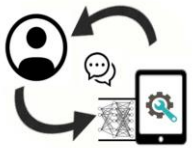
Benchmarking and Evaluating LLMs for Constraint Modelling

Lessons learned (so far)

Tsouros Dimos

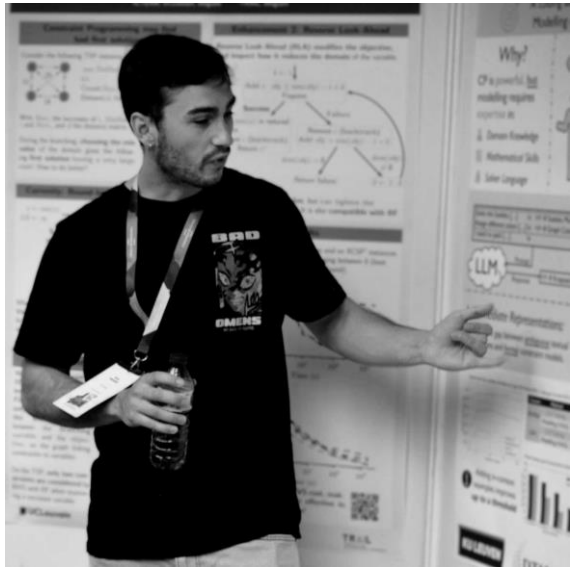
CPAIOR 2026 Masterclass, May 2026

H.F.R.I.
Hellenic Foundation for Research & Innovation



Most of the work presented here is joint work with:

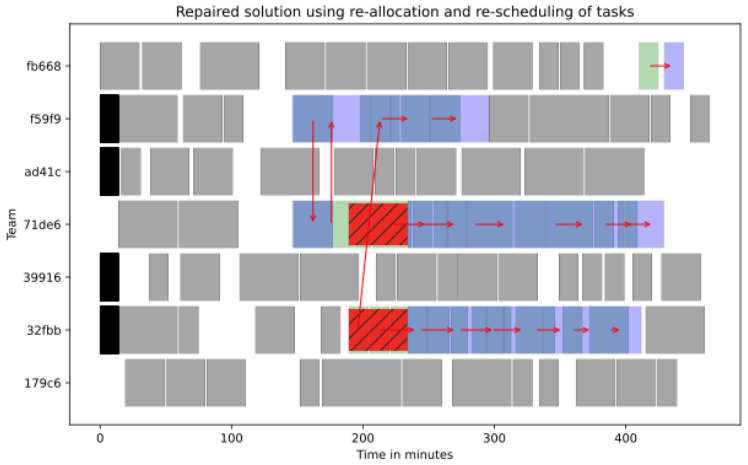
Kostis Michailidis

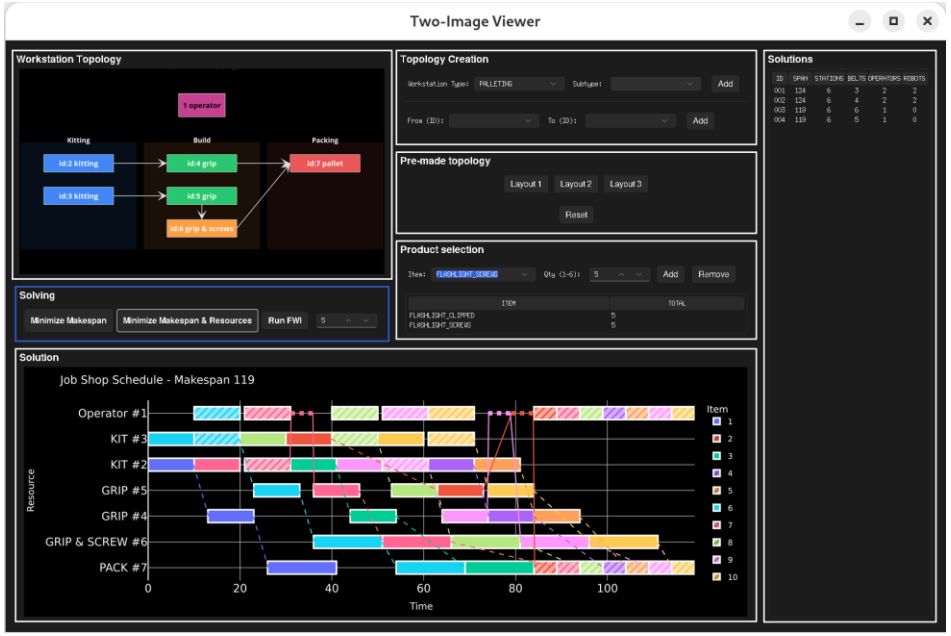


Tias Guns

Constraint Optimization Problems

	Week 1							Week 2							Total shifts
	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	
name															
Megan	F	D	D	D	D	F	F	D	D	F	F	D	D	D	9
Katherine	D	D	D	D	D	F	F	D	D	F	F	D	D	F	9
Robert	D	D	D	F	F	D	D	F	F	D	D	D	F	F	8
Jonathan	D	D	F	F	F	D	D	D	D	D	F	F	F	F	7
William	F	D	D	D	D	F	F	D	D	F	F	D	D	D	9
Richard	D	D	D	D	D	F	F	D	D	F	F	F	D	D	9
Kristen	F	F	D	D	D	F	F	D	D	D	F	F	D	D	8
Kevin	D	D	F	F	F	F	F	F	D	D	D	D	D	F	7
Cover D	5/5	7/7	6/6	5/4	5/5	2/5	2/5	6/6	7/7	4/4	2/2	5/5	6/6	4/4	14





Solved and visualized using the CPMpy library in python

Constraint Modeling

Model + Solve paradigm

- Decision Variables
- Domains
- Constraints
- Objective (optional)

Model → **Solver**

Constraint Modelling: An Example

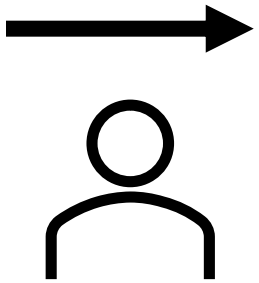
I am an actor with many offers.

Requirement 1: Only one movie per day.

Requirement 2: Maximize income.

Which movies to choose?

	Movie Title	Start Day	End Day
0	Tarjan of the Jungle	4	13
1	The Four Volume Problem	17	27
2	The President's Algorist	1	10
3	Steiner's Tree	12	18
4	Process Terminated	23	30
5	Halting State	9	16
6	Programming Challenges	19	25
7	Discrete Mathematics	2	7
8	Calculated Bets	26	31



Expert

Decision Variables:

$$movies_0, movies_1, \dots, movies_8$$

Domain:

$$movies_i \in \{0, 1\}$$

Constraints:

- $movies_0 + movies_2 \leq 1$
- $movies_0 + movies_3 \leq 1$
- $movies_0 + movies_5 \leq 1$
- [...]
- $movies_4 + movies_8 \leq 1$

Objective:

$$\max(\sum_{i=0}^{8} movies_i)$$

Source: http://www.hakank.org/minizinc/movie_scheduling.mzn
Source description: Steven S. Skiena's The Algorithm Design Manual

Constraint Modelling: An Example

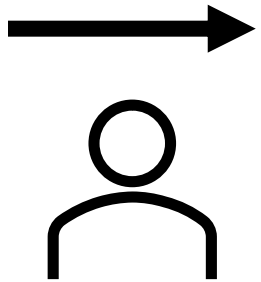
I am an actor with many offers.

Requirement 1: Only one movie per day.

Requirement 2: Maximize income.

Which movies to choose?

	Movie Title	Start Day	End Day
0	Tarjan of the Jungle	4	13
1	The Four Volume Problem	17	27
2	The President's Algorist	1	10
3	Steiner's Tree	12	18
4	Process Terminated	23	30
5	Halting State	9	16
6	Programming Challenges	19	25
7	Discrete Mathematics	2	7
8	Calculated Bets	26	31

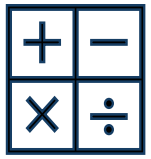



```python
from cpmpy import *

# Data (title, start, end)
movies = [["Tarjan of the Jungle", 4, 13],
    ["The Four Volume Problem", 17, 27],
    ["The President's Algorist", 1, 10],
    ["Steiner's Tree", 12, 18],
    ["Process Terminated", 23, 30],
    ["Halting State", 9, 16],
    ["Programming Challenges", 19, 25],
    ["Discrete Mathematics", 2, 7],
    ["Calculated Bets", 26, 31]]
num_movies = len(movies)

# Decision Variables
selected_movies = boolvar(shape=num_movies)  # 1 if the movie is selected, 0 otherwise

# CP Model
model = Model()

# Non-overlapping movie schedules
for i in range(num_movies):
    for j in range(num_movies):
        # Check if the intervals overlap for each pair of movies
        if i != j and movies[i][2] > movies[j][1] and movies[j][2] > movies[i][1]:
            # Constraint: the movies cannot be selected together
            model += ~(selected_movies[i] & selected_movies[j])

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

# Solve
model.solve()
print(selected_movies.value())
# Output:
# [False False False False False  True  True  True  True]
```

Source: http://www.hakank.org/minizinc/movie_scheduling.mzn
Source description: Steven S. Skiena's The Algorithm Design Manual

Constraint Solving technologies are powerful, but modeling requires expertise in:

Domain Knowledge



Mathematical Skills

Solver Formalisms

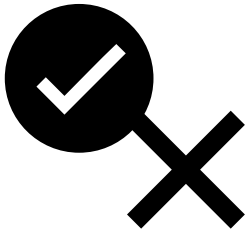
Making Constraint Solving More Accessible

Modelling Tools

CPMpy

Constraint Acquisition

Users provide or classify (non-)solutions



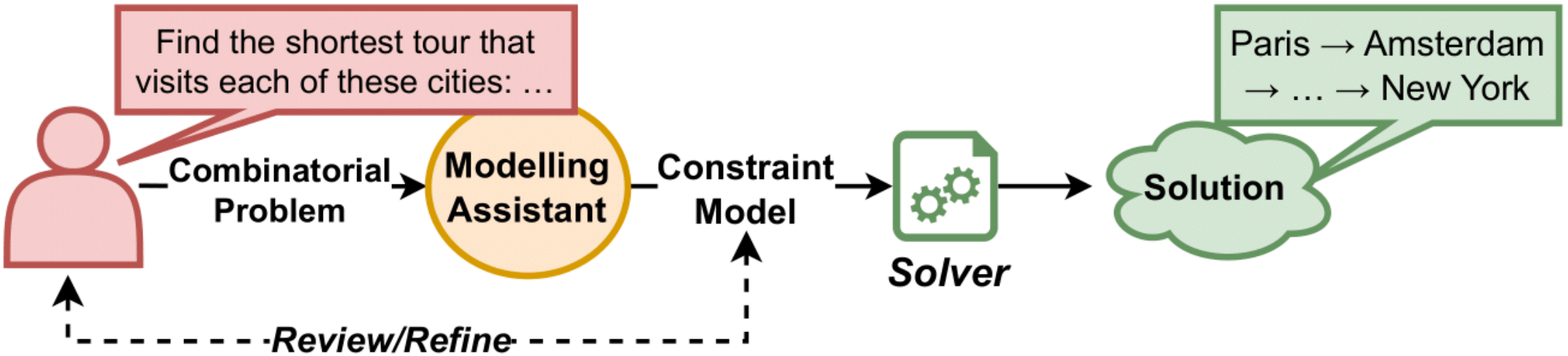
From Natural Language

Textual descriptions of combinatorial problems

LLMs assistants

- Rise of LLMs
- Chatbots and other apps everywhere!
- Coding assistants (e.g. Cursor) excel in general coding tasks

What about modeling assistants?



LLMs assistants

- Rise of LLMs
- Chatbots and other apps everywhere!
- Coding assistants (e.g. Cursor) excel in general coding tasks

What about modeling assistants?

Can Large Language Models model Constraint Problems?

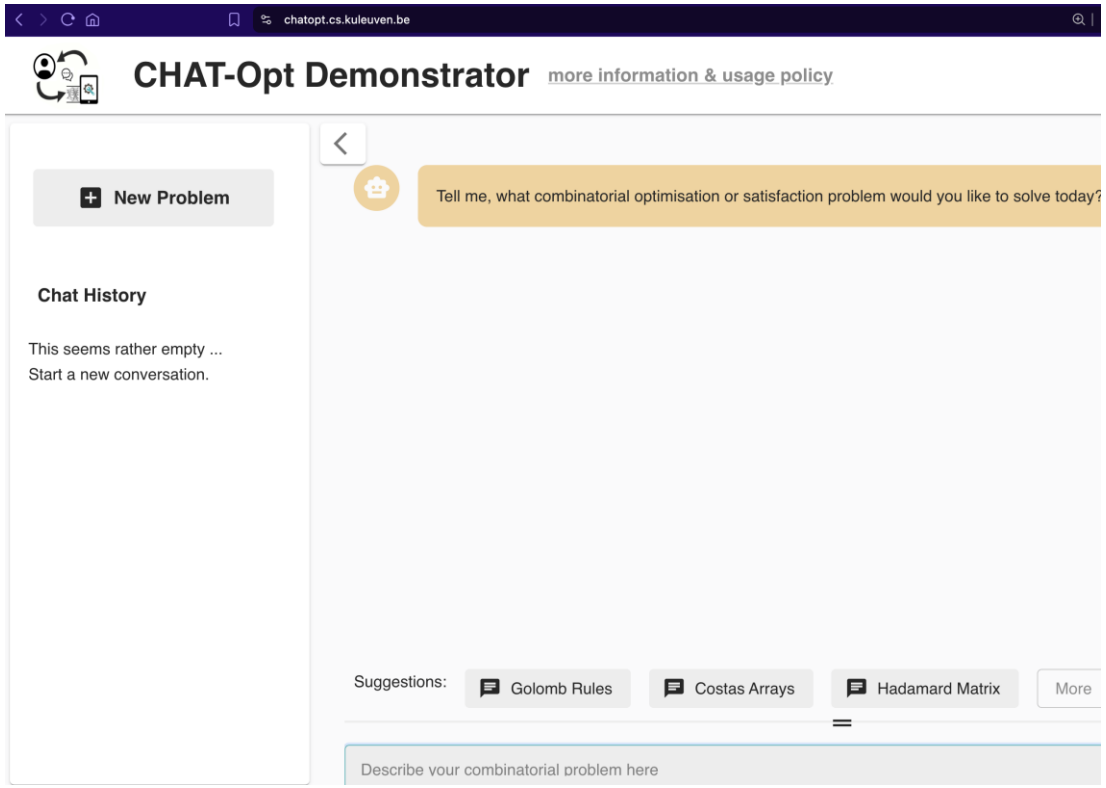
LLMs assistants

- Rise of LLMs
- Chatbots and other apps everywhere!
- Coding assistants (e.g. Cursor) excel in general coding tasks

What about modeling assistants?

Can Large Language Models model Constraint Problems?

Can Large Language Models model Constraint Problems?



ChatOpt Online Demo

https://chatopt.cs.kuleuven.be

European Research Council
Established by the European Commission

LLMs assistants

- Rise of LLMs
- Chatbots and other apps everywhere!
- Coding assistants (e.g. Cursor) excel in general coding tasks

What about modeling assistants?

How well ~~Can~~ Large Language Models model Constraint Problems?

LLMs for modeling optimization problems

- *NL4Opt*[1] NeurIPS competition
- Decomposition-based MILP synthesis[2]
- OptiMUS[3], ORLM[4] systems
- LLM-as-a-Judge[5]
- Text2Zinc[6]
- LLMZoo[7]
- OptiChat[8]

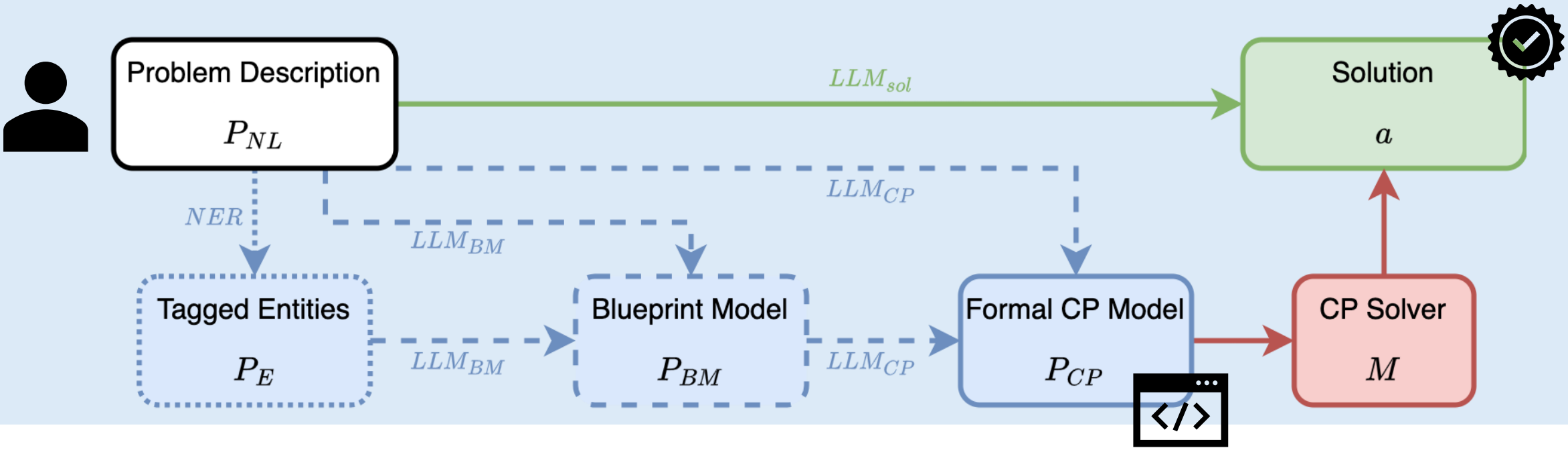
And many many many more .....

1. Ramamonjison et al. "Nl4opt competition: Formulating optimization problems based on their natural language descriptions" *(2022)*
2. Li et al. "Synthesizing mixed-integer linear programming models from natural language descriptions" (2023)
3. AhmadiTeshnizi et al. "Optimus: Scalable optimization modeling with (mi) lp solvers and large language models" (2024)
4. Huang et al. "ORLM: A Customizable Framework in Training Large Models for Automated Optimization Modeling" (2025)
5. Frechette et al. "LLM-as-a-Judge For Combinatorial Optimization Modelisations from Natural Language" (2025)
6. Singirikonda et al. "Text2zinc: A cross-domain dataset for modeling optimization and satisfaction problems in minizinc." (2025)
7. Crespin et al. "CP-Model-Zoo: A Natural Language Query System for Constraint Programming Models" (2025)
8. Chen et al. "OptiChat: Bridging Optimization Models and Practitioners with Large Language Models" (2025)

Part A

Our first attempts – GPT 3.5 Era

Decomposing the problem to different subtasks



[1] Michailidis, Tsouros, and Guns. "Constraint Modelling with LLMs Using In-Context Learning" CP2024

In-Context Learning (ICL)

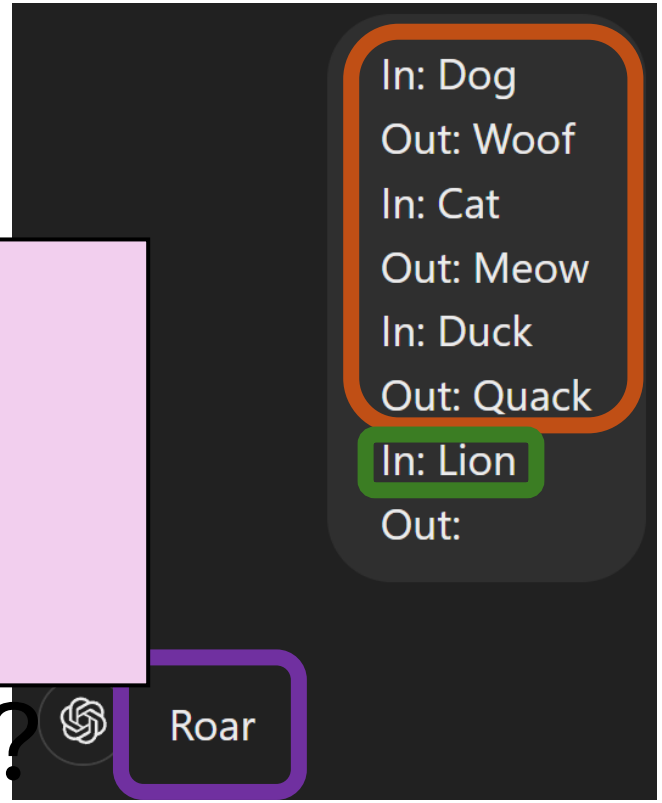
$$LLM(Prompt) = Response$$

In-Context Learning (ICL):
GPT 3.5 era!

- Task Learning:

$$LLM(C \oplus Q) \approx$$

1. What examples?
2. How many?
3. In what order?

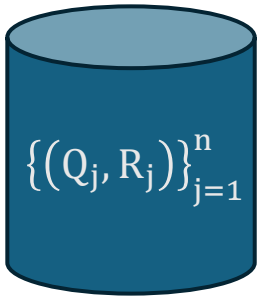


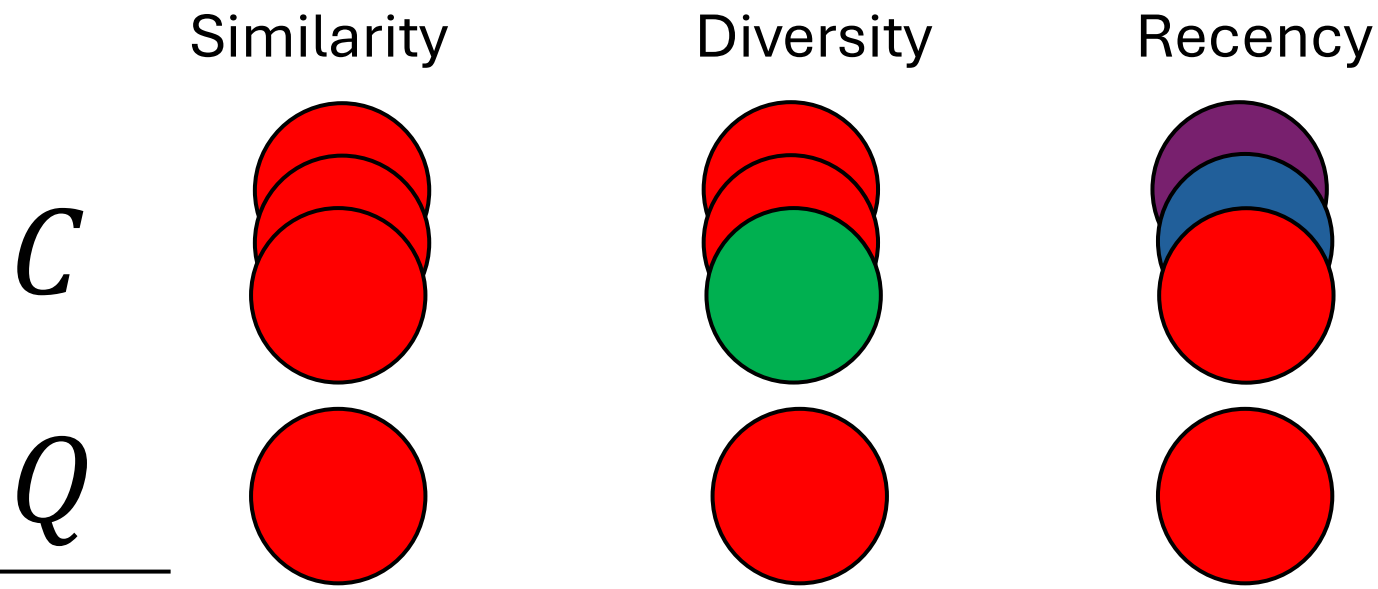
[1] Michailidis, Tsouros, and Guns. "Constraint Modelling with LLMs Using In-Context Learning" CP2024

How to choose C?

Baseline: Static/Random

Retrieval-Augmented ICL





[1] Michailidis, Tsouros, and Guns. "Constraint Modelling with LLMs Using In-Context Learning" CP2024

Important question!

How to evaluate our methods?

- Which *datasets*?
- Which evaluation *metrics*?

Ok, we got it.

Experiments: Datasets

- **NL4Opt**[1]: Linear Optimisation Problems
- **LGPs**[2]: Logic Grid Puzzles

} Homogeneous

- **Mixed**: 18 diverse problems drawn from a CP Modelling course.

} New

[1] Ramamonjison et al. Nl4opt competition: "Formulating optimization problems based on their natural language descriptions". (2023)
[2] Arindam Mitra and Chitta Baral. "Learning to automatically solve logic grid puzzles". (2015)

Evaluation Metrics

Declaration Accuracy

Model Accuracy

Solution Accuracy

```python
# Decision Variables
selected_movies = boolvar(shape=num_movies)  # 1 if the movie is selected, 0 otherwise

# CP Model
model = Model()

# Non-overlapping movie schedules
for i in range(num_movies):
    for j in range(num_movies):
        # Check if the intervals overlap for each pair of movies
        if i != j and movies[i][2] > movies[j][1] and movies[j][2] > movies[i][1]:
            # Constraint: the movies cannot be selected together
            model += ~(selected_movies[i] & selected_movies[j])

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

# Solve
model.solve()
print(selected_movies.value())
# Output:
# [False False False False False  True  True  True  True]
```

#Errors: CP Model could not be run

Intermediate Representations

Configuration: gpt-3.5-turbo-0125, 4-shot static ICL.

Dataset	Method	#Err	Sol. (%)	Decl. (%)	Model (%)
NL4Opt	CP	7	81.31	87.81	79.24
	+ BM	8	84.43	89.93	82.01
	+ NER	8	85.47	88.60	80.62
LGPs	CP	11	57.00	80.45	55.00
	+ BM	18	58.00	70.69	58.00
	+ NER	20	54.00	67.77	50.00

Linear Opt. Problems

For more complex problems:

BM: not really helping

NER: Variable Misclassification

Dataset	#Shots	CPMpy	+ BM
Mixed	2	33.33%	16.67%
	4	50.00%	50.00%
	8	50.00%	44.44%
	12	55.56%	50.00%

Configuration: gpt-3.5-turbo-0125, static ICL, Solution Accuracy.

Retrieval-Augmented ICL? Seems yes.

- ✓Relevance
- ✓Diversity
- ✓Recency

Retrieval	LGPs Sol. (%)	LGPs Decl. (%)	NL4Opt Sol. (%)	NL4Opt Decl. (%)
Static	57.00	80.45	81.31	87.81
Random	66.00	76.75	77.16	85.60
Similarity	68.00	85.34	85.12	87.72
R-Similarity	72.00	89.83	83.39	88.16
Diversity	66.00	83.09	84.08	87.99
R-Diversity	74.00	87.98	83.74	87.10
Random w/ Last-Similar	76.00	89.96	83.74	88.34

CP modelling. Config: gpt-3.5-turbo-0125, R-MMR $\lambda = 0.5$.

Evaluation challenge (Lesson learned #1)

- In general, evaluation needs **some form of mapping** between the decision variables.

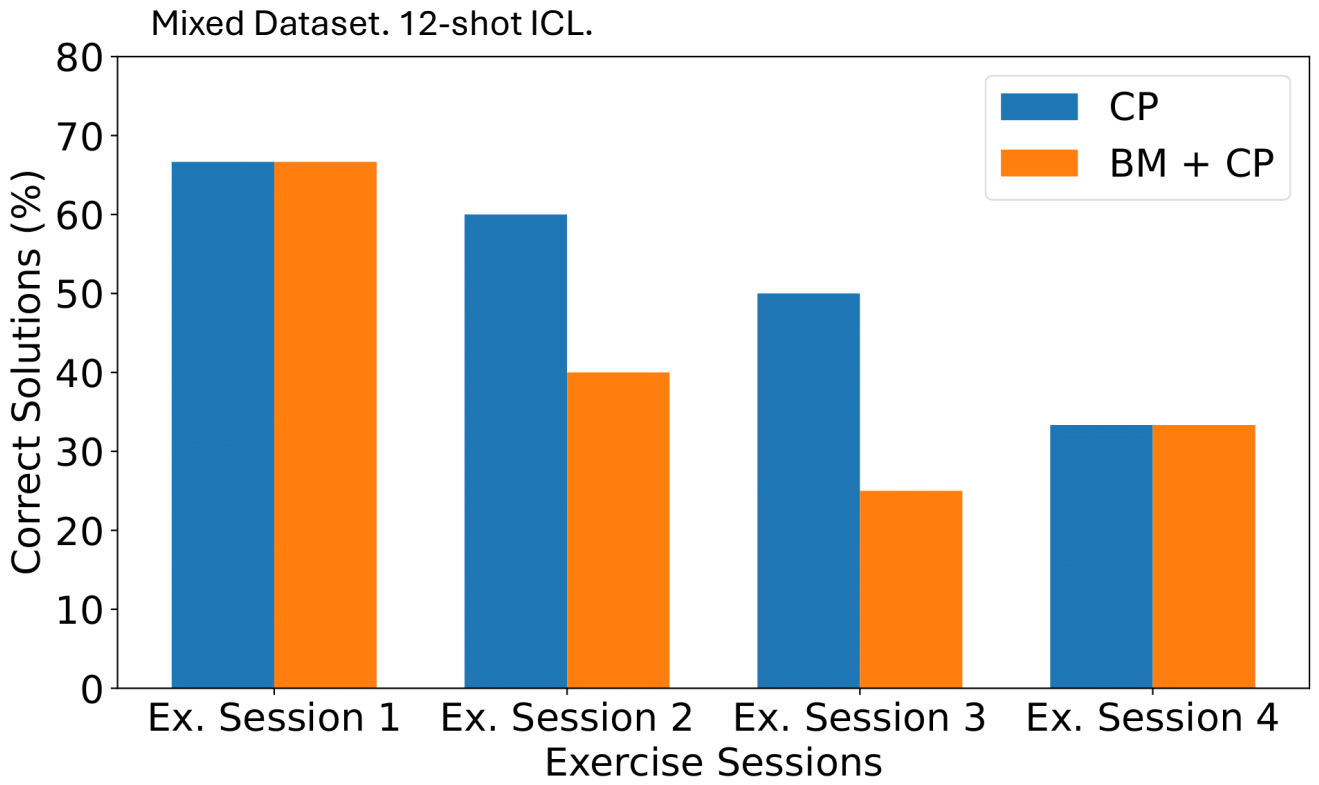
Choice of Decision Variables is free => How to check Model Accuracy and Decleration Accuracy ?

- Automatic process for mapping that worked for the *(homogenous)* NL4OPT and LGP datasets.
- But not for more complex and diverse problems

Focus on **Solution Accuracy:** Prompt LLMs to generate code that prints the solution in specific format.

Solution accuracy on diverse problems

- Few (similar) examples to retrieve from.
- More complex problems.
- Lower accuracy
- <40% in only fourth exercise!



Lesson learned #2: Need more complex and diverse problems to evaluate

Part B

Benchmarking LLM-driven Constraint Modelling

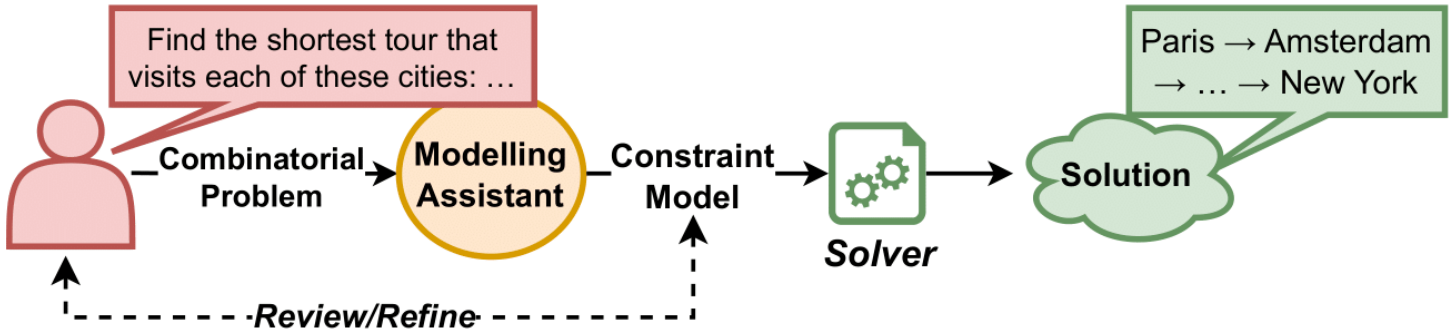
Existing LLM-based modelling datasets

- 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
- OptiBench => github.com/yangzhch6/ReSocratic
- CO-Bench => huggingface.co/datasets/CO-Bench/CO-Bench

Existing focus is mostly on LP, MI(L)P problems, many include homogenous problems, and are parallel efforts

Do we need another dataset?

Benchmarking LLM-based constraint modeling



- We need **representative benchmarks!**
- **Complex and Diverse**

Dataset of problems/models

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

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 KUL 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

```
# 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.

```
# 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:

```
# 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))
```

```
# 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 and more complex combinatorial 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 combinatorial problems dataset!

[1] Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "Constraint modelling with LLMs using in-context learning." (2024).

A lot of constraint 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**

Do we need another dataset?

We do need 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

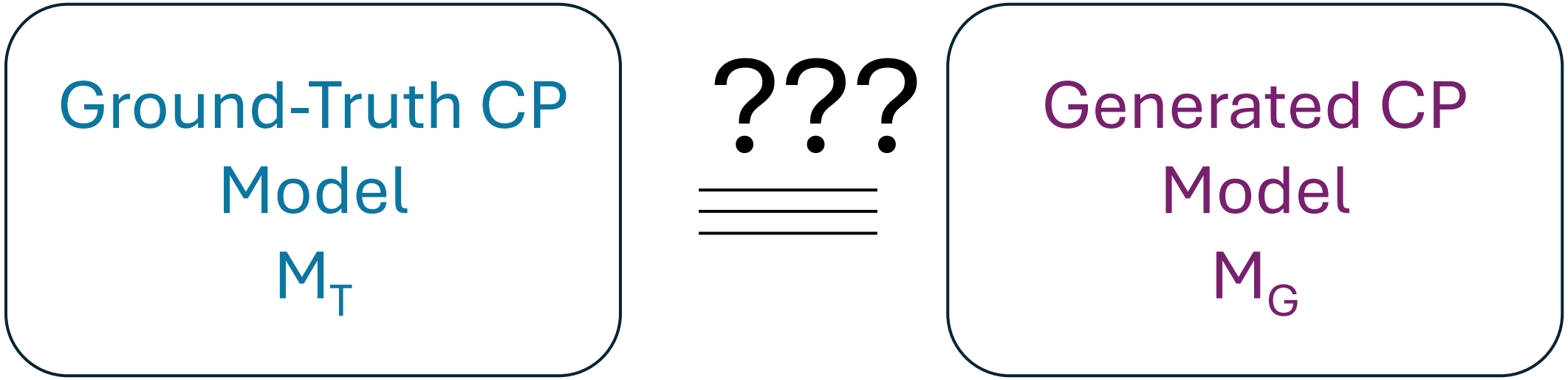
Result: CP-Bench

Making CP-Bench: Challenges

1. Evaluating the generated models! - from lesson learned #1
2. Selecting diverse problems (e.g. duplicate models that differ only in param values) - from lesson learned #2

Challenge 1: Evaluating the generated models

How can we conclude that a model is correct?



Challenge 1: Evaluating the generated models

```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 1: Evaluating the generated models

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

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**?
- Even our small CP Course-based dataset showed it is not straightforward!

[1] Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "Constraint modelling with LLMs using in-context learning." (2024).

Challenge 1: Evaluating the generated models

- **The Problem:** How to automatically evaluate correctness?
- **Our approach:** (Multi-instance/Multiple) 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 1: 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 of
 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))
```

Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "CP-Bench: Evaluating Large Language Models for Constraint Modelling“, ECAI, (2025)

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

Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "CP-Bench: Evaluating Large Language Models for Constraint Modelling“, ECAI, (2025)

Challenge 1: Evaluating the generated models

Importantly! CP-Bench only needs an (optimal) solution. Generated model can be in any framework!

Gurobipy

Google OR-Tools

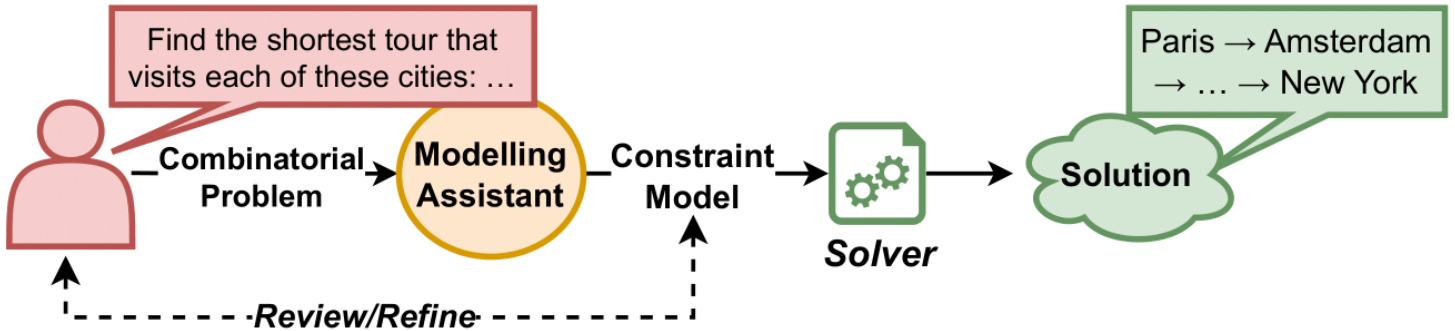
Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "CP-Bench: Evaluating Large Language Models for Constraint Modelling“, ECAI, (2025)

Challenge 2: 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 value
- 101 instances initially selected for CP-Bench

Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "CP-Bench: Evaluating Large Language Models for Constraint Modelling", ECAI, (2025)

LLMs as modelling assistants



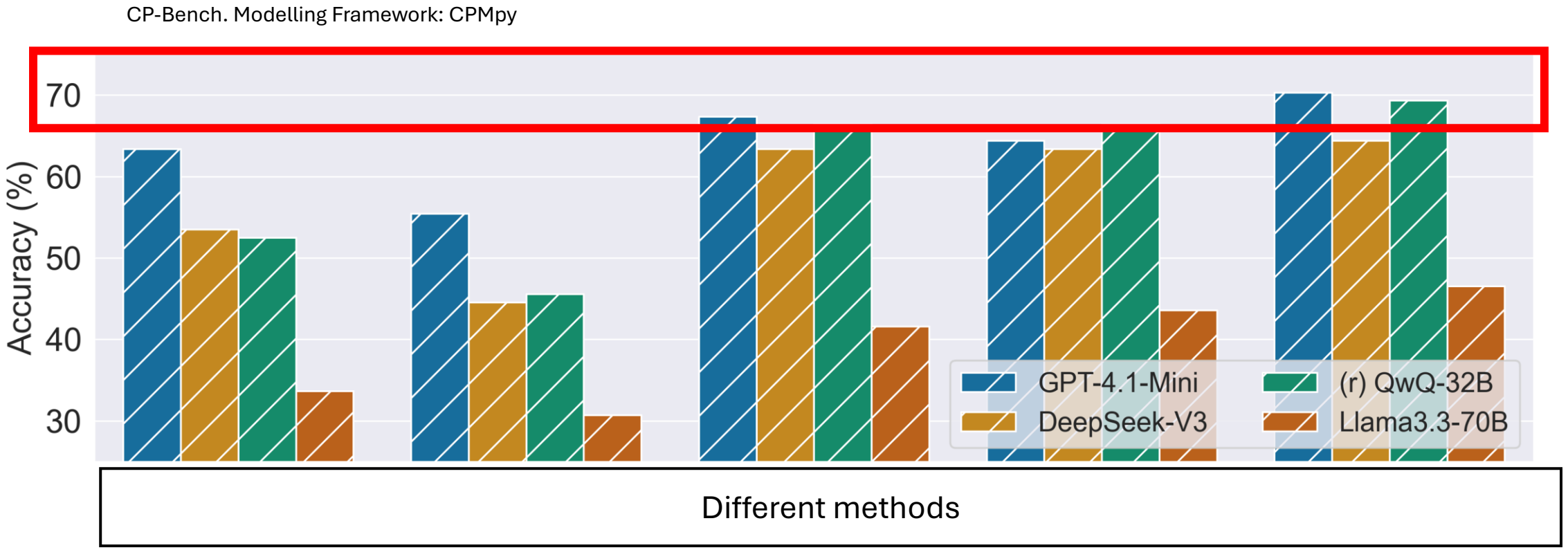
- **We need representative benchmarks!**

Dataset of problems/models

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

Some results on CP-Bench

Results on CP-Bench



Is 70% the ceiling? **No! More challenges than we thought**

Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "CP-Bench: Evaluating Large Language Models for Constraint Modelling“, ECAI, (2025)

Making CP-Bench: Challenges

1. Evaluating the generated models! - from lesson learned #1
2. Selecting diverse problems (e.g. duplicate models that differ only in param values) - from lesson learned #2

Making CP-Bench: Challenges

1. Evaluating the generated models! - from lesson learned #1
2. Selecting diverse problems (e.g. duplicate models that differ only in param values) - from lesson learned #2
3. **Fixing “bad” problem descriptions (e.g. underspecified, ambiguous etc.) -** Lesson learned #3

Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "CP-Bench: Benchmarking LLMs for CP model generation", LLM-Solve Workshop, (2025)

Challenge: Problem(atic) descriptions

Lesson learned #3

- 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 modeled by…”

Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "CP-Bench: Benchmarking LLMs for CP model generation", LLM-Solve Workshop, (2025)

Challenge: Problem(atic) descriptions

But still, the printing description (for solution accuracy evaluation) can be 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

Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "CP-Bench: Benchmarking LLMs for CP model generation", LLM-Solve Workshop, (2025)

Making CP-Bench: Challenges

1. Evaluating the generated models! - from lesson learned #1
2. Selecting diverse problems (e.g. duplicate models that differ only in param values) - from lesson learned #2
3. Fixing “bad” problem descriptions (e.g. underspecified, ambiguous etc.) – from lesson learned #3
4. **Correct Ground-Truth models** (precisely matching the NL description) – Lessons learned #4-5

Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "CP-Bench: Benchmarking LLMs for CP model generation", LLM-Solve Workshop, (2025)

Challenge: 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)

Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "CP-Bench: Benchmarking LLMs for CP model generation", LLM-Solve Workshop, (2025)

Challenge: Ground-Truth Models are not perfect

1. The model may make additional assumptions
   - e,g., “With 3 dollars you get 5 bananas”
   - Can you buy fruit individually or only in packs??

```
Model([the_sum == bananas+apples,
       # This don't work since "/" does integer division
       # 3*bananas/5 + 5*oranges/7 + 7*mangoes/9 + 9*apples/3 == 10
       # we multiply with 3*5*7*9=945 on both sides to weed out the
       3*bananas*189 + 5*oranges*135 + 7*mangoes*105 + 9*apples*315
       sum(x) == 100
       ])
```

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

```
bananas_bundles = cp.intvar(1, 100, name="bananas_bund
oranges_bundles = cp.intvar(1, 100, name="oranges_bund
mangoes_bundles = cp.intvar(1, 100, name="mangoes_bund
apples_bundles = cp.intvar(1, 100, name="apples_bundle

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

An LLM-generated model

Lesson learned #4

Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "CP-Bench: Benchmarking LLMs for CP model generation", LLM-Solve Workshop, (2025)

Challenge: Ground-Truth Models are not perfect

2. Include constraints not defined, e.g. symmetry-breaking constraints for faster solving (*but excluding feasible solutions…)*

```
# 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

Lesson learned #5

Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "CP-Bench: Benchmarking LLMs for CP model generation", LLM-Solve Workshop, (2025)

CP-Bench verified

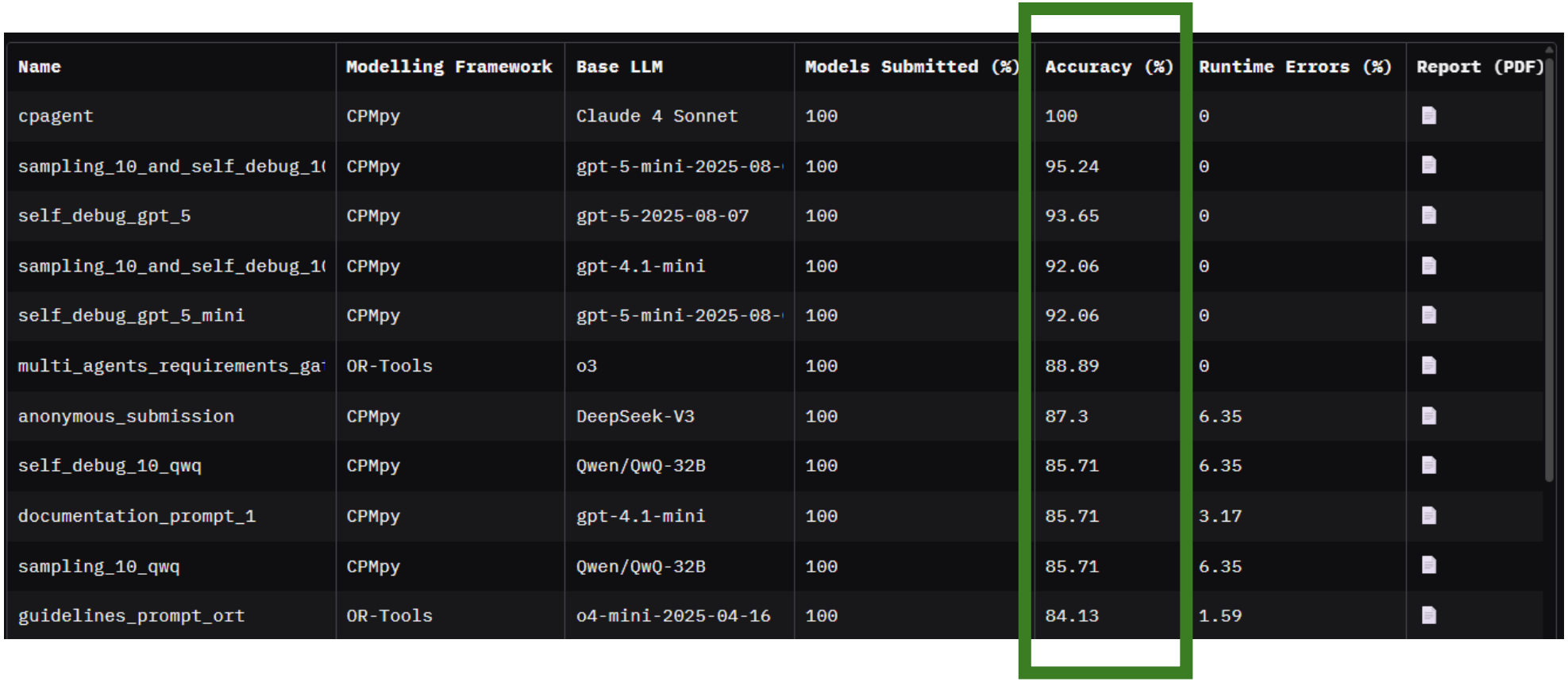
Verifying the dataset in-depth:

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

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

Michailidis, Kostis, Dimos Tsouros, and Tias Guns. "CP-Bench: Benchmarking LLMs for CP model generation", LLM-Solve Workshop, (2025)

CP-Bench *Verified*: Online Leaderboard



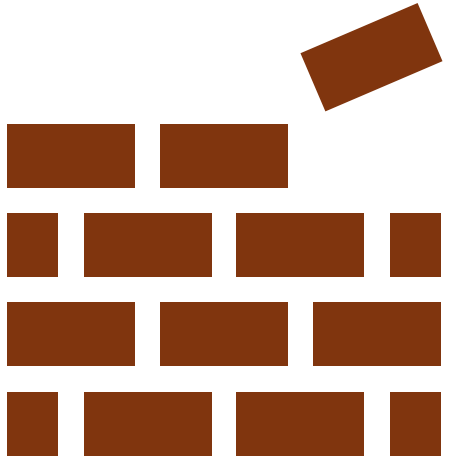
Name	Modelling Framework	Base LLM	Models Submitted (%)	Accuracy (%)	Runtime Errors (%)	Report (PDF)
cpagent	CPMpy	Claude 4 Sonnet	100	100	0	
sampling_10_and_self_debug_1(	CPMpy	gpt-5-mini-2025-08-(	100	95.24	0	
self_debug_gpt_5	CPMpy	gpt-5-2025-08-07	100	93.65	0	
sampling_10_and_self_debug_1(	CPMpy	gpt-4.1-mini	100	92.06	0	
self_debug_gpt_5_mini	CPMpy	gpt-5-mini-2025-08-(	100	92.06	0	
multi_agents_requirements_ga(	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	

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

Extending CP-Bench to DCP-Bench-Open

Extending with more problems:

- Currently: 164 *verified* & *diverse* instances



Part C

Evaluating LLM-driven Constraint Modelling

So how well can LLMs model constraint problems?

Constraint modeling instructions to LLMs

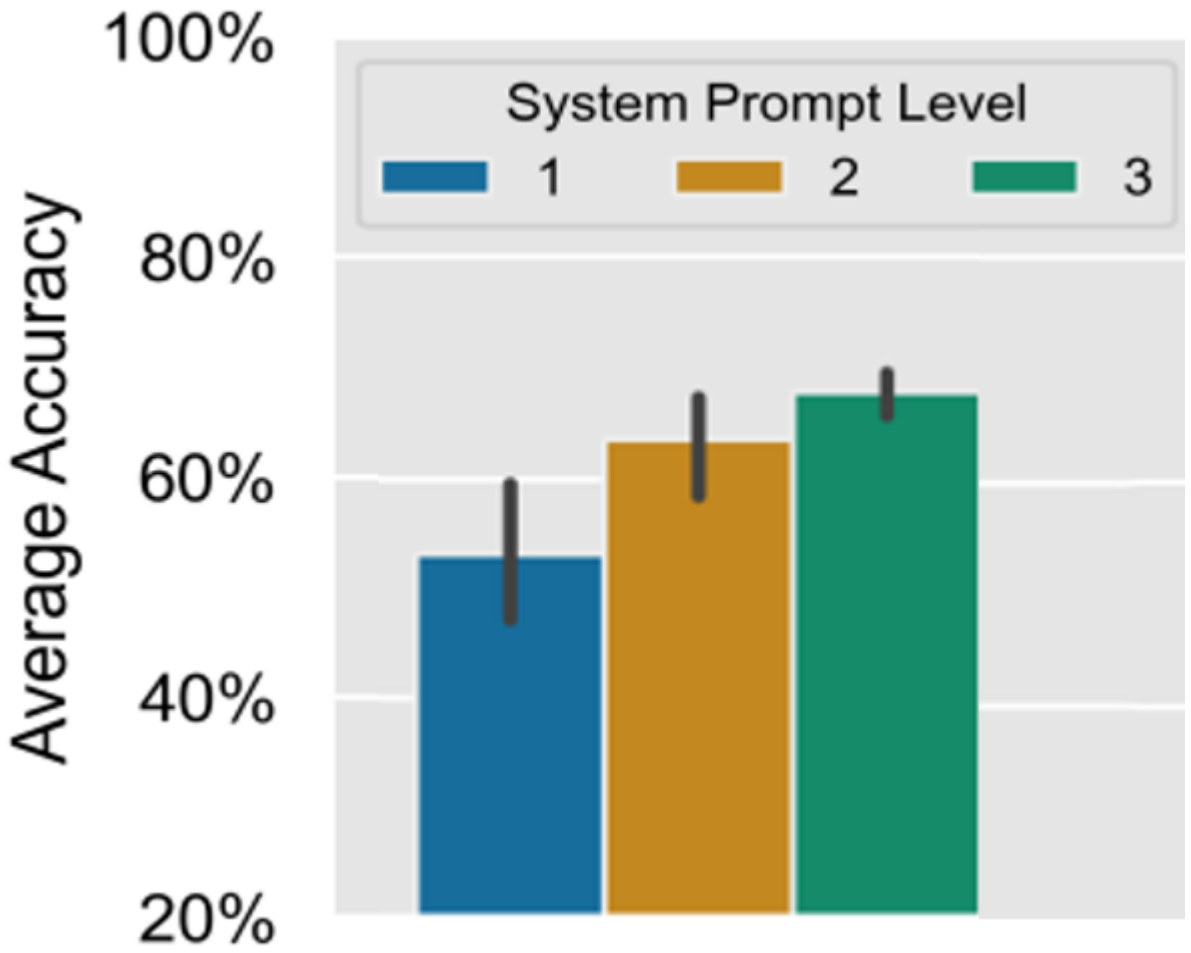
Evaluating effect of instructions in System Prompt.

3 Levels:

1. Basic prompt: Describe only the task
2. + Modeling guidelines: like guidelines to students
3. + Documentation of the modeling framework

Evaluating effect of instructions in System Prompt.

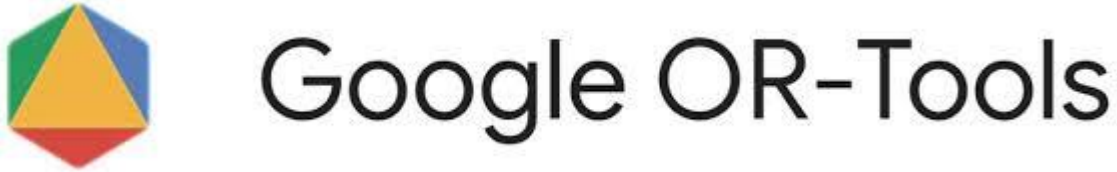
- Basic Prompt (1)
- Modelling Guidelines Prompt (2)
- Documentation Prompt (3)



What type of modelling framework can be used?

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

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



Evaluating the generated models

Importantly! CP-Bench only needs an (optimal) solution. Generated model can be in any framework!

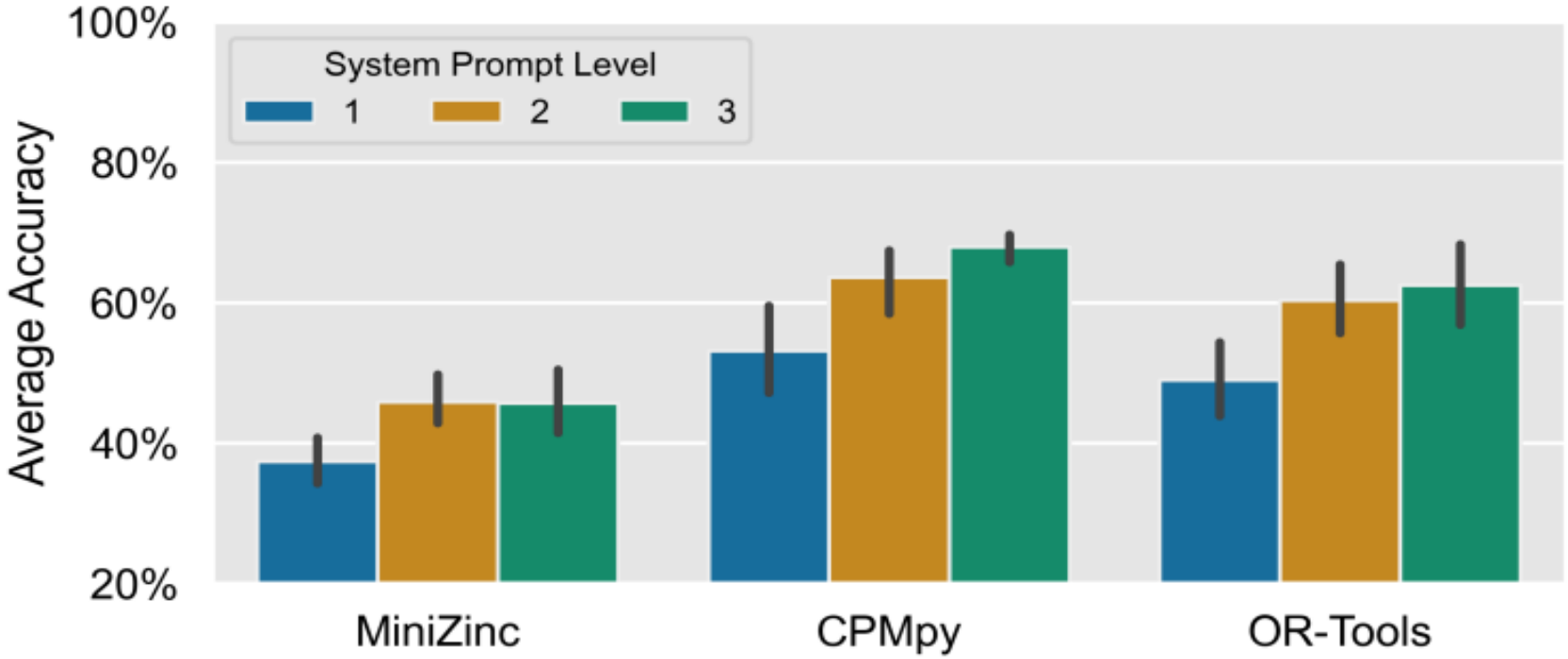
Google OR-Tools

What type of modelling framework can be used?

- Python-based vs. Domain-Specific

- High-level vs. Low-level

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



Lesson learned #6:

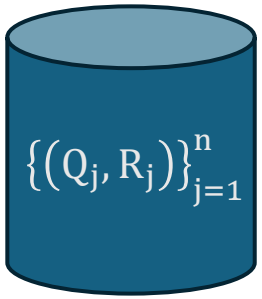
LLMs prefer **python**!

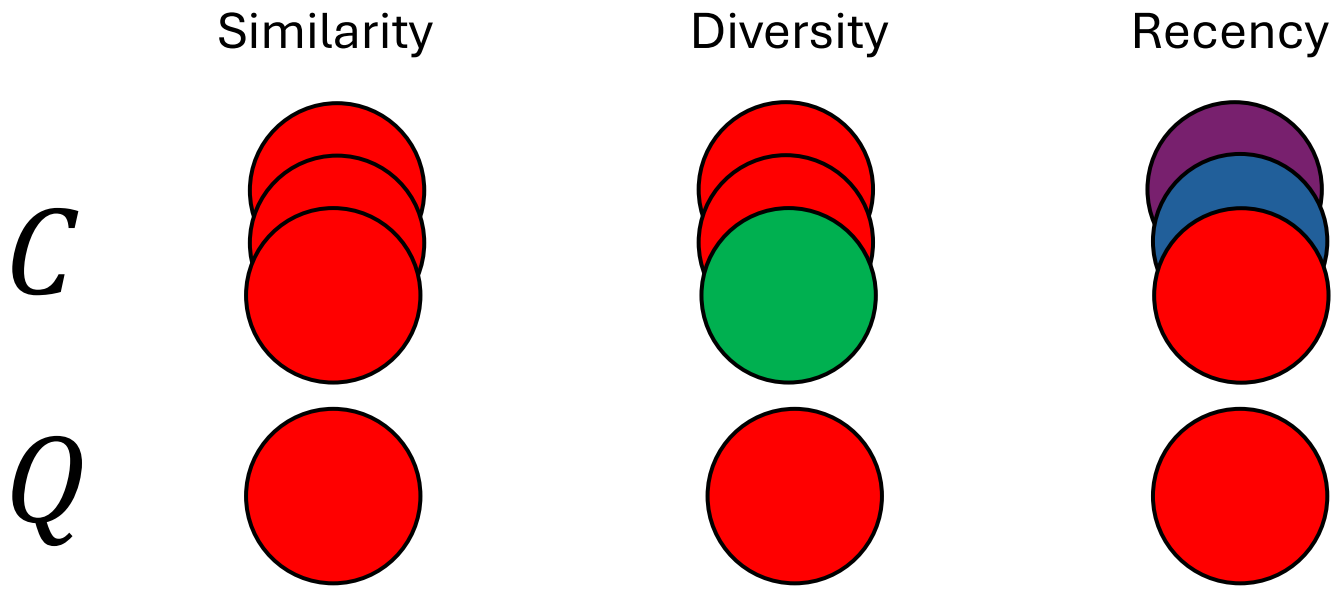
And LLMs prefer **high-level abstraction** …

What about In-Context Learning?

Baseline: Static/Random

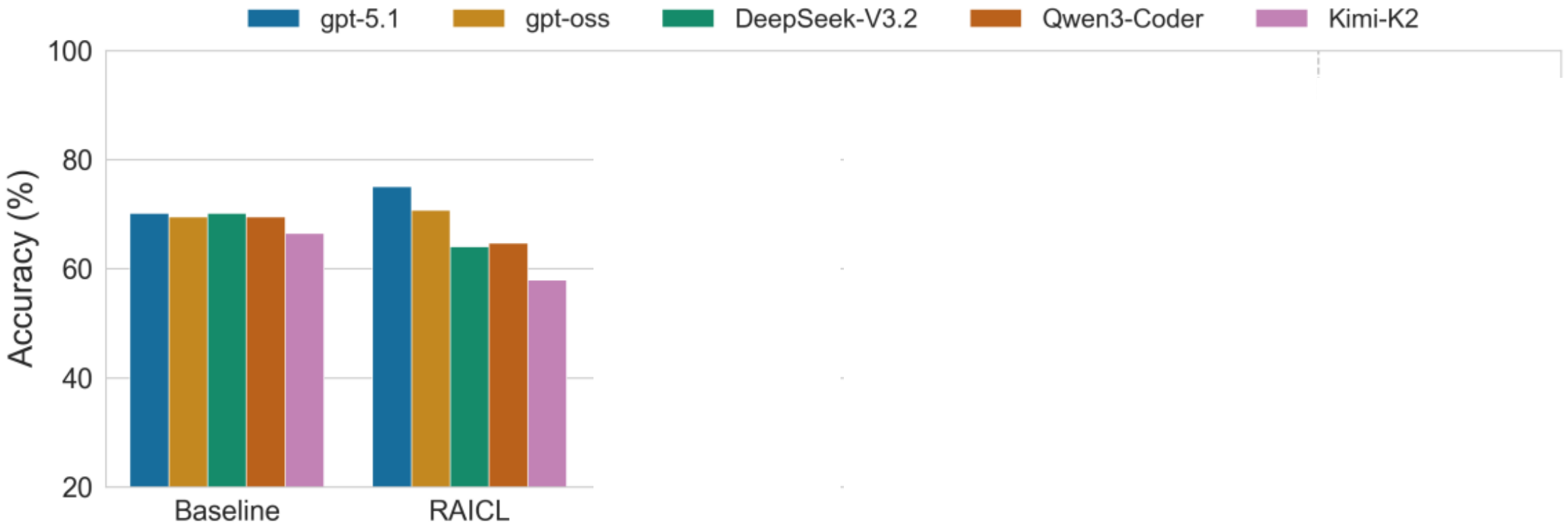
Retrieval-Augmented ICL





What about In-Context Learning?

- Results on DCP-Bench-Open (164 instances), CPMpy, system prompt 3

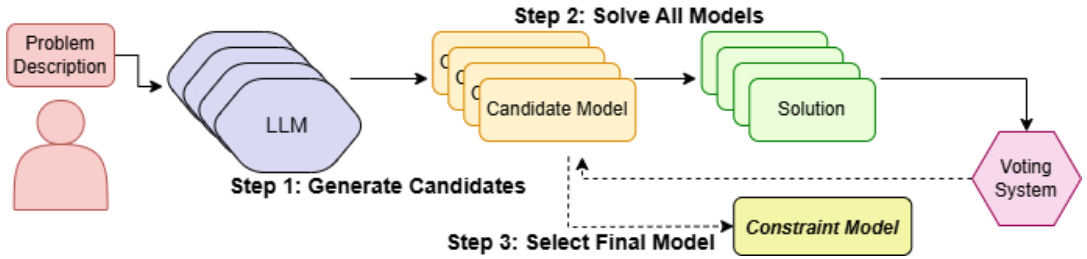


Lesson learned #7: In-context learning does not always help anymore.
Documentation prompt gives the important info + No homogenous problems

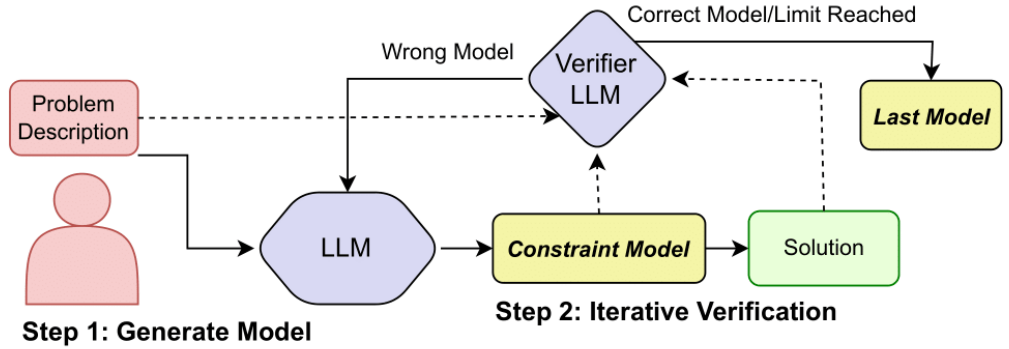
Inference-Time Compute Methods

Inspired from methods for coding assistants

1. Repeated Sampling



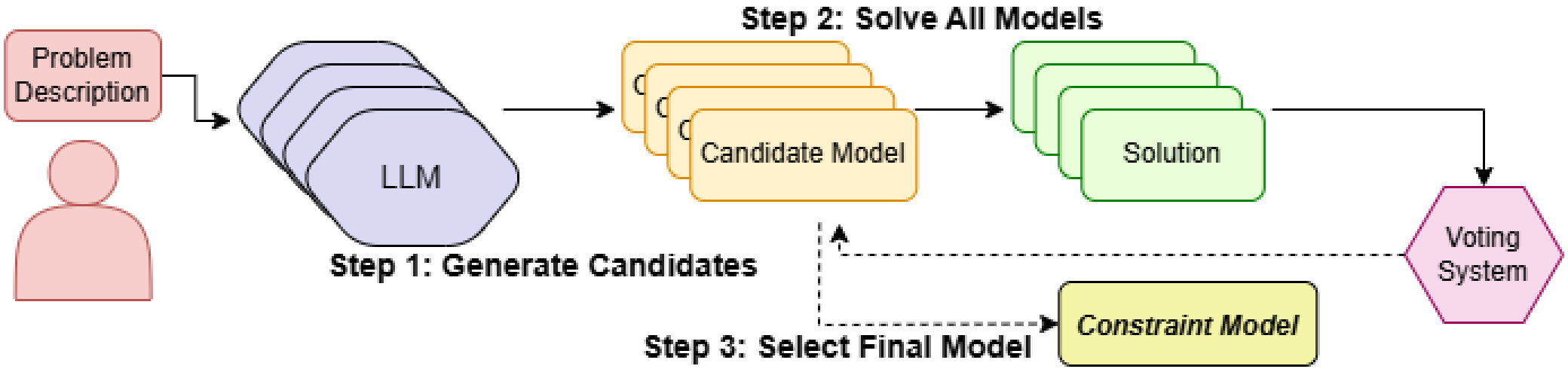
2. Self-Verification



+ *LLM internal reasoning*

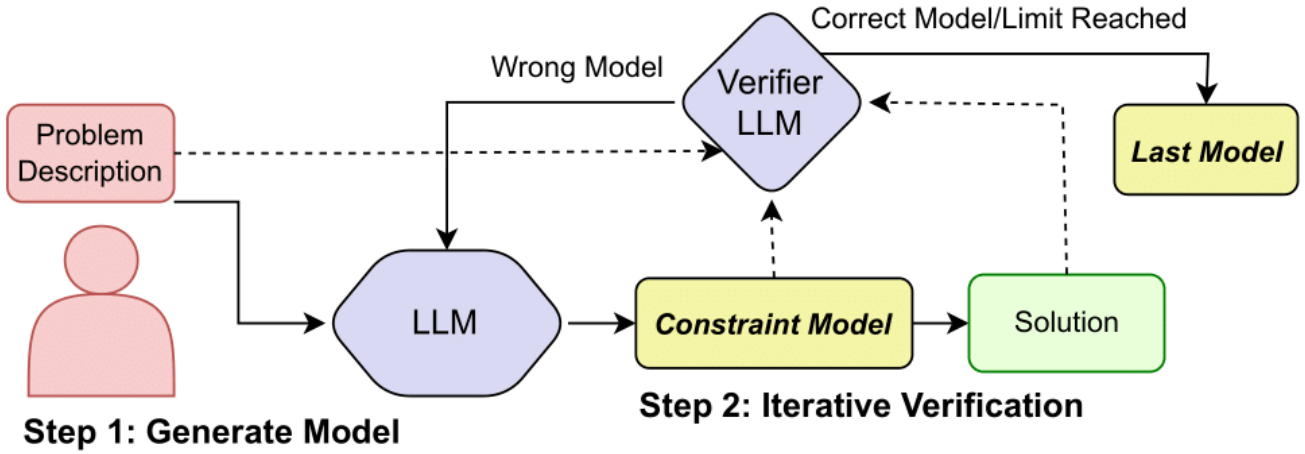
Repeated Sampling

- Exploiting variability in LLM outputs by increasing temperature
- Generate **N candidate models in parallel**
- **Solution Majority Voting** to select a model



Olausson, Theo, et al. "LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers." Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. 2023.

Self-Verification



- Modelling in a single pass is a hard task
- More **agentic** approach with tool calling
  - Execute the model and verify the output
  - Challenge: No unit tests as in imperative programming…

- *Self-Verification* prompt (v) includes 3 main criteria:
  - Runtime Integrity
  - Model Accuracy
  - Solution Output

Evaluating Inference-Time Compute Methods

- Results on DCP-Bench-Open (164 instances), CPMpy, system prompt 3

Huge improvement in gpt-5.1, but lower than baseline in other problems.

- Generic **internal reasoning** not always effective.

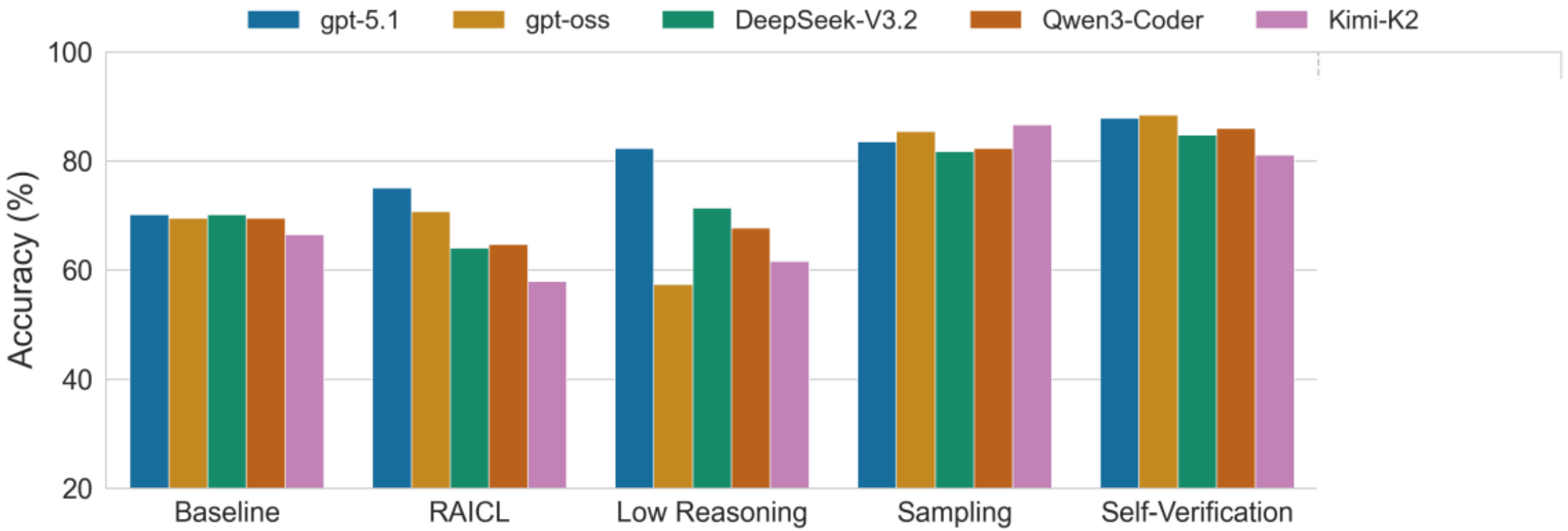
Evaluating Inference-Time Compute Methods

- Results on DCP-Bench-Open (164 instances), CPMpy, system prompt 3

Important gains with **sampling** in all LLMs (surprisingly?)

Evaluating Inference-Time Compute Methods

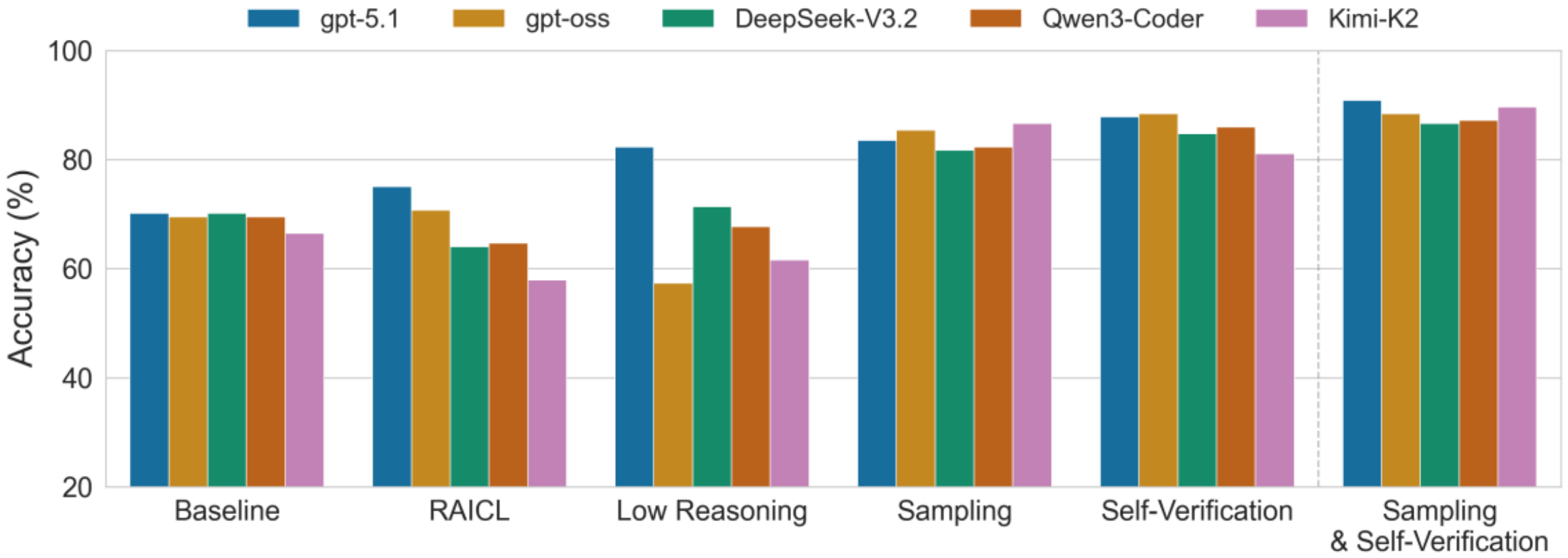
- Results on DCP-Bench-Open (164 instances), CPMpy, system prompt 3



Similar gains with **self-verification** of the generated model

Evaluating Inference-Time Compute Methods

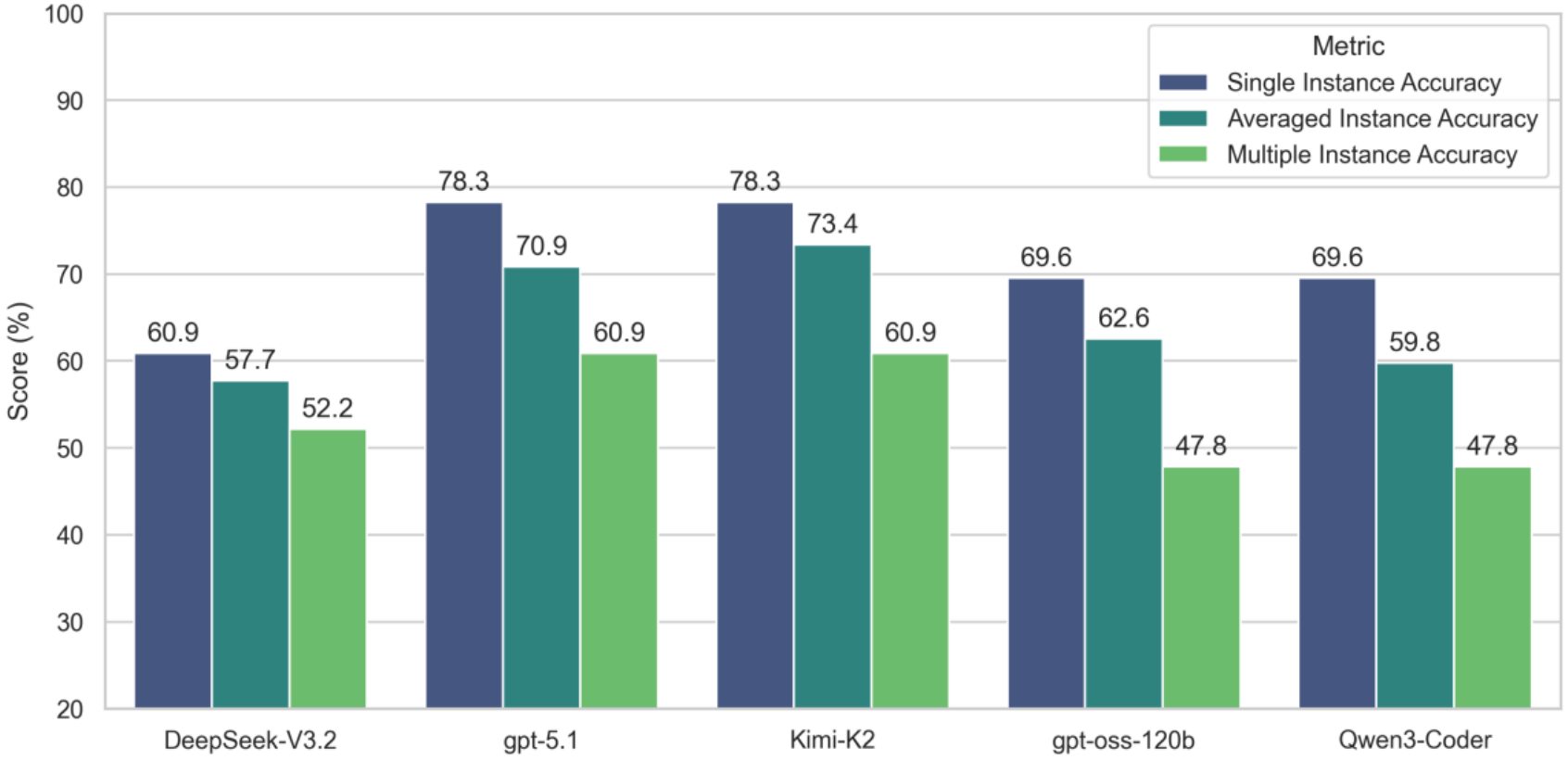
- Results on DCP-Bench-Open (164 instances), CPMpy, system prompt 3



Best performing: **combine sampling with self-verification**!
Lesson learned #8: Towards agentic approaches

What about multiple data instances?

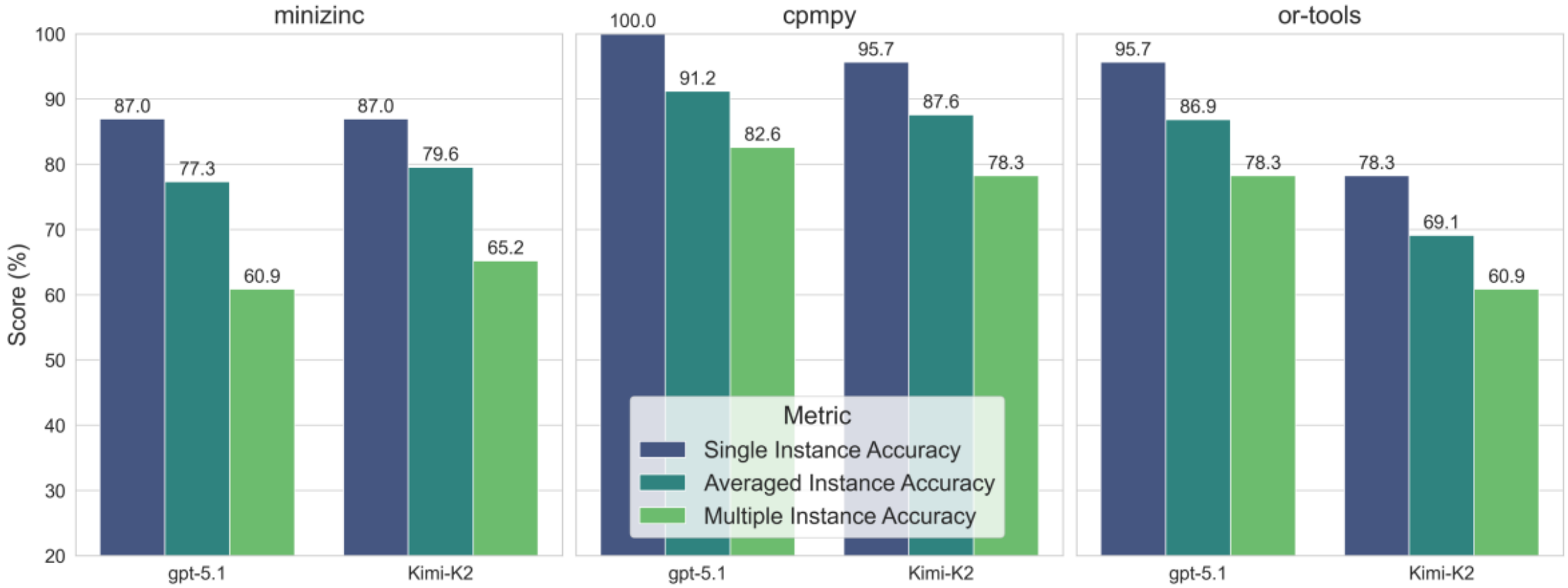
- Results on DCP-Bench-Open (23 instances), CPMpy, system prompt 3, baseline



Lesson learned #9: LLMs often overfit to the default data instance given in the prompt…

What about multiple data instances?

Results on DCP-Bench-Open (23 instances), CPMpy, system prompt 3, sampl+selfver

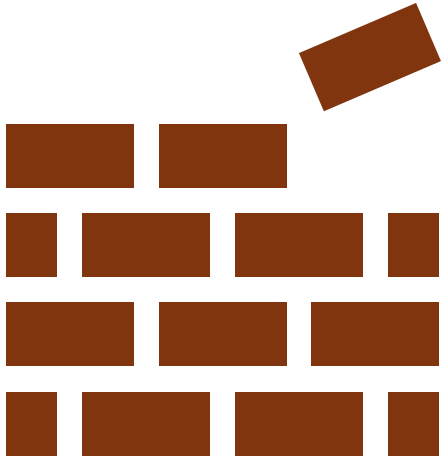


LLMs often overfit to the default data instance given in the prompt… Even with more advanced techniques

Part D

Conclusions

Conclusions (1)



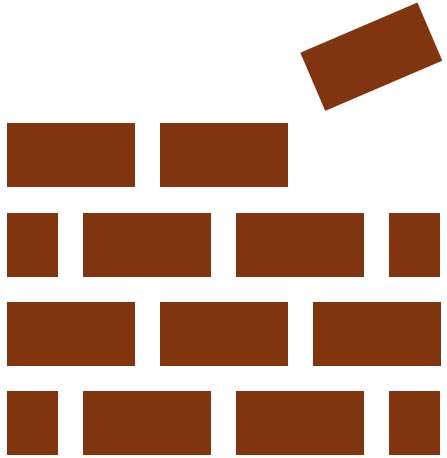
Benchmarking LLMs for constraint modeling is not straightforward

➢And can affect results a lot!

Should be a joint effort from the community

Extending CP-Bench to DCP-Bench-Open

- Extending with more problems:
  - Currently: 164 *verified* & *diverse* instances
- **Multi-instance evaluation** and more metrics
- **Evaluating more frameworks** and modelling paradigms



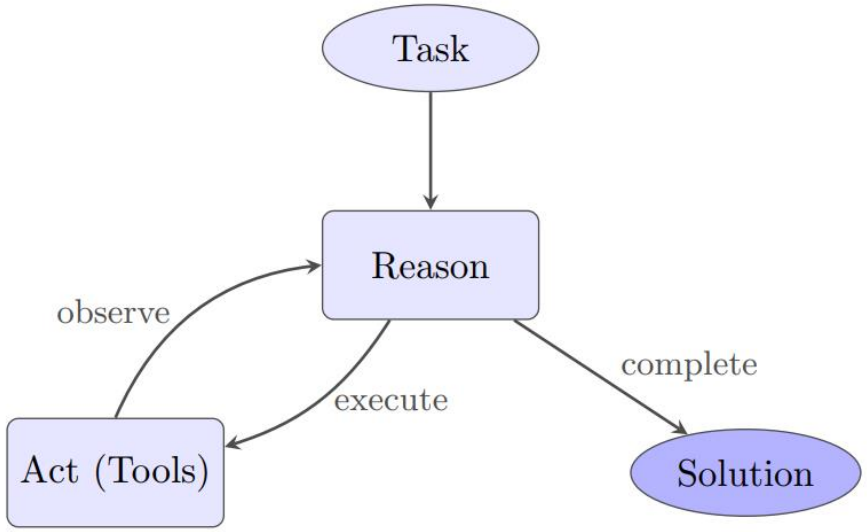
DCP-Bench-Open is **OPEN** to contributions from the community:

✓ More problems
✓ More instances per problem
✓ More runners in additional frameworks
✓ Evaluation metrics (accuracy, tokens, efficiency …)

https://github.com/DCP-Bench/DCP-Bench-

Conclusions (2)

- **Frameworks:** Python-based advantage over domain-specific
- **Richer prompts** => stronger LLM performance
- **Inference-Time Compute** methods can boost LLM performance
  - Towards agentic systems[1].



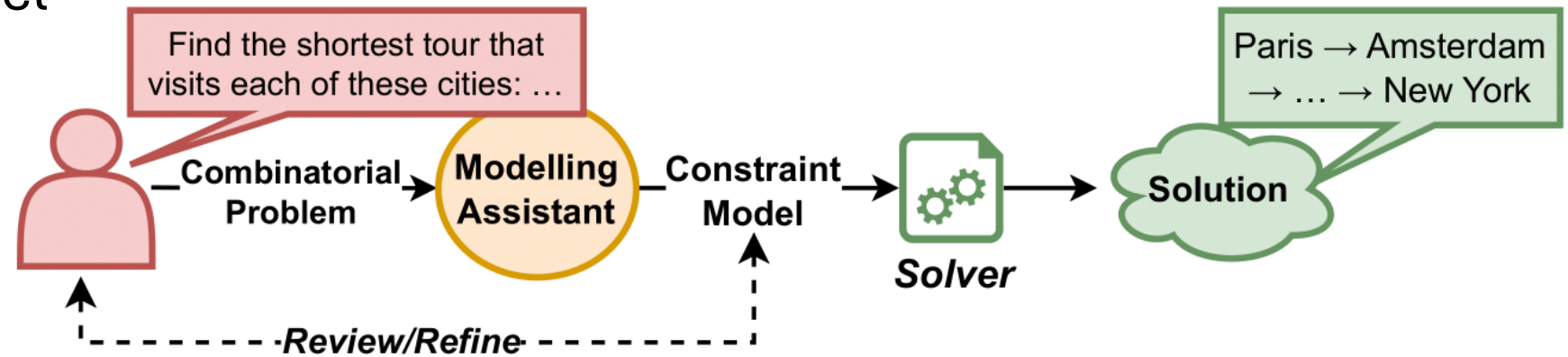
1. Stefan Szeider, “CP-Agent: Agentic Constraint Programming” (2025), *First in CP-Bench online leaderboard*

Promising avenues

LLM-based systems show good performance for one-shot modeling

But modeling is not a one-shot procedure![1]

- From ambiguous to unambiguous descriptions
- Users forget/miss constraints -> Interactively **acquire constraints**[2]
- Users need **explanations** / to understand (optimality of) solutions / to refine the model



1 Lawless et al. ""I Want It That Way": Enabling Interactive Decision Support Using Large Language Models and Constraint Programming"
2. Mechqrane et al. "Active Constraint Acquisition Using Large Language Models" (2026)

Promising avenues

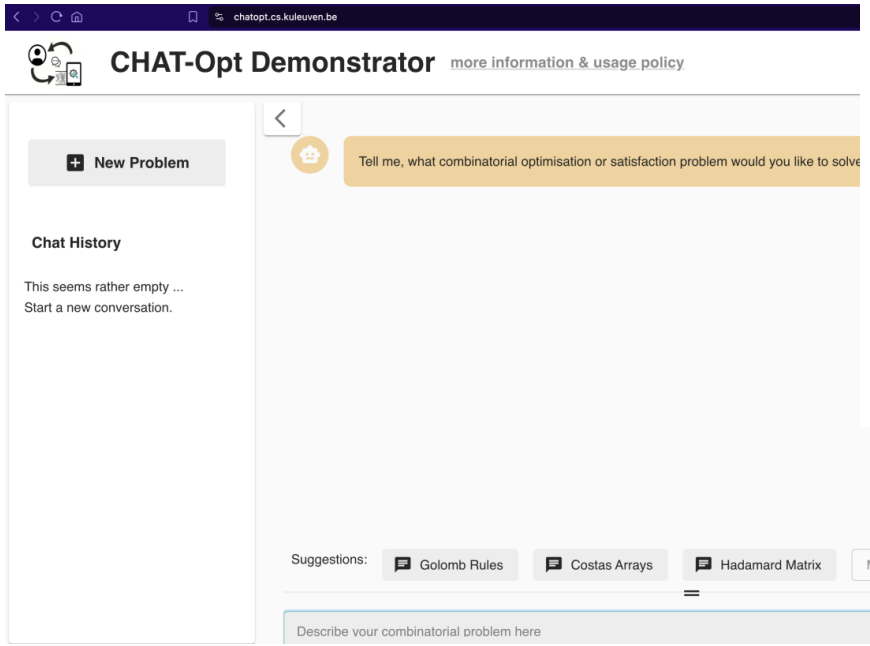
LLM-based systems show good performance for modeling

But users also need efficient models!

➢ LLM-based model reformulation for solving efficiency

DCP-Bench-Open
(repository)

CP-Bench
(paper)



ChatOpt Online Demo
https://chatopt.cs.kuleuven.be

Thank You!