The Best of Both Worlds

When Constraint Programming and Machine Learning Help Each Other

Hélène Verhaeghe

13 Mars 2025

ICTEAM, UCLouvain, helene.verhaeghe@uclouvain.be

Model + Search

- Goal: Find (optimal) solution wrt some constraints
- Pro: Exact method
- Con: Difficulties in dealing with huge inputs

(Big) Data + algorithms

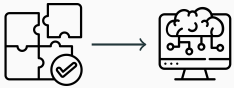
- Goal: Learn from examples
- Pro: Good with huge quantities of data
- Con: Difficulties to satisfy (hard) constraints in outputs

Can we get the best of both worlds?

Yes, by combining them!

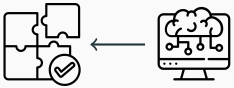
How to combine them?

UCLouvain



CP for ML

- Modeling ML problems
  (e.g., clustering using CP)
- Joint inference on NN output
  (e.g., visual sudoku problem)
- Improving the learning of NN
  (e.g., PLS experiment)

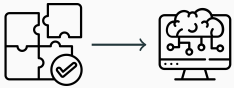


ML for CP

- Algorithm configuration
  (e.g., Sunny-CP solver)
- Learning to branch
  (e.g., SeaPearl project)
- Constraint acquisition
  (e.g., ConAcq approach)

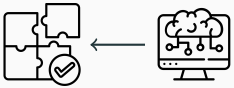
And many many other examples ...

UCLouvain



CP for ML

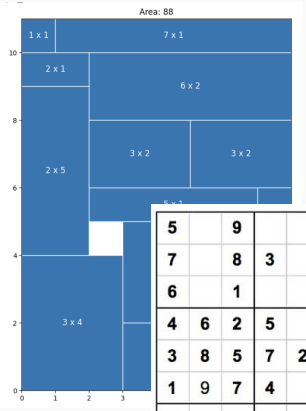
- Optimal decision trees



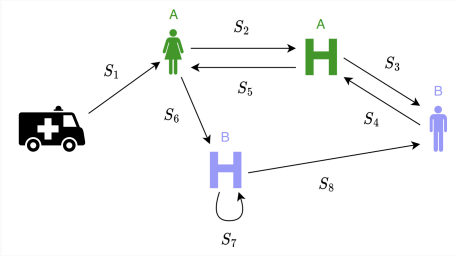
ML for CP

- Solving RCPSP using GNNs

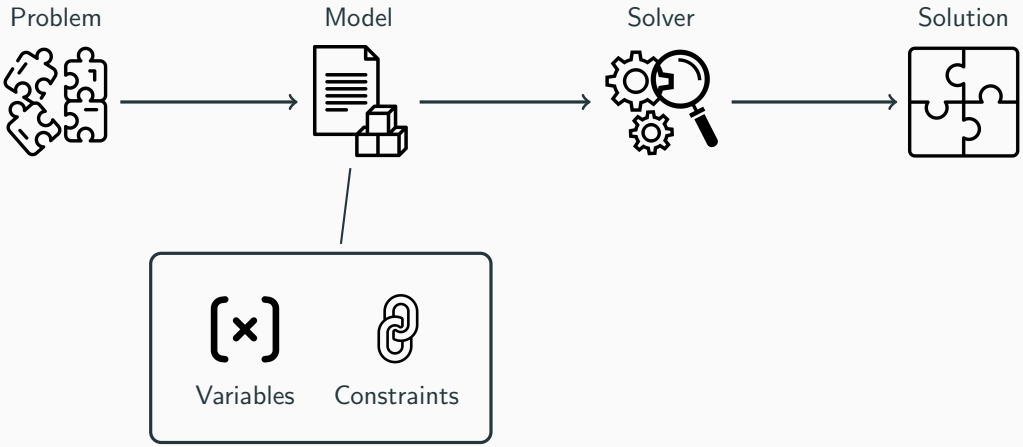
Constraint Programming

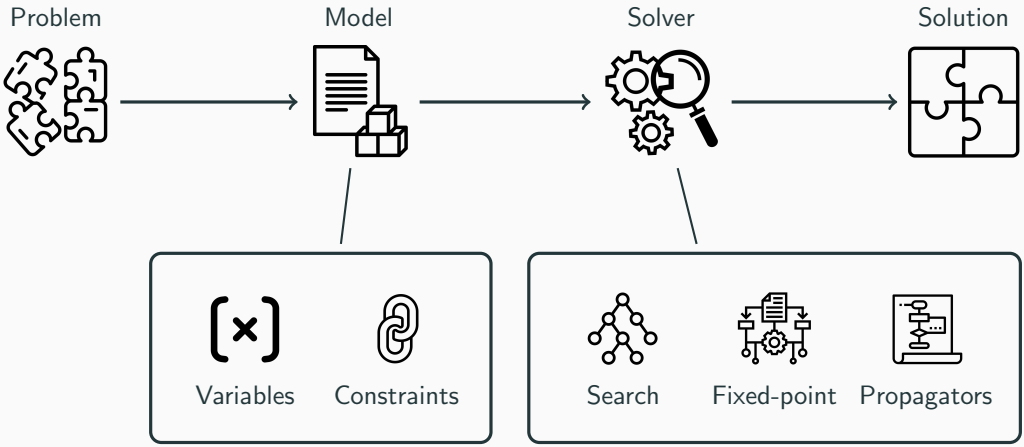


5		9				4		
7		8	3		4	9		
6		1				7	3	
4	6	2	5					
3	8	5	7	2	1	6	4	9
1	9	7	4		8	2		
2			1					4
		3		4			8	7
	7			5	3			6




Problem
Model
Solver
Solution


Problem
Model
Solver
Solution
Variables
Constraints


Problem
Model
Solver
Solution
Variables
Constraints
Search
Fixed-point
Propagators

UCLouvain

X $D(X)$

Goal of variables
Represent unknowns of the problem

UCLouvain

X	D(X)
integer	{...,-2,-1,0,1,2,...}
boolean	{true,false}

Goal of variables
Represent unknowns of the problem

UCLouvain

X	D(X)
integer	{...,-2,-1,0,1,2,...}
boolean	{true,false}
set	{{1,2,3},{1,2},{3,4},...}

Goal of variables
Represent unknowns of the problem

UCLouvain

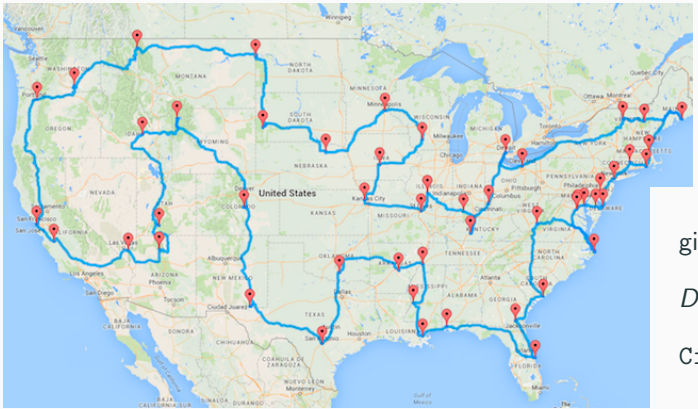
X D(X)

graph {  }

Goal of variables
Represent unknowns of the problem

Reference: **"CP(Graph): Introducing a Graph Computation Domain in Constraint Programming"**, by G. Dooms, Y. Deville, and P. Dupont, CP2005

Variable



Goal of global constraints
Capture a relation between a non-fixed number of variables

given $NY = 0$, $SF = 1$,...

$D(succ_i) = \{0, 1, ..., N-1\} \setminus \{i\}$

$\texttt{Circuit}([succ_{NY}, succ_{SF}, ...])$

Model

Goal of global constraints

Capture a relation between a non-fixed number of variables

given $NY = 0$, $SF = 1$,...

$D(succ_i) = \{0, 1, ..., N-1\} \setminus \{i\}$

$\texttt{Circuit}([succ_{NY}, succ_{SF}, ...])$

min $\sum$ dist$(i, succ_i)$

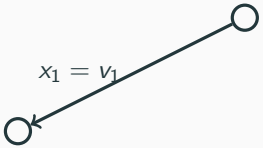
Model

Goal of the search
Explore the solution space

Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006

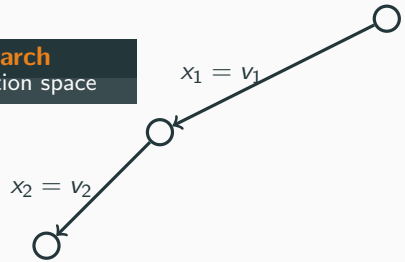
UCLouvain

Goal of the search
Explore the solution space



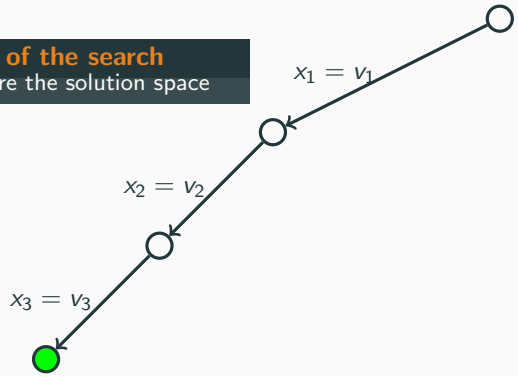
Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006

Goal of the search
Explore the solution space



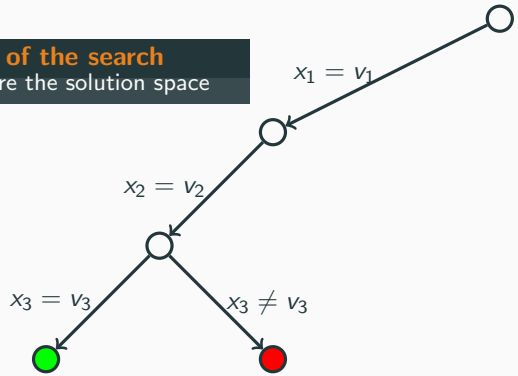
Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006

Goal of the search
Explore the solution space



Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006

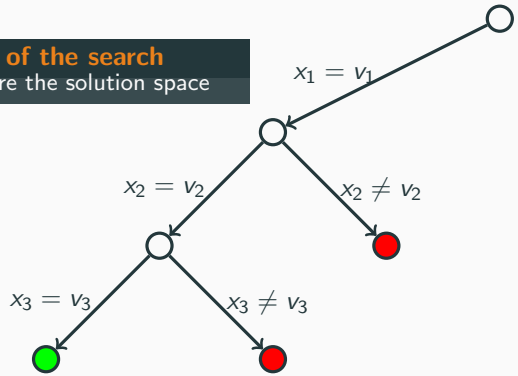
Goal of the search
Explore the solution space



Search

Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006

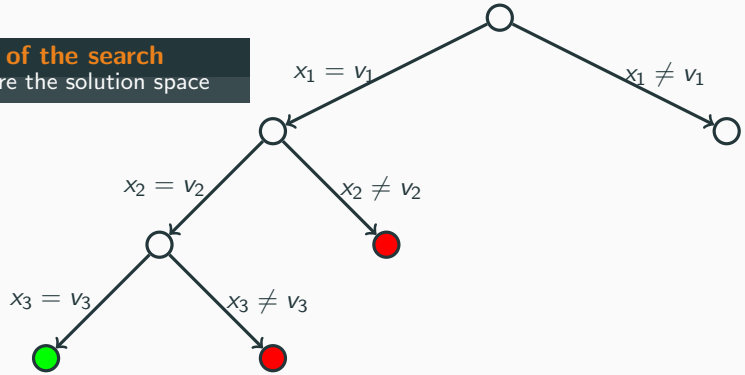
Goal of the search
Explore the solution space



Search

Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006

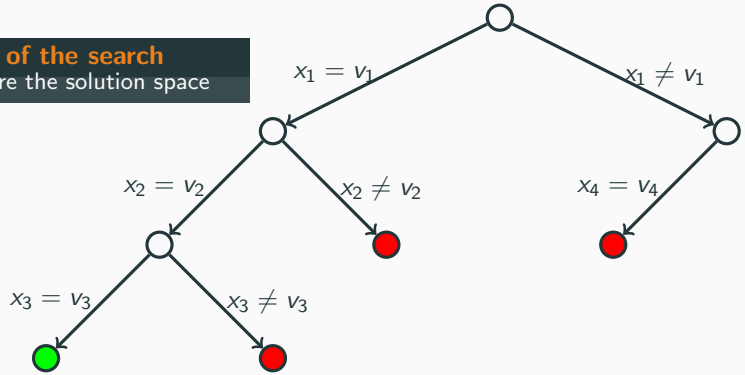
Goal of the search
Explore the solution space



Search

Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006

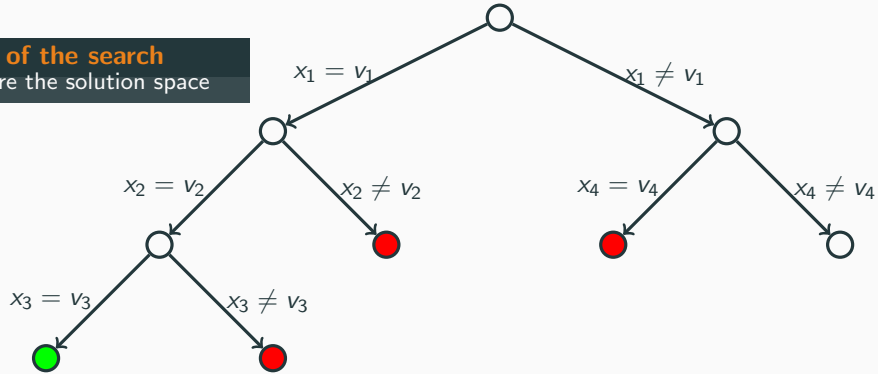
Goal of the search
Explore the solution space



Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006

UCLouvain

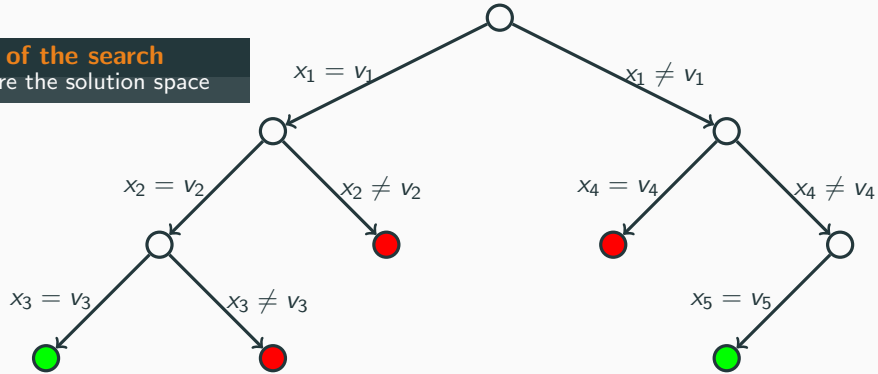
Goal of the search
Explore the solution space



Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006

UCLouvain

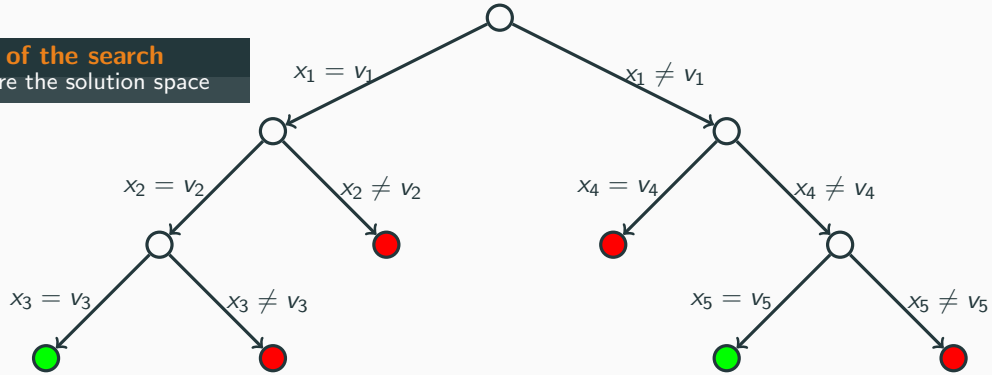
Goal of the search
Explore the solution space



Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006

UCLouvain

Goal of the search
Explore the solution space

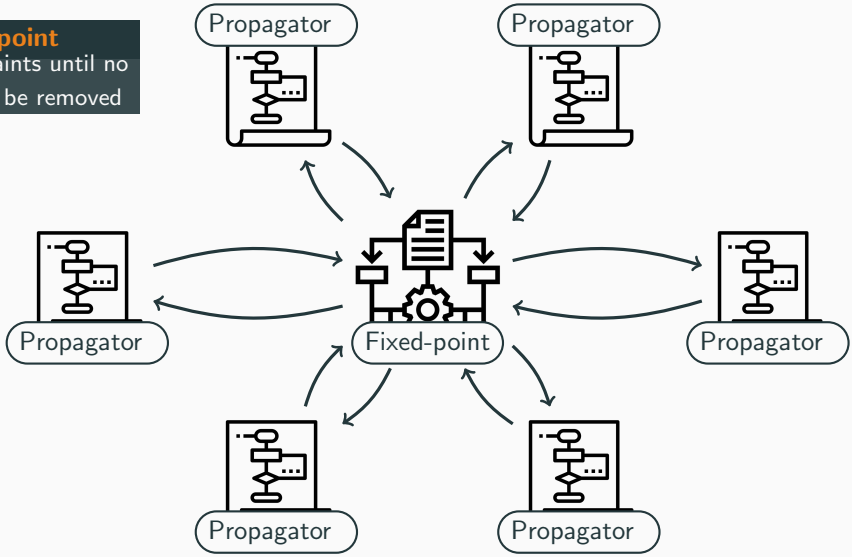


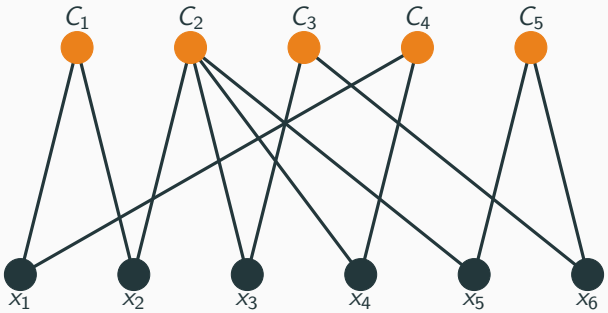
Search

Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006

Goal of fixed-point
Schedule constraints until no more values can be removed

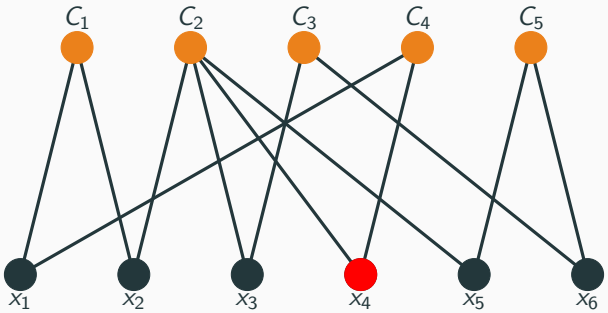
Goal of fixed-point
Schedule constraints until no more values can be removed





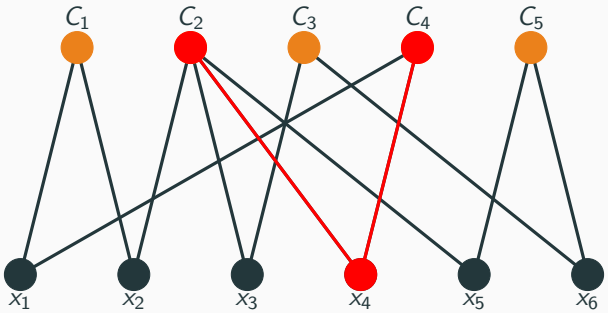
Propagating: Queue:

Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006



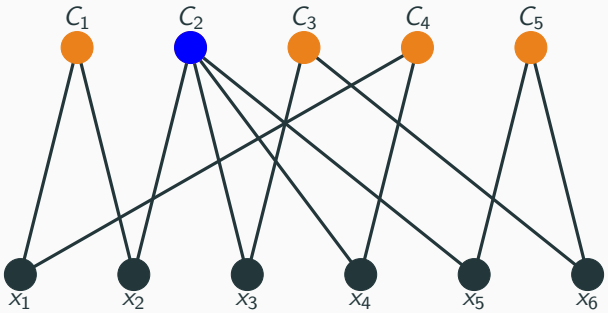
Propagating: ☐ Queue:

Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006



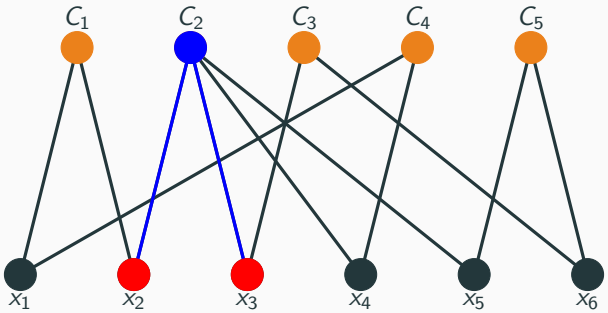
Propagating: □ Queue: C_2, C_4

Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006



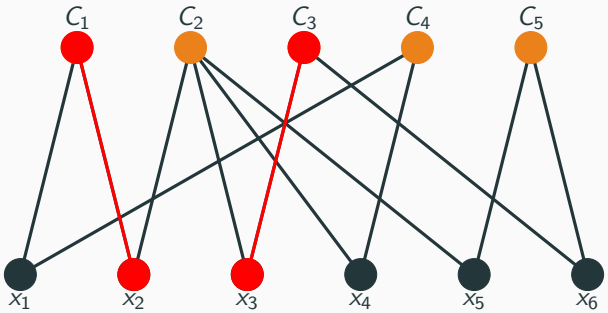
Propagating: $\boxed{C_2}$ Queue: C_4

Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006



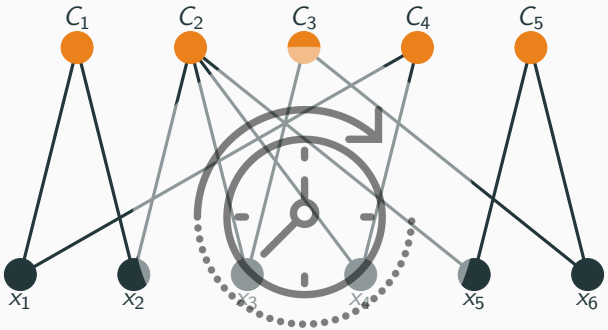
Propagating: $\boxed{C_2}$ Queue: C_4

Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006



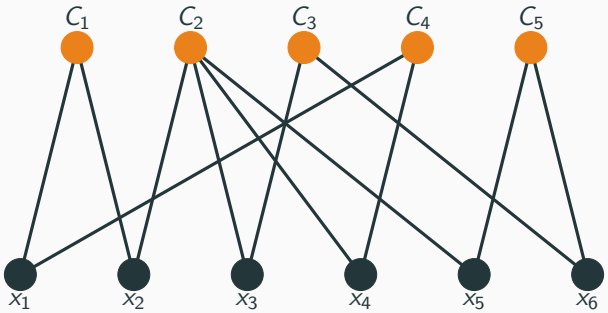
Propagating: □ Queue: C_4, C_1, C_3

Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006



Propagating: □ Queue: C_4, C_1, C_3

Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006

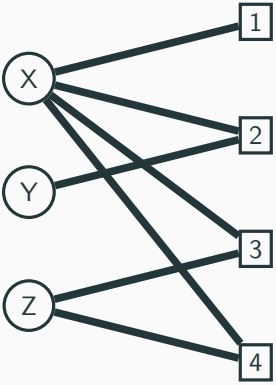


Propagating: Queue:

Reference: **"Handbook of constraint programming"**, by F. Rossi, P. Van Beek, T. Walsh, Elsevier 2006

UCLouvain

Goal of propagators
Filter invalid values

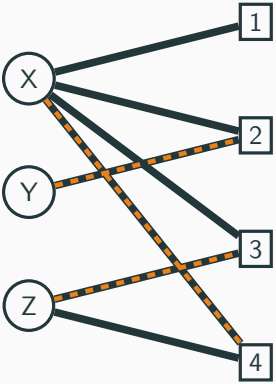


Propagator

Reference: **"A filtering algorithm for constraints of difference in CSPs"**, by J.-C. Régin, AAAI94

UCLouvain

Goal of propagators
Filter invalid values

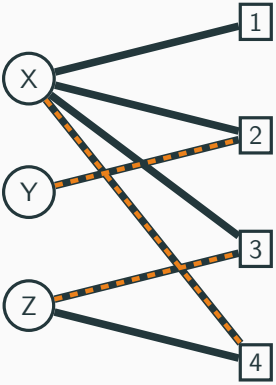


Propagator

Reference: **"A filtering algorithm for constraints of difference in CSPs"**, by J.-C. Régin, AAAI94

UCLouvain

Goal of propagators
Filter invalid values

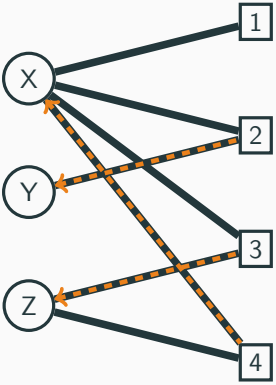



Propagator

Reference: **"A filtering algorithm for constraints of difference in CSPs"**, by J.-C. Régin, AAAI94

UCLouvain

Goal of propagators
Filter invalid values

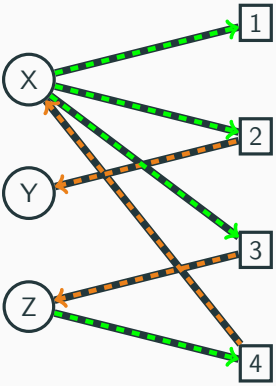



Propagator

Reference: **"A filtering algorithm for constraints of difference in CSPs"**, by J.-C. Régin, AAAI94

UCLouvain

Goal of propagators
Filter invalid values

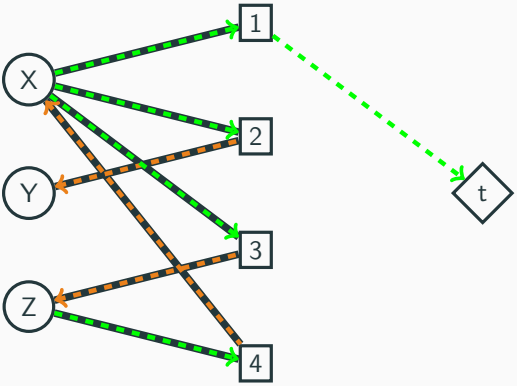



Propagator

Reference: **"A filtering algorithm for constraints of difference in CSPs"**, by J.-C. Régin, AAAI94

UCLouvain

Goal of propagators
Filter invalid values

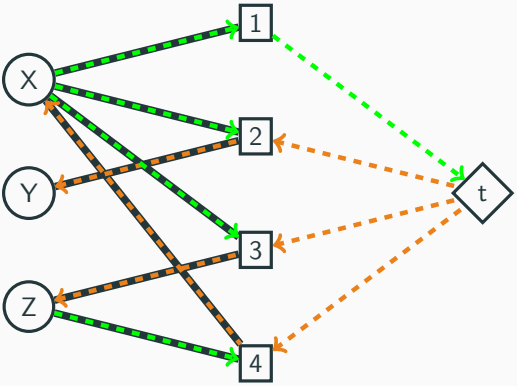


Propagator

Reference: **"A filtering algorithm for constraints of difference in CSPs"**, by J.-C. Régin, AAAI94

UCLouvain

Goal of propagators
Filter invalid values

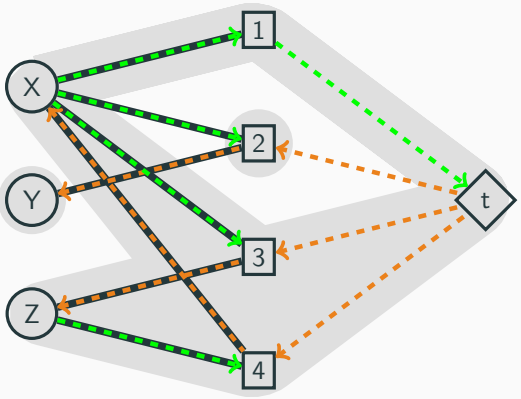


Propagator

Reference: **"A filtering algorithm for constraints of difference in CSPs"**, by J.-C. Régin, AAAI94

UCLouvain

Goal of propagators
Filter invalid values

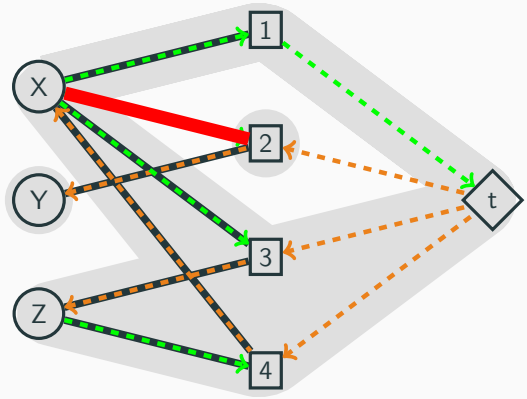


Propagator

Reference: **"A filtering algorithm for constraints of difference in CSPs"**, by J.-C. Régin, AAAI94

UCLouvain

Goal of propagators
Filter invalid values



Propagator

Reference: **"A filtering algorithm for constraints of difference in CSPs"**, by J.-C. Régin, AAAI94

Solving a problem using CP:

- **Step 1**: model the problem, using variables and constraints
- **Step 2**: solver searches for solution(s) using search strategies, fixed-point and propagators

Constraint programming

- is **modular** and **versatile**
- **can adapt** to one's needs
- relies on **well-defined structures** and mathematical **tools** such as graphs

CP applied to ML: Optimal Decision trees

Learning Optimal Decision Trees using Constraint Programming

Hélène Verhaeghe · Siegfried Nijssen · Gilles Pesant · Claude-Guy Quimper · Pierre Schaus

the date of receipt and acceptance should be inserted later

Abstract Decision trees are among the most popular classification models in machine learning. Traditionally, they are learned using greedy algorithms. However, such algorithms pose several disadvantages: it is difficult to limit the size of the decision trees while maintaining a good classification accuracy, and it is hard to impose additional constraints on the models that are learned. For these reasons,

UCLouvain

Database

f_1	f_2	f_3	...	f_n	c
1	0	1	...	1	+
0	1	0	...	1	−
1	1	0	...	0	+
0	0	0	...	0	+
1	0	0	...	0	+
0	1	1	...	1	−
1	1	1	...	0	−
⋮	⋮	⋮	⋱	⋮	⋮
1	1	1	...	1	+

- already a binary database

is green	produce gum	has flowers	poisonous?
yes	yes	no	+
no	yes	yes	−

- binarization required

height	age	F	sick?
134	34	1.45	+
178	23	3.66	−

height<150	height<180	F<1	...	sick?
yes	yes	no	...	+
no	yes	no	...	−

UCLouvain

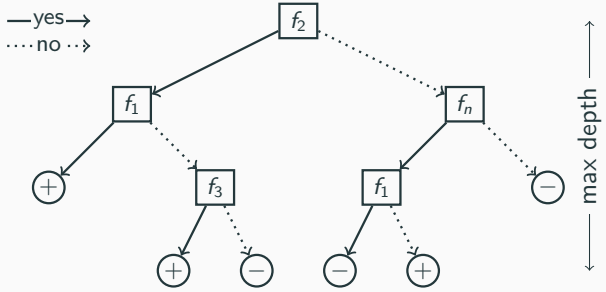
Database

f_1	f_2	f_3	$\dots$	f_n	c
1	0	1	$\dots$	1	$+$
0	1	0	$\dots$	1	$-$
1	1	0	$\dots$	0	$+$
0	0	0	$\dots$	0	$+$
1	0	0	$\dots$	0	$+$
0	1	1	$\dots$	1	$-$
1	1	1	$\dots$	0	$-$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
1	1	1	$\dots$	1	$+$

UCLouvain

Database

f_1	f_2	f_3	...	f_n	c
1	0	1	...	1	+
0	1	0	...	1	−
1	1	0	...	0	+
0	0	0	...	0	+
1	0	0	...	0	+
0	1	1	...	1	−
1	1	1	...	0	−
⋮	⋮	⋮	⋱	⋮	⋮
1	1	1	...	1	+



$$\min \sum (pred(i) - c(i))$$

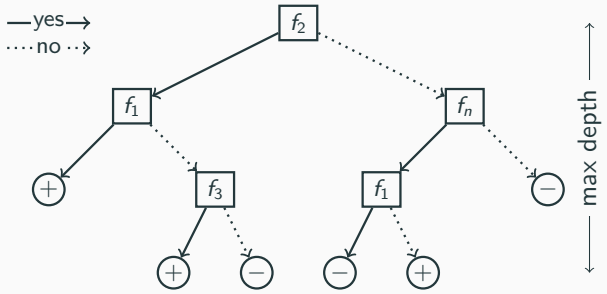
UCLouvain

Database

f_1	f_2	f_3	...	f_n	c
1	0	1	...	1	+
0	1	0	...	1	−
1	1	0	...	0	+
0	0	0	...	0	+
1	0	0	...	0	+
0	1	1	...	1	−
1	1	1	...	0	−
⋮	⋮	⋮	⋱	⋮	⋮
1	1	1	...	1	+

New sample

0	0	1	...	0	?



$$\min \sum (pred(i) - c(i))$$

The Problem: Learning Optimal Decision Trees

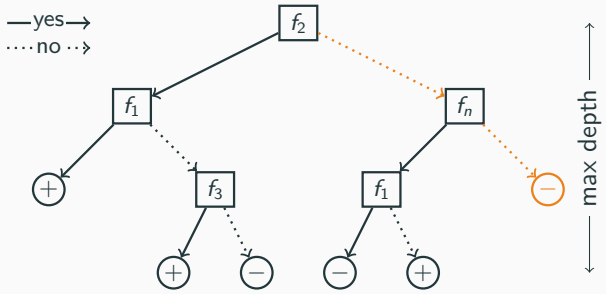
UCLouvain

Database

f_1	f_2	f_3	...	f_n	c
1	0	1	...	1	+
0	1	0	...	1	−
1	1	0	...	0	+
0	0	0	...	0	+
1	0	0	...	0	+
0	1	1	...	1	−
1	1	1	...	0	−
⋮	⋮	⋮	⋱	⋮	⋮
1	1	1	...	1	+

New sample

0	0	1	...	0	−

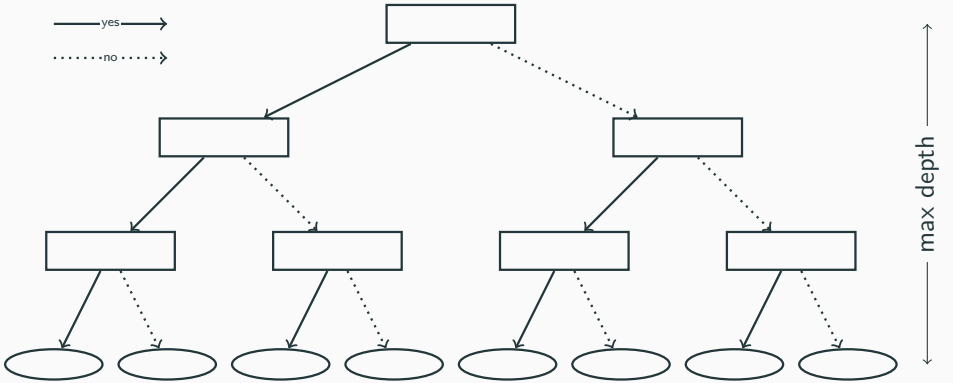


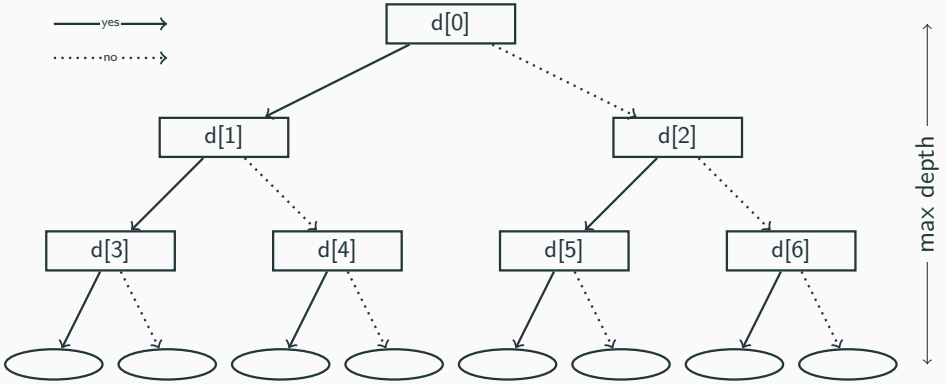
$$\min \sum (pred(i) - c(i))$$

Greedy methods:

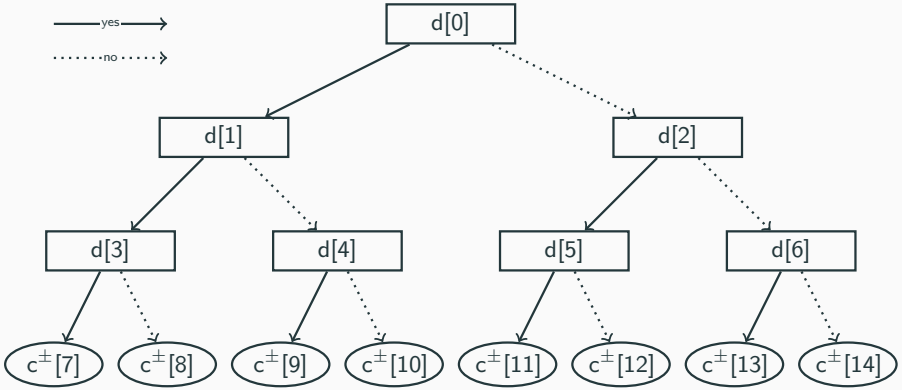
- ✔ easy construction
- ✗ hard to impose additional constraints
- ✗ potentially unnecessarily complex tree

- Mining optimal decision trees from itemset lattices, Nijssen, S., Fromont, E., 2007
- Minimising decision tree size as combinatorial optimisation, Bessiere, C., Hebrard, E., O'Sullivan, B., 2009
- Optimal constraint-based decision tree induction from itemset lattices,Nijssen, S., Fromont, É., 2010
- **Optimal classification trees**, Bertsimas, D., Dunn, J., 2017
- Learning optimal decision trees with sat, Narodytska, N., Ignatiev, A., Pereira, F., Marques-Silva, J., RAS, I., 2018
- Learning optimal and fair decision trees for non-discriminative decision-making, Aghaei, S., Azizi, M.J., Vayanos, P., 2019
- Learning optimal classification trees using a binary linear program formulation, Verwer, S., Zhang, Y., 2019


yes
no
max depth

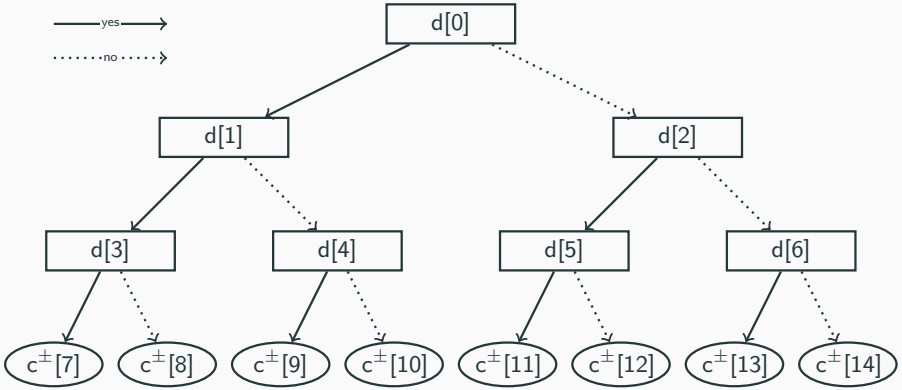


$dom(d[i]) = \{1, ..., n\}$



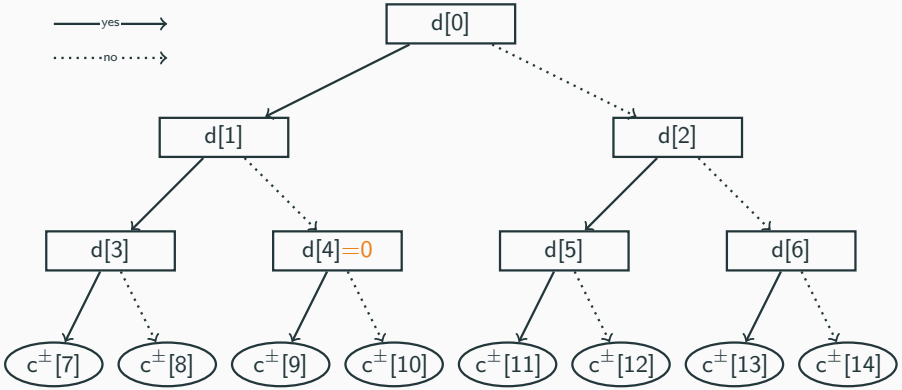
$dom(d[i]) = \{1, ..., n\}$

$dom(c[i]) = \{0, ..., N\}$



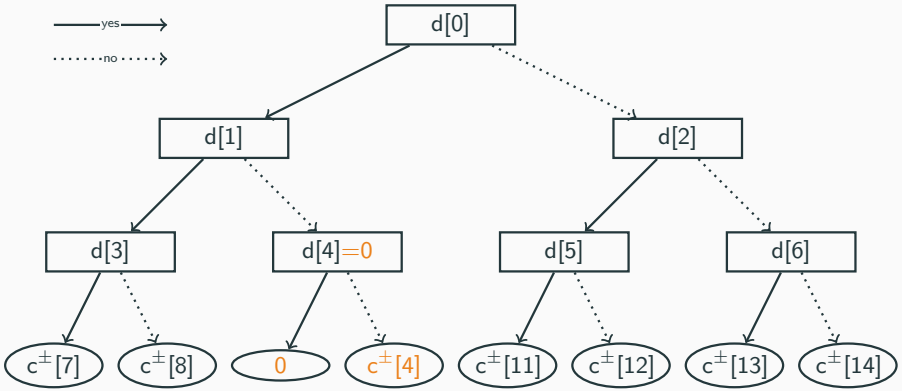
$dom(d[i]) = \{0, 1, ..., n\}$

$dom(c[i]) = \{0, ..., N\}$



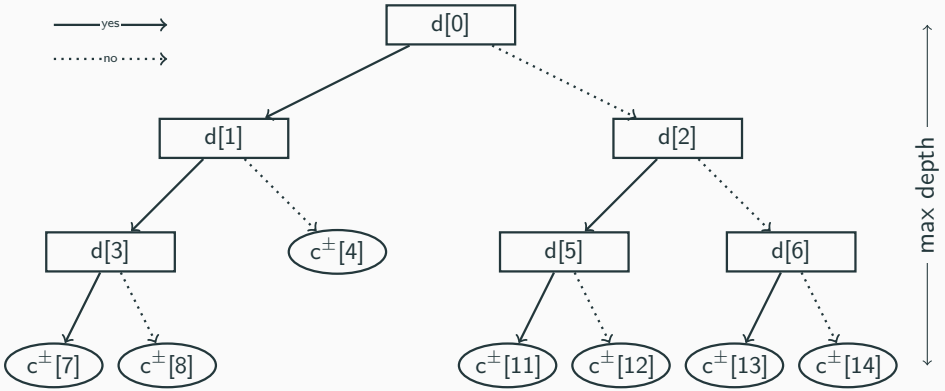
$dom(d[i]) = \{0, 1, ..., n\}$

$dom(c[i]) = \{0, ..., N\}$



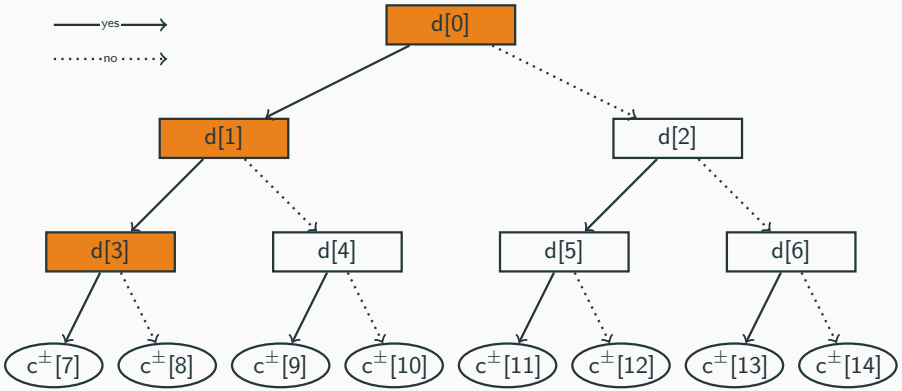
$dom(d[i]) = \{0, 1, ..., n\}$

$dom(c[i]) = \{0, ..., N\}$



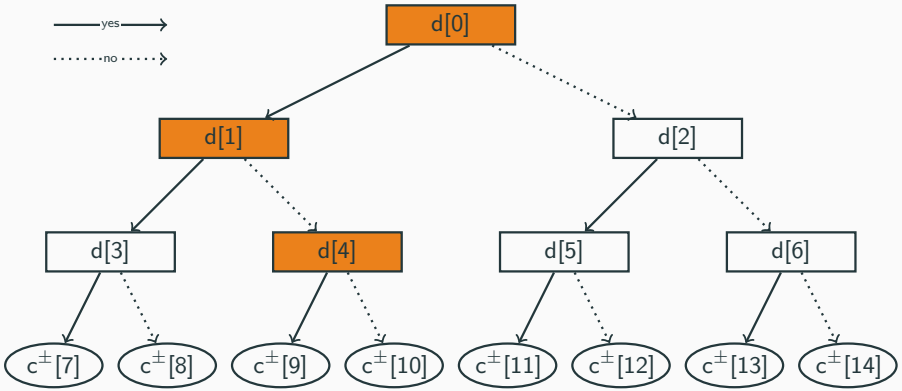
$dom(d[i]) = \{0, 1, ..., n\}$

$dom(c[i]) = \{0, ..., N\}$



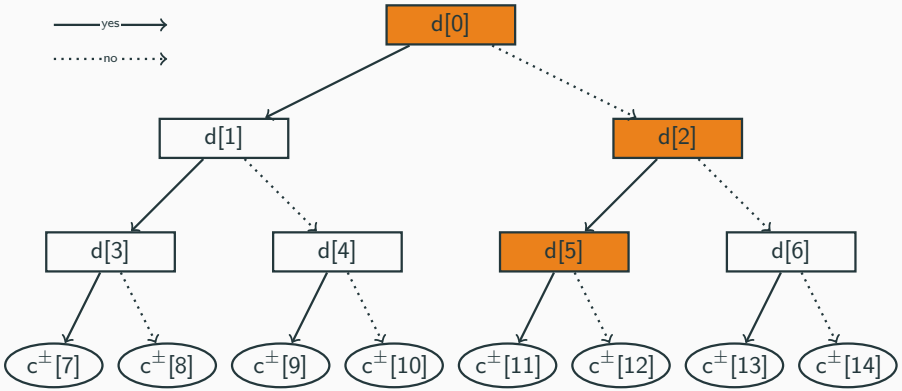
$dom(d[i]) = \{0, 1, ..., n\}$

$dom(c[i]) = \{0, ..., N\}$



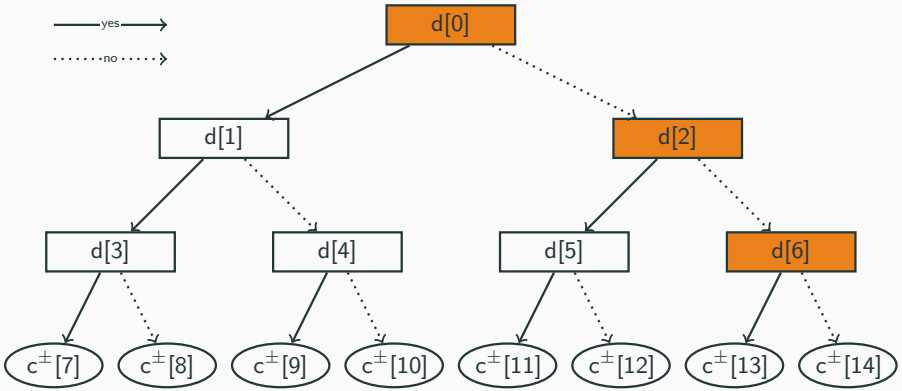
$dom(d[i]) = \{0, 1, ..., n\}$

$dom(c[i]) = \{0, ..., N\}$



$dom(d[i]) = \{0, 1, ..., n\}$

$dom(c[i]) = \{0, ..., N\}$



$dom(d[i]) = \{0, 1, ..., n\}$

$dom(c[i]) = \{0, ..., N\}$

UCLouvain

f_1	f_2	f_3	f_4
1	0	1	1
0	1	0	1
1	1	0	0
0	0	0	0
1	0	0	0
0	1	1	1
1	1	1	0
1	1	1	1

Features (Dense)				Counter
x_1	x_2	x_3	x_4	

P. Schaus, J. Aoga, and T. Guns. "Coversize: A global constraint for frequency-based itemset mining". In CP 2017.

UCLouvain

f_1	f_2	f_3	f_4
1	0	1	1
0	1	0	1
1	1	0	0
0	0	0	0
1	0	0	0
0	1	1	1
1	1	1	0
1	1	1	1

Features (Dense)				Counter
x_1	x_2	x_3	x_4	
0	1	0	1	

P. Schaus, J. Aoga, and T. Guns. "Coversize: A global constraint for frequency-based itemset mining". In CP 2017.

f_1	f_2	f_3	f_4
1	0	1	1
0	1	0	1
1	1	0	0
0	0	0	0
1	0	0	0
0	1	1	1
1	1	1	0
1	1	1	1

Features (Dense)				Counter
x_1	x_2	x_3	x_4	
0	1	0	1	3

P. Schaus, J. Aoga, and T. Guns. "Coversize: A global constraint for frequency-based itemset mining". In CP 2017.

f_1	f_2	f_3	f_4
1	0	1	1
0	1	0	1
1	1	0	0
0	0	0	0
1	0	0	0
0	1	1	1
1	1	1	0
1	1	1	1

Features (Dense)				Counter
x_1	x_2	x_3	x_4	
0	1	0	1	3

- Dense representation
- No feature rejection

UCLouvain

f_1	f_2	f_3	f_4
1	0	1	1
0	1	0	1
1	1	0	0
0	0	0	0
1	0	0	0
0	1	1	1
1	1	1	0
1	1	1	1

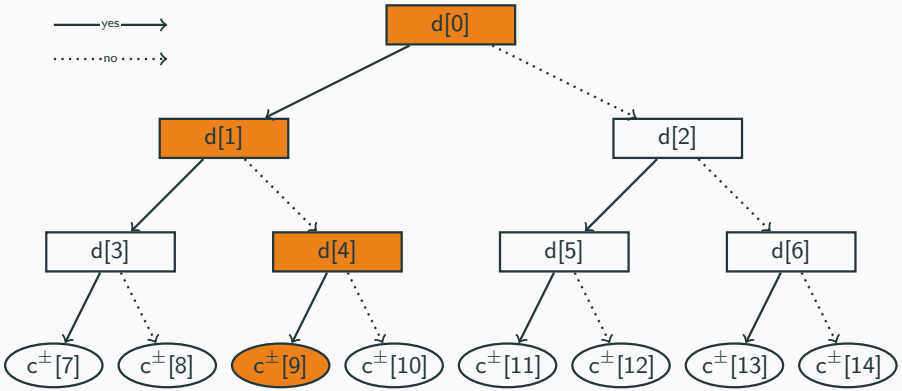
Features (Sparse)		Counter
y_1	y_2	
2	4	3

- ~~Dense representation~~
- No feature rejection

f_1	f_2	f_3	f_4
1	0	1	1
0	1	0	1
1	1	0	0
0	0	0	0
1	0	0	0
0	1	1	1
1	1	1	0
1	1	1	1

✔Features (Sparse)		✗Features (Sparse)	Counter
y_1	y_2	z_1	
2	4	3	1

- ~~Dense representation~~
- ~~No feature rejection~~



$Coversize(\{d[0], d[4]\}, \{d[1]\}, c^+[9])$

$Coversize(\{d[0], d[4]\}, \{d[1]\}, c^-[9])$

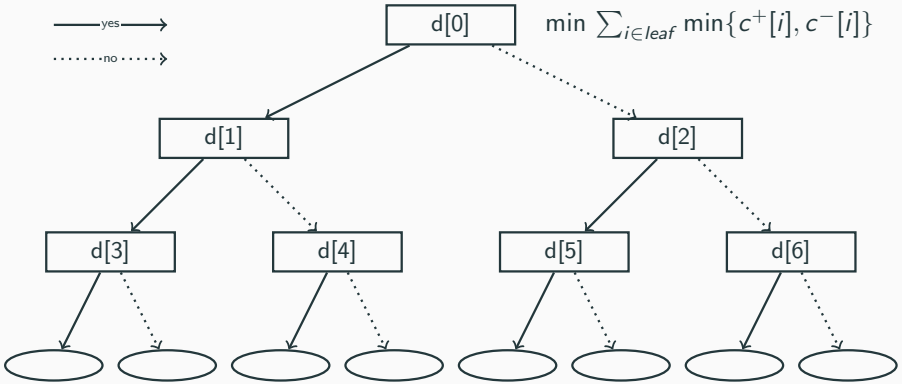
UCLouvain

- constraints imposing minimum at leaf

$$c^+[i] + c^-[i] \geq N_{min}$$

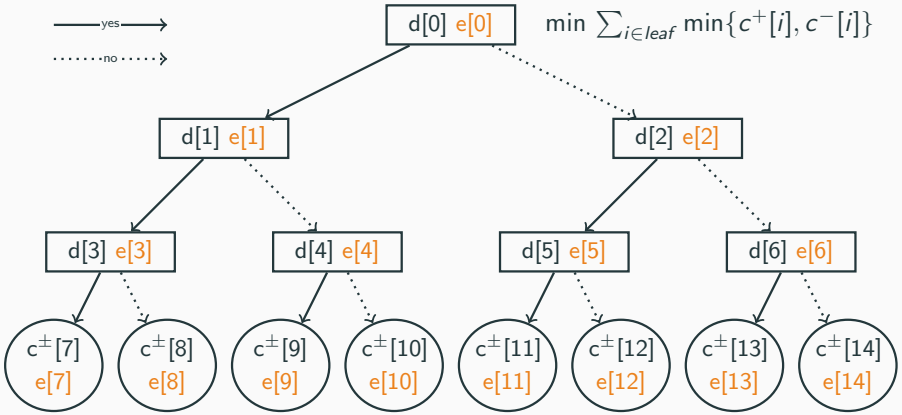
- constraints avoiding useless decisions

- redundant constraints improving speed

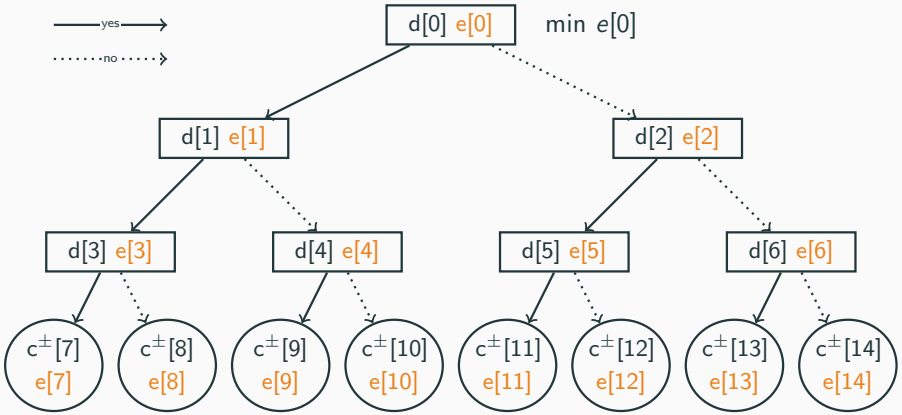


$dom(d[i]) = \{0, 1, ..., n\}$

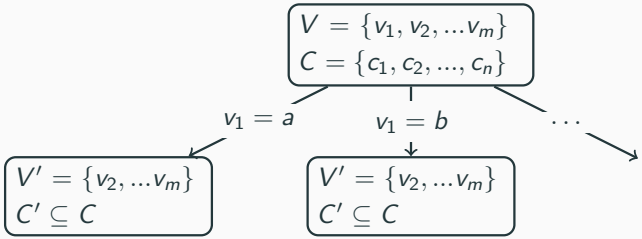
$dom(c[i]) = \{0, ..., N\}$



$dom(d[i]) = \{0, 1, ..., n\}$ $dom(c[i]) = \{0, ..., N\}$ $dom(e[i]) = \{0, ..., N\}$

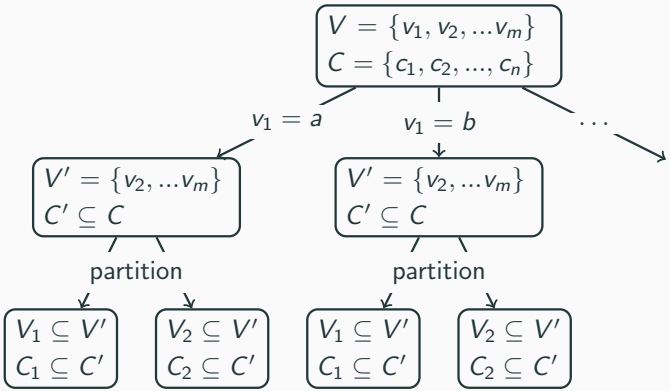


$dom(d[i]) = \{0, 1, ..., n\}$ $dom(c[i]) = \{0, ..., N\}$ $dom(e[i]) = \{0, ..., N\}$



OR nodes

SOL = SOL_1 or SOL_2 or ...

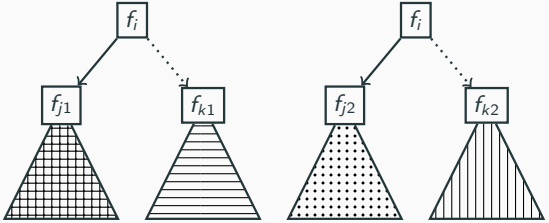


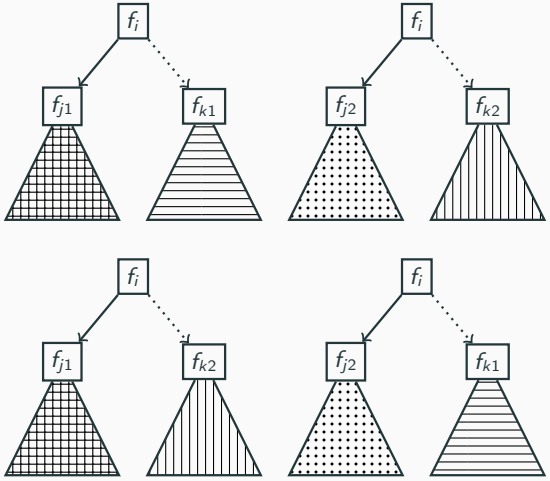
OR nodes

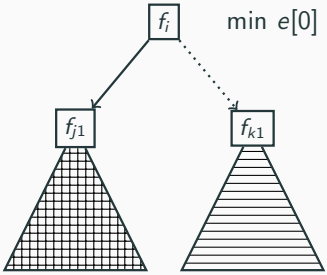
SOL = SOL_1 or SOL_2 or ...

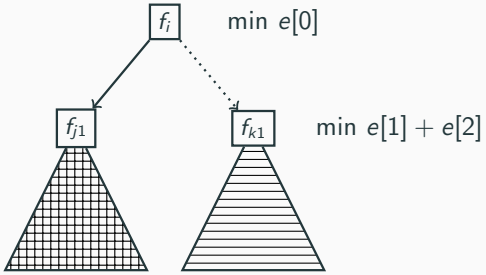
AND nodes

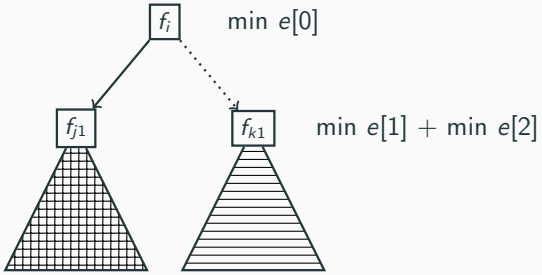
SOL = SOL_1 and SOL_2 and ...

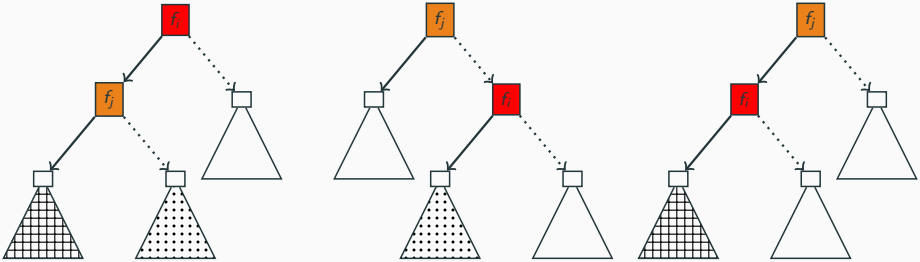

f_i
f_{j1}
f_{k1}
f_i
f_{j2}
f_{k2}

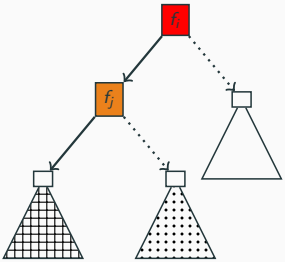

f_i
f_{j1}
f_{k1}
f_i
f_{j2}
f_{k2}
f_i
f_{j1}
f_{k2}
f_i
f_{j2}
f_{k1}

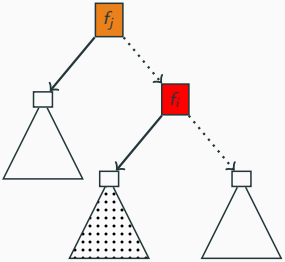

f_i
min $e[0]$
f_{j1}
f_{k1}

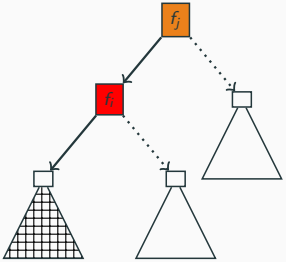

f_i
min $e[0]$
f_{j1}
f_{k1}
min $e[1] + e[2]$


f_i
min $e[0]$
f_{j1}
f_{k1}
min $e[1]$ + min $e[2]$


f_i
f_j
f_j
f_i
f_j
f_i







	yes	no	hash
	f_i f_j		$f_i, f_j -$
	f_i	f_j	$f_i - f_j$

UCLouvain

	$N_{\min} = 1$			$N_{\min} = 5$			
	DL8	BinOCT	CP	DL8	CP	CP-c	CP-m
Proven optimality	49(64%)	13(17%)	**57**(75%)	54(71%)	56(74%)	56(74%)	**58**(76%)
Best solution found	49(64%)	21(28%)	**76**(100%)	54(71%)	**74**(97%)	**74**(97%)	70(92%)
Fastest	23(30%)	11(14%)	**49**(64%)	28(37%)	**40**(53%)	33(43%)	22(29%)
Time out	27(36%)	63(83%)	**19**(25%)	22(29%)	21(28%)	21(28%)	**19**(25%)

23 instances, depths from 2 to 5, 10 min TO

DL8: Dynamic programming approach using frequent itemsets mining

BinOCT: MIP-based approach running on CPLEX

ML applied to CP: Learning Precedences with GNNs

Learning Precedences for Scheduling Problems with Graph Neural Networks

Hélène Verhaeghe
DTAI, KU Leuven, Leuven, Belgium[1]

Quentin Cappart
Polytechnique Montréal, Montreal, Canada

Gilles Pesant
Polytechnique Montréal, Montreal, Canada

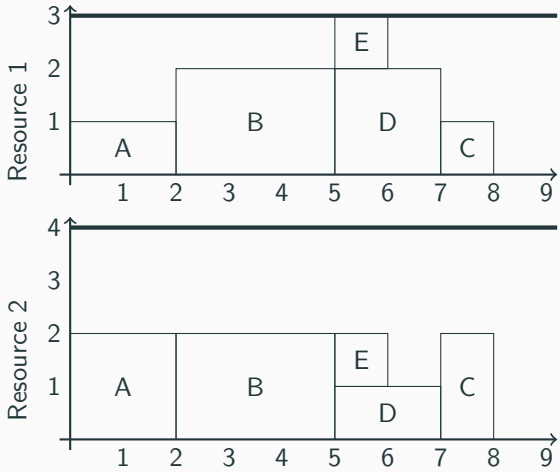
Claude-Guy Quimper
Université Laval, Quebec, Canada

Abstract

The *resource constrained project scheduling problem* (RCPSP) consists of scheduling a finite set of resource-consuming tasks within a temporal horizon subject to resource capacities and precedence relations between pairs of tasks. It is $\mathcal{NP}$-hard and many techniques have been introduced to improve the efficiency of CP solvers to solve it. The problem is naturally represented as a directed graph, commonly referred to as the *precedence graph*, by linking pairs of tasks subject to a precedence.

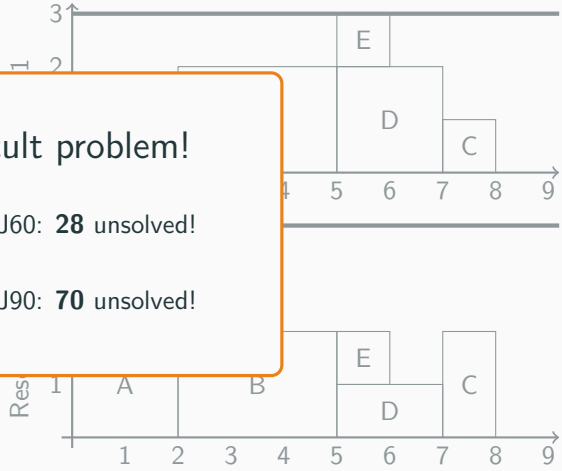
Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C

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



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 $C_{r_2} = 4$

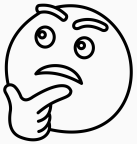


Difficult problem!

PSPlib J60: **28** unsolved!

PSPlib J90: **70** unsolved!

What if we had more information about the solution?

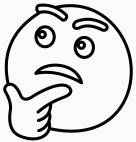


UCLouvain

What if we had more information about the solution?

- Guide toward solution
- Reduce search space

How do I get more information about the solution?



UCLouvain

How do I get more information about the solution?

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

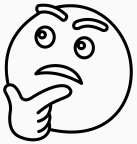
UCLouvain

How do I get more information about the solution?

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

Which ML tool?

Which information could be useful?



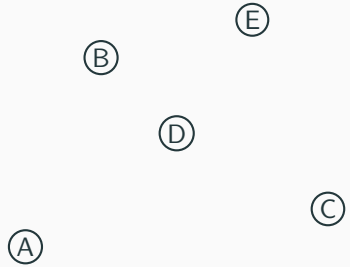
UCLouvain

Which information could be useful?

Precedences between pairs of tasks

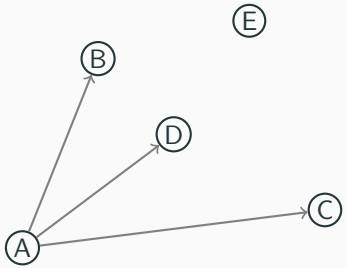
Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C

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



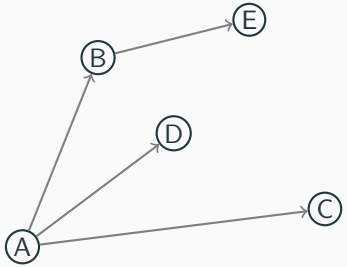
Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C

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



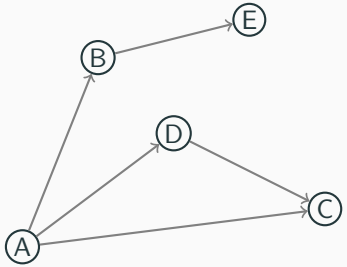
Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C

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



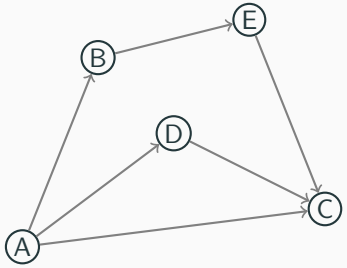
Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C

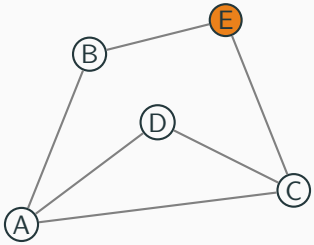
$C_{r_1} = 3$ and $C_{r_2} = 4$



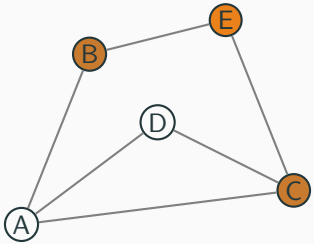
Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C

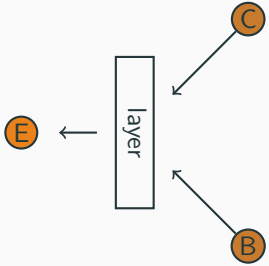
$C_{r_1} = 3$ and $C_{r_2} = 4$

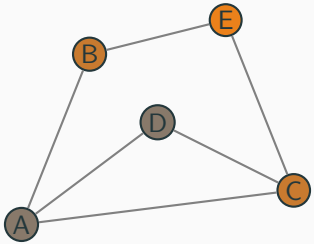


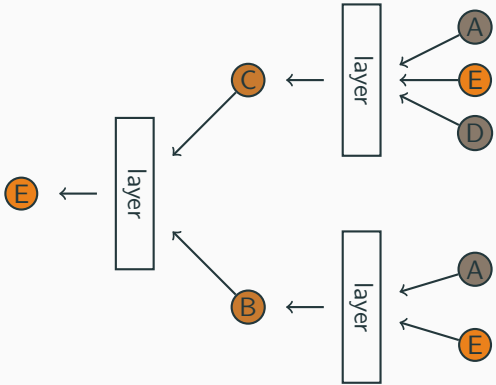

E
B
D
C
A

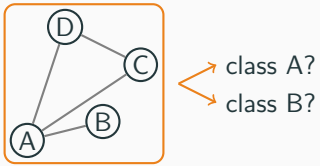

E


E
B
D
C
A

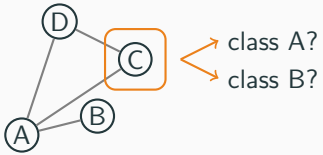

C
E
layer
B


E
B
D
C
A

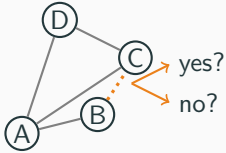

A
E
D
layer
C
E
layer
B
A
E
layer



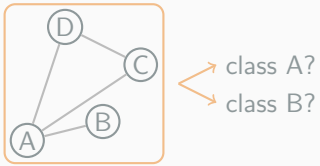
Graph classification



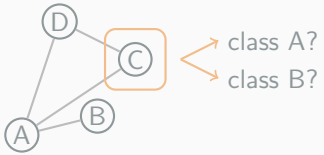
Node classification



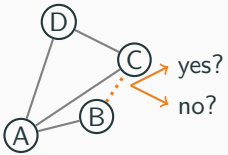
Link prediction



Graph classification



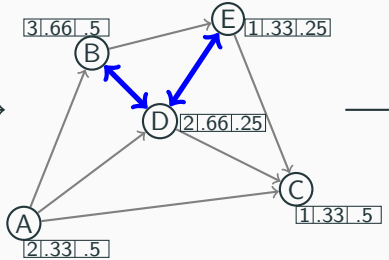
Node classification



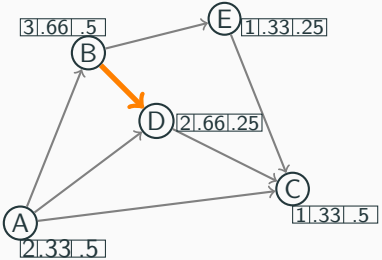
Link prediction

UCLouvain

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



GNN + MLP



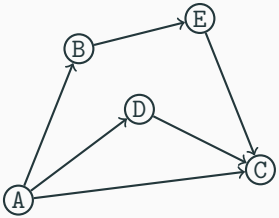
solver

solution

Candidate edge

UCLouvain

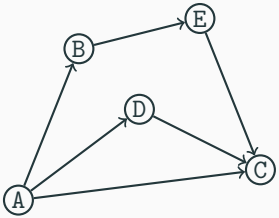
Instance



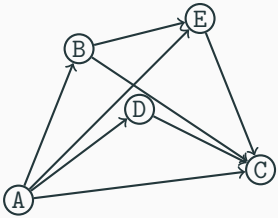
Candidate edge

UCLouvain

Instance



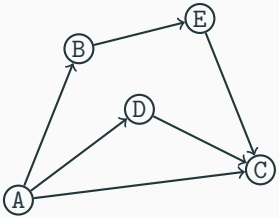
Transitive closure



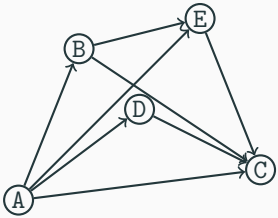
Candidate edge

UCLouvain

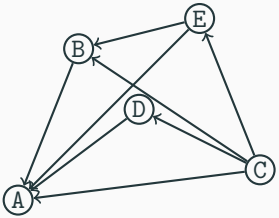
Instance



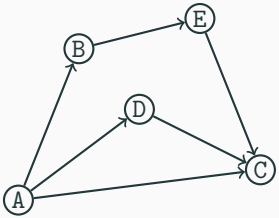
Transitive closure



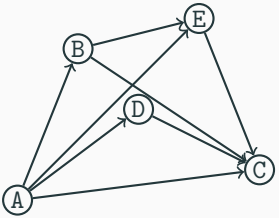
Avoided



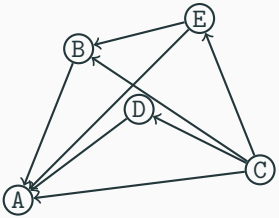
Instance



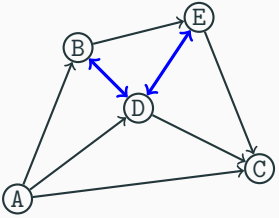
Transitive closure

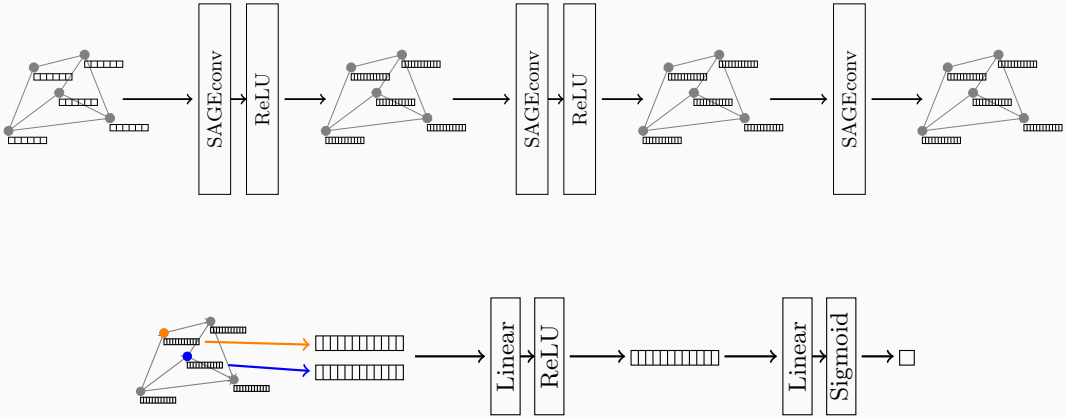


Avoided



Candidate




SAGEconv
ReLU
SAGEconv
ReLU
SAGEconv
Linear
ReLU
Linear
Sigmoid

UCLouvain

Additional precedences

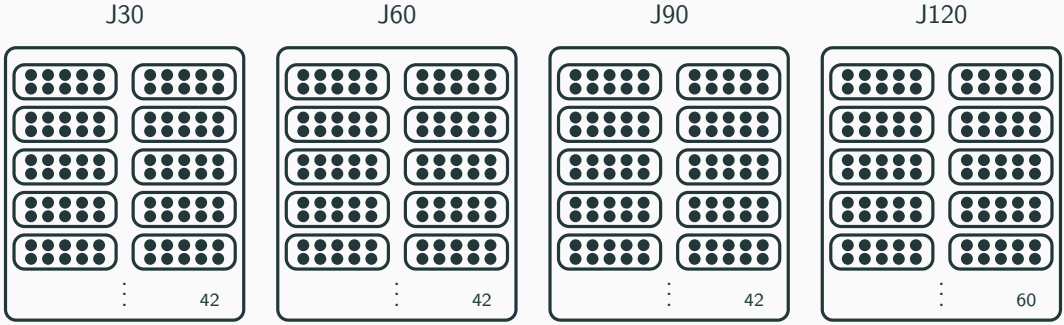
+ reduces search space
- restriction of the problem

Ordering heuristic

+ preserve solutions
- static ordering are slower

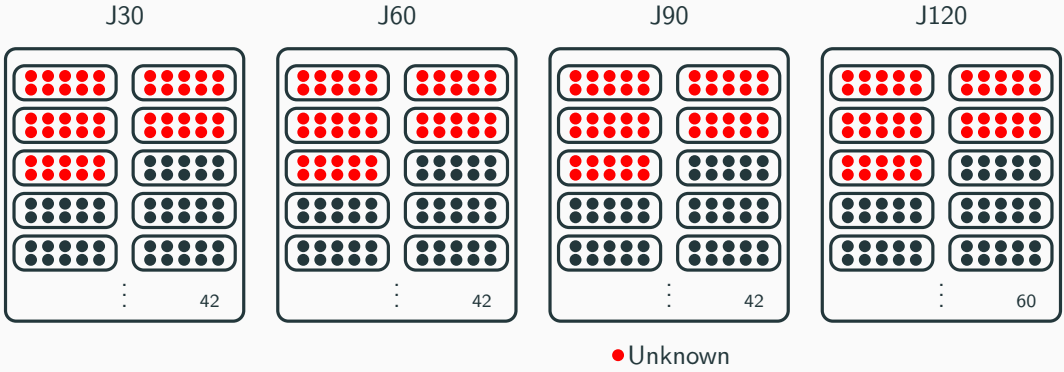
Benchmark - PSPlib

UCLouvain

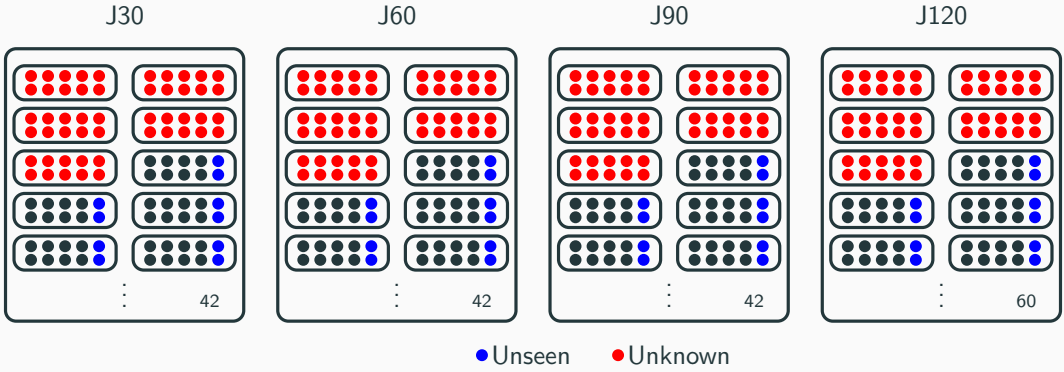


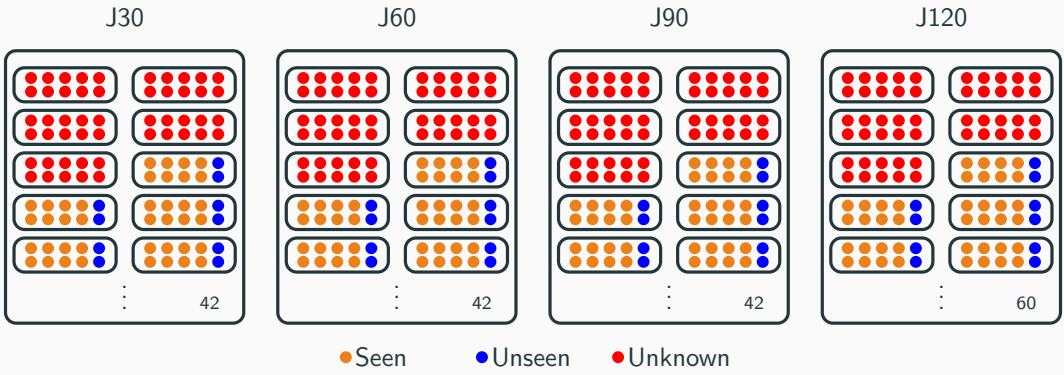
Benchmark - PSPlib

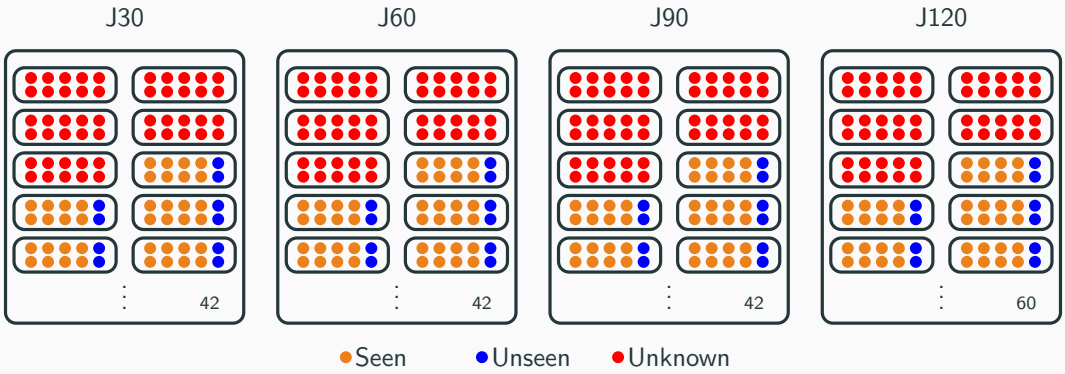
UCLouvain



UCLouvain




J30
J60
J90
J120
42
42
42
60
Seen
Unseen
Unknown



- 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

UCLouvain

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

	Model A ("*without*")				Model B ("*with*")			
	f1	*prec*	*rec*	*tn*	*f1*	*prec*	*rec*	*tn*
$SEEN_{J120}$							62	0.87
$UNSEEN_{J12}$							62	0.87
$UNKNOWN_{J}$							64	0.87
ALL_{J120}							62	0.87

Takeaway: Use best training set

Worse training metrics

Better use metrics

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

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

UCLouvain

	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
$\leq$J120 train	419/152	**125/386**	26/**469**	34/**459**	**433**/26	179/25	**8**/17	**2**/8
$\leq$J60 train	**421/141**	119/398	**27**/484	**35**/481	430/**17**	**187/18**	6/**10**	1/**4**

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

UCLouvain

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

Takeaway: very good generalization

Equivalent training metrics

Equivalent use metrics

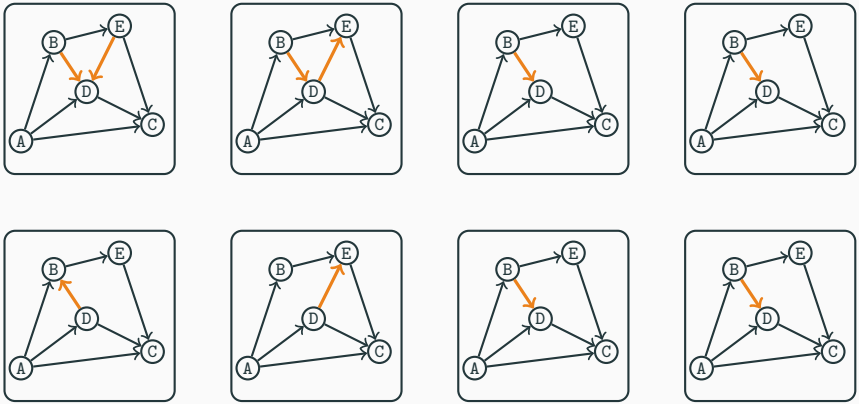
	1s	sed for ordering 10m	1h
≤J120 train	419/1	**8**/17	**2**/8
≤J60 train	**421/1**	6/**10**	1/**4**

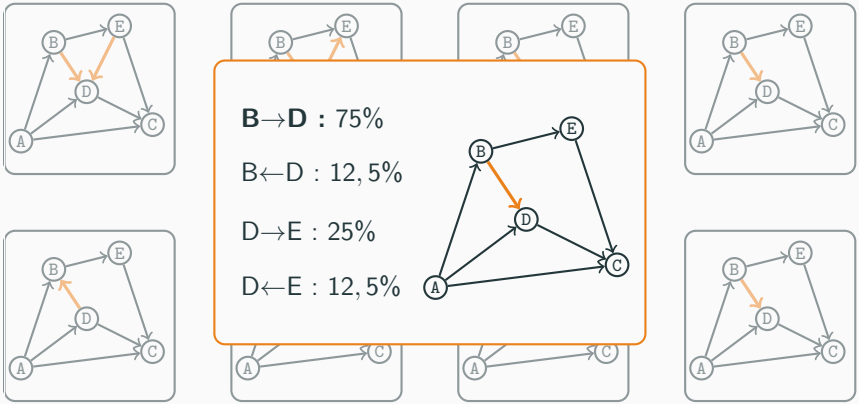
Setting: Training on ≤J60 instances; evaluation on J120 instances, with vsids/sbps


Sol 1


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

Aggregating solutions




B→D : 75%
B←D : 12, 5%
D→E : 25%
D←E : 12, 5%
A
B
C
D
E

UCLouvain

	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

UCLouvain

	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

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

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

Conclusion

- Hybridization is beneficial
- Opens up a lot of new approaches for solving problems
- Allows to benefits from the best of both families of methods

Thank you for listening!

Any questions?

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