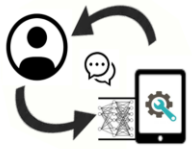
LLM-Guided Evolutionary Search for Model Reformulation towards Solver Efficiency

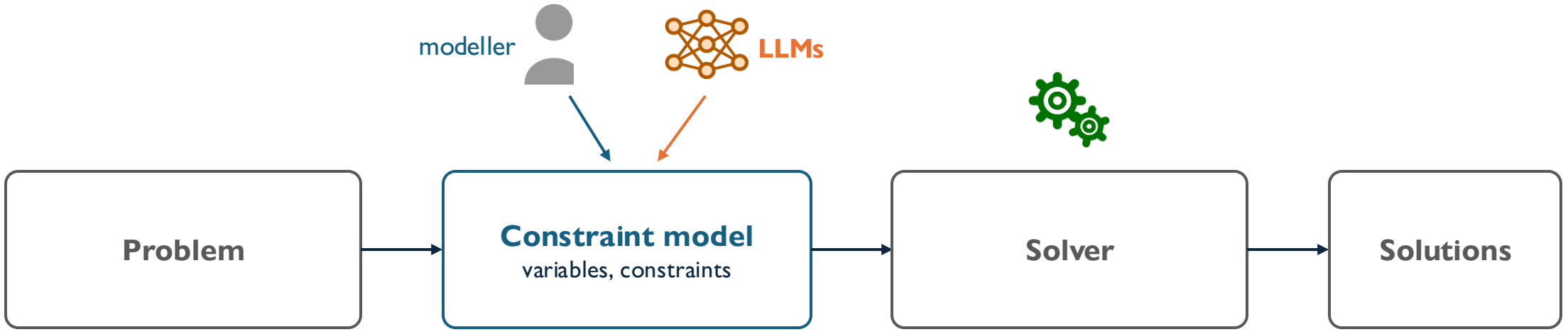
Kostis Michailidis[1], Dimos Tsouros[2], Nguyen Dang[3], Tias Guns[1]

[1]DTAI, KU Leuven [2]University of Western Macedonia [3]University of St Andrews

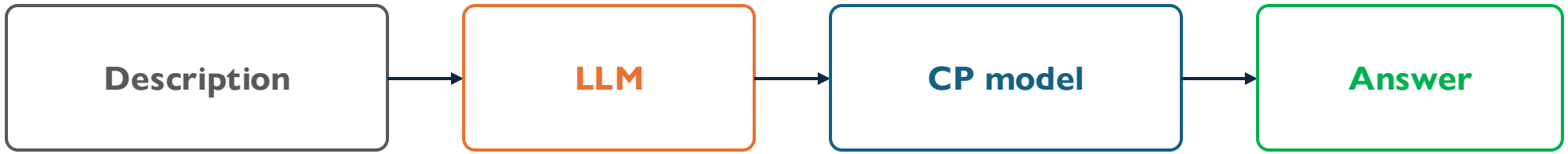
LLMs meet Constraint Solving | CP 2026 Workshop | Lisbon, 19 July 2026



Model-and-solve



LLMs can already write correct models



Name	Modelling Framework	Base LLM	Models Submitted (%)	Accuracy (%)
cpagent	CPMpy	Claude 4 Sonnet	100	100
ml_research_loop_p18	CPMpy	Codex-assisted deterministic CPMPy solver expansi	98.41	95.24
sampling_10_and_self_debug_1(	CPMpy	gpt-5-mini-2025-08-07	100	95.24
self_debug_gpt_5	CPMpy	gpt-5-2025-08-07	100	93.65
sampling_10_and_self_debug_1(	CPMpy	gpt-4.1-mini	100	92.06
self_debug_gpt_5_mini	CPMpy	gpt-5-mini-2025-08-07	100	92.06
multi_agents_requirements_ga	OR-Tools	o3	100	88.89

CP-Bench Verified leaderboard: solution accuracy of generated models.

Correctness is largely handled. Solving time is not measured.

Related Work

Model Reformulation:

- **SavileRow** Nightingale et al. 2017
- **Streamliners** Spracklen et al. 2023
- **Conjure** Akgün et al. 2022

LLM-generated Streamliners:

- **StreamLLM** Voboril et al. 2025

Model reformulation towards solving efficiency

Same problem, different models

Car sequencing: place cars on a line so that no option exceeds its capacity in any window.

slow the baseline model

```
# one integer per position
car = intvar(0, n_cls-1, shape=n)

# demand per class
for c in classes:
    m += Count(car, c) == demand[c]

# option use needs a lookup
for o in options:
  for w in windows(o):
    m += sum(req[car[p], o]
             for p in w) <= cap[o]
```

fast another viewpoint

```
# Boolean class-position matrix
y = boolvar(shape=(n, n_cls))

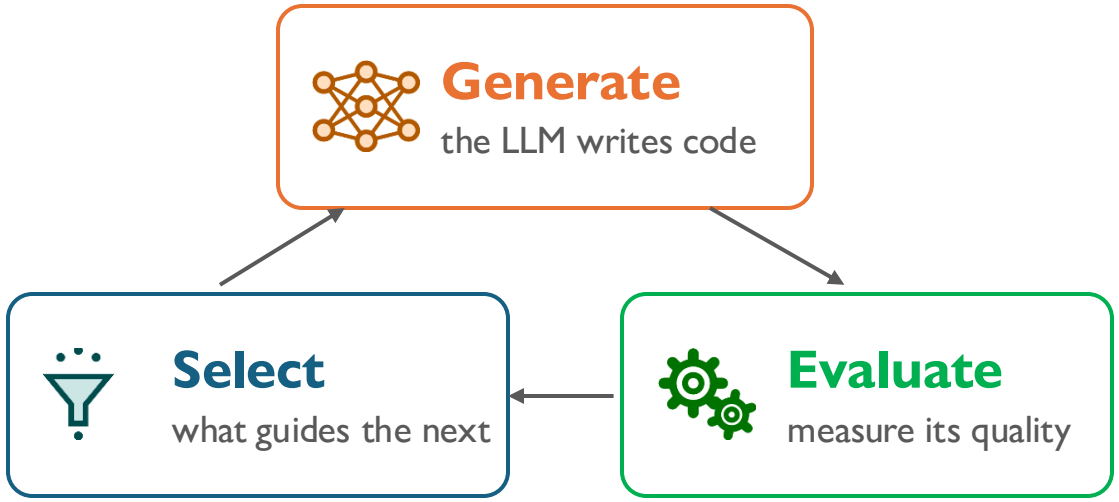
# exactly one class per position
m += [sum(y[p, :]) == 1 for p in pos]
m += [sum(y[:, c]) == demand[c] ...]

# option use is now linear in y
use[o, p] = sum(req[c, o] * y[p, c]
                for c in classes)
for o in options:
  m += sum(use[o, w]) <= cap[o]
```

The solver behaves differently on these models.

LLM-guided evolutionary search for faster models

Background: Automatic Heuristic Design (AHD)



Evolutionary search over code

FunSearch Romera-Paredes et al. 2024

EoH Liu et al. 2024

ReEvo Ye et al. 2024

LLaMEA van Stein and Bäck 2024

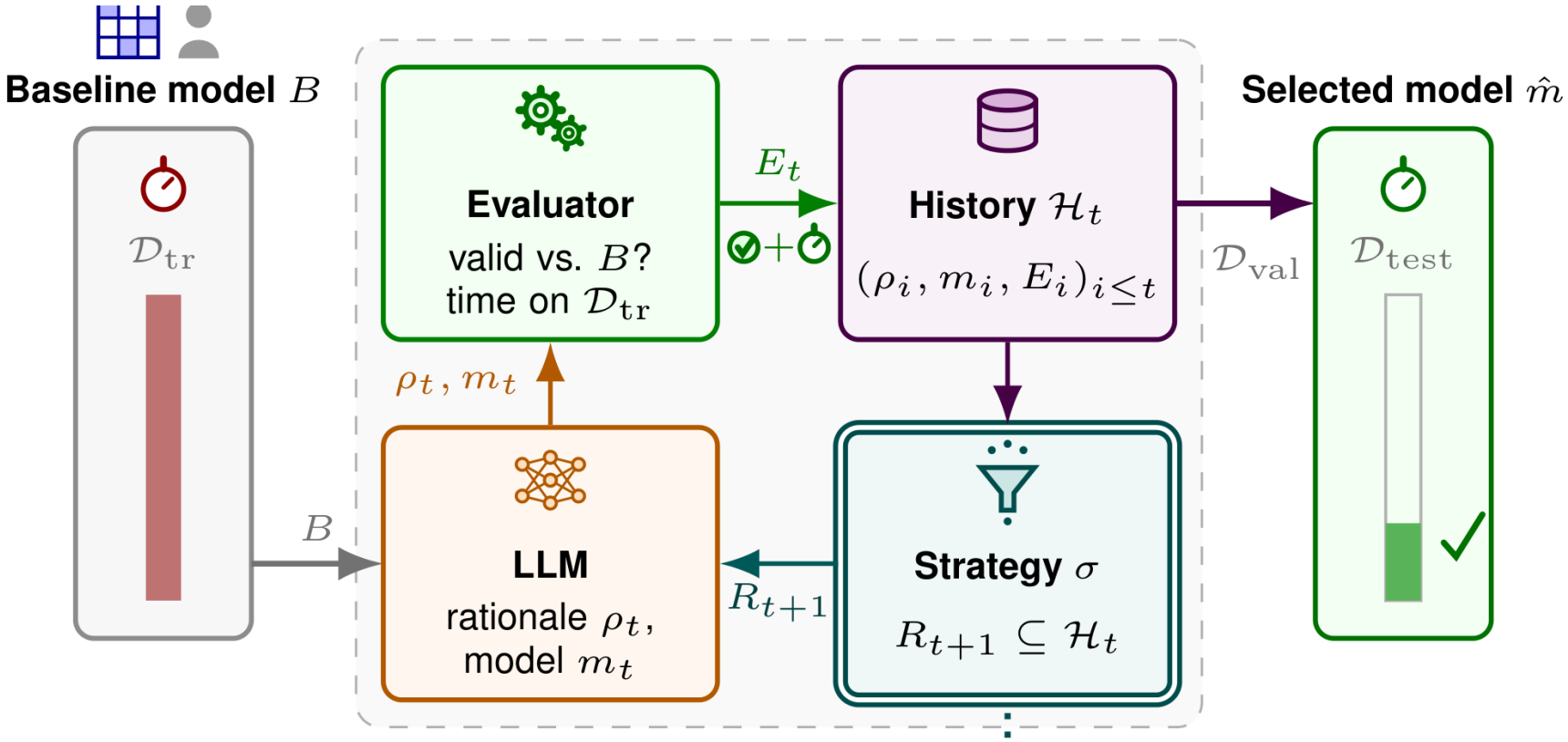
and many more frameworks

All of them evolve imperative algorithms.

Adapting it to constraint models

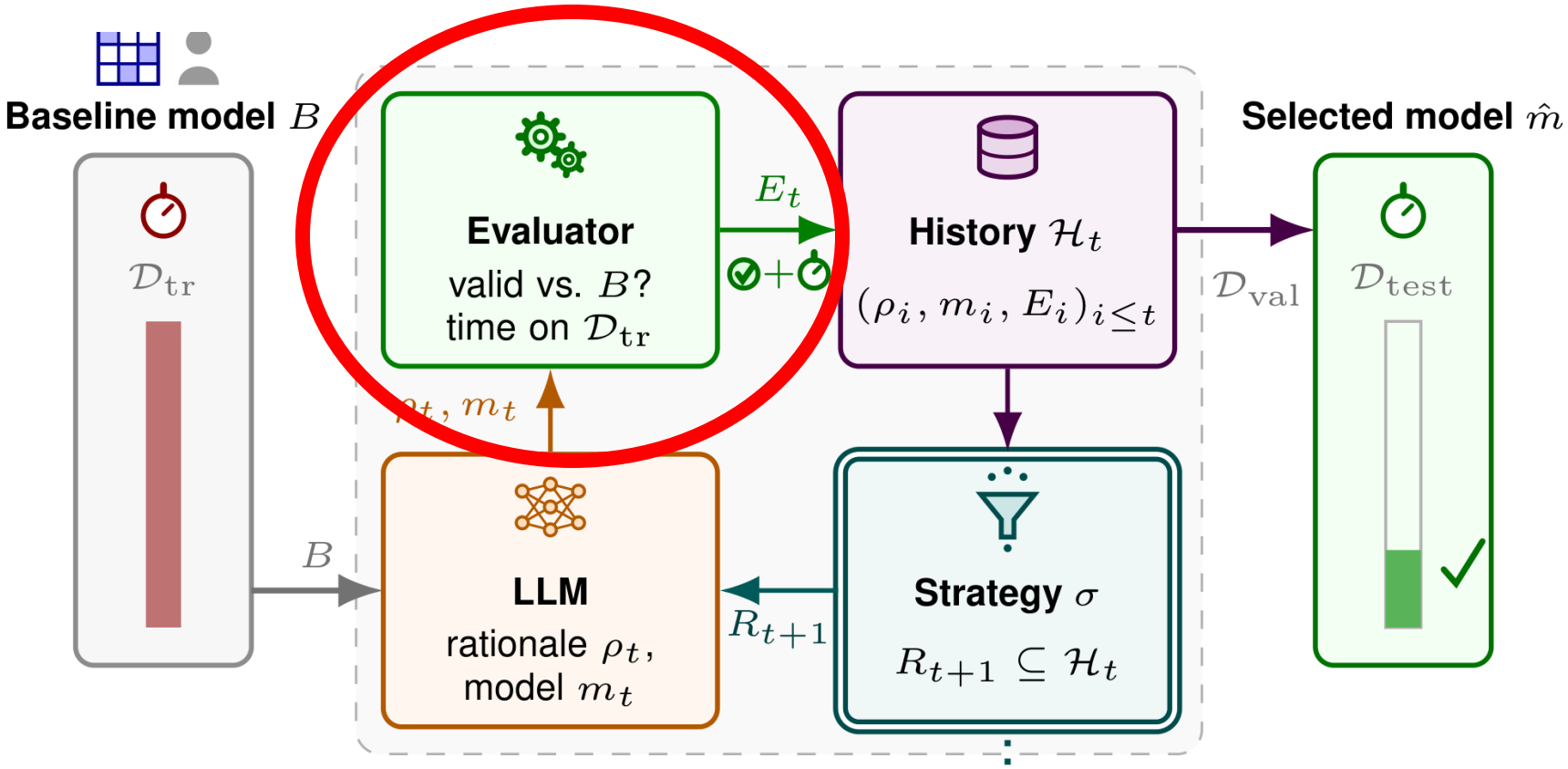
	Automatic Heuristic Design	Our work
Output	an imperative heuristic	→ a declarative CP model
Metric	objective value	→ solving time on training instances
Correctness	the code runs	→ model must remain **valid**, i.e. its first solution must be correct

Reformulation loop



what context should the LLM see in each iteration?

Reformulation loop



Candidate Evaluation

1. Validity

```
# 1. the candidate solves
model.solve()
ans = extract(model)
```

add as constraint

```
# 2. the baseline must admit it
B += [x[i] == ans[i] for i in X]
B.solve()   # SAT -> accept
```

- Solution-level accuracy

2. Performance

solving time on the training instances, speedup compared to the baseline

Solution-level versus *semantic equivalence*

Baseline

```
car = intvar(0, n_cls-1, shape=n)
Count(car, c) == demand[c]
sum(req[car[p], o] for p in w) <= cap[o]
```

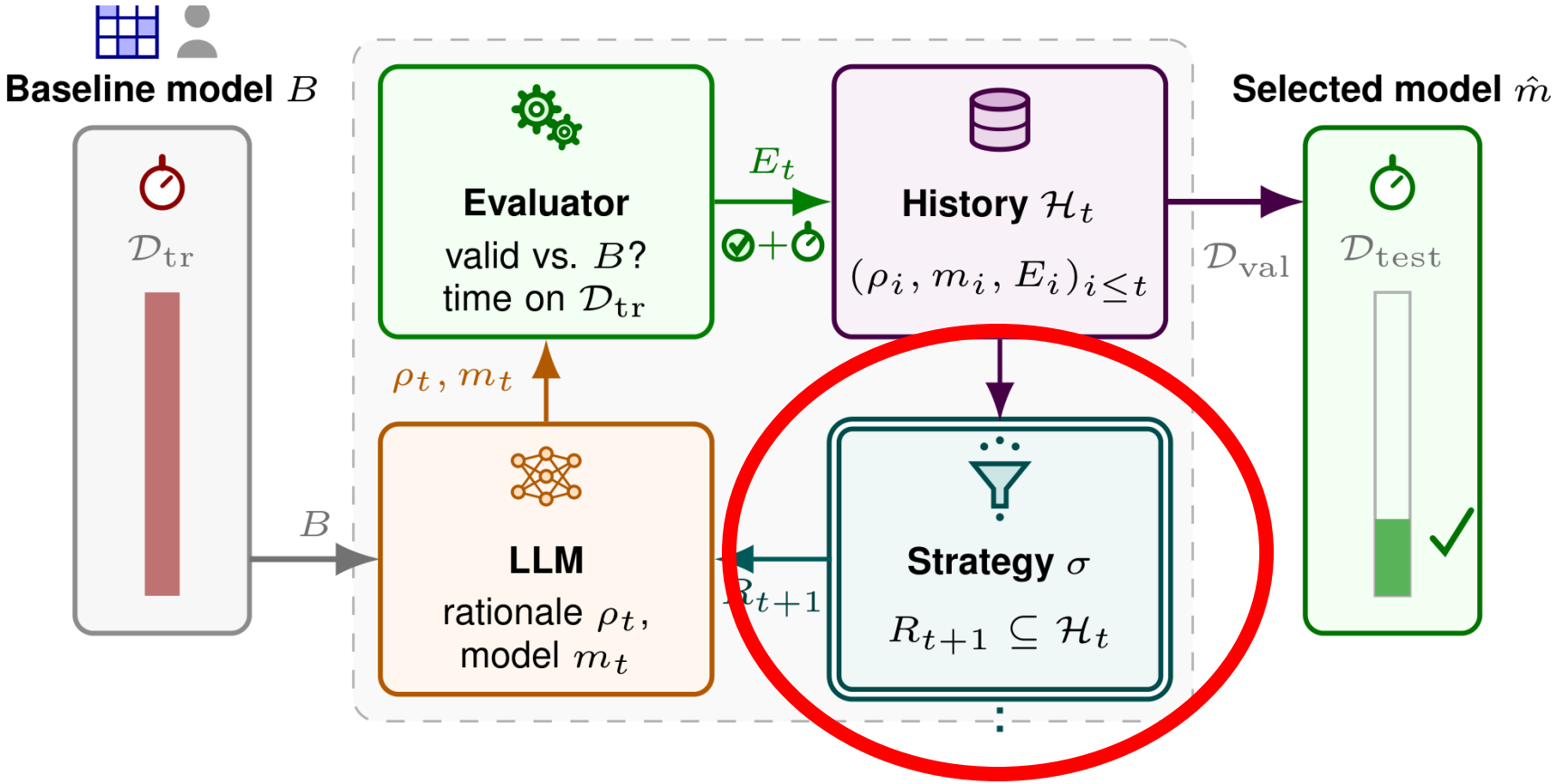
Candidate

```
y = boolvar(shape=(n, n_cls))
sum(y[p, :]) == 1
use[o, p] = sum(req[c, o] * y[p, c] ...)
sum(use[o, w]) <= cap[o]
```

Model-level/Semantic equivalence:

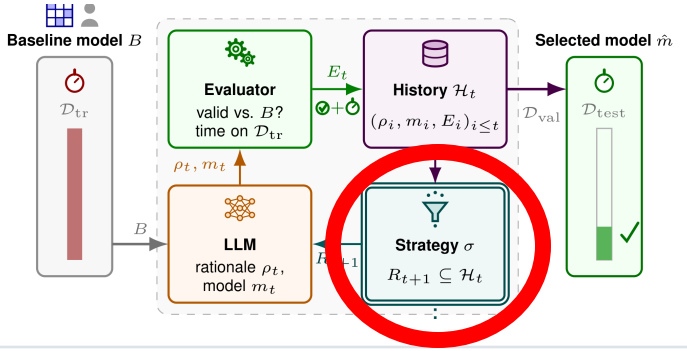
- Constraint-by-constraint comparison not possible: different variables etc.
- Full solution-set comparison can be very expensive.
- Reverse check is non-trivial: we do not control the generated model's variables.

Reformulation loop



what context should the LLM see in each iteration?

Retention Strategies



Sampling	nothing
Full-history	everything that fits
Last-k	the k most recent attempts
Top-k	the k fastest attempts
Hybrid	last-k and top-k

Evolution of Heuristics

a population of parents, and operators that say how to "use" them

Profile-Diverse Retention

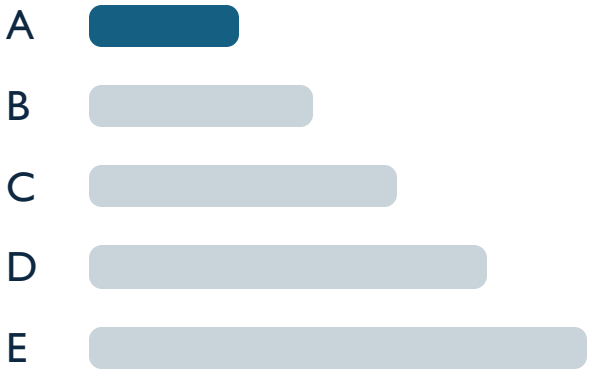
selects attempts based on both **speed** and for behaving **differently**

EoH: population and operators

An evolutionary loop: keep a population, sample parents, vary them.

Population

e.g. the 5 fastest valid models



bar = training runtime

sample 1–2 parents

$\propto$ 1 / runtime

faster is likelier

Prompt "Operators": applied in a fixed cycle

e1	2 parents	propose a distinct approach
e2	2 parents	combine them
m1	1 parent	make a local change
m2	1 parent	change a solver-facing component

Evolution of Heuristics (EoH) by Liu et al. 2024

Profile-Diverse Retention (PDR)

Every valid model has a runtime profile over the training instances.

m1

fast total runtime

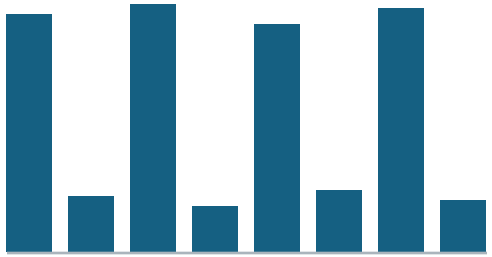
✓ **retained**

m2

similar shape, slower

✗ dropped

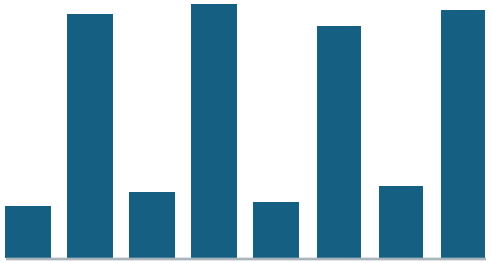
m3



different behaviour

✓ **retained**

m4



different behaviour

✓ **retained**

Retain a small set that is **fast** and **behaviourally diverse** using Maximal Marginal Relevance (MMR).

$$\arg \max_{m \in \mathcal{A}_t \setminus R} \left[\lambda q_t(m) + (1 - \lambda) \min_{u \in R} d_t(m, u) \right]$$

One iteration

EoH + feedback prompt | car sequencing | iteration 12

SYSTEM

fixed context

```
## Role        ...reformulate a given CPMpy baseline model to maximize solving efficiency.
## Rules       Keep build_model(data) -> (model, extractor) and the same input/output contract.

## Baseline Model      def build_model(data):  ...

## Baseline Training Runtimes   1: 1.39s   2: 1.98s   ...   8: 18.81s
```

USER

dynamic context

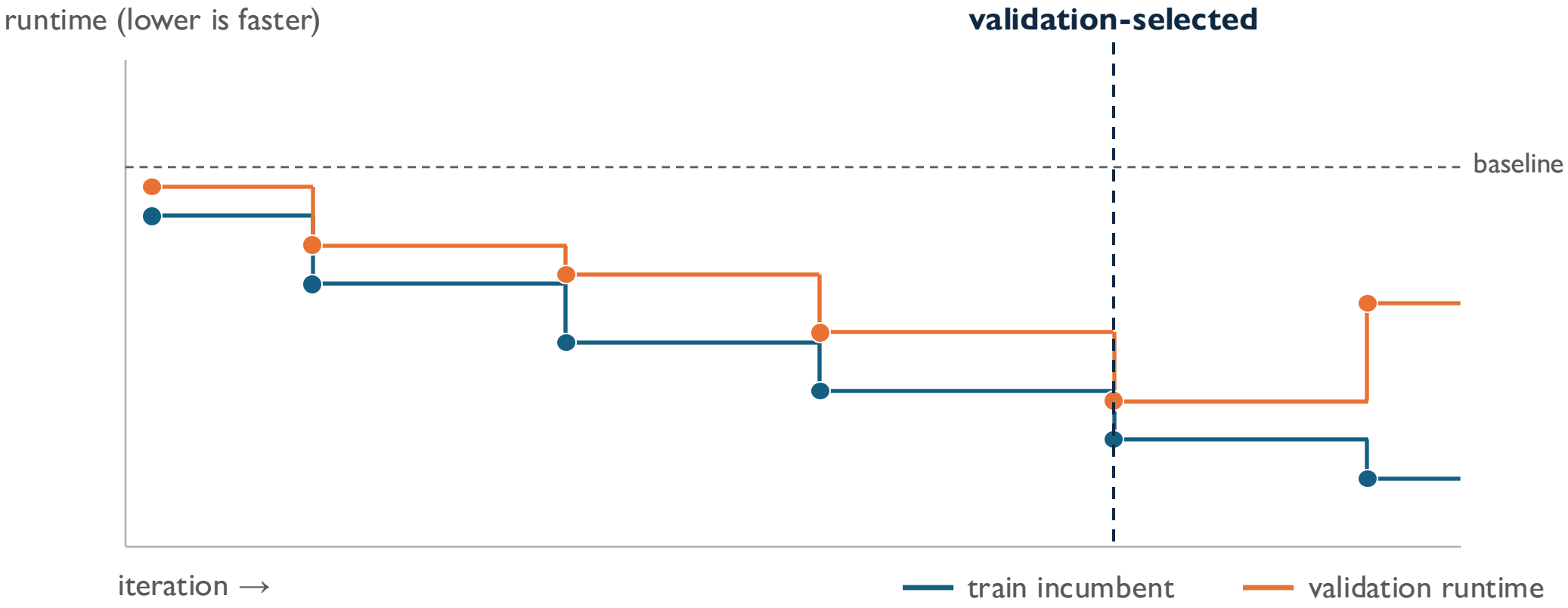
```
## Parent Models
### Parent 1
## Rationale  ...   ## Final Model
```python ...
### Measured Performance
20.97s vs 48.95s | 2.33x

### Parent 2
## Rationale  ...   ## Final Model
```python ...
### Measured Performance
17.74s vs 48.95s | 2.76x
```

operator e2

```
## Task   Generate a new model using common or compatible ideas from the
          parent models above...
```

Validation: Which model do we return?



The fastest model on the **training** instances is not always the one that transfers to **held-out** instances.

Results

Setup

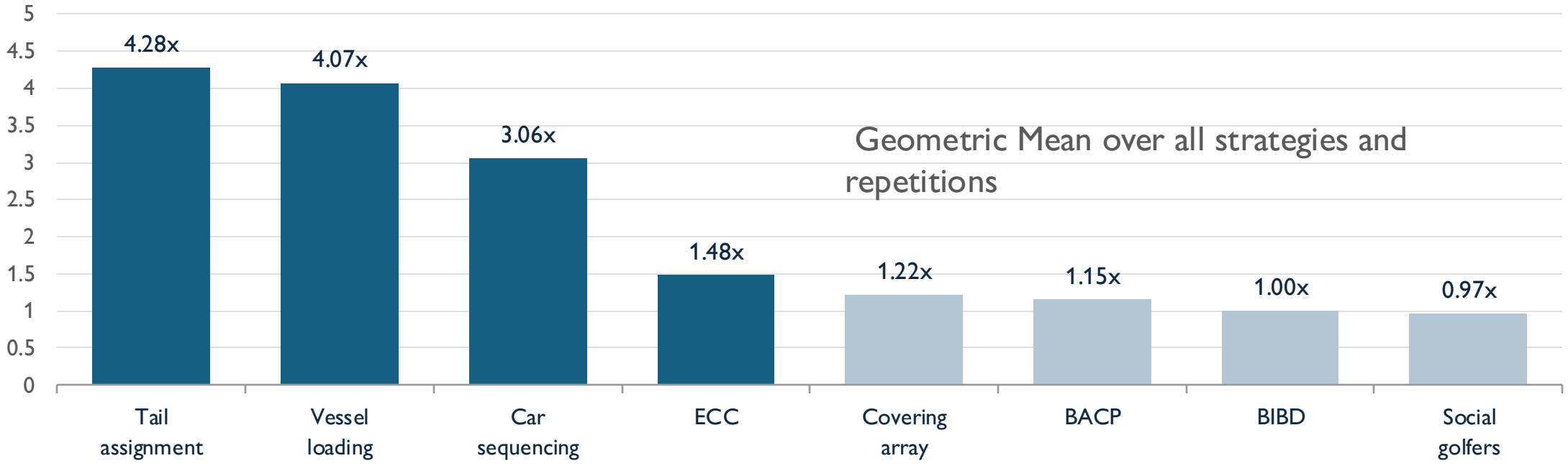
Problems	8 from CSPLib	car sequencing, vessel loading, social golfers, BIBD, BACP, error-correction codes, covering arrays, tail assignment
Model & solver	CPMpy + OR-Tools (single-core)	
Generator	DeepSeek V4 Flash (high reasoning effort, 1M-token context)	
Search	3 repetitions × 100 iterations per strategy/problem pair	

Instances: AutoIG pools, remeasured on our baseline models (Dang et al. 2022; Spracklen et al. 2023).

8 runtime bins → 1 train + 1 validation + 2 test from each = 8 / 8 / 16 per problem

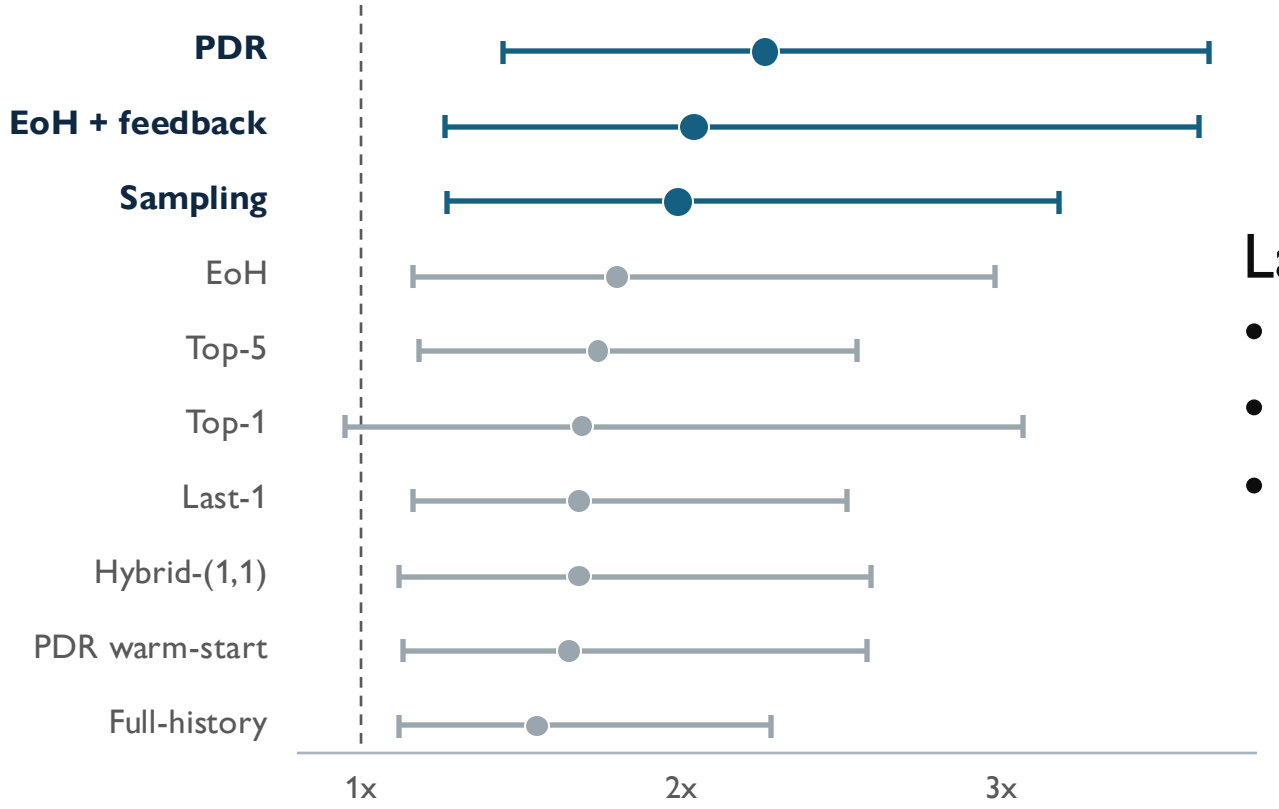
Speedup per problem

144 of 240 searches return a faster model | 67 baseline returned · 29 did not transfer



- **Every returned model stayed valid**
- **Variance in gain per problem is large**

Speedup per strategy

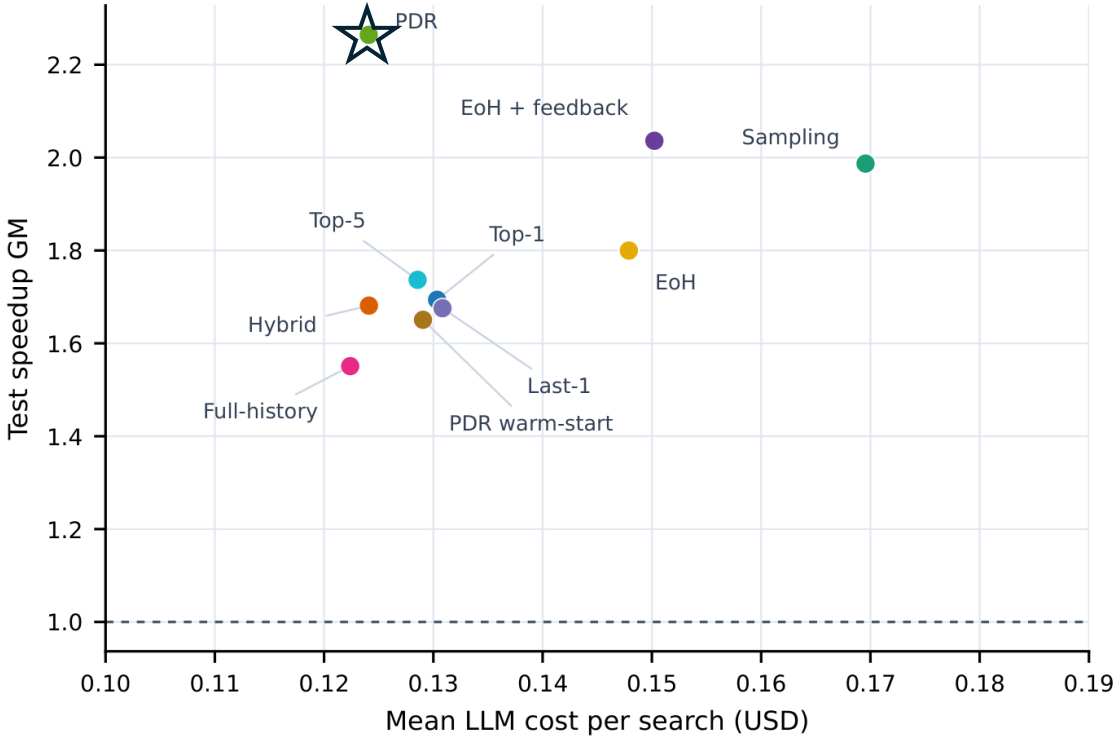


Held-out GM speedup, 95% cluster-bootstrap CI.

Largest speedups:

- behavioural diversity
- measured feedback evolution
- (surprisingly) Repeated Sampling

Cost per search



Sampling most expensive

it re-derives everything: 0.50M reasoning tokens

Full-history cheapest

6.7M input tokens, 98% cache hits

PDR best speedup, 27% cheaper than Sampling

Do LLMs use modelling techniques?

An example: car sequencing

Baseline

```
# one integer per position
car = intvar(0, n_cls-1,shape=n)
sum(req[car[p], o]
    for p in w) <= cap[o]
```

A: viewpoint change

```
# Boolean class-position
y = boolvar(shape=(n, n_cls))
use[o,p] = sum(req[c,o]
     * y[p,c] for c in cls)
```

Artigues et al. 2014

B: option profiles

```
# merge equal classes
prof = group(classes,key=req)
y = boolvar(shape=(n,n_prof))
```

Dincbas et al. 1988

- Pattern A appears in **29 of 29** winning car-sequencing searches, pattern B in 6 of them.
- Best model: **5.8x** (Full-history, A+B).

Modelling patterns observed

Problem	Most frequent pattern (A)	Second recurring pattern (B)	Faster models	Best model (strategy, patterns)
Car sequencing	Boolean class–position matrix viewpoint (Artigues et al. 2014) (29/29)	Aggregation of interchangeable types by option profile (Dincbas, Simonis, and Van Hentenryck 1988) (6/29)	29/30	$5.8\times$ (Full-history, A+B)
Vessel loading	Reified pairwise spatial disjunctions (Moffitt and Pollack 2006) (26/29)	Binary orientation variables for 90° rotation (Andrade and Birgin 2013) (26/29)	29/30	$11.8\times$ (PDR, A+B)
Social golfers	Symmetry breaking via first-week ordering or fixed golfer–group assignments (Liu, Löffler, and Hofstedt 2019) (9/9)	Global-cardinality constraints for group sizes (Régin 1996) (4/9)	9/30	$1.7\times$ (Sampling, A)
BIBD	Equivalent incidence-matrix formulations (2/2)	–	2/30	$1.2\times$ (EoH+feedback, A)
BACP	Prerequisite-derived bounds on course-period domains (Schaus, Deville et al. 2008) (11/13)	Redundant cumulative constraints for period capacities (Aggoun and Beldiceanu 1993) (2/13)	13/30	$3.9\times$ (Top-5, A)
ECC	Symmetry breaking via lexicographic ordering of codewords (Flener et al. 2002) (15/24)	Tabulation of pairwise symbol-distance contributions (Cena et al. 2023) (12/24)	24/30	$3.0\times$ (EoH+feedback, B)
Covering array	Initial-column symmetry breaking or streamlining (7/8)	Base-g encoding of column-wise t-tuples (Hnich et al. 2006) (6/8)	8/30	$250.7\times$ (PDR, A+B)
Tail assignment	Boolean variables over feasible flight arcs (15/30)	Set-membership constraints on successor variables (12/30)	30/30	$11.5\times$ (Top-1, A)

Conclusions & Discussion

Summary

- An evolutionary framework for model reformulation with wide speedups
- PDR: retains fast and behaviourally diverse models
- Validation-based selection avoids overfit
- Many reformulations use established modelling techniques

Discussion: Limitations/Future Work

Correctness

- Equivalent models or accepted solution?
- For equivalence: solution enumeration or bidirectional check is necessary.

Search

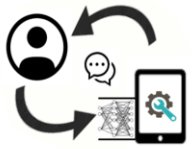
- Known techniques as explicit operators? E.g. “use sym-breaking to improve…”
- PDR: Diversity from solver-internal features?
- Retain failures, or a compressed ledger/summary?
- Free choice of solver and configuration too?

Evaluation

- More LLMs, solvers, optimization problems…
- How to select training instances? Fast or distributed?
- One model, or a portfolio?

Thank you!


erc
European Research Council
Established by the European Commission




DTAI
Declarative Languages &
Artificial Intelligence


KU LEUVEN