Learning Precedences for Scheduling Problems with Graph Neural Networks

Hélène Verhaeghe[1], Quentin Cappart[2], Gilles Pesant[2], Claude-Guy Quimper[3]

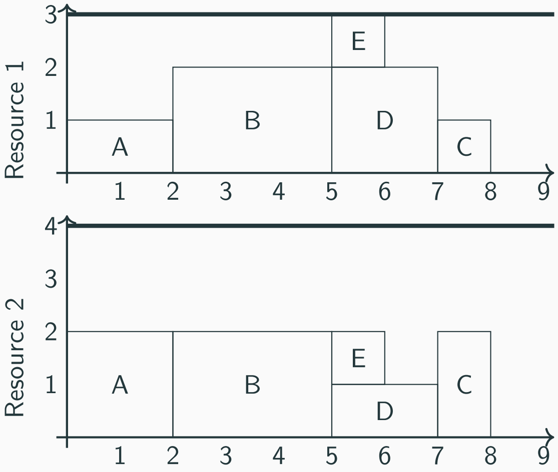
4 September 2024

[1] UCLouvain, Louvain-la-Neuve, Belgium, *helene.verhaeghe@uclouvain.be*
[1] KULeuven, Leuven, Belgium
[2] Polytechnique Montréal, Montréal, Canada
[3] Université Laval, Quebec, Canada

Resource-Constrained Project Scheduling Problem (RCPSP)

Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C

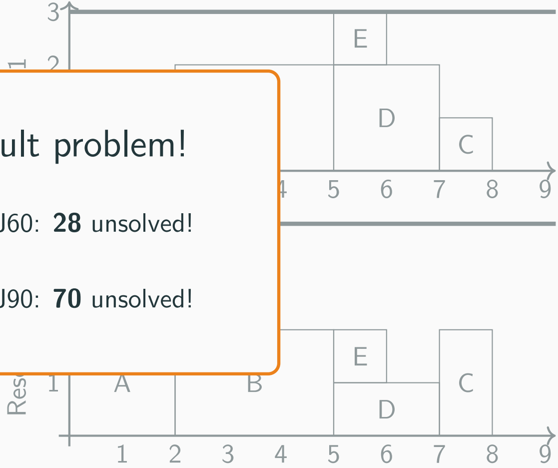
$C_{r_1} = 3$ and $C_{r_2} = 4$



Resource-Constrained Project Scheduling Problem (RCPSP)

Task	p_i	c_{ir_1}	c_{ir_2}
A	2	1	2
B	3	2	2
C	1	1	2
D	2	2	1
E	1	1	1

$C_{r_1} = 3$ and $C_{r_2} = 4$



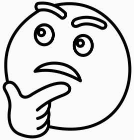
Difficult problem!

PSPlib J60: **28** unsolved!

PSPlib J90: **70** unsolved!

What if?

What if we had more information about the solution?



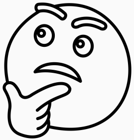
KU LEUVEN

What if we had more information about the solution?

- Guide toward solution
- Reduce search space

How?

How do I get more information about the solution?



How?

How do I get more information about the solution?

How about Machine Learning?
Can we learn to predict information?

How do I get more information about the solution?

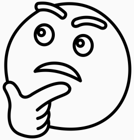
How about Machine Learning?

Can we learn to predict information?

Which ML tool?

KU LEUVEN

Which information could be useful?



Which information?

KU LEUVEN

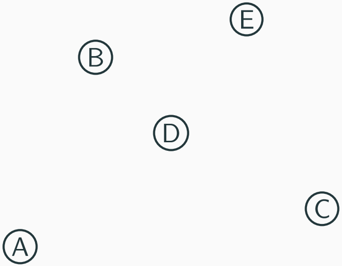
Which information could be useful?

Precedences between pairs of tasks

RCPSP as a graph: the precedence graph

Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C

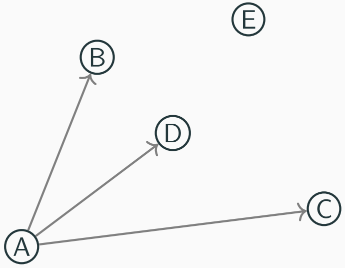
$C_{r_1} = 3$ and $C_{r_2} = 4$



RCPSP as a graph: the precedence graph

Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C

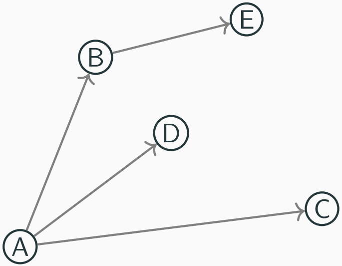
$C_{r_1} = 3$ and $C_{r_2} = 4$



RCPSP as a graph: the precedence graph

Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C

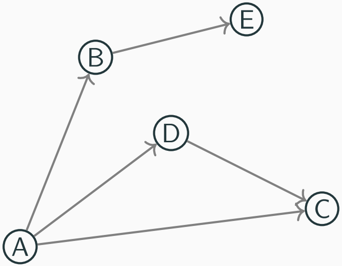
$C_{r_1} = 3$ and $C_{r_2} = 4$



RCPSP as a graph: the precedence graph

Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C

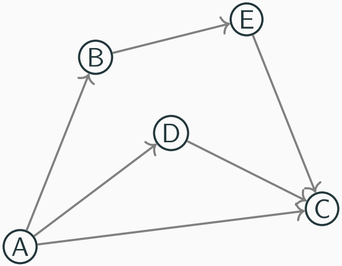
$C_{r_1} = 3$ and $C_{r_2} = 4$



RCPSP as a graph: the precedence graph

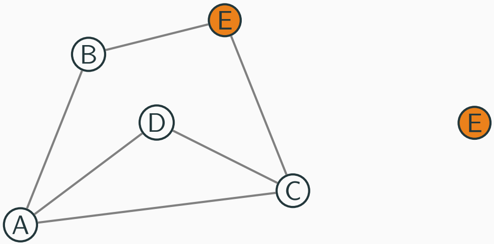
Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C

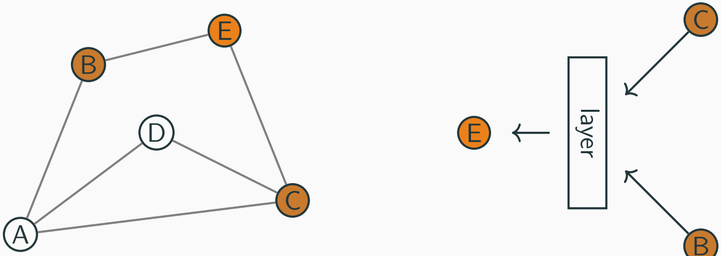
$C_{r_1} = 3$ and $C_{r_2} = 4$



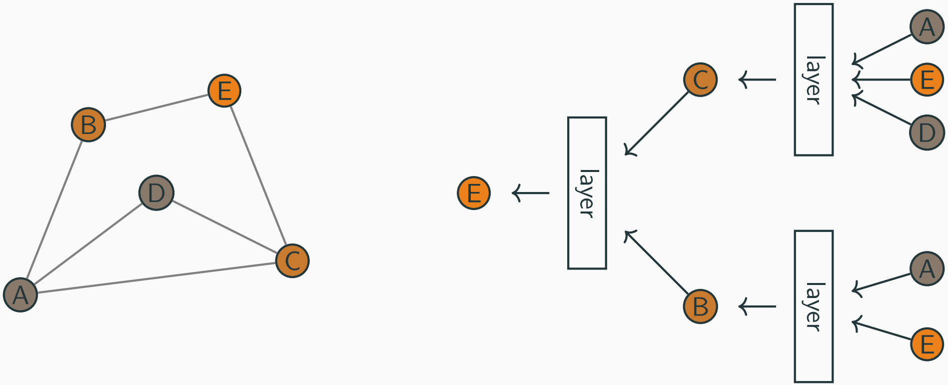
Graph Neural Networks

Graph Neural Networks: Embeddings

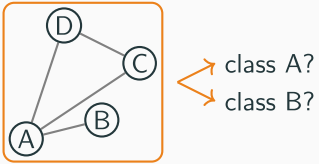



E
B
D
C
A
C
layer
E
B

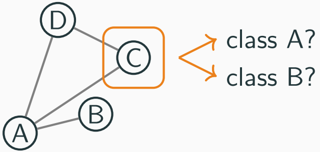
Graph Neural Networks: Embeddings



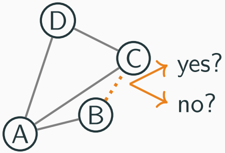
Graph Neural Networks: Tasks



Graph classification

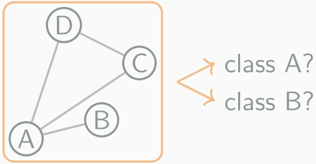


Node classification

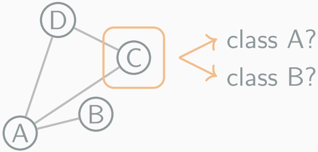


Link prediction

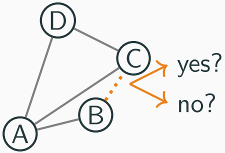
Graph Neural Networks: Tasks



Graph classification



Node classification

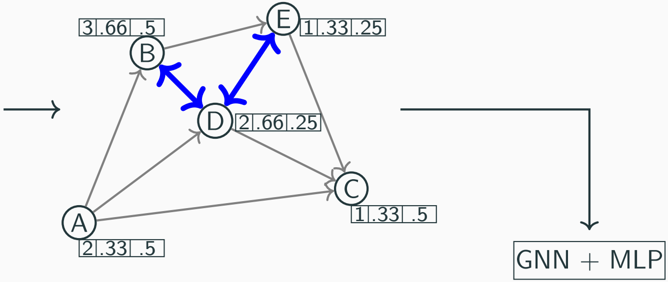


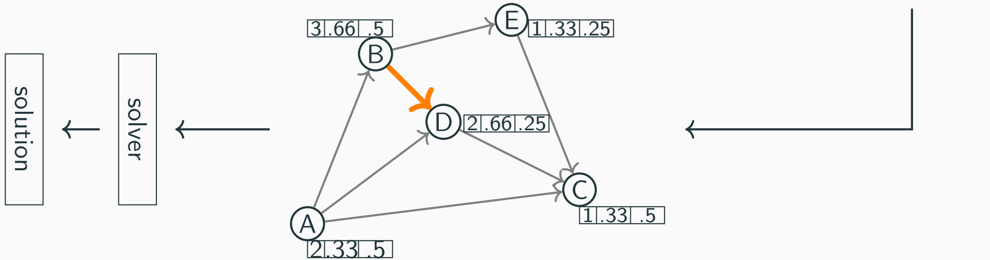
Link prediction

Methodology

Our method

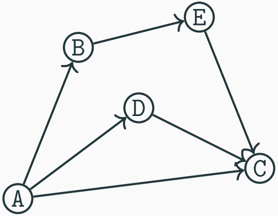
Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C



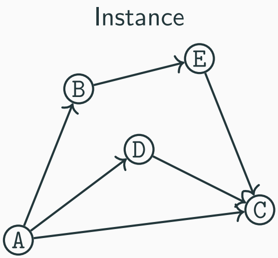


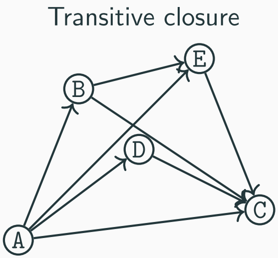
Candidate edge

Instance



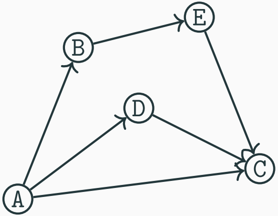
Candidate edge



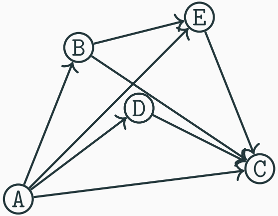


Candidate edge

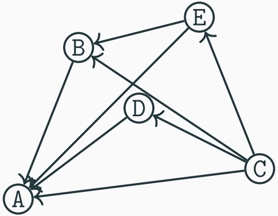
Instance



Transitive closure

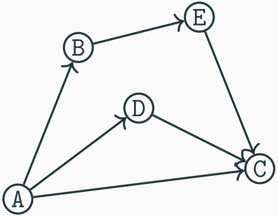


Avoided

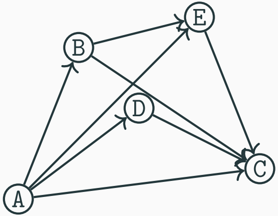


Candidate edge

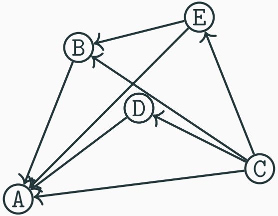
Instance



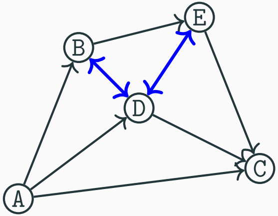
Transitive closure

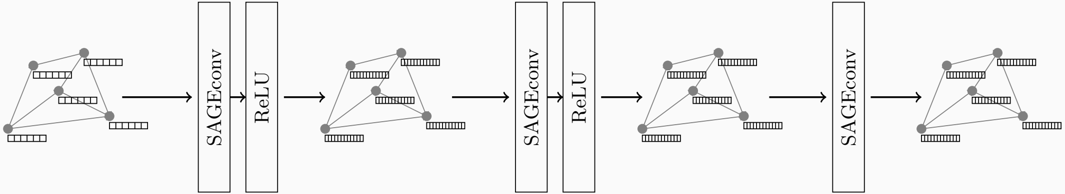


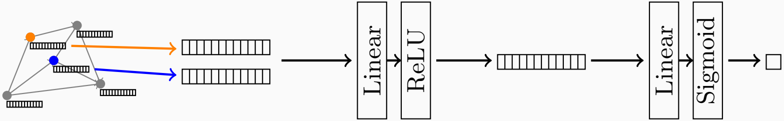
Avoided



Candidate




SAGEconv
ReLU
SAGEconv
ReLU
SAGEconv


Linear
ReLU
Linear
Sigmoid

Use of prediction

Additional precedences

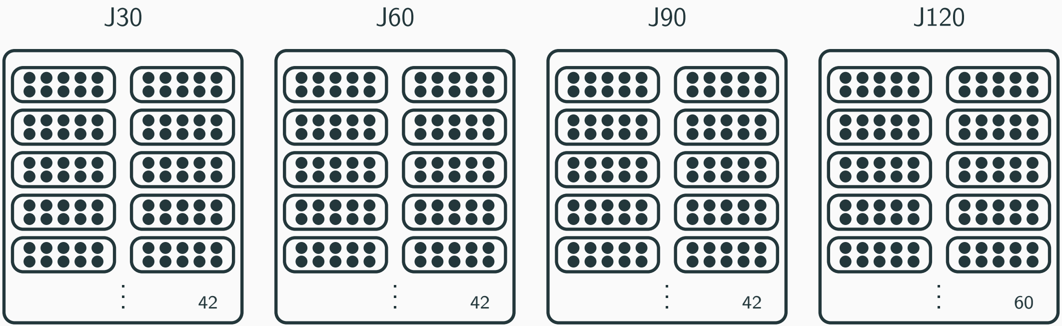
+ reduces search space
- restriction of the problem

Ordering heuristic

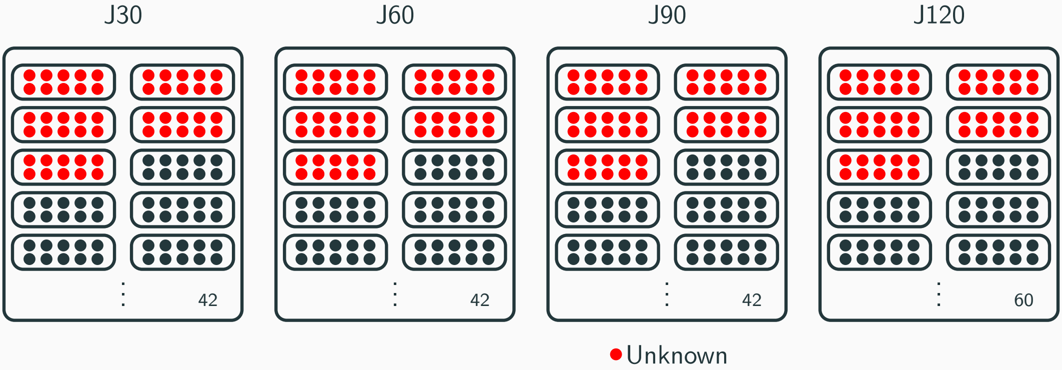
+ preserve solutions
- static ordering are slower

Results

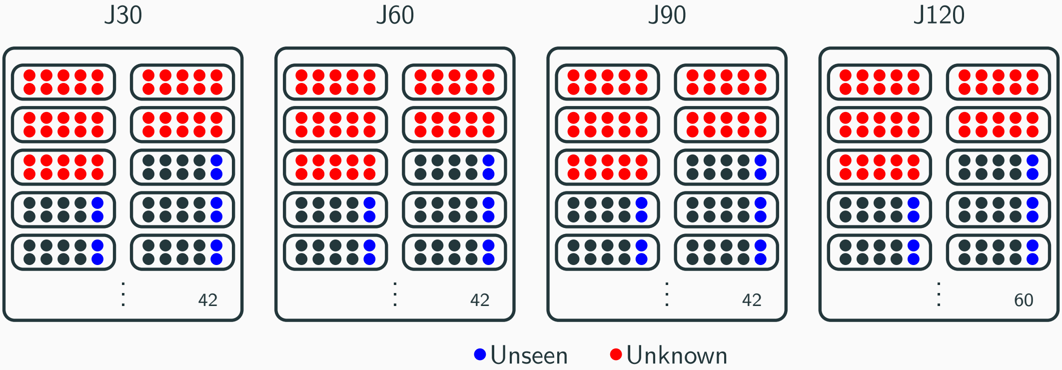
Benchmark - PSPlib



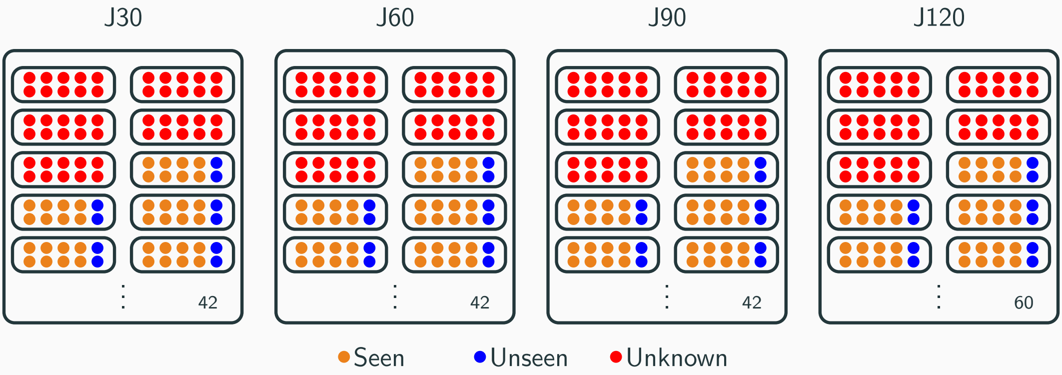
Benchmark - PSPlib



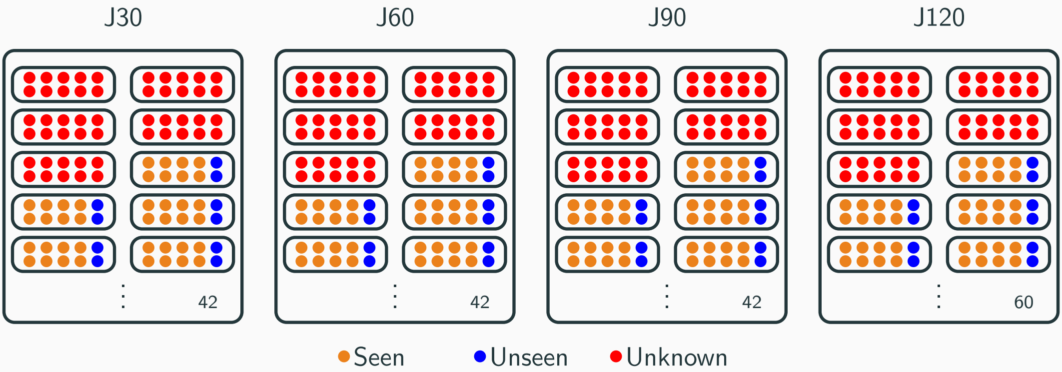
Benchmark - PSPlib



Benchmark - PSPlib



Benchmark - PSPlib



- solutions at 1h time out
- 10-fold cross validation
- chuffed solver

- two CP models:
  - one with early task first heuristic
  - one with sbps/vsids search heuristic

Quality of the training set

	Model A ("*without*")				Model B ("*with*")			
	f1	*prec*	*rec*	*tn*	*f1*	*prec*	*rec*	*tn*
SEEN_{J120}	0.79	0.89	0.71	0.91	0.71	0.82	0.62	0.87
UNSEEN_{J120}	0.78	0.89	0.70	0.92	0.71	0.83	0.62	0.87
UNKNOWN_{J120}	0.80	0.90	0.72	0.92	0.72	0.84	0.64	0.87
ALL_{J120}	0.79	0.89	0.71	0.91	0.71	0.83	0.62	0.87

	Predictions used as constraints				Predictions used for ordering			
	to=1s	to=1m	to=10m	to=1h	to=1s	to=1m	to=10m	to=1h
Model A	411/163	108/454	24/533	28/526	413/45	172/39	6/25	1/13
Model B	**419/152**	**125/386**	**26/469**	**34/459**	**433/26**	**179/25**	**8/17**	**2/8**

Setting: Training on ≤J120 instances generated with or without vsids/sbps; evaluation on J120 instances, with vsids/sbps

Quality of the training set

	Model A ("*without*")				Model B ("*with*")			
	f1	*prec*	*rec*	*tn*	*f1*	*prec*	*rec*	*tn*
SEEN_{J120}	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	…62	0.87
UNSEEN_{J120}	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	…62	0.87
UNKNOWN_{J120}	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	…64	0.87
ALL_{J120}	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	…62	0.87

Takeaway: Use best training set

Worse training metrics

Better use metrics

	t…	[illegible]	…ed for ordering	
			to=10m	to=1h
Model A	41…	[illegible]	6/25	1/13
Model B	**41…**	[illegible]	**8/17**	**2/8**

Setting: Training on $\leq$J120 instances generated with or without vsids/sbps; evaluation on J120 instances, with vsids/sbps

Generalizing - models trained with smaller instances

	f1	*prec*	*rec*	*tn*
SEEN_{J120}	0.72	0.82	0.64	0.86
UNSEEN_{J120}	0.72	0.83	0.64	0.87
UNKNOWN_{J120}	0.73	0.83	0.65	0.87
ALL_{J120}	0.72	0.83	0.64	0.87

	Predictions used as constraints				Predictions used for ordering			
	1s	1m	10m	1h	1s	1m	10m	1h
≤J120 train	419/152	**125/386**	26/**469**	34/**459**	**433**/26	179/25	**8**/17	**2**/8
≤J60 train	**421/141**	119/398	**27**/484	**35**/481	430/**17**	**187/18**	6/**10**	1/**4**

Setting: Training on ≤J60 instances; evaluation on J120 instances, with vsids/sbps

Generalizing - models trained with smaller instances

	f1	*prec*	*rec*	*tn*
SEEN_{J120}	0.72	0.82	0.64	0.86

Takeaway: very good generalization

Equivalent training metrics

Equivalent use metrics

	1s	…	sed for ordering 10m	1h
$\leq$J120 train	419/1	…	**8**/17	**2**/8
$\leq$J60 train	**421/1**	…	6/**10**	1/**4**

Setting: Training on $\leq$J60 instances; evaluation on J120 instances, with vsids/sbps

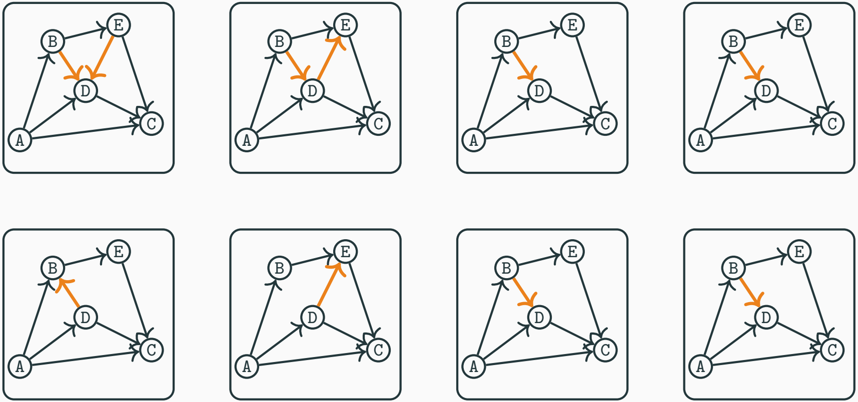
Aggregating solutions

Sol 1

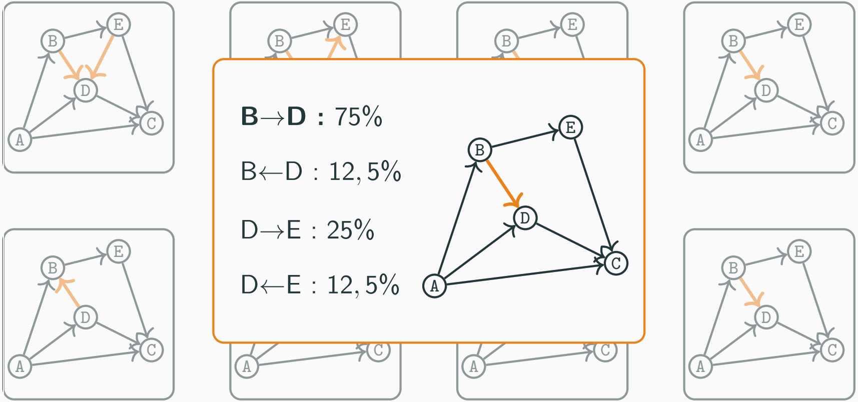
Aggregating solutions

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

Aggregating solutions



Aggregating solutions



B→D : 75%

B←D : 12, 5%

D→E : 25%

D←E : 12, 5%

Aggregating solutions

KU LEUVEN

	f1	prec	rec	tn
ALL_{J30}	0.67	0.80	0.57	0.86
ALL_{J60}	0.68	0.79	0.59	0.84
ALL_{J90}	0.67	0.78	0.59	0.83
ALL_{J120}	0.72	0.83	0.64	0.87

	Predictions used as constraints				Predictions used for ordering			
	1s	1m	10m	1h	1s	1m	10m	1h
1-sol	421/141	119/398	27/484	35/481	430/17	187/18	6/10	**1**/4
Aggregate	**436/108**	**154/332**	**46/427**	**48/424**	**451/0**	**193/1**	**7/2**	0/**2**

Setting: Training on $\leq$J60 instances, with aggregate of up to 100 solutions (70% threshold); evaluation on J120 instances, with vsids/sbps

Aggregating solutions

	f1	prec	rec	tn
ALL_{J30}	0.67	0.80	0.57	0.86
ALL_{J60}	0.68	0.79	0.59	0.84

Takeaway: More useful precedences

Equivalent training metrics

Improved use metrics

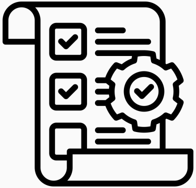
		ed for ordering	
	1s	10m	1h
1-sol	421/14	6/10	**1**/4
Aggregate	**436/10**	**7/2**	0/**2**

Setting: Training on ≤J60 instances, with aggregate of up to 100 solutions (70% threshold); evaluation on J120 instances, with vsids/sbps

Conclusion

Conclusion

KU LEUVEN



- Solver independent solution
- Good learning capabilities
- Very good generalization, allowing easier dataset generation
- Noise reduction when training on aggregate solutions
- Better earlier solutions

Going further

- Other RCPSPs benchmarks
- Other generalization aspects (# resources,...)
- Updating prediction during the search
- Adapt on other problems with underlying graphs

Thank you for listening!

Any questions?

`https://hverhaeghe.bitbucket.io/`