Airbus Amber

Solving scheduling problems with Constraint Programming and Graph Neural Networks

ANITI Days – 16-17 November 2023

Guillaume Povéda, Florent Teichteil-Koenigsbuch

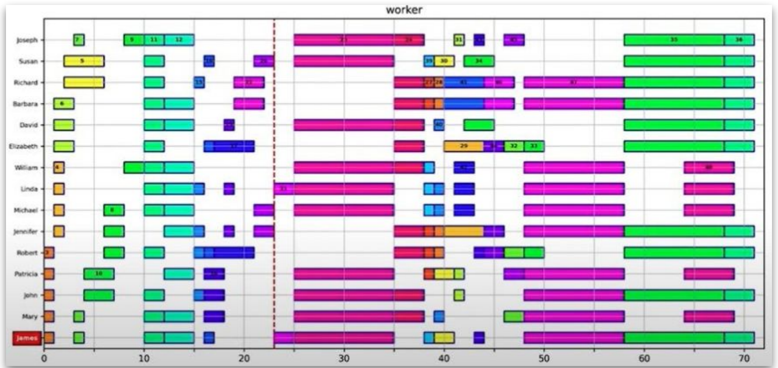
Manufacturing scheduling @Airbus

ANITI Université Fédérale Toulouse Midi-Pyrénées

Tuples TRUSTWORTHY AI

AIR Airbus AI Research

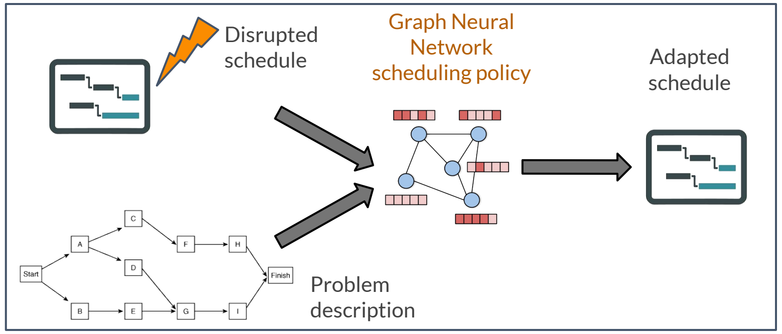
1 Cutting-edge algorithms to optimize the **nominal** manufacturing workflow



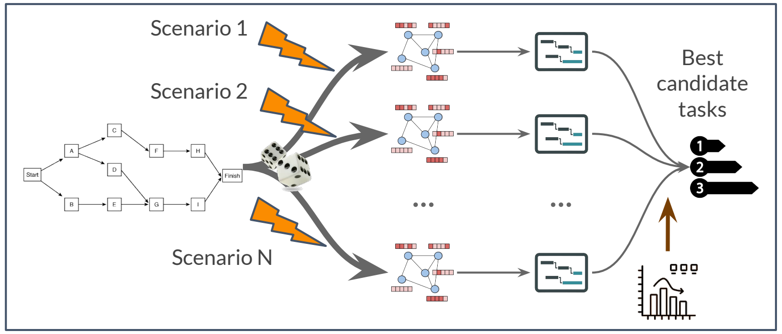
- Multi-skill worker allocation
- Versatile time constraint modeling between tasks
- Non-renewable and renewable resource sharing

2 Deep learning rescheduling strategies for fast **disruption** management & adaptation

- Pre-trained rescheduling policy
- Nearly instantaneous adaptation of current schedule to disruption events
- Possibility to improve adapted schedule as time allows



3 Massive uncertain multi-scenario analysis & **robust** decision-making



- Fast scenario evaluation and prediction by inferring the deep learning rescheduling policy
- Robust continuous scheduling based on real-time statistical analysis and aggregation of multiple scenarii and criteria

Part I	Part II

Part I
Large-Scale Scheduling with Limited Resources and Complex Relations in Practice

Scheduling in industry

Problems usually look like classical JSP/RCPSP but with additional constraints : preemption of tasks, calendar shifts and multiskill workforce.

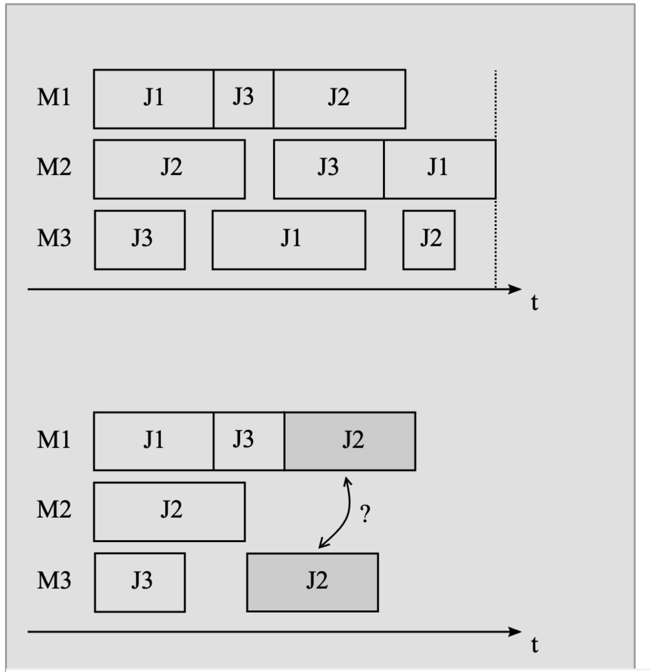
Borreguerro, Portoleau et al. 2021]

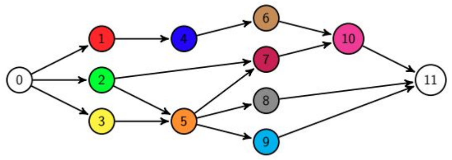
Job shop scheduling problem
-Unitary resource
-Precedence constraint between subpart of jobs

Flexible shop scheduling problem
-Job shop + flexibility of resource allocation for some jobs.

Resource constrained project scheduling problem (RCPSP)
- Job shop with “cumulative” resource (!=1), modeling workforce, tool, area, energy..

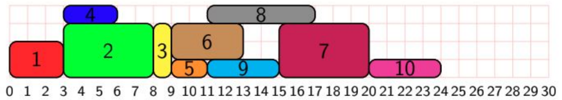
Multi-skill RCPSP
- Adding worker skills modelisation + skills requirements to each activities : increasing realism of real problems





$|R| = 1, B = 4, \mathcal{T} = [0, 30)$

i	p_i	b_i
1	3	2
2	5	3
3	1	3
4	3	1
5	2	1
6	4	2
7	5	3
8	6	1
9	4	1
10	4	1



Limitation of academical scheduling problems

Jobshop and RCPSP fail to model **(multi) skills individual workers**

1. **Preemption** of tasks is usually impossible in such optimisation models
2. **Generalized precedence constraints** are usually not considered in classical model

<u>Contributions :</u>

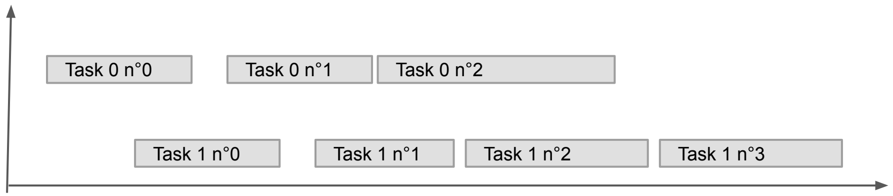
- Efficient constraint programming modeling
- Generic Large Neighborhood Search
- Capitalisation in open-source libraries

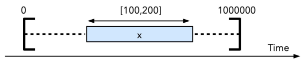
More :
Partially Preemptive Multi Skill/Mode Resource-constrained Project Scheduling with Generalized Precedence Relations and Calendars, CP2023, Povéda, Alvarez, Artigues

Constraint programming for complex scheduling

	1	2		3	4	5	6	7
	3	4	5		6	1	8	2
		1		5	8	2		6
		8	6					1
	2				7		5	
		3	7		5		2	8
	8			6		7		
2		7		8	3	6	1	5

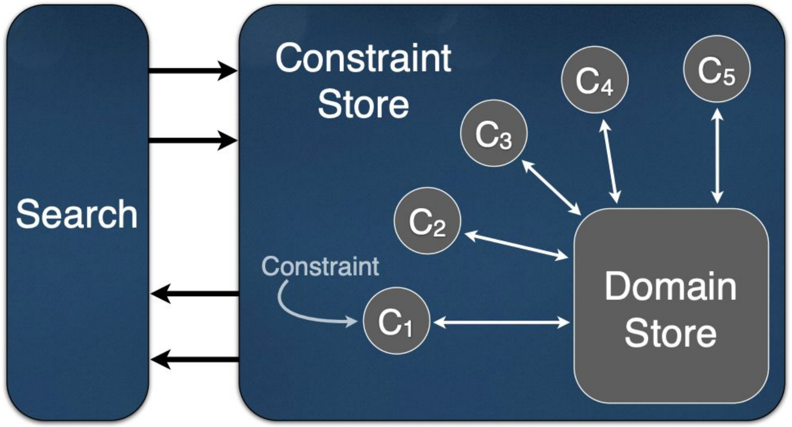
CPMODEL
array[1..9, 1..9] of var 1..9: grid;
forall i in row:
    allDifferent(grid[i, :]);
forall i in column:
    allDifferent(grid[:, i]);
…





CPMODEL
array[1..Nbtasks, 1..N] of var Interval: schedule;

forall i,j in Precedence:
    schedule[j,1].start >= schedule[i,N].end
….



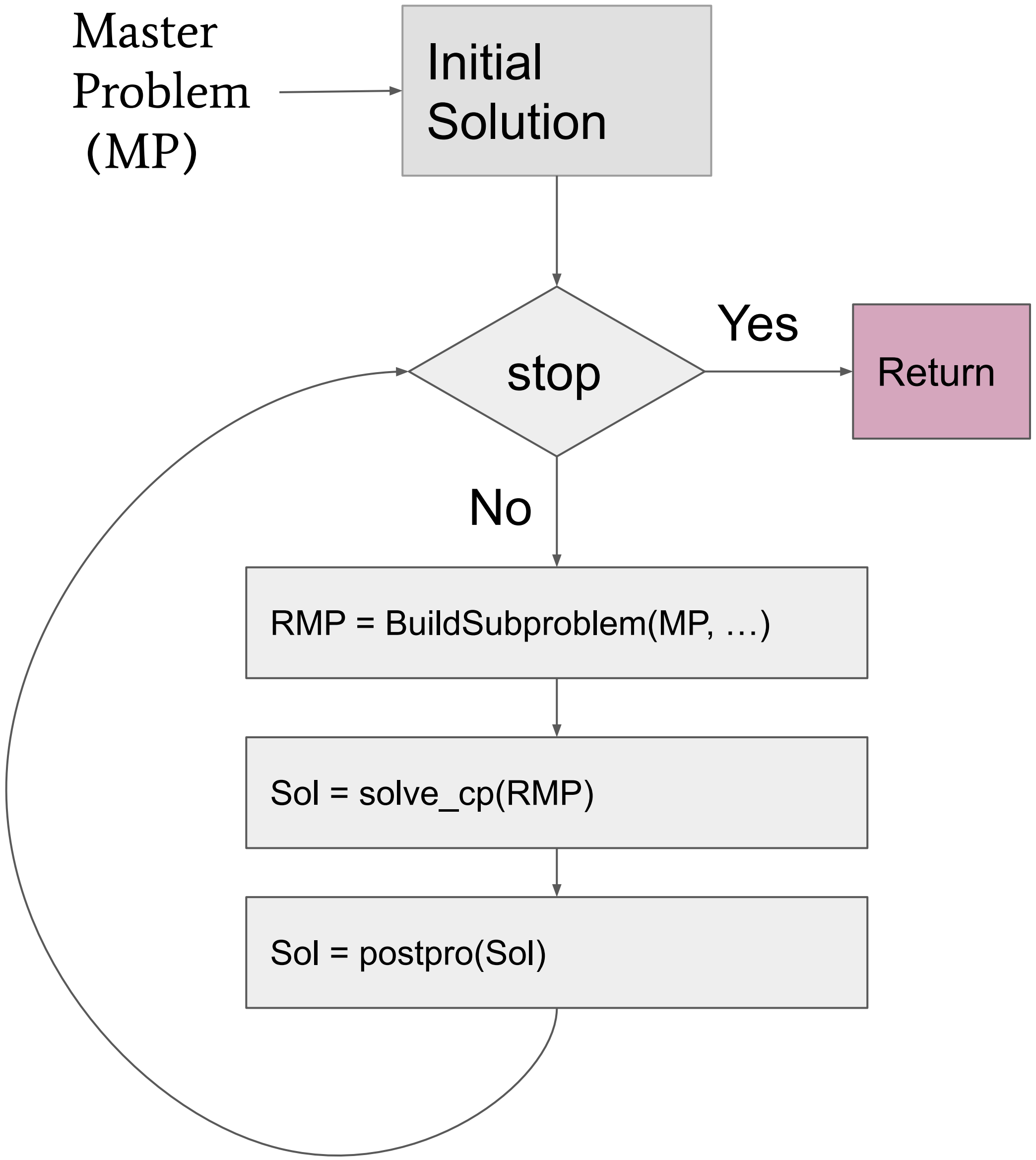
Some features of the developed CP model :
- Use of optional chain of interval variables for the preemption feature
- Softening hard generalized precedence constraints

Large neighborhood search

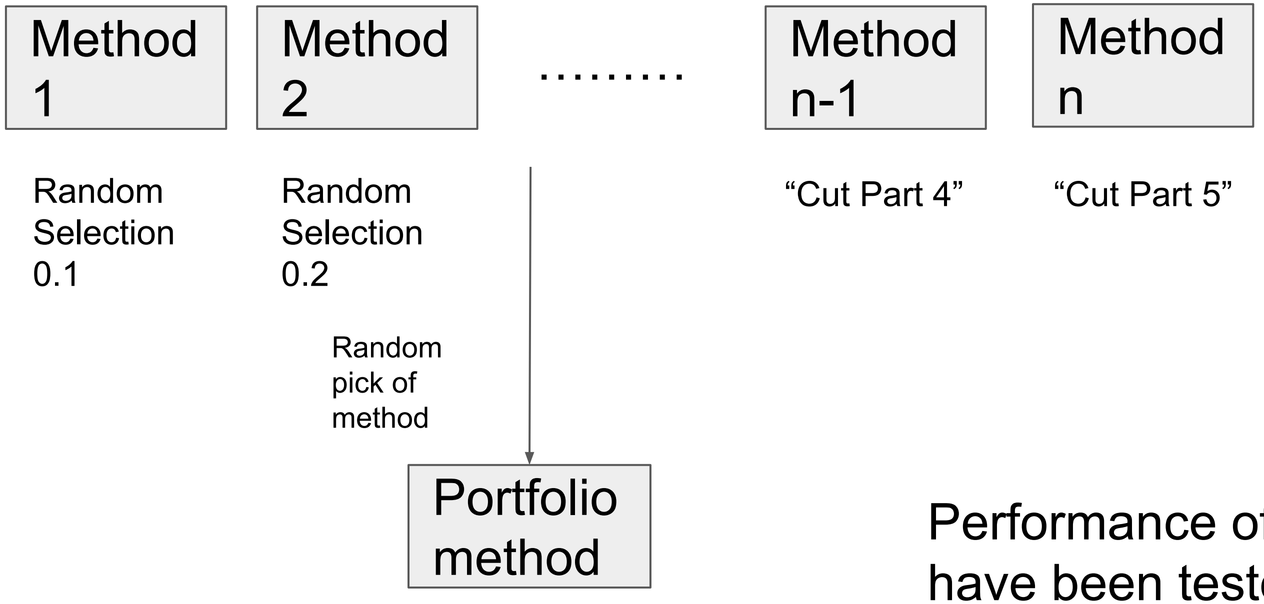
Algorithm 1 Generic Large Neighborhood Search Algorithm

Begin

```
Y* = ∞, X* = None
(X^0, Y^0) = (X*, Y*) = initial_solution(P)
iter = 0
repeat
  RMP = buildsubproblem(MP, X^iter)
  X^(iter+1), Y^(iter+1) = solve(RMP)
  if Y^(iter+1) ≤ Y* then
    X* ← X^(iter+1)
    Y* ← Y^(iter+1)
  iter ← iter + 1
until stop criterion is met
return X*, Y*
```



Portfolio evaluation



Method	j1201_1	j1201_2	j1201_3	j1201_4	j1201_5
RS(0.1)	116.6	131.3	138.6	106.5	131.3
RS(0.2)	112.2	129.8	136.8	106.0	128.3
RS(0.3)	110.4	128.4	134.3	105.4	120.4
RS(0.4)	108.9	118.0	132.6	103.3	116.7
Cut(2)	**105**	118.0	130	102.0	**113.0**
Cut(3)	107	118.0	135	**101.0**	118.0
Cut(4)	108	126.0	131	106.0	132.0
Cut(5)	111	128.0	136	106.0	130.0
Cut(6)	111	129.0	137	106.0	130.0
Mixing	106.5	**115.0**	**128.3**	102.0	**113.0**

Performance of LNS solver with different subproblem methods have been tested:
- random and cut methods taken individually
- portfolio of previous methods (called Mixing in the results table)

Mixing method achieved the most consistent performance (best or second best results) on our few testing instance

Capitalisation of optimisation models/solvers : open-source libraries

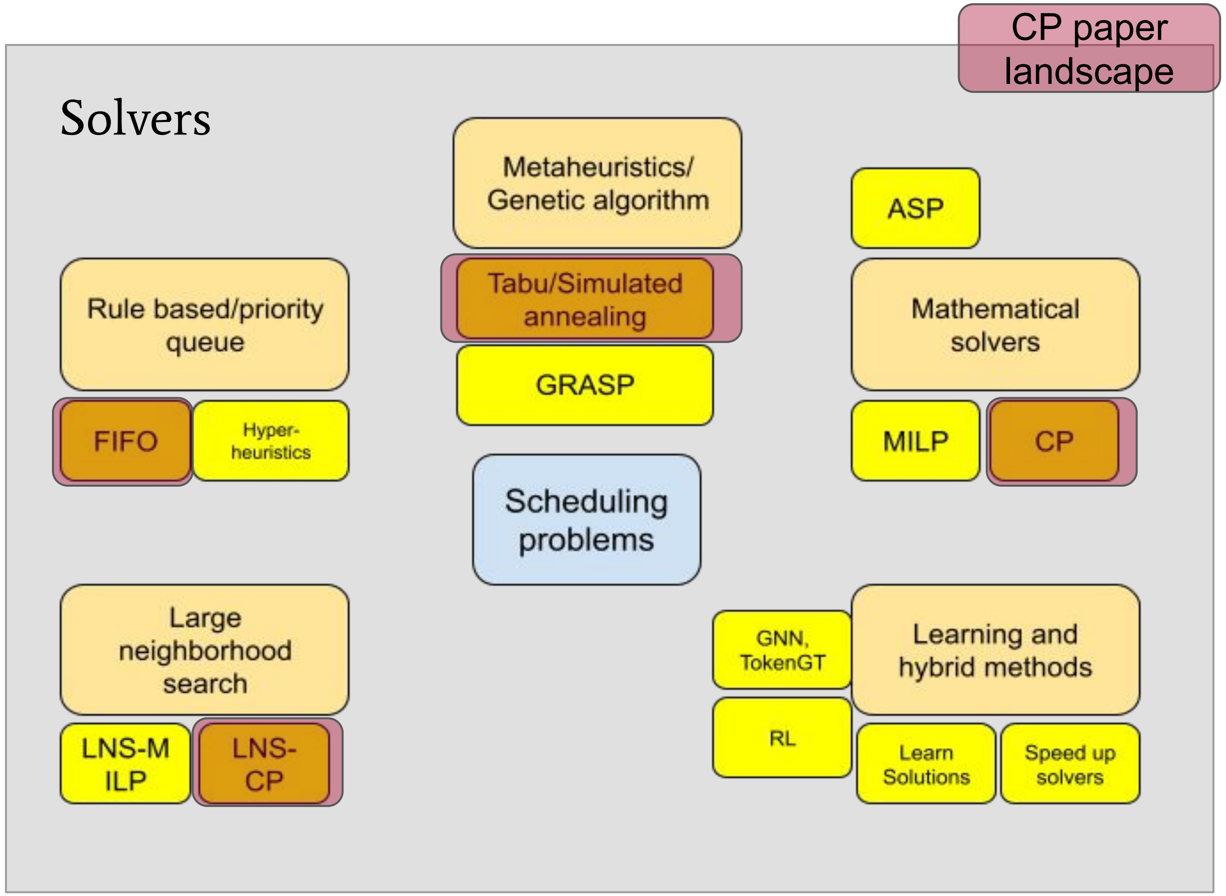
One library to **capitalize**/benchmark different solving methods for discrete optimisation problems.

https://github.com/airbus/discrete-optimization
https://github.com/airbus/scikit-decide

Easy example of use :
rcpsp_problem = parse(file)
results = solve(rcpsp_problem, solver=CPSolver)

Problem definition
- evaluate(solution)
- satisfy(solution)
- ...

Solve()



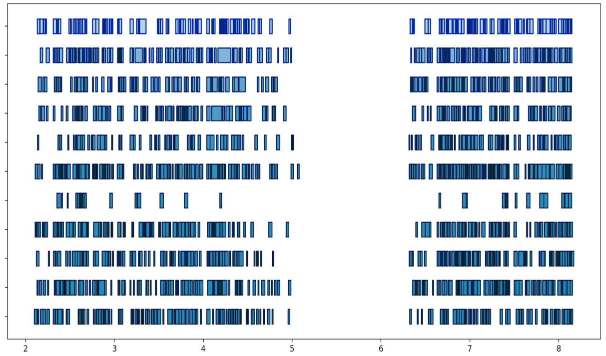
Now used in 3 publications around scheduling :

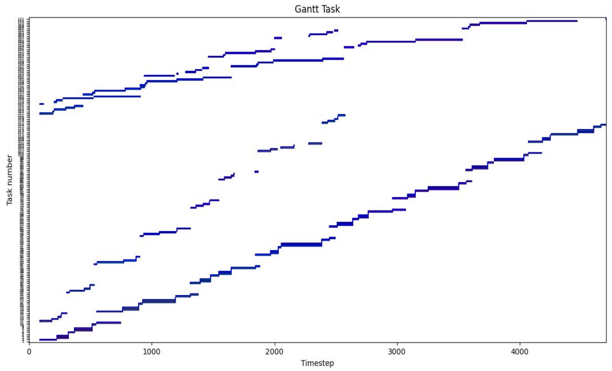
-"***An Empirical Evaluation of Permutation-Based Policies for Stochastic RCPSP***", Olivier Regnier-Coudert, Guillaume Povéda, GECCO 2021

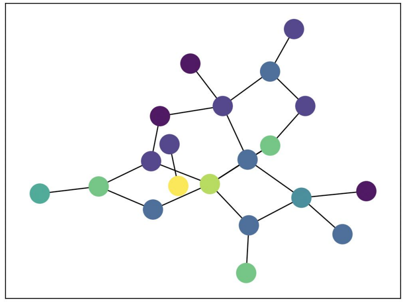
-"***Fast and Robust Resource-Constrained Scheduling with Graph Neural Networks***" Teichteil-Königsbuch, F., Povéda, G., González de Garibay Barba, G., Luchterhand, T., & Thiébaux, S., ICAPS 2023

- '**Partially Preemptive Multi Skill/Mode Resource-constrained Project Scheduling with Generalized Precedence Relations and Calendars**', Povéda, Alvarez, Artigues, CP2023,

Capitalisation of optimisation models/solvers : open-source libraries







<u>Main interest</u>:

1) Benchmark solvers on the same problem but from different communities (LP, CP, Metaheuristics, soon ML)
2) Combine easily solvers in some more complex pipeline (→ such as the LNS we describe)
3) Educational purpose for combinatorial optimization introduction

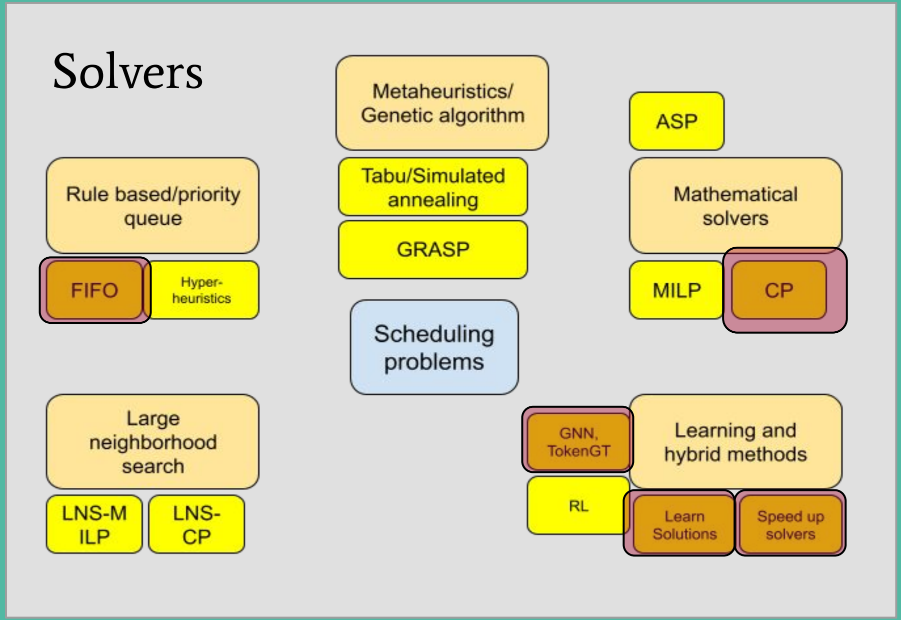
<u>Main problem implemented</u>:
Workforce allocation problems, routing, scheduling (JSP, RCPSP and variants..)..

<u>Example of solvers binded</u>:

https://github.com/airbus/discrete-optimization
https://github.com/airbus/scikit-decide

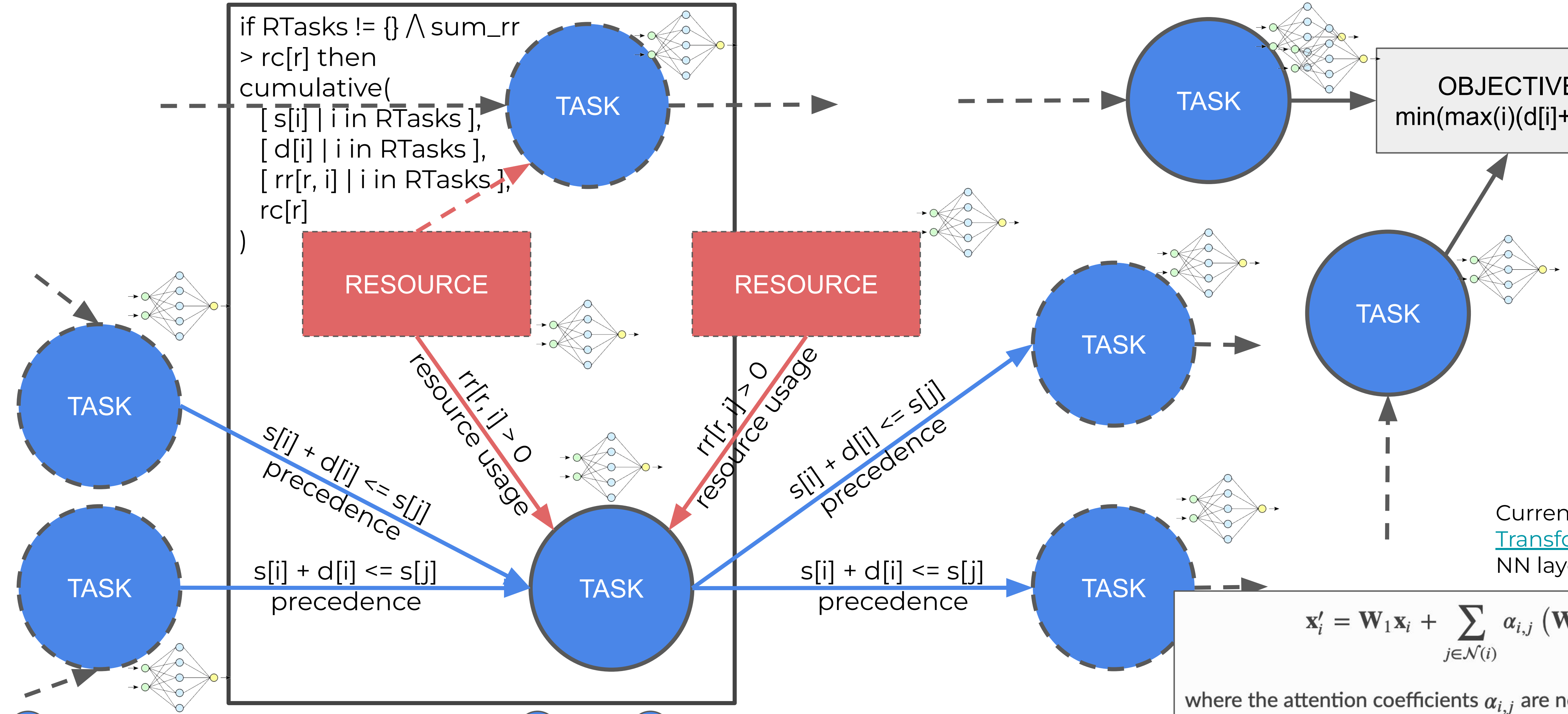
Part II
Frugal Learning of Deep Learning Scheduling Heuristics

With the help of model-based solvers



RCPSP represented as a Graph (Neural Network)

Cumulative (special case: non-overlapping)



Currently using TransformerConv as NN layers:

$$\mathbf{x}'_i = \mathbf{W}_1 \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} \left(\mathbf{W}_2 \mathbf{x}_j + \mathbf{W}_6 \mathbf{e}_{ij} \right),$$

where the attention coefficients $\alpha_{i,j}$ are now computed via:

$$\alpha_{i,j} = \mathrm{softmax} \left(\frac{(\mathbf{W}_3 \mathbf{x}_i)^\top (\mathbf{W}_4 \mathbf{x}_j + \mathbf{W}_6 \mathbf{e}_{ij})}{\sqrt{d}} \right)$$

Task node encoding: [0, 1, 0, duration]

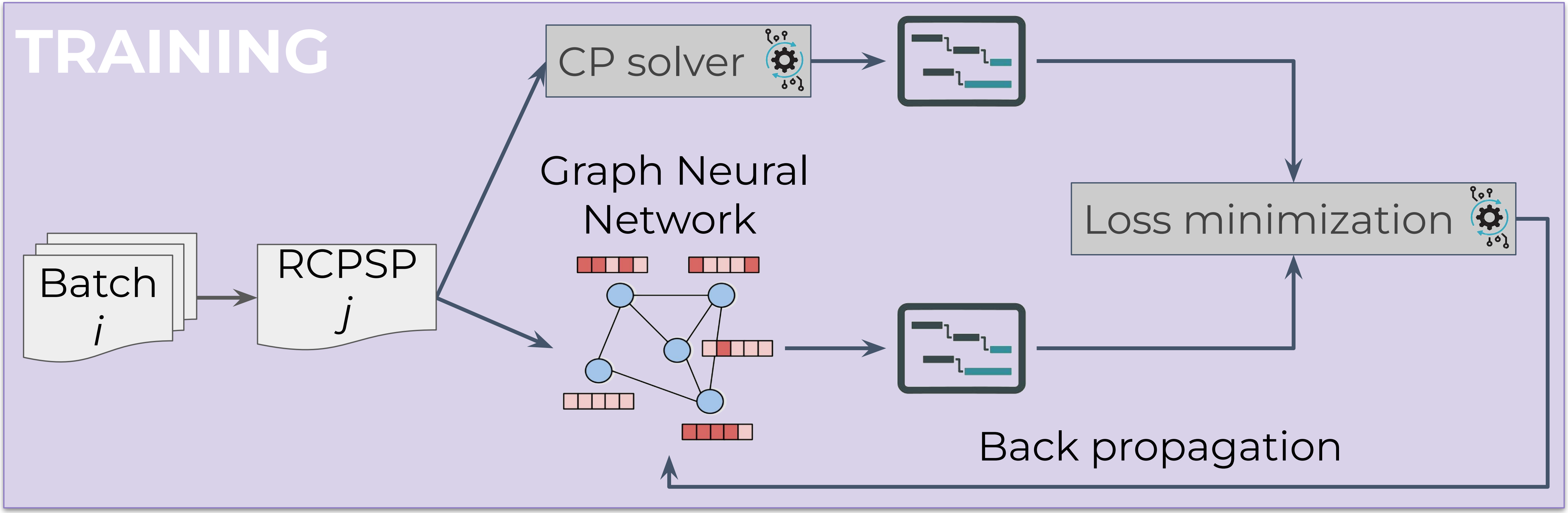
Resource node encoding: [1, 0, #resources, 0]

Precedence edge encoding: [1, 0, 0, 0, 0]

Resource edge encoding: [0, 1, 0, 0, #consumed]

Hybridizing CP+GNN: our SIREN training algorithm

(80% of 2040 RCPSP instances from PSPLIB)

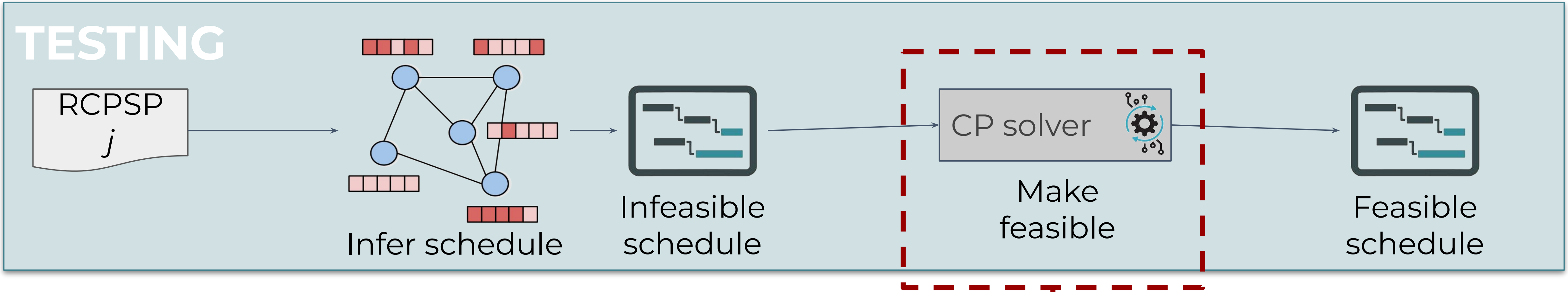


Don't learn to mimic the CP solver but learn to directly produce schedules with a Graph Neural Network structure specific to all RCPSP problems

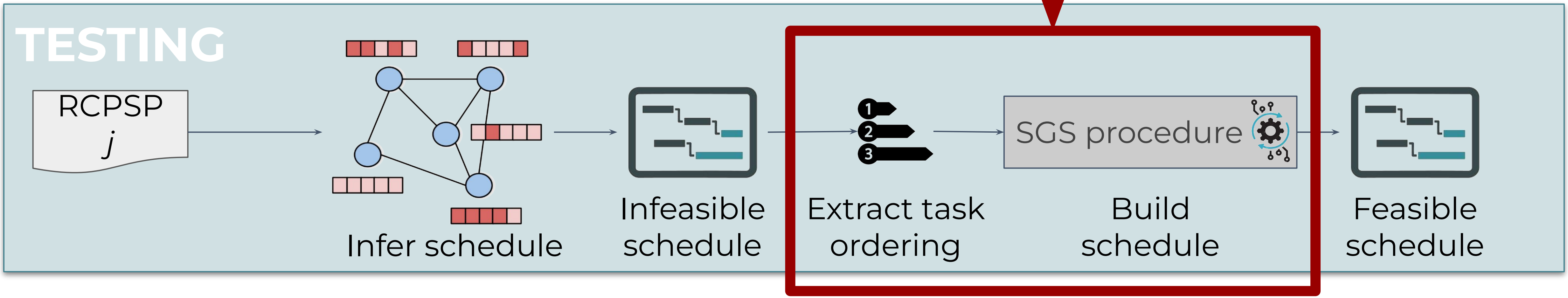
AIRBUS

Testing phase: our SIREN inference algorithm

Idea 1 (not working well)



Idea 2 (working well) SIREN

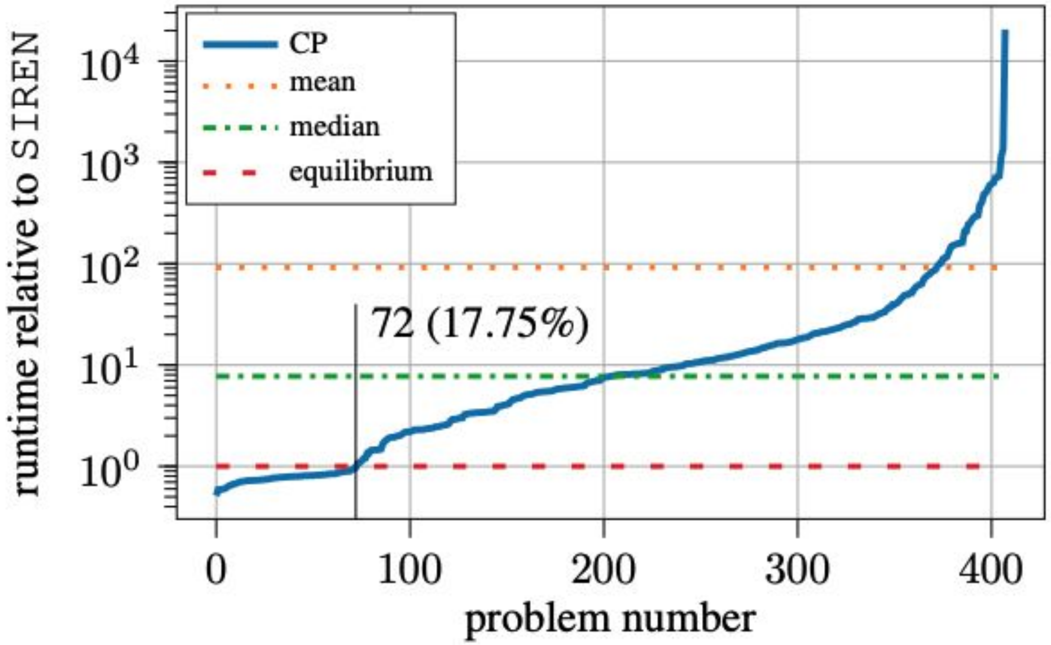


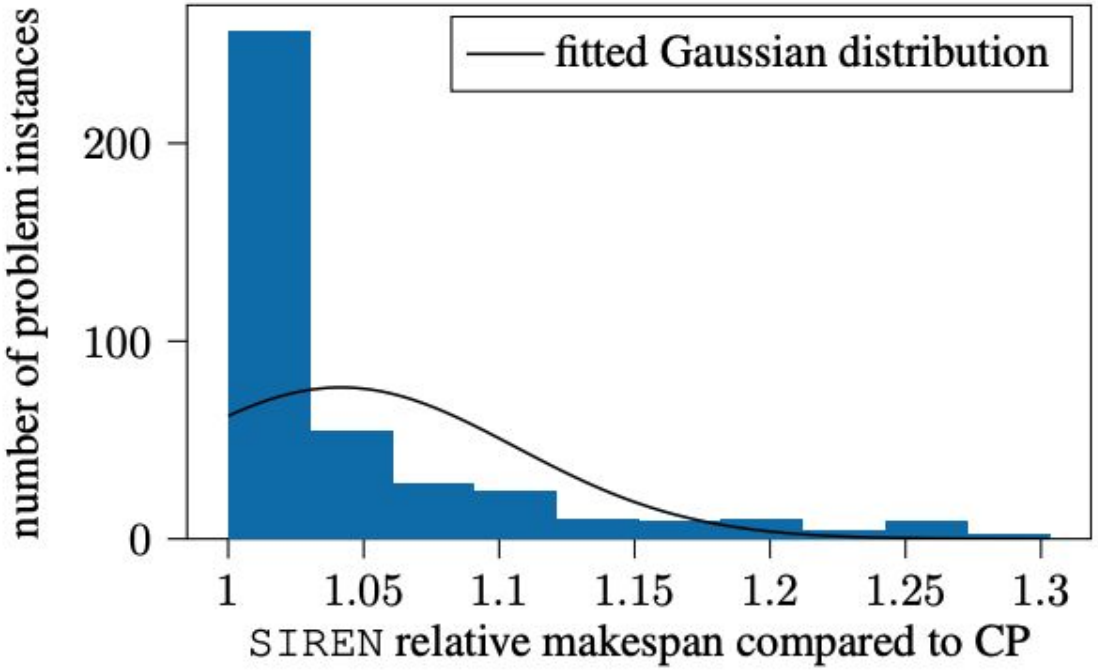
SGS is way faster than CP!

CP + GNN-SGS : testing statistics

(20% of 2040 RCPSP instances from PSPLIB)

Protocol: evaluate vanilla CP solver time to get same quality solution as GNN+SGS solver, then compare with GNN+SGS solver time





Using ResTransformer with 256 hidden neurons and 50000 epochs

- **In more than 82% of problems CP-SAT takes more time than SIREN to achieve a solution of comparable quality.**
- **In over 40% of the cases, CP-SAT's computational overhead ranges from 10 times up to over 20,000 times the computation time of SIREN.**

AIRBUS

CP + GNN-SGS (SIREN) vs custom ordering heuristics

(20% of 2040 RCPSP instances from PSPLIB)

3 heuristics: DUM, MDPR, CCPM are all using SGS with a different task ordering

- DUM : [1, 2 … N] : order by index of task
- MDPR : Order by maximum of descendants in the precedence graph.
- CCPM : Order using critical path method outputs.

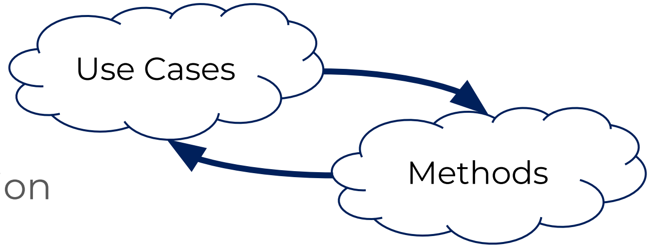
	relative % overcost compared to best CP solution					
algorithm	mean	std	25%	50%	75%	max
DUM	12.07	10.12	0.0	11.81	20.46	36.79
MDPR	7.72	7.22	0.0	6.59	12.70	36.29
CCPM	6.21	7.34	0.0	2.21	11.87	30.95
Ours SIREN	4.16	6.45	0.0	0.93	5.73	30.32

is systematically better

Table 2: Statistics of relative overcost compared to best CP solutions on the 408 test instances. (% are for percentiles)

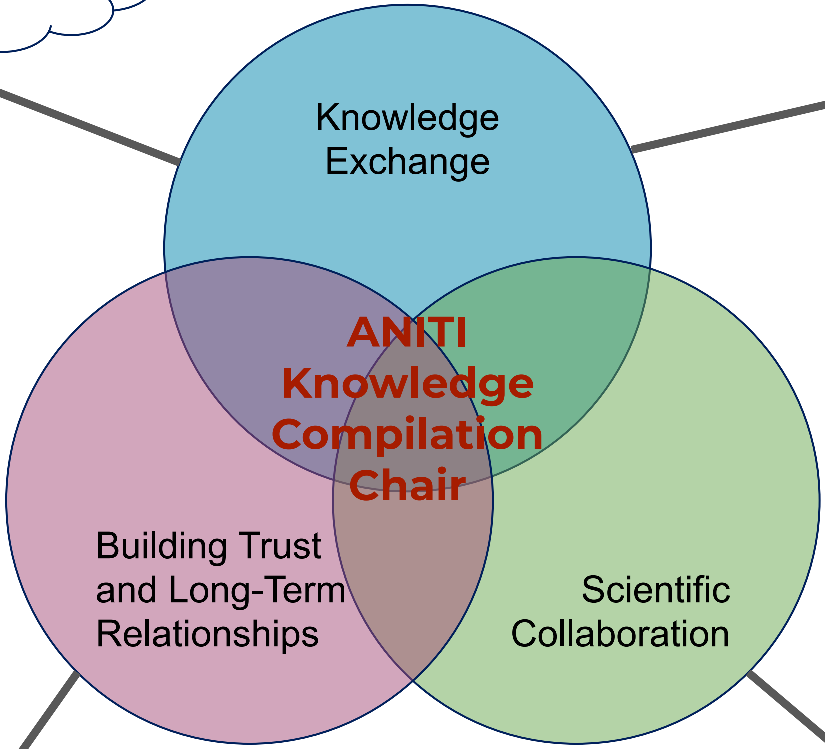
Using ResTransformer with 256 hidden neurons and 50000 epochs

Final words: thank you ANITI-1.0!



Cross-Fertilization

Discussions
➔ **Seminars**
➔ **Social activities** ☕



Get-To-Know and to work together 🙌
➔ **TUPLES** project
➔ **ANITI-2 HEROIC** chair proposal

Innovative methods for solving scheduling problems inspired by airbus manufacturing applications
➔ **ICAPS-23 paper:** Hybrid DL/CP
➔ **CP-23 paper:** LNS/CP