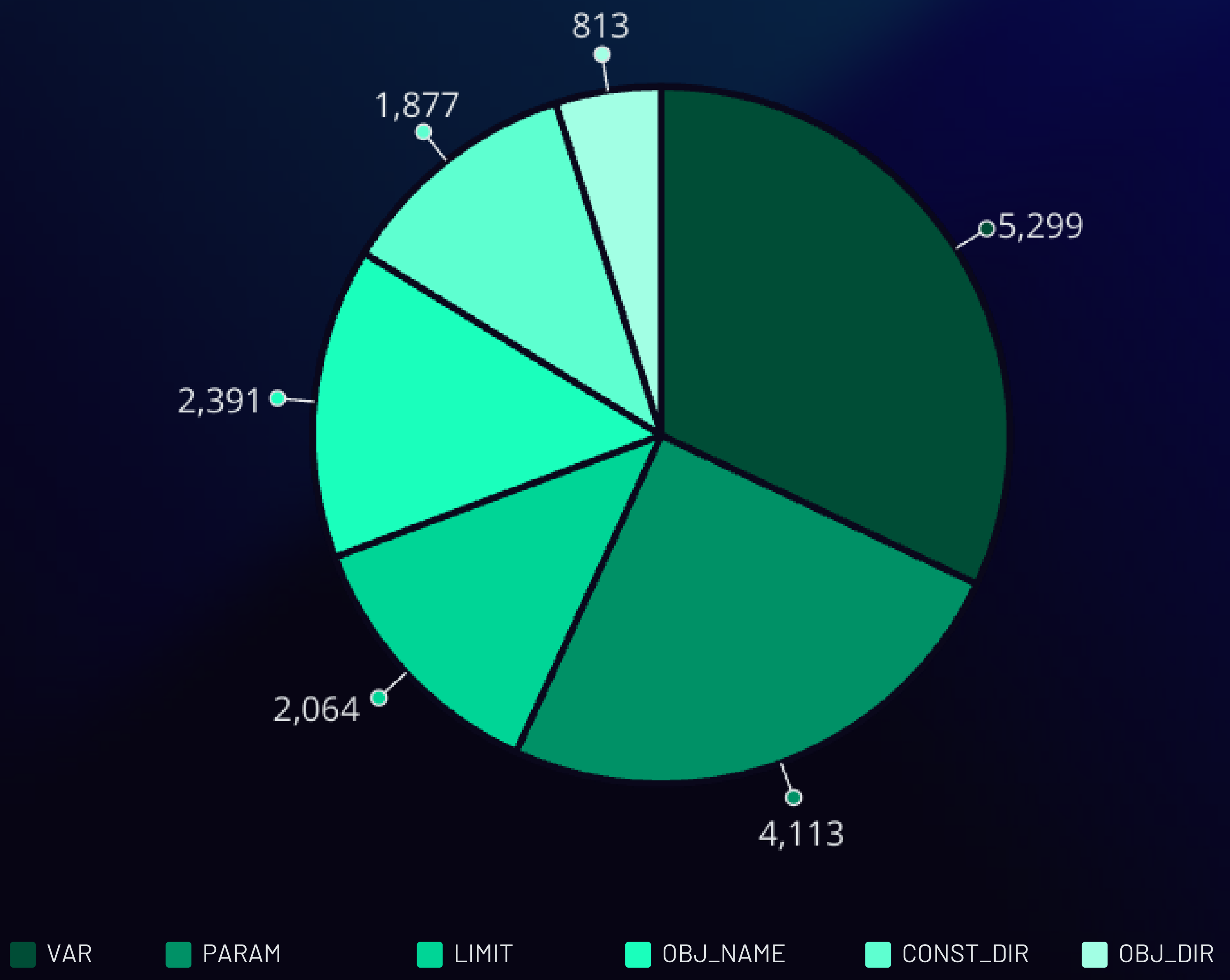
Replacing 80% of masked words with the **MASK token**, 10% with random words, and 10% with the original word to create robust training examples.

Self-Supervised Training

Training the model in a **self-supervised fashion** to predict the masked words, helping it learn **optimization-specific vocabulary** and patterns.

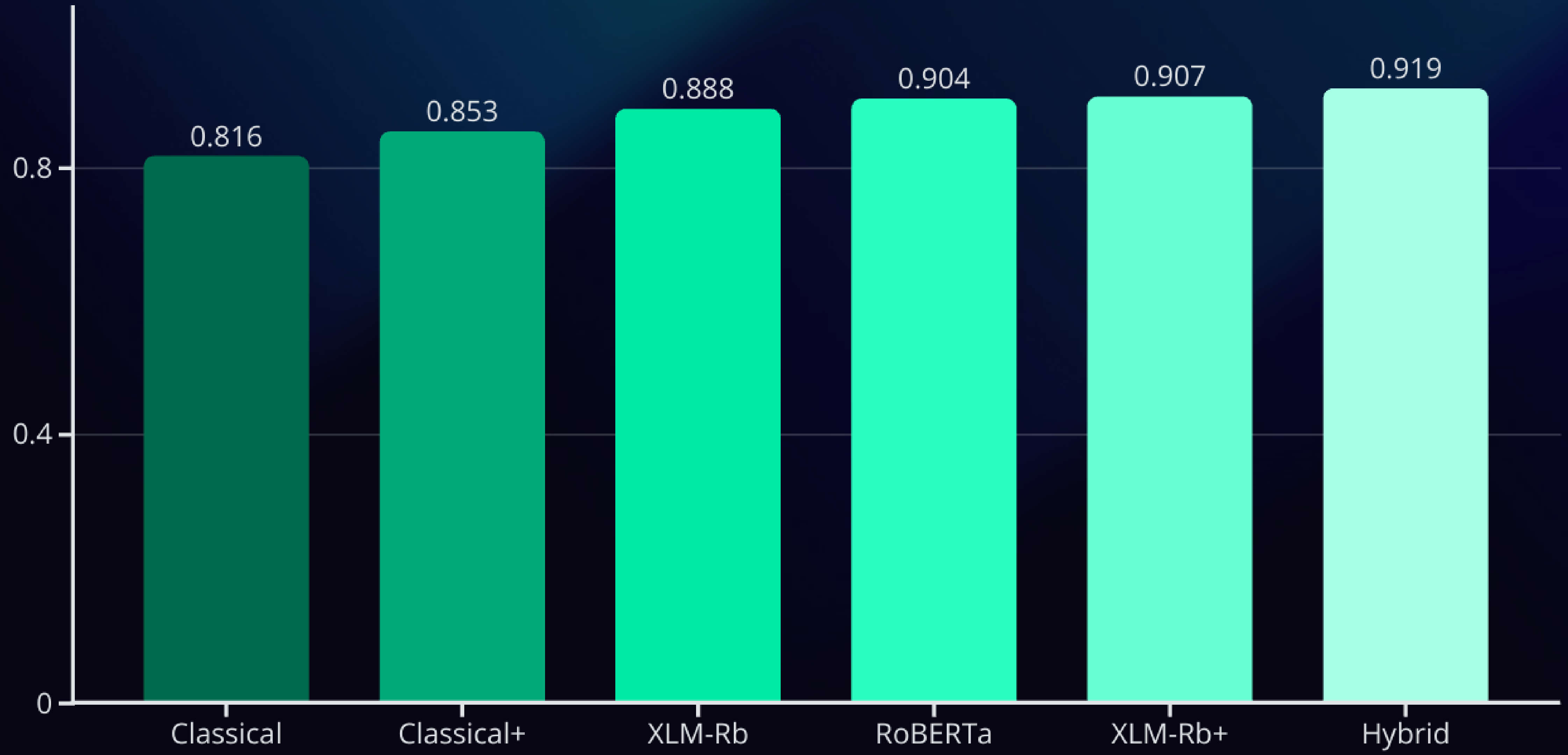
Experimental Setup

- Experiments on a benchmark dataset of **linear programming word problems**.
- This dataset contains **1,101 samples annotated** with six entity types: variable (VAR), parameter (PARAM), limit (LIMIT), constraint direction (CONST_DIR), objective direction (OBJ_DIR), and objective name (OBJ_NAME).
- The problems in the dataset span **six domains** grouped into **source domains**: advertising, investment, sales **target domains**: production, science, transportation.
- Training set consists samples **only from source domains**, while development and test sets include samples from both source and target domains in a 1:3 ratio.
- Variables (VAR) are the most common entity type, followed by parameters (PARAM) and objective names (OBJ_NAME). Objective direction (OBJ_DIR) is the least frequent entity type.

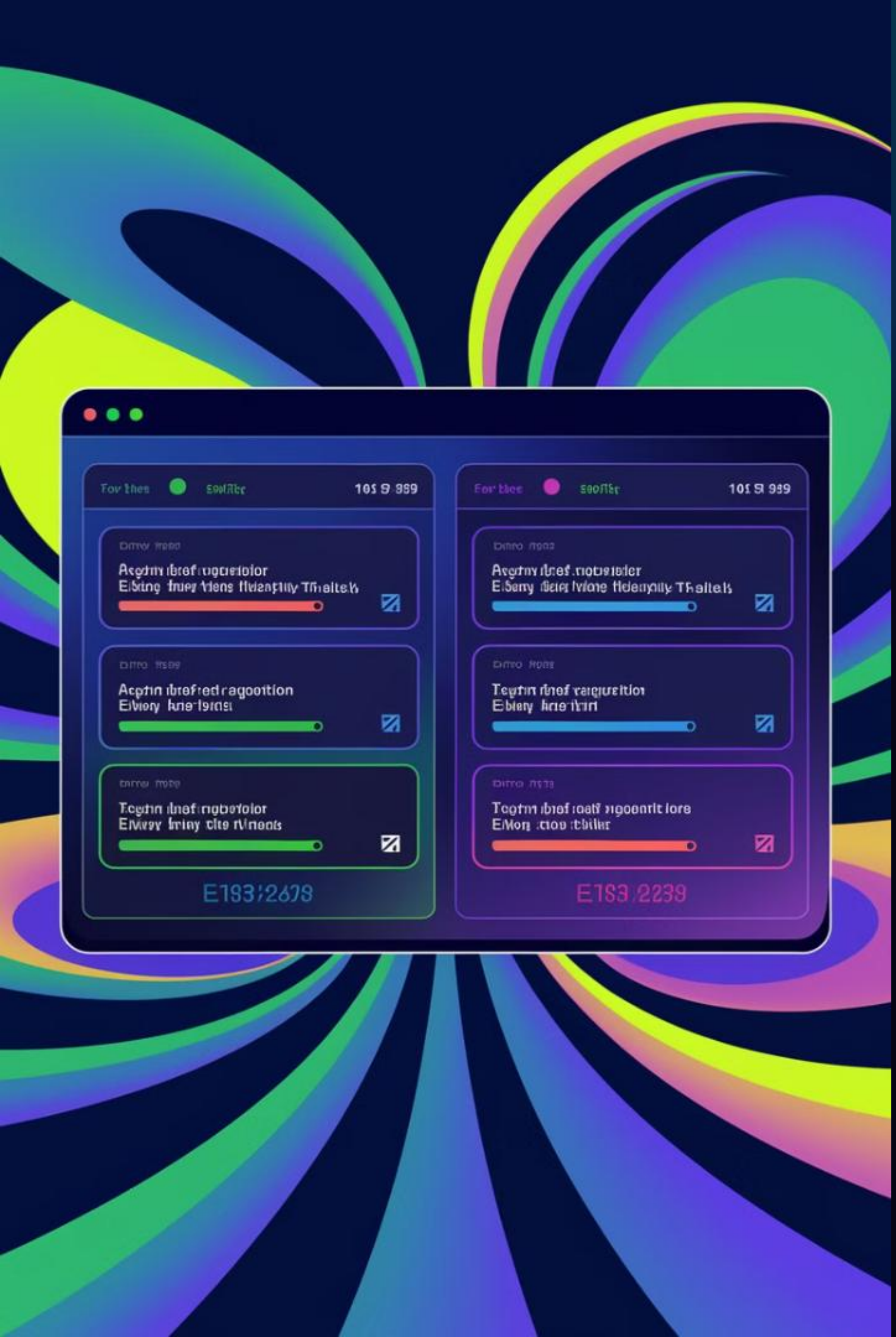


Ramamonjison et. al., NL4Opt Competition: Formulating Optimization Problems Based on Natural Language Descriptions

Experimental Results



The **Hybrid Approach** combining classical feature engineering with optimization-fine-tuned language models achieves the best performance with a micro-averaged F1 score of **0.919**. This represents a significant improvement over the baseline classical approach (0.816) and the previous state-of-the-art (0.888). The **most challenging entity** to identify is the objective name (OBJ_NAME), where the hybrid approach shows the largest improvement over other methods.



Comparison with Large Language Models (GPT-4)

1

Zero-Shot GPT-4

Direct application of GPT-4 without examples achieves only **0.546** F1 score, struggling with entity boundaries and disambiguation.

2

Few-Shot Learning

Adding examples improves performance significantly, with five examples reaching **0.838** F1 score, demonstrating the importance of in-context learning.

3

Hybrid Approach

Our dedicated Ner4Opt hybrid solution (**0.919** F1) still outperforms even few-shot GPT-4, highlighting the value of specialized approaches for optimization tasks.

Ner4Opt for Modeling Assistants

44.44%

Without Annotations

GPT-4 with problem description only
MiniZinc model generation

65.66%

With Ner4Opt Annotations

GPT-4 with problem description + Ner4Opt
MiniZinc model generation

Ner4Opt Open-Source Library

Library Features

Simple API for extracting optimization entities from text, with options to select different model types and confidence thresholds.

OBIE Output Format

Returns a list of dictionaries, each containing entity information including start/end indices, text, entity type, and confidence score.

Pre-trained Resources

Source code, training protocols, and **pre-trained models** are all publicly available through GitHub and Hugging Face.

pip install ner4opt

https://huggingface.co/spaces/skadio/ner4opt

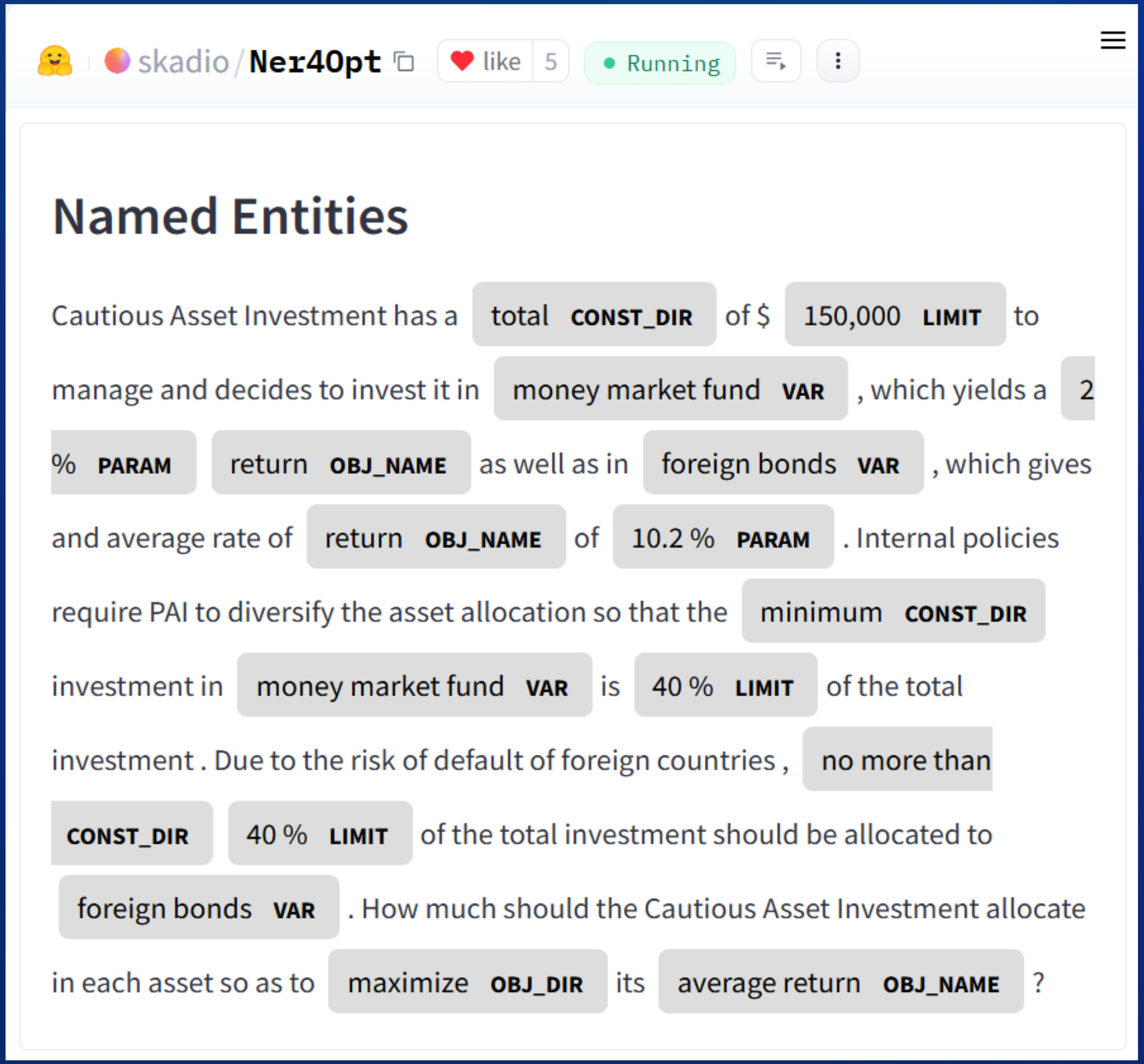
Interactive Demo

User Interface

Demo interface with a model selection panel, **a text input field for problem description** and an output panel that highlights identified entities with color-coded tags.

Model Selection

Users can choose between **different model** types (lexical, lexical-plus, semantic, hybrid) to compare performance on the same input text.



Hugging Face Spaces Demo

https://huggingface.co/spaces/skadio/ner4opt

ChatOpt Integration

Integration

Designed for easy integration with optimization solvers and modeling frameworks, envisioned as part of **Holy Grail 2.0** constraint programming modeling assistant.

- *Holy Grail 2.0: From Natural Language to Constraint Models, CP 2023, D. Tsouros, H. Verhaeghe, S. Kadıoğlu, T. Guns*
- *Constraint modelling with LLMs using in-context learning, CP 2024, K. Michailidis, D. Tsouros, T. Guns*

ChatOpt Integration

Integra

Designed t
modeling f
constraint

U

You are designing an office space with two types of desks: long desks and short desks. You can spend at most $2000. Long desks cost $300, take up 10 square feet of space, and seat 6 employees. Short desks cost $100, take up 4 square feet of space, and seat 2 employees. The office can have at most 200 square feet of desks. How many of each desk should you buy in order to maximize the seating availability?

B

Variables:

- Number of long desks: Long
- Number of short desks: Short

Objective: Maximize seating availability: 6 * Long + 2 * Short

Constraints:

- Total cost should not exceed $2000: 300 * Long + 100 * Short <= 2000
- Total space taken by desks should not exceed 200 square feet: 10 * Long + 4 * Short <= 200

Pseudo Model

Generate An Executable CPMpy Model

- ❑ *Holy Grail 2.0: From Natural Language to Constraint Models, CP 2023, D. Tsouros, H. Verhaeghe, S. Kadıoğlu, T. Guns*
- ❑ *Constraint modelling with LLMs using in-context learning, CP 2024, K. Michailidis, D. Tsouros, T. Guns*

ChatOpt Integration

Integra

Designed

modeling

constraint

You are designing an office space with … 0 square feet of space, and seat 6 employees. Short desks co… esks. How many of each desk should you buy in order to maxim…

U

B

Variables:

- Number of long desks:
- Number of short desks

Objective: Maximize seating

Constraints:

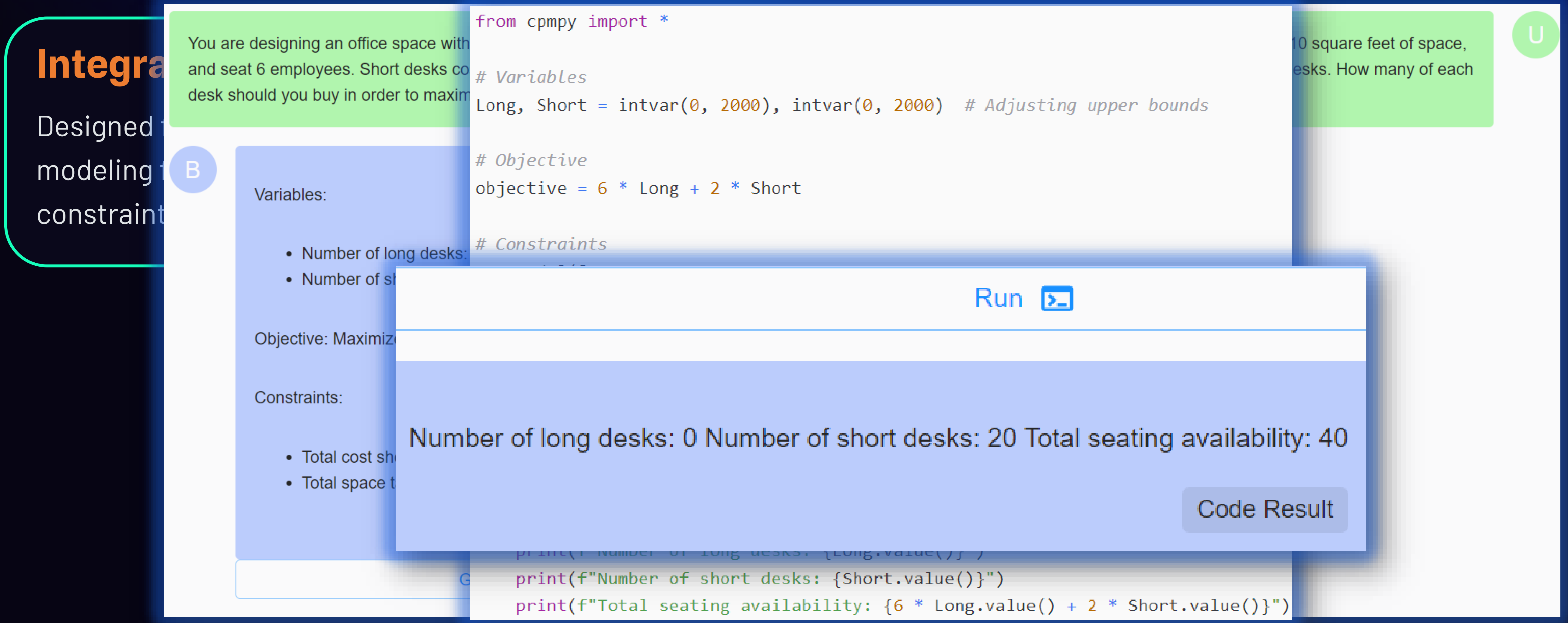
- Total cost should not e
- Total space taken by d

G

```python
from cpmpy import *

# Variables
Long, Short = intvar(0, 2000), intvar(0, 2000)  # Adjusting upper bounds

# Objective
objective = 6 * Long + 2 * Short

# Constraints
m = Model([
    300 * Long + 100 * Short <= 2000,  # Cost constraint
    10 * Long + 4 * Short <= 200  # Space constraint
])

# Maximizing seating availability
m.maximize(objective)

# Solve the model
if m.solve():
    print(f"Number of long desks: {Long.value()}")
    print(f"Number of short desks: {Short.value()}")
    print(f"Total seating availability: {6 * Long.value() + 2 * Short.value()}")
```


- *Holy Grail 2.0: From Natural Language to Constraint Models, CP 2023, D. Tsouros, H. Verhaeghe, S. Kadıoğlu, T. Guns*
- *Constraint modelling with LLMs using in-context learning, CP 2024, K. Michailidis, D. Tsouros, T. Guns*

ChatOpt Integration



- Holy Grail 2.0: From Natural Language to Constraint Models, CP 2023, D. Tsouros, H. Verhaeghe, S. Kadıoğlu, T. Guns
- Constraint modelling with LLMs using in-context learning, CP 2024, K. Michailidis, D. Tsouros, T. Guns

Constraints'24

Constraints (2024) 29:261–299
https://doi.org/10.1007/s10601-024-09376-5

NER4OPT: named entity recognition for optimization modelling from natural language

Serdar Kadıoğlu[1,2] · Parag Pravin Dakle[1] · Karthik Uppuluri[1] · Regina Politi[2] · Preethi Raghavan[1] · SaiKrishna Rallabandi[1] · Ravisutha Srinivasamurthy

github.com/skadio/ner4opt

pip install ner4opt

What’s Next?

Integration with Solvers

Embedding Ner4Opt directly into optimization platforms to enable natural language interfaces for model creation.

Domain Adaptation

Extending the approach to specialized fields like supply chain, finance, and healthcare with domain-specific entity types.

Interactive Modeling

Developing conversational interfaces that use Ner4Opt to clarify ambiguities and refine optimization models through dialogue.

Text2Zinc

A unified cross-domain dataset curated for LLM co-pilots and an associated leaderboard to evaluate strategies to generate MiniZinc models from natural language text.

Our Contributions

Holy Grail 2.0

Blueprint for optimization modelling co-pilots with feedback loop and user interactivity.
Tsouros et. al., 2023

Ner4Opt

A principled approach to extracting components of optimization models such as the objective, variables, and constraints from free-form natural language text.
Kadioglu et. al, Constraints'24

Text2Zinc

A **unified cross-domain dataset** curated to work with **LLM co-pilots** and **leaderboard** to evaluate strategies to generate **MiniZinc models** from free-from natural language text. *Singirikonda et. al., AAAI'25*

Gala: Global LLM Agents

A global agentic approach with multiple specialized LLM agents decompose the modeling task by global constraint type.
Cai et. al. NeurIPS'25

Text2Zinc: Motivation

Driving Progress

Datasets and benchmarks **fuel progress** in various domains: Computer Vision, NLP, and SAT, CP, MIP, RecSys, etc.

Room for Improvement

Current problem datasets have **potential for improvement** for integration with language models.

Structured Information & Metadata

Models and natural language descriptions of problems have been documented heavily but seldom occur together. Crucial **metadata is unavailable**.

Existing Resources

Optimization

NL4OPT

- Linear programming problems
- No separation between problem description and data
- Relatively easy instances

NLP4LP

- Extends NL4OPT
- Introduces MIPs
- Evaluated with GurobiPy and cvxpy

ComplexOR

- Standard OR Problems
- Evaluated with GurobiPy

Satisfaction

Logic Grid Puzzles

- Introduces satisfaction problems in the form of logic grid puzzles

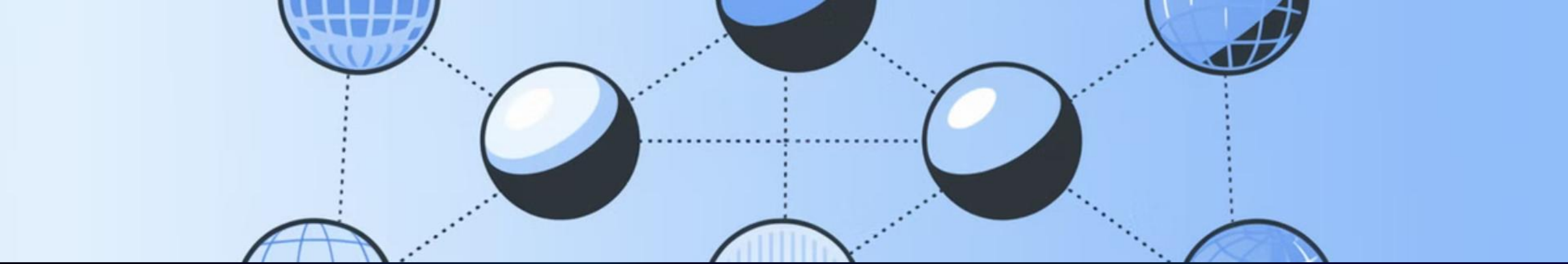
CSPLib

- CP and Satisfaction problems
- Not designed to work with ML or LLMs

Hakank's Models

- Extensive set of constraint models in various languages
- Does not capture metadata

** Massive thank you to the community for contributing these valuable resources!*



Text2Zinc: Addressing Dataset Gaps

1

Cross-Domain

- Focus on combining both **optimization & satisfaction** problems.
- **The first (and currently only)** dataset to incorporate LP, MIP, and CP problems.

2

Unified Format

- Unifies existing datasets.
- **Clear separation** of problem description & instance data.

3

Solver Agnostic

- Enables **solver agnostic** approaches.
- MIP, CP, SAT, LCG through MiniZinc.

4

Data Augmentation

- Clear and concise descriptions.
- Input and output specification.
- **Metadata** generation.
- Manual verification.

S. Kadıoğlu et. al. Text2Zinc: A Cross-Domain Dataset for Modeling Optimization and Satisfaction Problems in MiniZinc

MiniZinc: Solver-Agnostic Modeling

High-Level Language

MiniZinc supports both discrete and continuous optimization and satisfaction problems with an intuitive, declarative syntax.

Multiple Backends

Solver-agnostic design communicates with CP, MIP, and SAT solvers through FlatZinc compilation—write once, run anywhere.

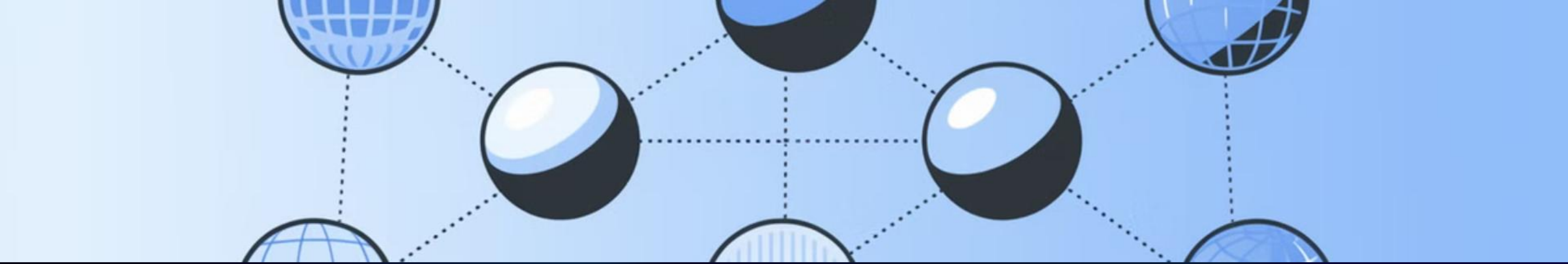
Global Constraints

Powerful abstractions like all_different simplify modeling, replacing numerous pairwise constraints with single declarations.

MiniZinc Example: All Different Constraint

```
% pairwise binary inequalities
all_different(array[int] of var int:x)=
    forall(i,j in index_set(x) where i < j)
        x[i] != x[j]

% built-in global constraint
all_different(array[int] of var int:x)=
    gecode_all_different(x) % native Gecode version
```

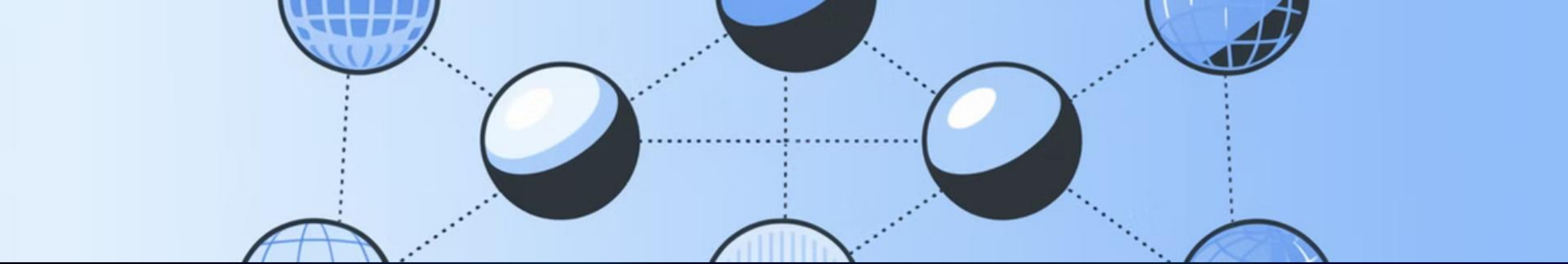
Global constraints allow users to **leverage higher-level abstractions** rather than focusing on low-level decomposition, significantly simplifying the modeling process.



Text2Zinc: Example Timetabling Problem – Description


```
"description": "Lecture timings need to be scheduled for courses across a limited number of periods. Each
    course requires a specific number of lectures and can only be assigned to certain periods due to
    availability constraints. Some courses have conflicts due to having common students and cannot be scheduled
    at the same time. Additionally, there is a limited number of rooms that can be used and thus a maximum
    number of lectures that can occur simultaneously. How can we allocate lectures to periods while ensuring all
     constraints are met?",
"identifier": "or_lp_ip_scheduling_problem_2",
"metadata": {
  "name": "Timetable Problem", "domain": "Scheduling", "objective": "satisfy", "source": "hakank", "constraints
      ": [
    "forall", "<=", "+", "=", "sum"]
  }
}
```

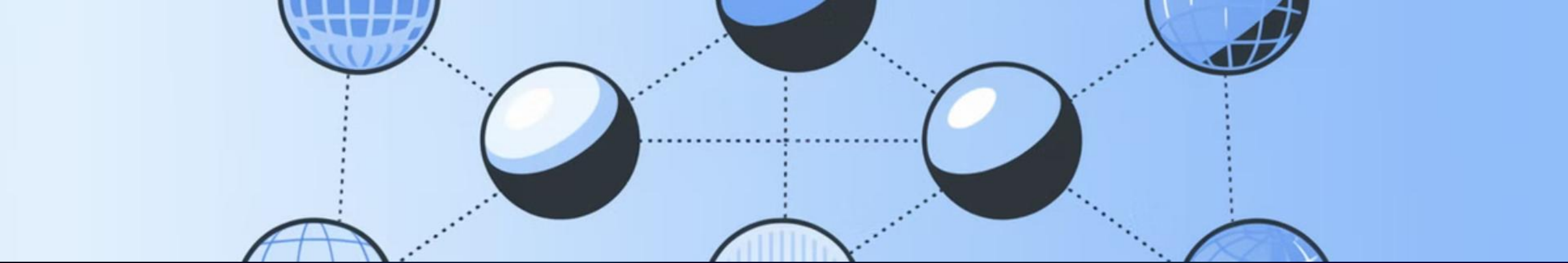

Figure 2 **An example input with description, parameters, metadata, and output fields.**



Text2Zinc: Example Timetabling Problem – Model

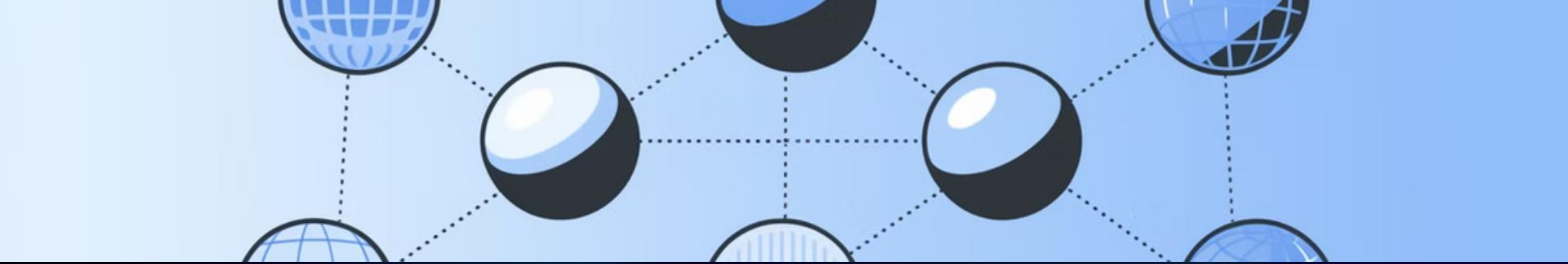
model.mzn

```
include "globals.mzn";

% Input parameters
int: courses;
int: periods;
int: rooms;

array[1..courses, 1..periods] of int: available;
array[1..courses, 1..courses] of int: conflict;
array[1..courses] of int: requirement;

% Decision variables
array[1..courses, 1..periods] of var 0..1: timetable;
```



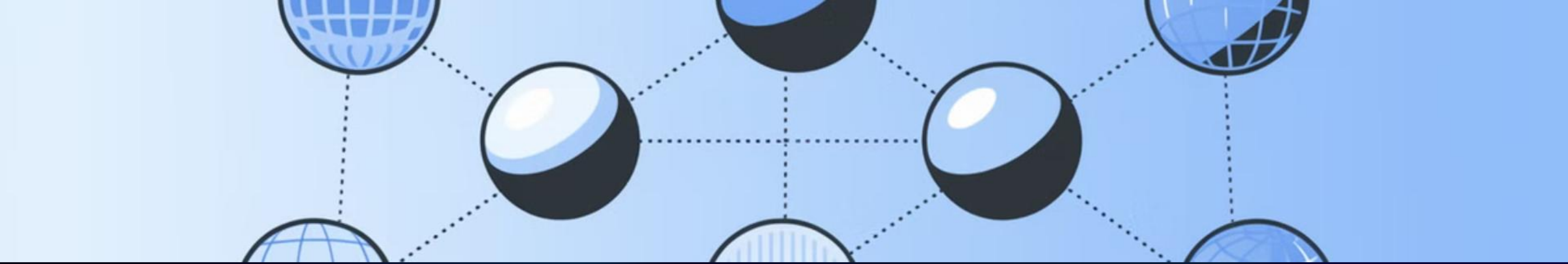
Text2Zinc: Example Timetabling Problem – Model

```
constraint
   % 1. Conflicts: Courses with common students must not be scheduled at the same time
   forall(c1, c2 in 1..courses where c1 < c2) (
      if conflict[c1, c2] = 1 then
         forall(p in 1..periods) (
            timetable[c1, p] + timetable[c2, p] <= 1
         )
      else
         true
      endif
   )
   % 2. Availabilities: Courses can only be scheduled in available periods
   /\
   forall(c in 1..courses, p in 1..periods) (
      if available[c, p] = 0 then
         timetable[c, p] = 0
```



Text2Zinc: Example Timetabling Problem – Model

```
% 3. Rooms: At most `rooms` lectures can be scheduled per period
/\
forall(p in 1..periods) (
   sum([timetable[c, p] | c in 1..courses]) <= rooms
)
% 4. Number of lectures per course must match the requirement
/\
forall(c in 1..courses) (
   sum([timetable[c, p] | p in 1..periods]) = requirement[c]
);
```



Text2Zinc: Example Timetabling Problem – Input & Output

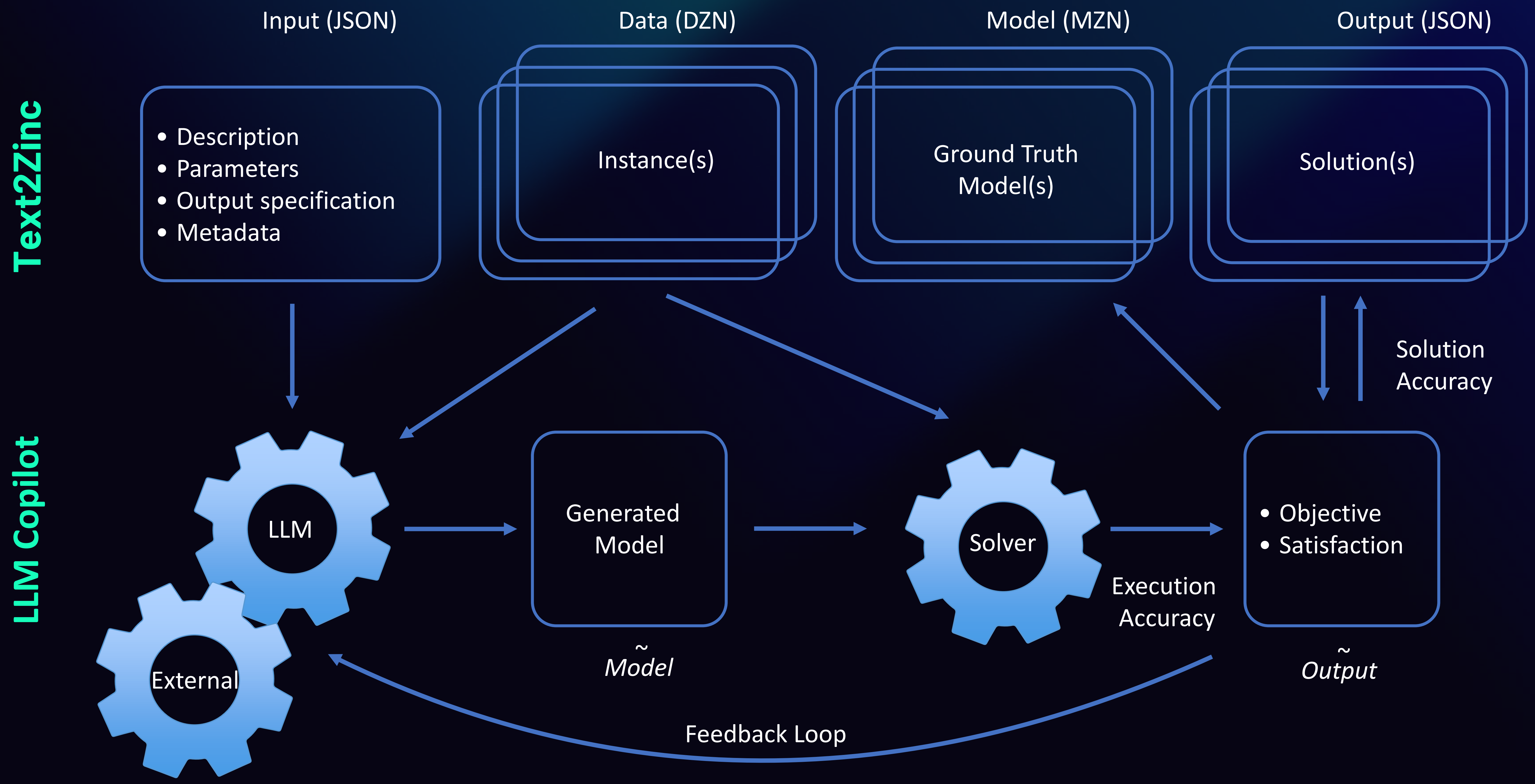
data.dzn

```
int: courses = 5;
int: periods = 20;
int: rooms = 2;
array[1..courses, 1..periods] of int:
     available = array2d(1..courses,
    1..periods, [
    % 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
        7 8 9 0
      0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
          1, 0, 1, 1, 0, 1, 1, 1, 1,
      1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1,
          1, 1, 1, 1, 1, 1, 1, 1, 1,
      0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
          1, 0, 1, 1, 1, 1, 0, 1, 1,
      1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1,
          1, 1, 1, 1, 1, 1, 0, 1, 1,
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
          1, 1, 1, 1, 1, 1, 1, 1, 1
```

output.json

```
{
"timetable": [
   [0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0,
      1, 0, 1, 0, 0, 0, 0, 0, 0],
   [1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
      0, 1, 0, 1, 1, 1, 1, 1, 1],
   [0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
      1, 0, 1, 1, 1, 1, 0, 1, 1],
   [0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1,
      0, 0, 0, 0, 0, 0, 0, 0, 0],
   [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
      0, 1, 0, 0, 0, 0, 1, 0, 0]
 ]
 }
```

Text2Zinc: A Unified Approach



Text2Zinc Statistics

567

Total Problems

Natural language instances

110

Manually Verified

High-quality curated data

11

Problem Domains

Diverse aplication areas covered

Our dataset includes instances of mixed of **LP, MIPs, and CP problems** across **6 different sources**
Providing a comprehensive benchmark for natural language to constraint model translation.

Text2Zinc Co-Pilot Approaches

Out-of-the-box LLM

Vanilla prompting, zero-shot, few-shot performance

Single vs. Multi-Call

Chain-of-Thought

Improved reasoning through step-by-step problem-solving

Structured Prediction

Grammar-based model generation to enforce LLM output

Knowledge Graph

Leveraging structured knowledge as intermediary representation

Text2Zinc Initial Results (GPT-4)

Solution Approach	Execution Accuracy	Solution Accuracy	LLM Calls
Baseline	0.33	0.17	1
Chain-of-Thought (CoT)	0.57	0.28	1
Knowledge Graph	0.48	0.26	2
CoT + Code Validation	0.57	**0.28**	2
CoT + Grammar Validation	0.63	0.23	3
CoT + Code & Grammar Val.	**0.70**	0.25	3
Compositional	0.44	0.20	4
Compositional + Code Val.	0.44	0.21	5

[Hugging Face Text2Zinc Leaderboard]

Text2Zinc Initial Results (GPT-4o Reasoning)

Solution Approach	Execution Accuracy	Solution Accuracy	LLM Calls
Baseline	0.33	0.17	1
Chain-of-Thought (CoT)	0.57 – **0.61**	0.28 – **0.35**	1
Knowledge Graph	0.48	0.26	2
CoT + Code Validation	0.57 – **0.81**	**0.28** – **0.41**	2
CoT + Grammar Validation	0.63	0.23	3
CoT + Code & Grammar Val.	**0.70** – **0.74**	0.25 – **0.40**	3
Compositional	0.44	0.20	4
Compositional + Code Val.	0.44	0.21	5

Hugging Face Text2Zinc Leaderboard

General Observations

1 Execution-Solution Gap

Consistently lower solution accuracies across strategies indicate **complexity of modeling** expertise

2 Compilation Issues

Syntax errors are primary cause of execution failures, attributed to LLM's limited exposure to **MiniZinc's specialized syntax**

3 Information Sweet Spot

Both too little and too much information can be detrimental, suggesting an optimal **level of context** exists

4 Reasoning vs. Structure

Superior **performance of CoT and compositional** approaches suggests how information is processed matters more than quantity provided

informs
https://pubsonline.informs.org/journal/ijds

INFORMS JOURNAL ON DATA SCIENCE

Vol. 00, No. 0, Xxxxx 0000, pp. 000–000

ISSN 2694-4022, EISSN 2694-4030

Text2Zinc for Large-Language Models as Modeling Co-Pilots

Serdar Kadıoğlu
Department of Computer Science, Brown University; AI Center of Excellence, Fidelity Investments, serdark@cs.brown.edu

Akash Singirikonda
Department of Computer Science, Brown University, akash_singirikonda@brown.edu

Karthik Uppuluri
AI Center of Excellence, Fidelity Investments, karthik.uppuluri@fmr.com

hf.co/datasets/skadio/text2zinc

hf.co/spaces/skadio/text2zinc-leaderboard

What's Next?

1. **Call-to-Action: Dataset Expansion**

   Encourage community contributions to create more comprehensive resources

2. **Context Engineering**

   Explore alternative intermediate representations and semantic graphs

3. **Model Improvements**

   Develop specialized LLMs for optimization and satisfaction tasks

4. **Agentic Frameworks**

   Investigate potential of agentic approaches in capturing nuances of optimization modeling problems

Our Contributions

Holy Grail 2.0

Blueprint for **optimization modelling co-pilots** with feedback loop and user interactivity.
Tsouros et. al., 2023

Ner4Opt

A principled approach to **extracting components of optimization models** such as the objective, variables, and constraints from free-form natural language text.
Kadioglu et. al, Constraints'24

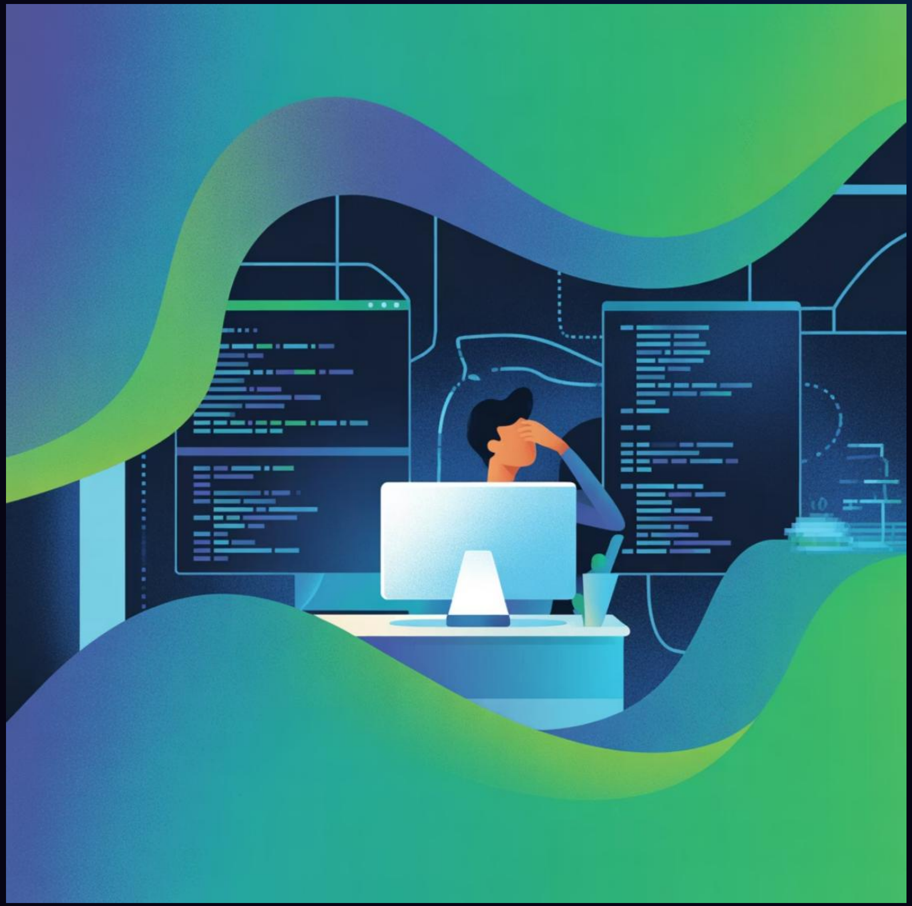
Text2Zinc

A **unified cross-domain dataset** curated to work with **LLM co-pilots** and **leaderboard** to evaluate strategies to generate **MiniZinc models** from free-from natural language text. *Singirikonda et. al., AAAI'25*

Gala: Global LLM Agents

A **global agentic approach** with multiple specialized LLM agents decompose the modeling task by global constraint type.
Cai et. al. NeurIPS'25

The Challenge: Precise Logic & Domain Knowledge



Current State

Notoriously difficult to translate into correct MiniZinc models. This process demands both **logical reasoning** and **modeling expertise**.

Problem

General-purpose prompting often **fails to capture** all variables and constraints correctly, especially for harder optimization problems.

Current approaches using and even powerful LLMs are **"not yet a push-button technology"** for generating combinatorial models from text.

Motivation

This motivates research into more structured and guided methods that **break problems into manageable pieces.**

Multi-Agents for LLM Co-Pilots

Multi-step and multi-agent frameworks have emerged as promising solutions for natural language optimization tasks. By dividing complex problems into manageable pieces, each LLM handles a **simpler reasoning challenge,** potentially reducing overall complexity.

01

Chain-of-Experts

Assigns **multiple LLM experts** with specific roles (interpreting text, formulating components, coding, verifying) coordinated by a central conductor Xiao et al. [2023]

02

OptiMUS System

Uses **LLM-based agent** to iteratively identify parameters, write constraints, and debug linear program models AhmadiTeshnizi et al. [2023]

03

Promising Results

These approaches considerably **improve over single-LLM methods** on complex operations research problems.

Breaking Down the Complexity Further

Current Limitations

Existing multi-agent approaches still inherit the **full problem complexity** rather than focusing on tractable sub-tasks.

Our Novelty

Gala centers around **global constraints**: high-level CP primitives like all_different that capture common patterns.

Agents meets Constraint Programming

all_different

Enforces distinct values across variables

cumulative

Enforces scheduling resource capacity over time

global_cardinality

Limits how many variables take each value

circuit

Creates a Hamiltonian circuit

Key Advantage

Gala aligns and **combines the key strength of Constraint Programming with Agentic Frameworks**, turning model generation into a collaboration of focused experts.

Global Constraints

High-level primitives that concisely **represent recurring patterns** such as distinct, resource limits, ordering, and counting found across planning, scheduling, assignment, and configuration problems.

Gala Agents

Each specialized agent **focuses only on detecting and encoding one constraint type,** simplifying the reasoning task dramatically.

Agentic Solution: Gala Framework

Gala is an agentic framework that addresses text-to-model translation with **global agents:** multiple specialized LLM agents **decompose** the modeling task by **global constraint type.**

Decomposition

Problem divided into smaller, well-defined sub-tasks.

Specialized Agents

Each agent detects and generates code for a specific class of global constraint.

Integration

Assembler agent integrates constraint snippets into complete model.

How Agents Work?

Specialized LLM Agents

LLM agent with a specialized prompt for each global constraint.
Each agent receives the full problem description, but its **instructions are local**: detect if the constraint is present and produce the MiniZinc snippet

Binary Classification + Code

Each agent performs binary classification (constraint present or not) followed by code generation. The agent is instructed **not to produce any other modeling elements** beyond its constraint.

No Broader Modeling

By isolating each agent's focus, we simplify the reasoning task. Unlike previous agentic approaches, **our agents do not need to understand the entire problem structure**, only whether a specific pattern appears.

> *"You are a MiniZinc modeling assistant specialized in detecting and modeling all_different constraints.*
> *Given a problem description, decide whether it requires one or more all_different constraints.*
> *If it does, generate only MiniZinc code specifying the all_different constraint and its variables.*
> *If it does not, return FALSE."*

The Assembler: Bringing It All Together

Assembler LLM takes over once constraint-specific agents return code snippets. Prompted as a MiniZinc modeler tasked with compiling a complete and coherent model.

01

Declare Variables

Define all decision variables and domains, renaming or merging for consistency

02

Analyze Constraints

Decide whether to include provided global constraints

03

Fill Gaps

Add remaining constraints from text not covered by hints

04

Define Objective

Determine optimization objective or satisfy goal

05

Finalize Model

Append solver boiler plate and output format

Key Benefit: Much of the heavy lifting is done by specialized snippets, the assembler focuses on gluing components together and writing boilerplate code.

Initial Results: Evaluation Framework

We conduct an initial evaluation of Gala, focusing on **two critical aspects** of the system's performance:

1

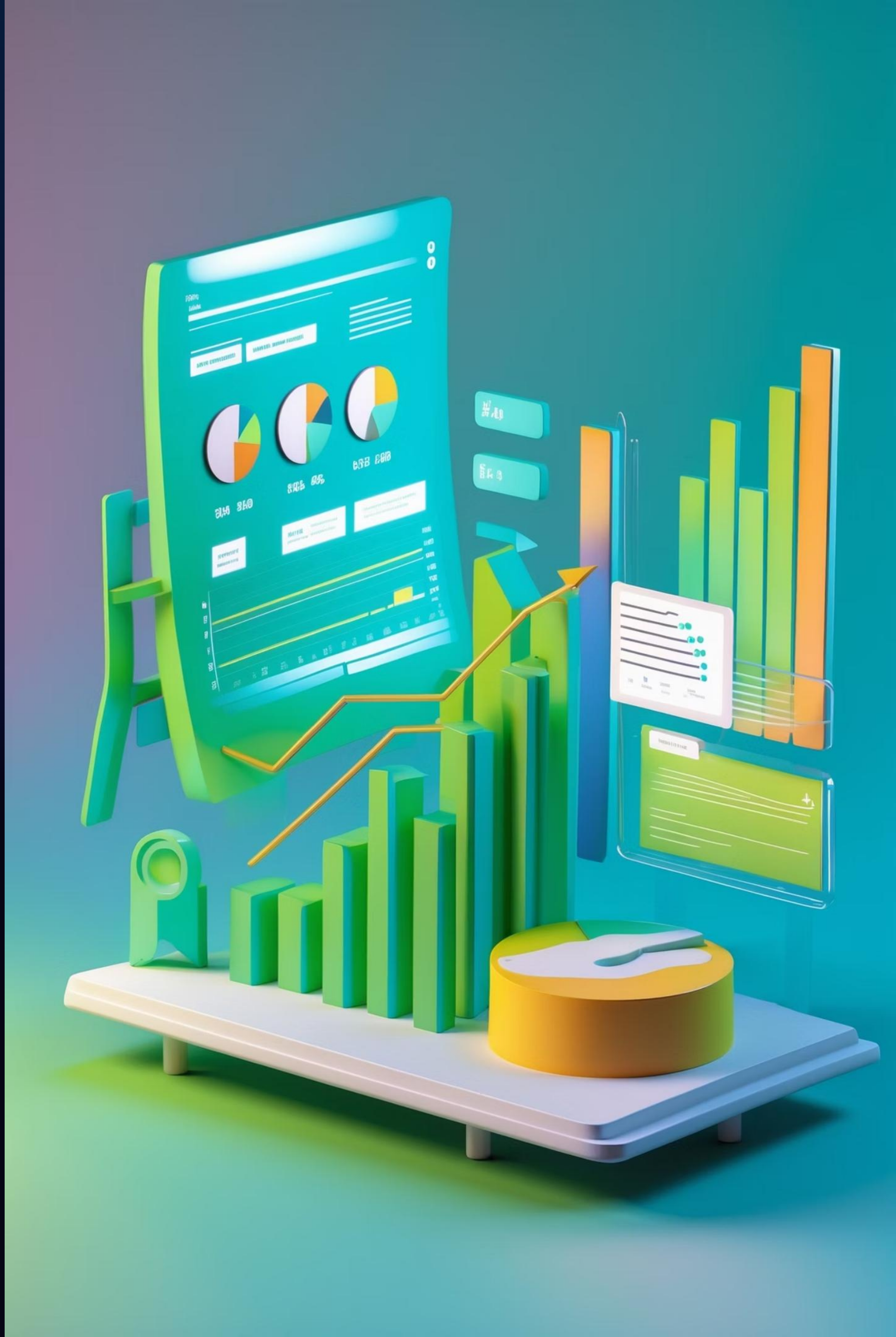
Global Agent Detection

The ability of global agents to correctly detect global constraints in problem descriptions.

2

End-to-End Performance

The end-to-end performance of the agentic pipeline compared to baseline prompting strategies and chain-of-thought (CoT).



Global Agent Detection Performance

We evaluate detection performance for seven global constraints using Phi4 on all 567 Text2Zinc instances

Constraint Type	Detection Rate (%)	False Detection Rate (%)
circuit	**100**	2.75
all_different	**88.1**	14.42
increasing	**78.6**	4.67
global_cardinality	**77.8**	17.43
lex_less	**71.4**	6.6
count	67.9	**28.3**
cumulative	58.3	17.59

Strong Detection Rates

Overall, our agents achieve detection rates around 70% to 80%, with circuit achieving perfect 100% detection.

Room for Improvement

False detection rates are generally low for rarer constraints. The main exception is `count` **(28.3%),** Distinguishing counting patterns from numerical constraints remains a challenge.

Gala End-to-End Modeling Performance

We compare Gala with direct prompting (baseline) and COT on 110 verified Text2Zinc instances.

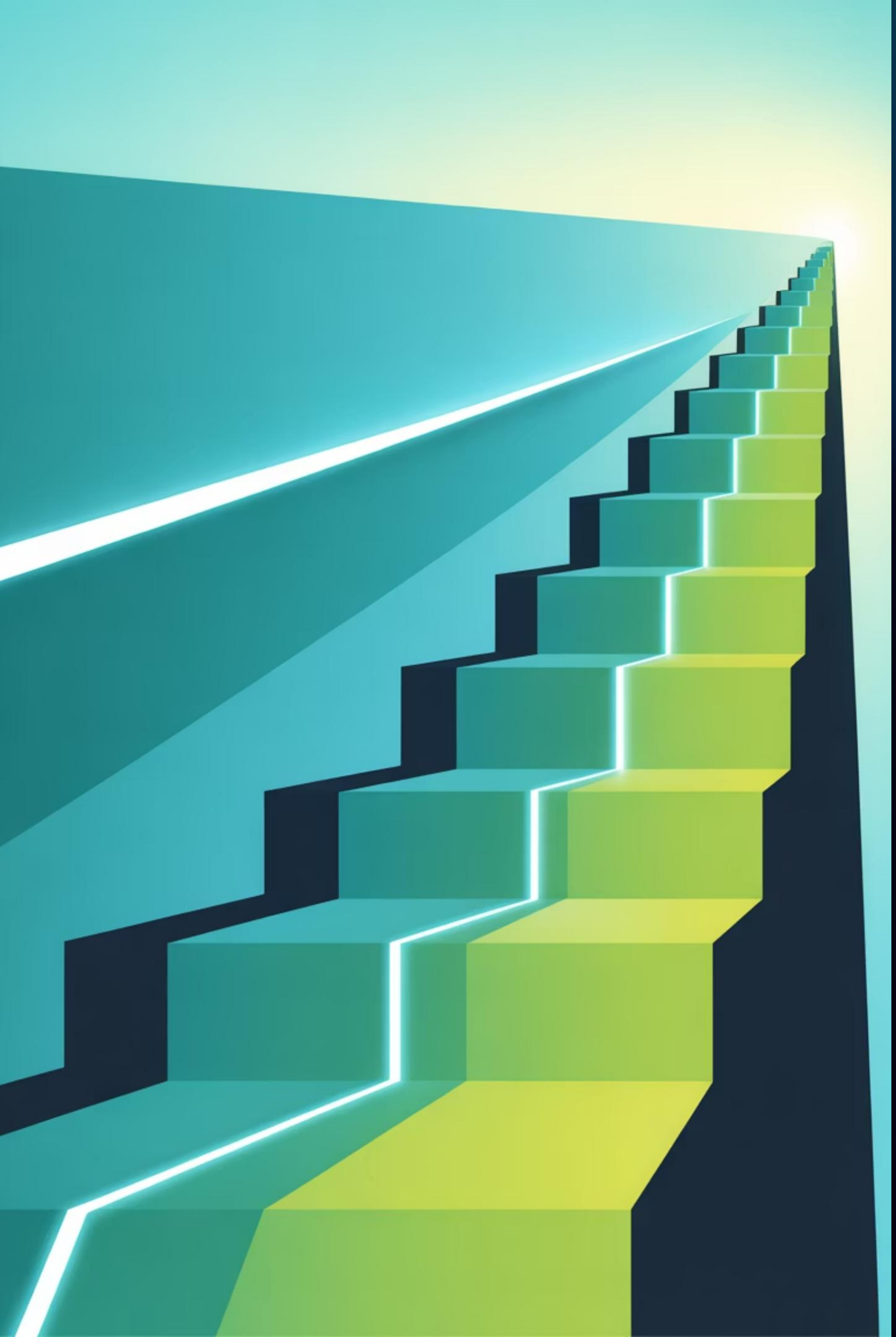
Model & Strategy	Execution Rate (%)	Solve Rate (%)
o3-mini Gala	**57.27**	**32.73**
o3-mini CoT	52.73	30.91
gpt-4o-mini Gala	**33.64**	**17.27**
gpt-4o-mini CoT	31.82	12.73
gpt-oss:20b Gala	**17.27**	8.18
gpt-oss:20b CoT	16.36	**10**
gpt-oss:20b baseline	11.81	7.27

Consistent Performance

Gala **consistently outperform** CoT on stronger models (o3-mini, GPT-4o-mini) across execution and solve

Decomposition Advantage

Gains come from agentic assembly with **no model tuning, prompt optimization**, or hyperparameter tuning.



Potential Enhancements

Optimize Global Agents

Replace hand-crafted prompts with **systematic optimization**, **curated few-shot** exemplars, and adversarial negatives. Consider **fine-tuning per global constraint** to boost precision and recall.

Unblock the Assembler

Add supervisor to **extract variables and objectives** before delegation. Build systematic error taxonomy to map where agents succeed or fail, driving targeted fixes and **feedback loops**.

Scale Evaluation

Run **stronger LLMs** (e.g., GPT-5) and sweep both open- and closed-weight models. Benchmark on **global constraint rich datasets** to better showcase the approach.

MLxOR@NeurIPS'25

GALA: Global LLM Agents for Text-to-Model Translation

Junyang Cai[1], **Serdar Kadıoğlu**[2,3], **Bistra Dilkina**[1]

[1]Department of Computer Science, University of Southern California

[2]AI Center of Excellence, Fidelity Investments

[3]Department of Computer Science, Brown University

caijunya@usc.edu, serdark@cs.brown.edu, dilkina@usc.edu

hf.co/datasets/skadio/text2zinc

hf.co/spaces/skadio/text2zinc-leaderboard

Emerging Literature

- **Co-Pilot Position Papers:** Holy Grail 2.0 (Tsouros et.al. 2023), Co-Pilot Manifesto (Wasserkrug et al. 2025)
- Learning natural **language interfaces** with neural models, Li Dong, PhD thesis, 2019
- **LGPSolver:** Solving logic grid puzzles automatically, ACL'20
- Automatic formulation and optimization of **linear problems** from a structured paragraph, ICSCT'21
- Early efforts (Ramamonjison et al. 2022) focused on linear programming problems using entity recognition and logical forms
- Synthesizing **MIP models** from NL, Qingyang Li et. al., 2023
- **Latex2Solver** turns input .tex files into optimization models + symbolic model UI, Ramamonjison et.al, ACL'23
- **OptiGuide:** LLMs for Supply Chain Optimization, Microsoft, 2023
- **MeetMate:** Enabling interactive decision support using LLMs and CP, Lawless et al., 2023
- **Logic.py:** Software verification through logic (Kesseli et. al. 2025)
- Towards an Automatic Optimization Model Generator Assisted with GPT, Almonacid, 2023
- **LM4OPT:** Unveiling the Potential of LLMs in Formulating Mathematical Optimization Problems, 2024
- **Agents:** Multi-agent chain-of-experts (Xiao et al. 2023) **Optimus** (AhmadiTeshnizi et al. 2024) specific to Gurobi and cvxpy
- **Data scarcity** for LPs in PuLP addressed by data augmentation, leveraging CodeT5 (Prasath and Karande 2023)
- **OR-Instruct:** Training custom LLMs through solver specific OR-Instruct (Huang et al. 2024a).
- **MAMO benchmark** (Huang et al. 2024b), focusing on LLMs' mathematical modeling processes rather than solution correctness
- **Streamliners:** LLMs for generating streamliners in CP using MiniZinc (Voboril et al. 2024)
- **RAG:** In-context learning and RAG to build CPMPY constraint models (Michailidis et al. 2024)
- **Privacy:** Domain-specific applications, focusing on supply chain optimization while preserving data privacy (Li et al. 2023)
- **Infeasible:** Diagnosing infeasible optimization problems through interactive conversations (Chen et al. 2023)
- **Generation:** optimization problems from scratch (Jiang et al. 2025)
- **MCP:** Model context protocol to integrate LLMs and with symbolic solvers (Szeider 2025)

Strategic Pillars of Enterprise AI @ Fidelity AI Center

AI Learning from Offline Data

Robust, scalable, reproducible features from structured, unstructured, and semi-structured datasets.

Selective, TextWiser, Seq2Pat

AI for Learning from Online Feedback

Adaptive, real-time, A/B testing systems that continuously learn from user interaction.

Mab2Rec, MABWiser

AI for Decision Making

Large-scale, integrated, (meta) solvers for resource management and optimization.

Forge, Balans, PathFinder

AI for Automated Assistants

Extraction and translation of natural language into downstream tasks and intents for human-computer interaction

Gala, Ner4Opt, Text2Zinc, iCBS (w/ Amazon)

Responsible AI

Horizontal capabilities for explainability, evaluation, fairness, and bias mitigation across all systems

Jurity, BoolXAI (w/ Amazon)

Open-Source AI at Scale: Establishing an Enterprise AI Strategy [AI Magazine'25]

skadio.github.io