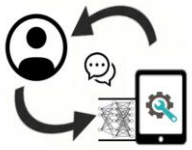
Based on paper (accepted at ECAI 2025):
CP-Bench: Evaluating Large Language Models for Constraint Modelling

CP-Bench: Benchmarking LLMs for CP Model Generation

Kostis Michailidis, Dimos Tsouros, Tias Guns

DTAI, KU Leuven, Belgium

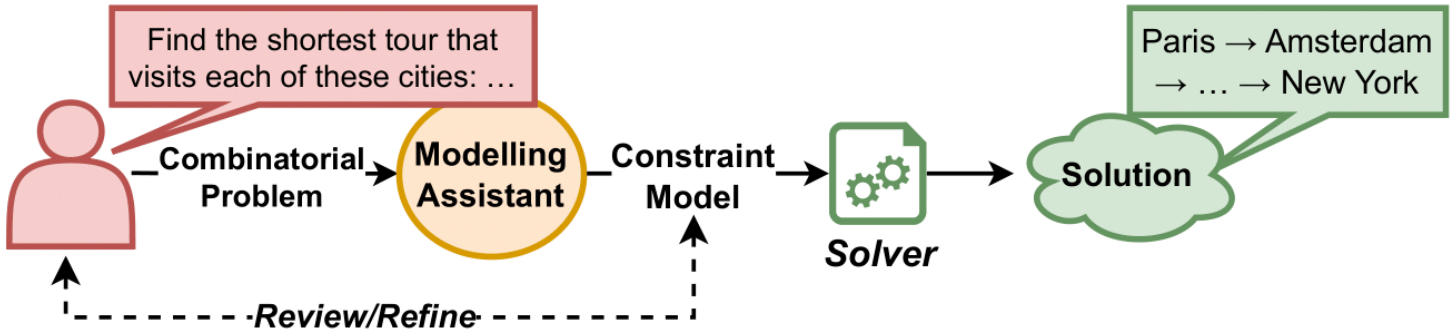
LLMs meet Constraint Solving | CP/SAT 2025 Workshop



why do we care about benchmarking LLMs for modelling?

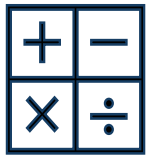
LLMs as modelling assistants

- Coding assistants (e.g. Copilot) excel in general coding tasks
- What about CP Modelling?



CP is powerful, but modelling requires expertise in:

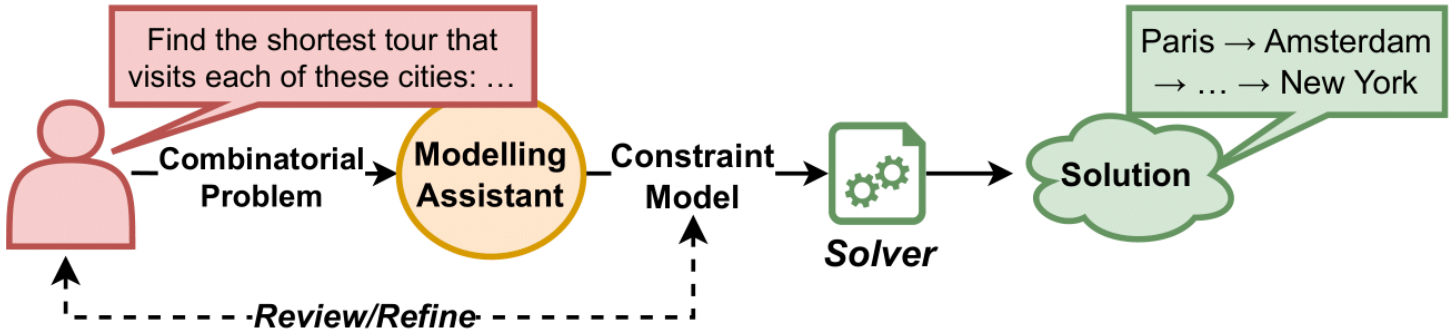
Domain Knowledge



Mathematical Reasoning

Solver Syntax & Semantics

LLMs as modelling assistants



- What about CP Modelling?
- **We need representative benchmarks!**

Dataset of problems/models

Evaluation Challenge: Multiple variable/modelling choices (viewpoints) & multiple (optimal) solutions

Evaluation Challenge:
No unit tests…
How to (semantically) verify?

why another dataset?

Existing LLM-based modelling benchmarks

- NL4Opt => nl4opt.github.io
- NLP4LP => huggingface.co/datasets/udell-lab/NLP4LP
- MAMO => huggingface.co/datasets/CardinalOperations/MAMO
- IndustryOR => huggingface.co/datasets/CardinalOperations/IndustryOR
- BWOR => huggingface.co/datasets/SJTU/BWOR
- ComplexOR => github.com/xzymustbexzy/Chain-of-Experts
- Text2Zinc => huggingface.co/datasets/skadio/text2zinc
- CPEval => github.com/Yuliang795/LLMs-CP-CPEVAL

Existing focus is mostly on LP, MIP problems, or parallel efforts

What about existing modelling datasets?

We already investigated LLM-driven modelling[1] with these datasets:

- **NL4Opt[2]**: Linear Optimisation Problems
- **LGPs[3]**: Logic Grid Puzzles
- **Mixed CP**: 18 modelling problems from a Constraint Programming course.

Let's see some examples…

[1] Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "Constraint modelling with LLMs using in-context learning." 30th International conference on principles and practice of constraint programming. 2024.
[2] Ramamonjison, Rindranirina, et al. "Nl4opt competition: Formulating optimization problems based on their natural language descriptions." NeurIPS 2022 competition track. PMLR, 2023.
[3] Mitra, Arindam, and Chitta Baral. "Learning to automatically solve logic grid puzzles." Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015.

What about existing modelling datasets?

An NL4Opt example:

A park transports its visitors around either by scooter or rickshaw. A scooter can carry 2 people while a rickshaw can carry 3 people. To avoid excessive pollution, at most 40% of the vehicles used can be rickshaws. If the park needs to transport at least 300 visitors, minimize the total number of scooters used.

```python
# At most 40% of the vehicles used can be rickshaws:
m += Rickshaw <= 0.4 * (Scooter + Rickshaw)
# The park needs to transport at least 300 visitors:
m += 2 * Scooter + 3 * Rickshaw >= 300

# Minimize the total number of scooters used:
m.minimize(Scooter)
```

LGP clues examples:

Ultra Hex is Gabe Grant. Bane either started in 2007 or in 2009. The four people are Deep Shadow, the hero who started in 2007, the hero who started in 2009 and Matt Minkle.

```python
# Ultra Hex is Gabe Grant.
m += (ultra_hex == gabe_grant)

# Criminal Bane is either in 2007 or in 2009.
m += Xor([criminal_bane == _2007, criminal_bane == _2009])

# The four people are Deep Shadow, 2007, 2009 and Matt
m += AllDifferent([deep_shadow, _2007, _2009, matt_minkle])
```

Mixed CP examples:

```python
# For each pair of movies, check if the intervals overlap
for i in range(num_movies):
    for j in range(num_movies):
        # Check if the intervals overlap
        if (i != j  # Different movies
            and movies[i]['interval'][1] > movies[j]['interval'][0]
            and movies[j]['interval'][1] > movies[i]['interval'][0]
        ):
            # If they overlap, they cannot be selected together
            m += selected_movies[i] + selected_movies[j] <= 1

# Objective: Maximize the number of selected movies
m.maximize(sum(selected_movies))
```

```python
# No two adjacent nodes can both be in the independent set
for i, neighbors in enumerate(adjacency_list):
    for neighbor in neighbors:
        # Adjust for 1-based indexing in the data
        neighbor_idx = neighbor - 1
        # At least one of its endpoints is not in the independent set
        m += ~(nodes[i] & nodes[neighbor_idx])

# Objective: Maximize the number of nodes in the independent set
m.maximize(sum(nodes))
```

What about existing modelling datasets?

We already investigated LLM-driven modelling[1] with these datasets:

- **NL4Opt[2]**: Linear Optimisation Problems
- **LGPs[3]**: Logic Grid Puzzles

} Homogeneous

Small number of unique constraints

- **Mixed CP**: 18 modelling problems from a Constraint Programming course.

} Diverse

Large variation in constraint types

[1] Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "Constraint modelling with LLMs using in-context learning." 30th International conference on principles and practice of constraint programming. 2024.
[2] Ramamonjison, Rindranirina, et al. "Nl4opt competition: Formulating optimization problems based on their natural language descriptions." NeurIPS 2022 competition track. PMLR, 2023.
[3] Mitra, Arindam, and Chitta Baral. "Learning to automatically solve logic grid puzzles." Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015.

Diverse problems offer a bigger challenge[1]

Mixed CP examples:

```python
# For each pair of movies, check if the intervals overlap
for i in range(num_movies):
    for j in range(num_movies):
        # Check if the intervals overlap
        if (i != j  # Different movies
            and movies[i]['interval'][1] > movies[j]['interval'][0]
            and movies[j]['interval'][1] > movies[i]['interval'][0]
        ):
            # If they overlap, they cannot be selected together
            m += selected_movies[i] + selected_movies[j] <= 1

# Objective: Maximize the number of selected movies
m.maximize(sum(selected_movies))
```

```python
# No two adjacent nodes can both be in the independent set
for i, neighbors in enumerate(adjacency_list):
    for neighbor in neighbors:
        # Adjust for 1-based indexing in the data
        neighbor_idx = neighbor - 1
        # At least one of its endpoints is not in the independent set
        m += ~(nodes[i] & nodes[neighbor_idx])

# Objective: Maximize the number of nodes in the independent set
m.maximize(sum(nodes))
```

Towards a larger diverse CP dataset!

[1] Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "Constraint modelling with LLMs using in-context learning." 30th International conference on principles and practice of constraint programming. 2024.

A lot of CP model repositories exist, but…

E.g.: CSPLib, PyCSP3 examples, MiniZinc examples and many more…

Can't we just use them as they are to evaluate LLMs on modelling?

No!

1. **Unstructured problem descriptions & models**
2. **No standardized metric for evaluating a model**

why another ~~dataset~~ benchmark?

1. **Unstructured problem descriptions & models**
2. **No standardized metric for evaluating a model**

Our solution:

1. Collect and formalize instances (problem description + model)
2. Define an evaluation metric

Result: CP-Bench

making CP-Bench

Challenge 1: Selecting diverse problems

Challenge 1: Selecting diverse problems

- Extending the mixed CP course-based dataset…
- Additional sources:
  - **CSPLib** => csplib.org
  - **Hakan's repository** => hakank.org
  - **CPMpy repository** => github.com/CPMpy
- Important: Maintaining diversity
  - E.g. Duplicate models that only differ in their parameter values ✖
- 101 instances selected

Challenge 1: Selecting diverse problems

Table 1. Comparison of CP-Bench with existing datasets for evaluating LLM-generated CP models of combinatorial problems. Unique constraints count distinct constraint operators and their argument structures (arity and type, e.g., variable/constant/expression).

Benchmark	#	Constraints		Decision Variables		# Unique Constraints	# Opt. Problems
		Average	Min/Max	Average	Min/Max		
NL4Opt	289	2.90 ±0.81	2/5	2.02 ±0.13	2/3	27	All
LGPs	100	38.47 ±15.60	7/69	12.00 ±0.00	12/12	8	None
CP-Bench *[ours]*	101	84.76 ±274.90	1/2017	51.69 ±125.03	3/962	241	30

making CP-Bench

Challenge 2: Problem(atic) descriptions

Challenge 2: Problem(atic) descriptions

- Are underspecified, ambiguous:
  - e.g., “With 3 dollars, you get 5 apples…”
  - Ambiguity: Can you also get 7 apples, or just 5, 10, 15, 20 etc. ?
- Contain irrelevant information
  - e.g. “A variant of this problem is to…”
- Contain limited information
  - e.g. “The problem is the Isomorphic Dice. Model it.”
- Provide modelling guidelines
  - e.g., “This problem can be modelled by…”

making CP-Bench

Challenge 3: Ground-Truth Models are not perfect

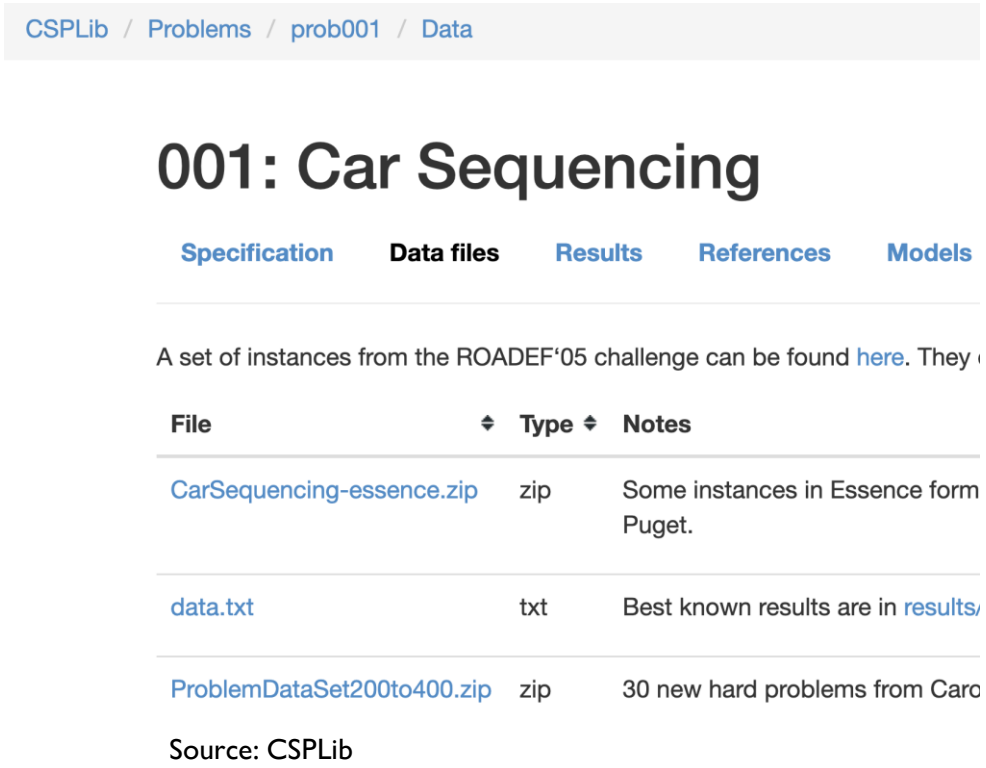
Challenge 3: Ground-Truth Models are not perfect

Our goal is that each ground-truth model:

1. Precisely represents the problem description
2. Is self-contained (i.e. instantiated and executable)

Challenge 3: Ground-Truth Models are not perfect

1. Data/Parameter Values (often missing or external)



Source: CSPLib

Challenge 3: Ground-Truth Models are not perfect

2. The model may make additional assumptions
   - e,g., “With 3 dollars you get 5 apples”

```
Model([the_sum == bananas+apples,
       # This don't work since "/" does integer division
       # 3*bananas/5 + 5*oranges/7 + 7*mangoes/9 + 9*apple
       # we multiply with 3*5*7*9=945 on both sides to wee
       3*bananas*189 + 5*oranges*135 + 7*mangoes*105 + 9*a
       sum(x) == 100
       ])
```

Source: https://www.hakank.org/cpmpy/bananas.py

```
bananas_bundles = cp.intvar(1, 100, name="bananas_bundles")
oranges_bundles = cp.intvar(1, 100, name="oranges_bundles")
mangoes_bundles = cp.intvar(1, 100, name="mangoes_bundles")
apples_bundles = cp.intvar(1, 100, name="apples_bundles")

# Total fruits and total cost
total_fruits = (5 * bananas_bundles) + (7 * oranges_bundles)
total_cost = (3 * bananas_bundles) + (5 * oranges_bundles) +
```

An LLM-generated model

Challenge 3: Ground-Truth Models are not perfect

3. Symmetry-breaking constraints that optimize the solving

```
# Symmetry breaking
# lexicographic ordering of rows
for r in range(N - 1):
    b = boolvar(N + 1)
    model += b[0] == 1
    model += b == ((matrix[r] <= matrix[r + 1]) &
                   ((matrix[r] < matrix[r + 1]) | b[1:] == 1))
    model += b[-1] == 0
```

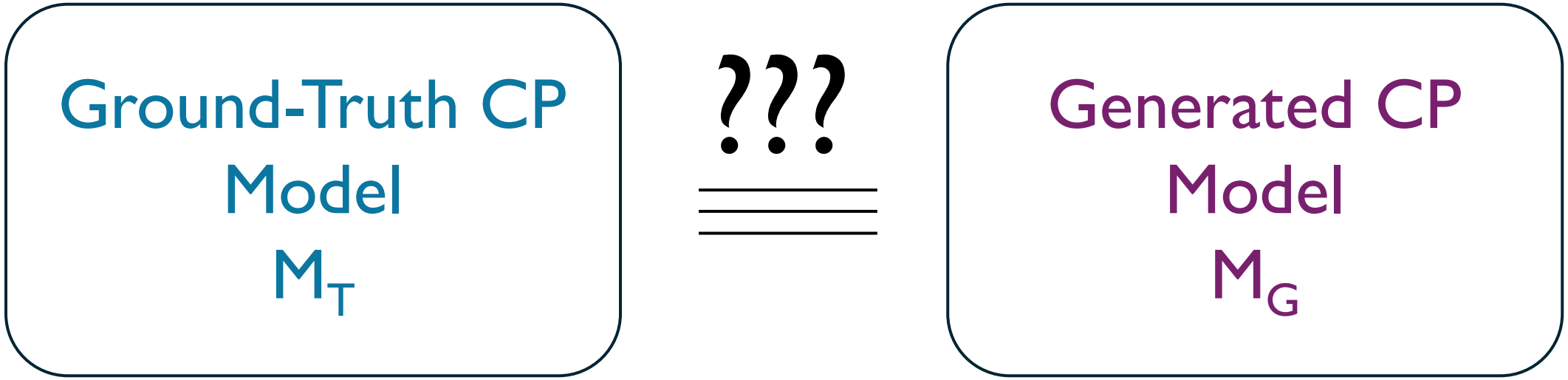
Source: https://github.com/CPMpy/cpmpy/blob/master/examples/csplib/prob050_diamond_free.py

making CP-Bench

Challenge 4: Evaluating the generated models

Challenge 4: Evaluating the generated models

How can we conclude that a model is correct?



Making CP-Bench: Evaluation (Example)

```python
# Parameters
n = 2  # Number of pack sizes
packs = [7, 13]  # Pack sizes
max_val = target * 2  # Arbitrary max limit of pack counts

# Decision variables
counts = intvar( lb: 0, max_val, shape=n, name="counts")  # C
total = intvar( lb: 0, max_val * n, name="total")  # Total nu

# Model
model = Model([
    total == (counts * packs).sum(),
    total >= target,
])

# Objective
model.minimize(total)

# Solve the model
model.solve()
```

Ground-Truth

```python
# Decision Variables
# Number of 7-packs and 13-packs
counts = cp.intvar( lb: 0, target, shape=2, name="counts")

# Constraints
# The total number of beers must be at least the target
model += counts[0]*7 + counts[1]*13 >= target
# Objective: minimize the total number of beers minus th
total_beers = counts[0]*7 + counts[1]*13
model.minimize(total_beers - target)

# Solve and print
if model.solve():
    solution = {'counts': counts.value().tolist()}
```

Generated

Challenge 4: Evaluating the generated models

Model Equivalence[1]: $(M_T \rightarrow M_G) \wedge (M_G \rightarrow M_T)$

Ideal, but it requires that Decision Variables are "connected" (channeling of M_G to M_T):

- What if M_G uses a different viewpoint?
- What if M_G is expressed in a different framework?

[1] Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "Constraint modelling with LLMs using in-context learning." 30th International conference on principles and practice of constraint programming. 2024.

Challenge 4: Evaluating the generated models

- **The Problem:** How to automatically evaluate correctness?
- **Our approach:** Solution Accuracy
  1. The LLM generates a model.
  2. The model is executed to get a solution*.
  3. Is solution valid and optimal against our ground-truth model?

$$a_G \in Sol(C_T) \qquad f(a_G) = f(a_T)$$

Challenge 4: Evaluating the generated models

Printing Description: Prompting with pre-specified key names for the solution printing!

Problem Description:

```
"""
Given a target number of beers, find the closest combination
 7-packs and 13-packs that meets or exceeds the target.

Print the counts of 7-packs and 13-packs used (counts).
"""
```

Generated Model:

```
# Decision Variables
# Number of 7-packs and 13-packs
counts = cp.intvar( lb: 0, target, shape=2, name="counts")

# Constraints
# The total number of beers must be at least the target
model += counts[0]*7 + counts[1]*13 >= target
# Objective: minimize the total number of beers minus the
total_beers = counts[0]*7 + counts[1]*13
model.minimize(total_beers - target)

# Solve and print
if model.solve():
    solution = {'counts': counts.value().tolist()}
    print(json.dumps(solution, indent=4))
```

Making CP-Bench: Evaluation (Example)

```python
# Parameters
n = 2  # Number of pack sizes
packs = [7, 13]  # Pack sizes
max_val = target * 2  # Arbitrary max limit of pack counts

# Decision variables
counts = intvar( lb: 0, max_val, shape=n, name="counts")  # C
total = intvar( lb: 0, max_val * n, name="total")  # Total nu

# Model
model = Model([
    total == (counts * packs).sum(),
    total >= target,
])

# Objective
model.minimize(total)

# Solve the model
model.solve()
```

Ground-Truth

Adding the solution as a constraint

```python
# Decision Variables
# Number of 7-packs and 13-packs
counts = cp.intvar( lb: 0, target, shape=2, name="counts")

# Constraints
# The total number of beers must be at least the target
model += counts[0]*7 + counts[1]*13 >= target
# Objective: minimize the total number of beers minus th
total_beers = counts[0]*7 + counts[1]*13
model.minimize(total_beers - target)

# Solve and print
if model.solve():
    solution = {'counts': counts.value().tolist()}
    print(json.dumps(solution, indent=4))
```

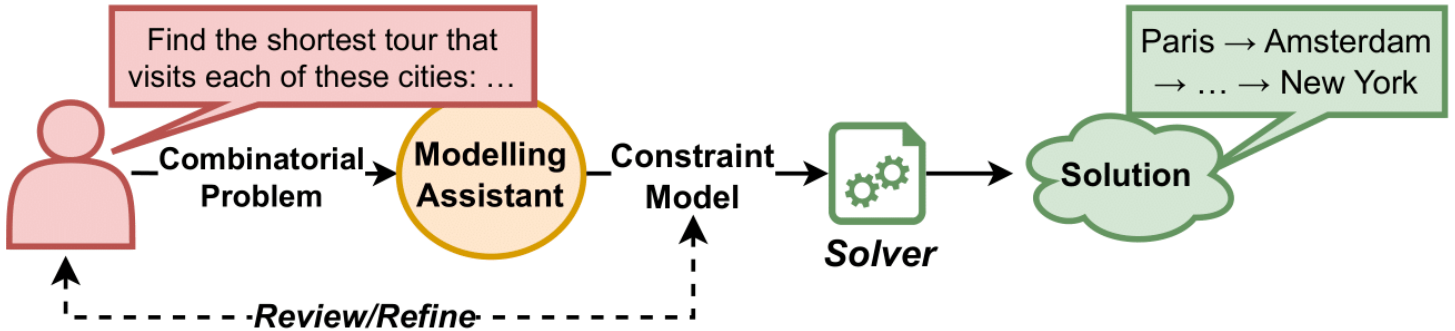
Generated

Challenge 4: Evaluating the generated models

But still, the printing description can be also underspecified:

1. Indexing misalignment
   - 0-based vs. 1-based
2. Matrices orientation
   - Rows vs. Columns
3. Value types
   - 2D boolean array vs. 1D integer array

LLMs as modelling assistants



- **We need representative benchmarks!**

Dataset of problems/models

Evaluation Challenge: Multiple variable/modelling choices (viewpoints), multiple (optimal) solutions

Evaluation Challenge: No unit tests… How to (semantically) verify?

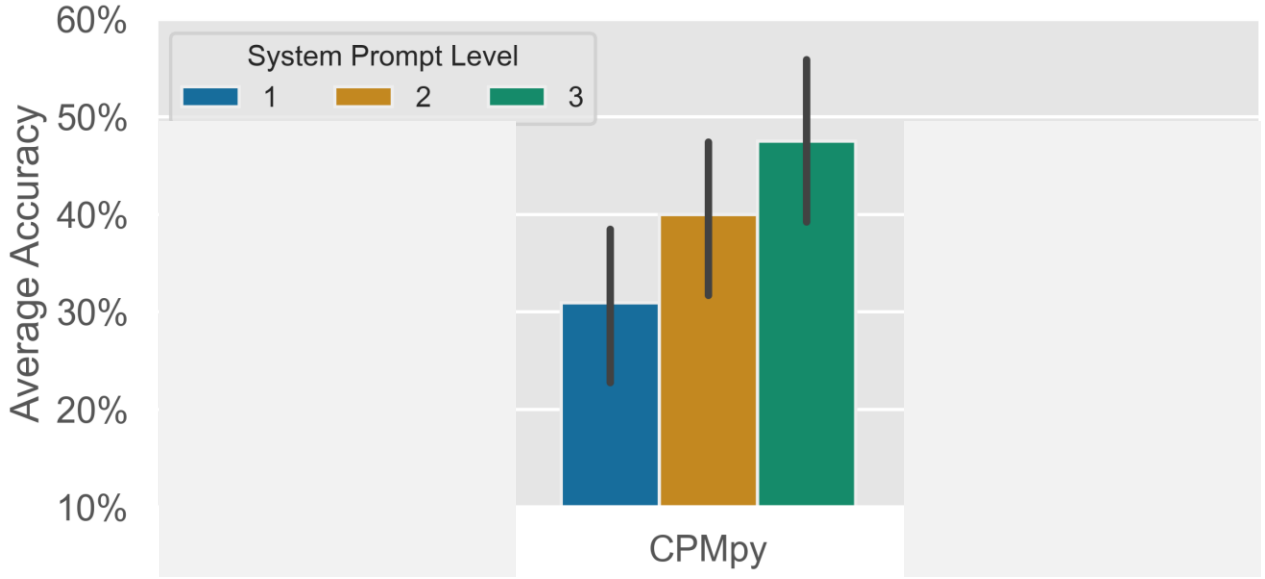
Based on paper (accepted at ECAI 2025):
CP-Bench: Evaluating Large Language Models for Constraint Modelling

results

Can LLMs actually model?

Results

What should the prompt contain?



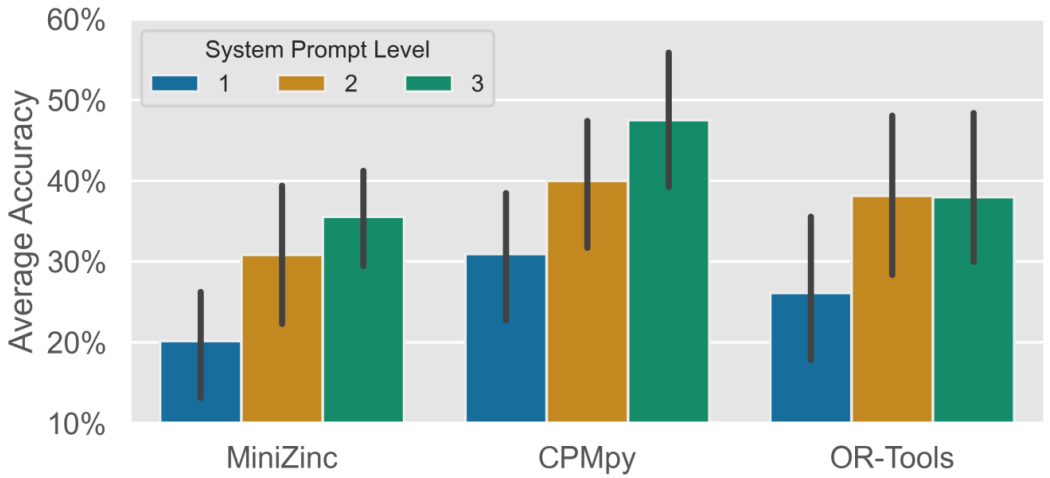
- Basic Prompt (1)
- Modelling Guidelines Prompt (2)
- Documentation Prompt (3)

Results

What type of modelling framework can be used?

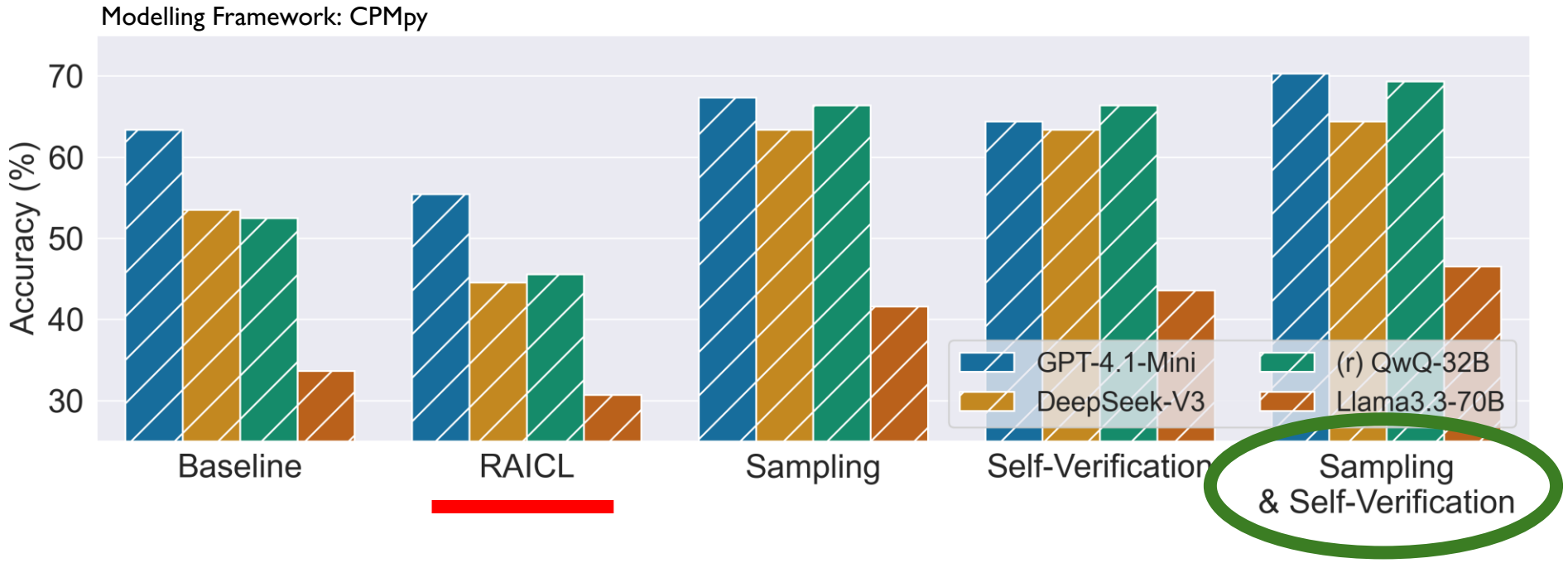
Framework	Abstraction Level	Interface Type
MiniZinc	High	Domain-Specific
CPMpy	High	Python
OR-Tools	Medium	Python

- Python-based vs. Domain-Specific
- High-level vs. Low-level



Results

1. Can In-Context Learning improve accuracy?
2. Can simpler Test-Time Scaling methods?



Is 70% the ceiling?

Verifying the dataset in-depth:

- Overconstrained Ground-Truth models (symmetry breaking)
- Solution printing underspecifications (indexing, value types, etc.)
- Modelling errors

CP-Bench *Verified*: a subset of 63 verified instances

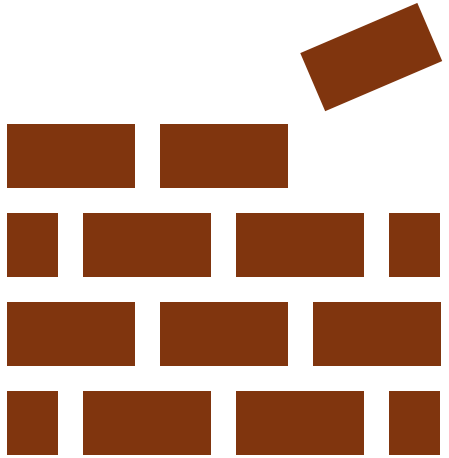
CP-Bench *Verified*: Online Leaderboard

Name	Modelling Framework	Base LLM	Models Submitted (%)	Accuracy (%)	Runtime Errors (%)	Report
sampling_10_and_self_debug_1(	CPMpy	gpt-5-mini-2025-08-(	100	95.24	0	
self_debug_gpt_5	CPMpy	gpt-5-2025-08-07	100	93.65	0	
sampling_10_and_self_debug_1(	CPMpy	gpt-4.1-mini	100	92.06	0	
self_debug_gpt_5_mini	CPMpy	gpt-5-mini-2025-08-(	100	92.06	0	
multi_agents_requirements_gat	OR-Tools	o3	100	88.89	0	
anonymous_submission	CPMpy	DeepSeek-V3	100	87.3	6.35	
self_debug_10_qwq	CPMpy	Qwen/QwQ-32B	100	85.71	6.35	
documentation_prompt_1	CPMpy	gpt-4.1-mini	100	85.71	3.17	
sampling_10_qwq	CPMpy	Qwen/QwQ-32B	100	85.71	6.35	
guidelines_prompt_ort	OR-Tools	o4-mini-2025-04-16	100	84.13	1.59	
documentation_prompt_gpt_5_m:	CPMpy	gpt-5-mini-2025-08-(	100	80.95	6.35	
guidelines_prompt_mnz	MiniZinc	o4-mini-2025-04-16	100	65.08	20.63	

https://huggingface.co/spaces/kostis-init/CP-Bench-Leaderboard

Extending CP-Bench (in progress)

- Extending with more instances:
  - Fixing all the aforementioned issues
  - Diversity should be preserved
  - Qualitative Verification
  - Target: 150 verified & diverse instances
- Additional correctness metrics?



Discussion

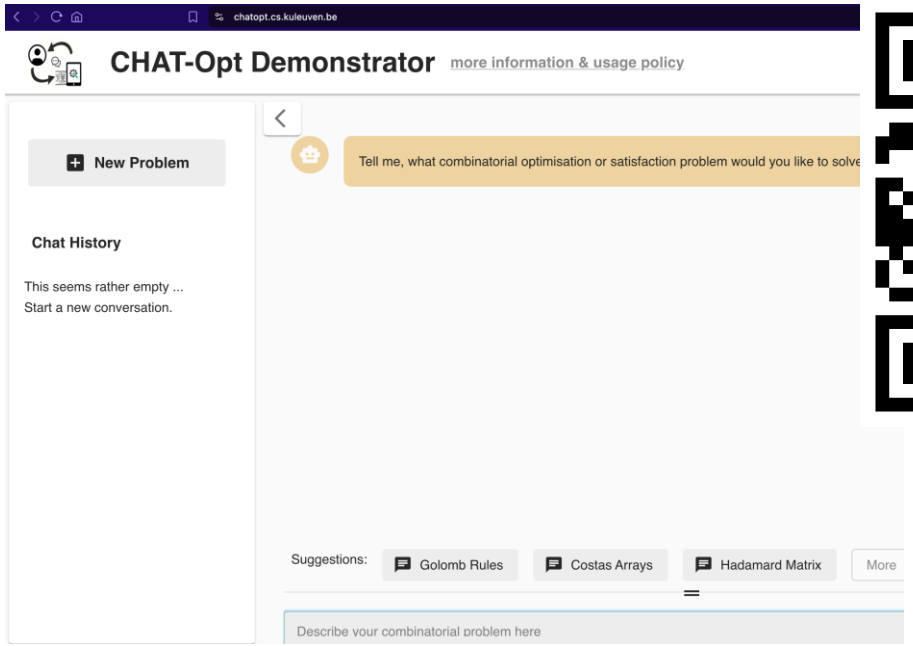
- Evaluation: Is solution accuracy enough?
- Modeling with large or external data sources?
- Correct model vs *good* model (e.g. efficiently solvable, interpretable)?

- From model+solve to interactive model refinement?
- Underspecified models & guiding the user in the problem formulation

Thank You!

Leaderboard

Paper



ChatOpt Online Demo

https://chatopt.cs.kuleuven.be

Panel: “Are LLMs the missing piece to get us to the Holy Grail?”