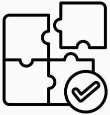
Best of both worlds: synergies between Constraint Programming and Machine Learning

Hélène Verhaeghe

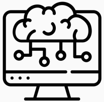
14 February 2024

KULeuven, Leuven, Belgium, *helene.verhaeghe@kuleuven.be*

Introduction



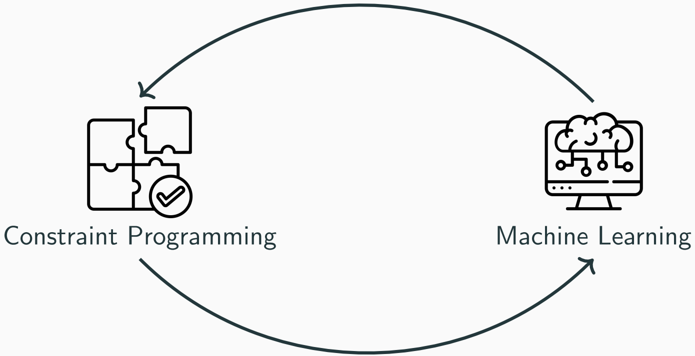
Constraint Programming



Machine Learning

Introduction

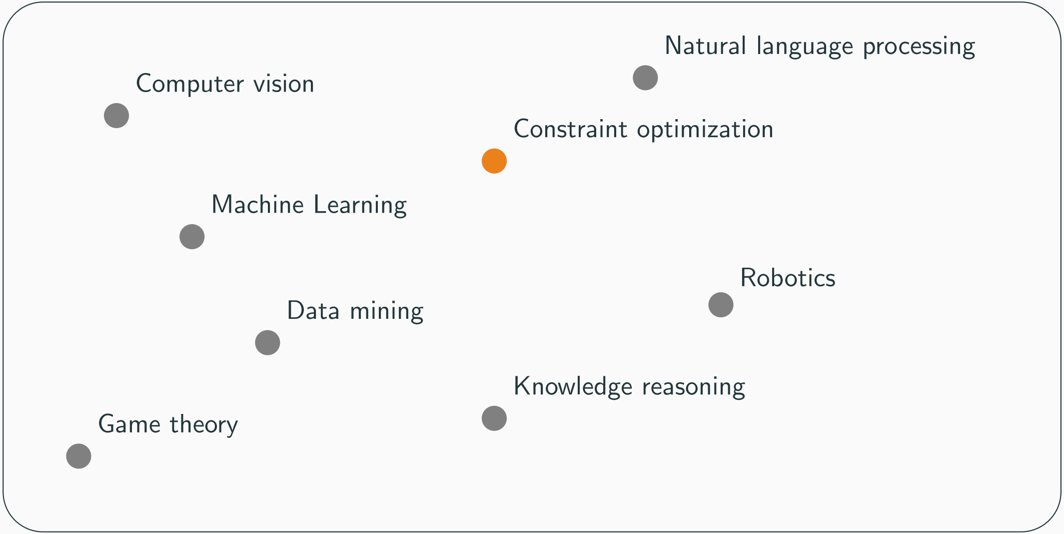
Introduction



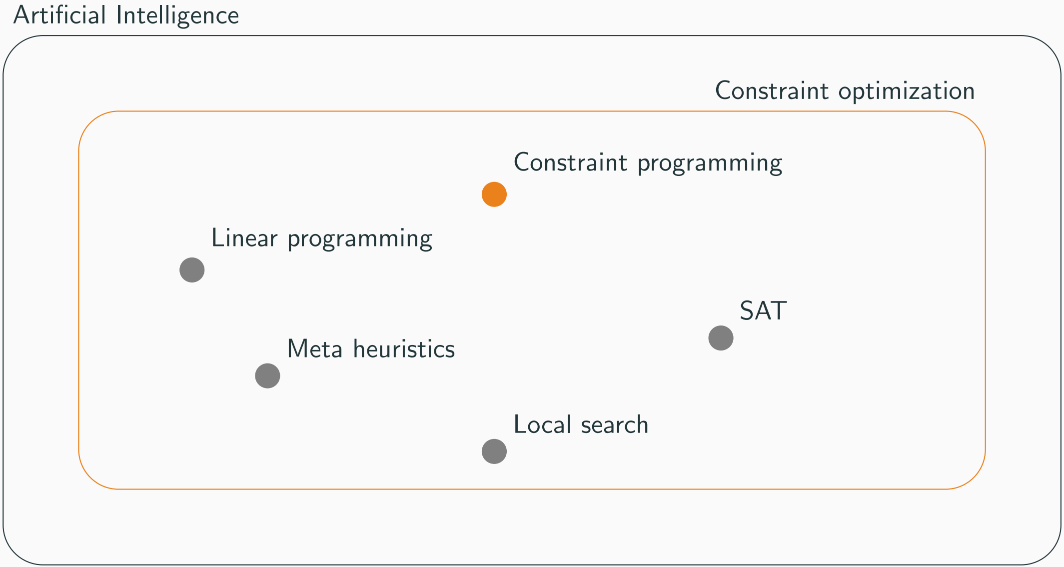
Constraint Programming

Constraint Programming within AI

Artificial Intelligence



Constraint Programming within AI



Constraint programming within Optimization

Complete search	Incomplete search
• Branch & Bound • **Constraint programming** • Integer programming • Sat programming	• Local search • Large neighborhood search • Genetic algorithms • Meta heuristics
Pros: • Optimality guarantees	Pros: • Fast
Cons: • Takes time	Cons: • No optimality guarantees

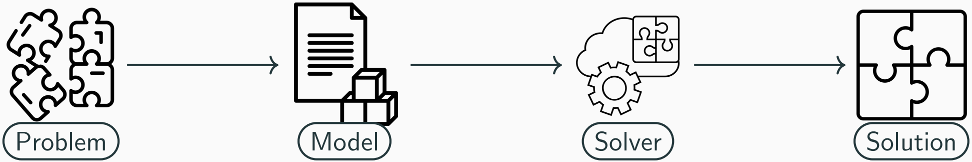
What technique to choose? Depends on the goal, get the best solution or get a solution quickly!

What is constraint programming?

"Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it." — E. Freuder

Constraint programming

In Constraint Programming, we model in a declarative way the desired solution, and then the computer/solver finds the solution.



Model

- Variables: X, Y, Z,...

Model

- Variables: X, Y, Z,...
- Domains: $\{1, 2\}$, $\{true, false\}$, $\{bleu, rouge, ...\}$

Model

- Variables: X, Y, Z,...
- Domains: $\{1, 2\}$, $\{true, false\}$, $\{bleu, rouge, \ldots\}$
- Constraint:
  - arithmetic: $X + Y = Z$, $X \leq Y$
  - logic: $A \wedge B$, $A \vee B$
  - global: $AllDifferent(X, Y, Z)$, $Circuit(X_1, X_2, X_3)$

Example - Sudoku

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

$$X_{i,j} \in \{1,2,3,4,5,6,7,8,9\}$$

$$X_{i,j} = grid_{i,j} \quad \forall grid_{i,j} \neq \emptyset$$

Example - Sudoku

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

$$X_{i,j} \in \{1,2,3,4,5,6,7,8,9\}$$

$$X_{i,j} = grid_{i,j} \quad \forall grid_{i,j} \neq \emptyset$$

$$AllDifferent(X_{i,1}, \ldots, X_{i,9}) \quad \forall 1 \leq i \leq 9$$

Example - Sudoku

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

$$X_{i,j} \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$X_{i,j} = grid_{i,j} \quad \forall grid_{i,j} \neq \emptyset$$

$$AllDifferent(X_{i,1}, ..., X_{i,9}) \quad \forall 1 \leq i \leq 9$$

$$AllDifferent(X_{1,j}, ..., X_{9,j}) \quad \forall 1 \leq j \leq 9$$

Example - Sudoku

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

$$X_{i,j} \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

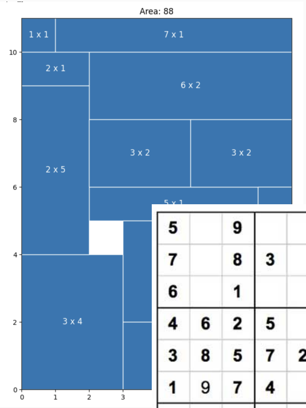
$$X_{i,j} = grid_{i,j} \quad \forall grid_{i,j} \neq \emptyset$$

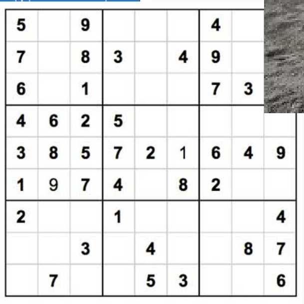
$$AllDifferent(X_{i,1}, \ldots, X_{i,9}) \quad \forall 1 \leq i \leq 9$$

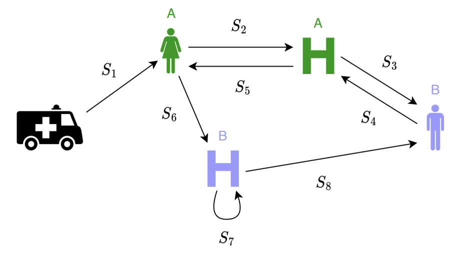
$$AllDifferent(X_{1,j}, \ldots, X_{9,j}) \quad \forall 1 \leq j \leq 9$$

$$AllDifferent(X_{3k,3l}, X_{3k+1,3l}, \ldots, X_{3k+2,3l+2}) \quad \forall 0 \leq k, l < 3$$

Examples of use of CP

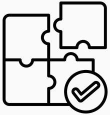






Hybridization ML/CP

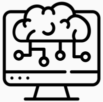
Model + Search



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

Machine Learning

(Big) Data + algorithms



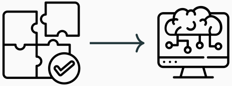
- Goal: Learn from examples
- Pro: Good with huge quantities of data
- Con: Difficulties to satisfy (hard) constraints in outputs

Best of both worlds

Can we get the best of both worlds?

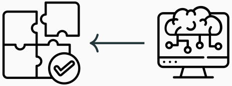
Yes, by combining them!

How to combine them?



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)

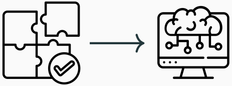


ML for CP

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

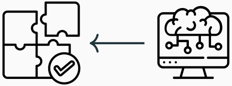
And many many other examples ...

Examples of such integration



CP for ML

- Optimal decision trees
- CP-BP for learning



ML for CP

- Solving RCPSP using GNNs
- Generic Graph Representation

When CP helps ML: Optimal decision trees

The Problem: Learning Optimal Decision Trees

KU LEUVEN

Database

f_1	f_2	f_3	$\ldots$	f_n	c
1	0	1	$\ldots$	1	$+$
0	1	0	$\ldots$	1	$-$
1	1	0	$\ldots$	0	$+$
0	0	0	$\ldots$	0	$+$
1	0	0	$\ldots$	0	$+$
0	1	1	$\ldots$	1	$-$
1	1	1	$\ldots$	0	$-$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
1	1	1	$\ldots$	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	$\ldots$	sick?
yes	yes	no	$\ldots$	$+$
no	yes	no	$\ldots$	$-$

The Problem: Learning Optimal Decision Trees

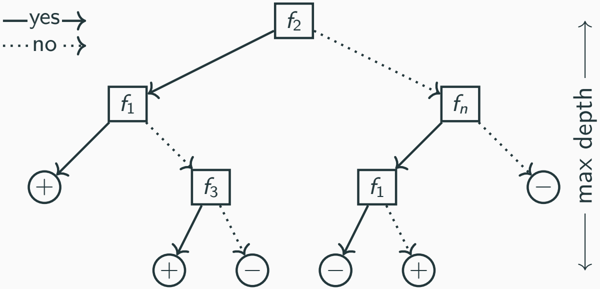
Database

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	$+$

The Problem: Learning Optimal Decision Trees

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	$+$



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

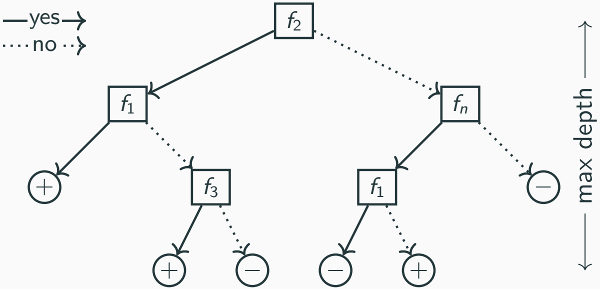
The Problem: Learning Optimal Decision Trees

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	+

New sample

0	0	1	$\dots$	0	?



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

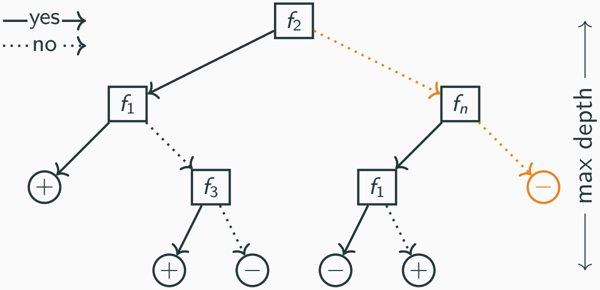
The Problem: Learning Optimal Decision Trees

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	$+$

New sample

0	0	1	$\dots$	0	$-$



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

Motivations

KU LEUVEN

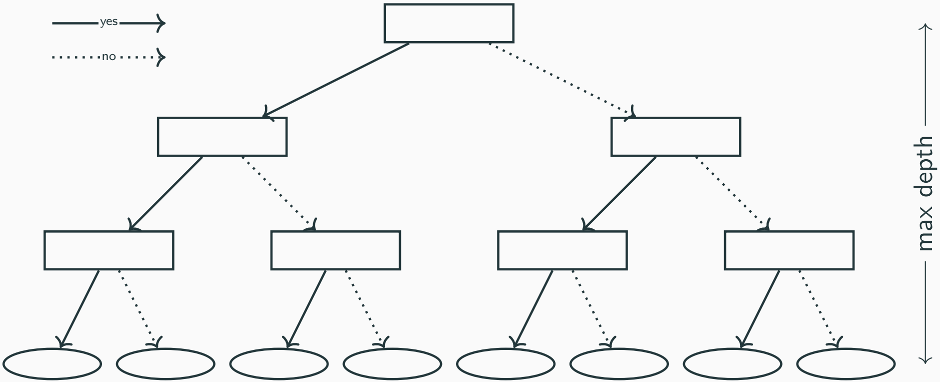
Greedy methods:

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

Motivations

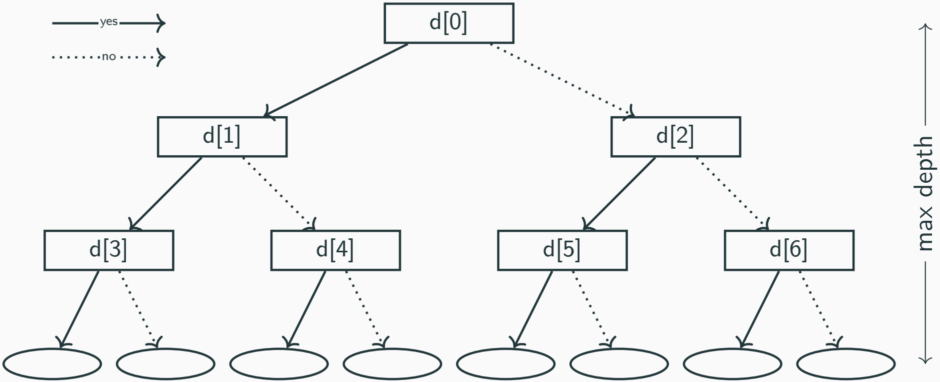
- Mining optimal decision trees from itemset lattices, Nijssen, S., Fromont, E., 2007
- Minimising decision tree size as combinatorial optimisation, Bessiere, C., Hebrard, E., O'Sullivan, B., 2009
- Optimal constraint-based decision tree induction from itemset lattices,Nijssen, S., Fromont, É., 2010
- **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

Model - Decision variables



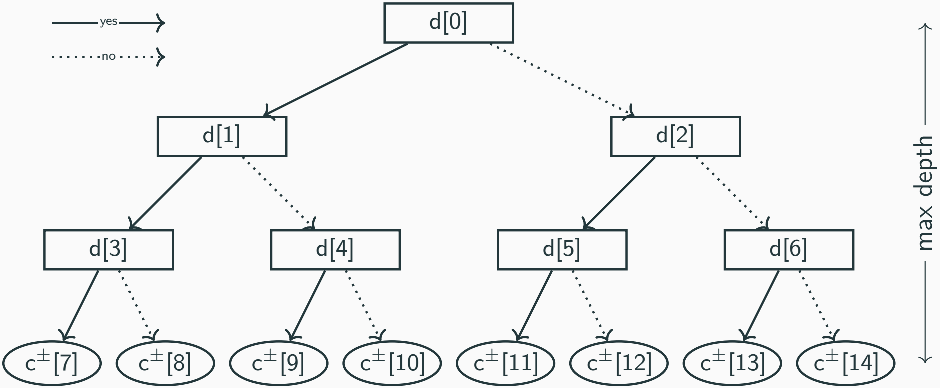
Model - Decision variables

KU LEUVEN



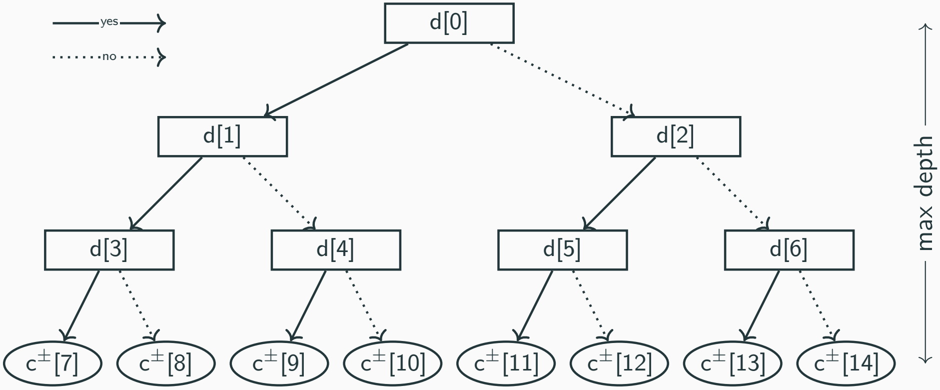
$$dom(d[i]) = \{1, ..., n\}$$

Model - Decision variables



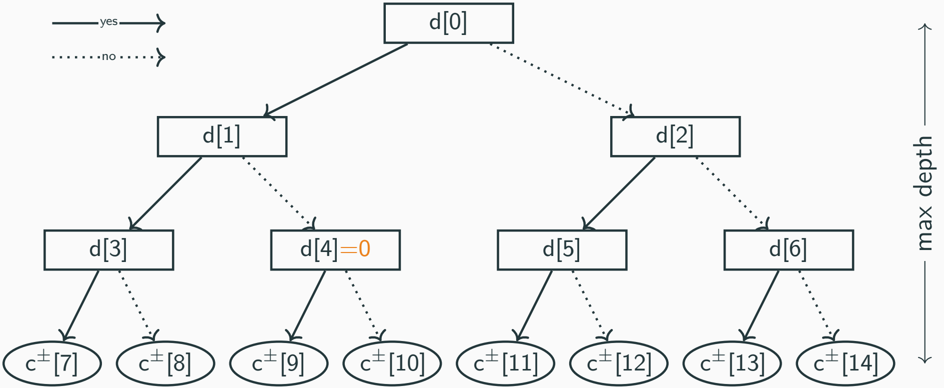
$dom(d[i]) = \{1, ..., n\}$ $dom(c[i]) = \{0, ..., N\}$

Model - Unbalanced Tree



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

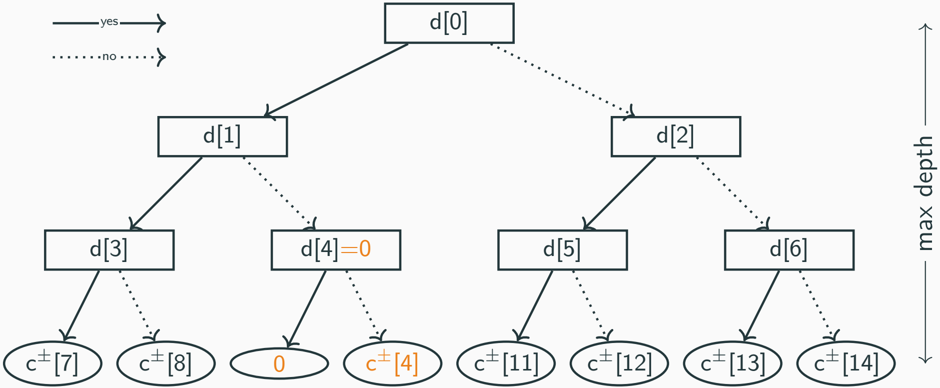
Model - Unbalanced Tree



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

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

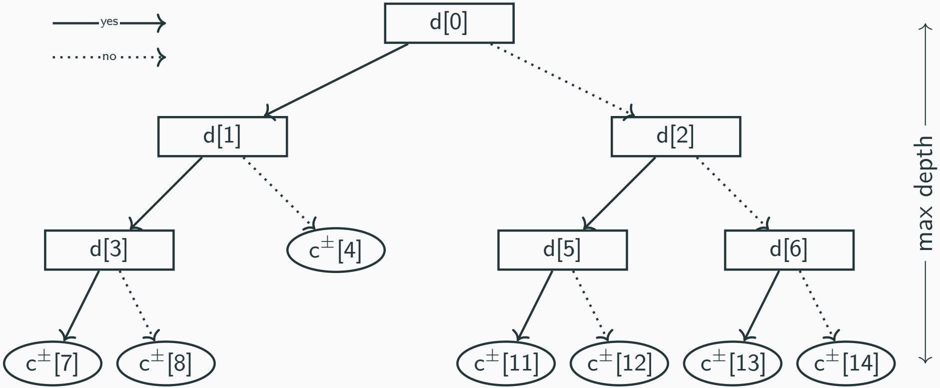
Model - Unbalanced Tree



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

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

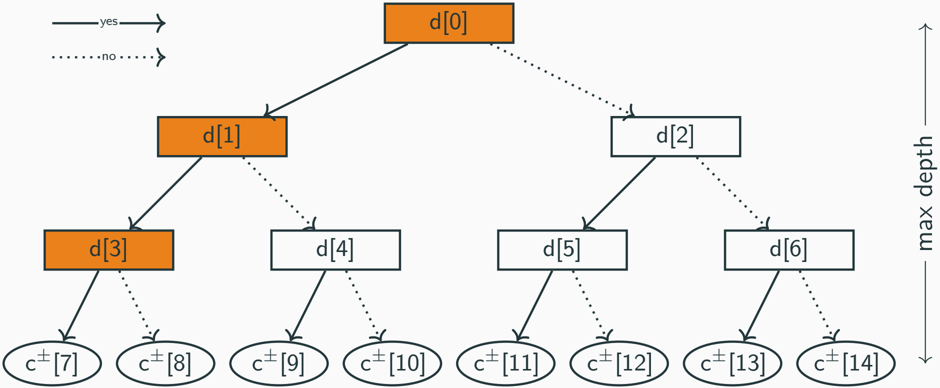
Model - Unbalanced Tree



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

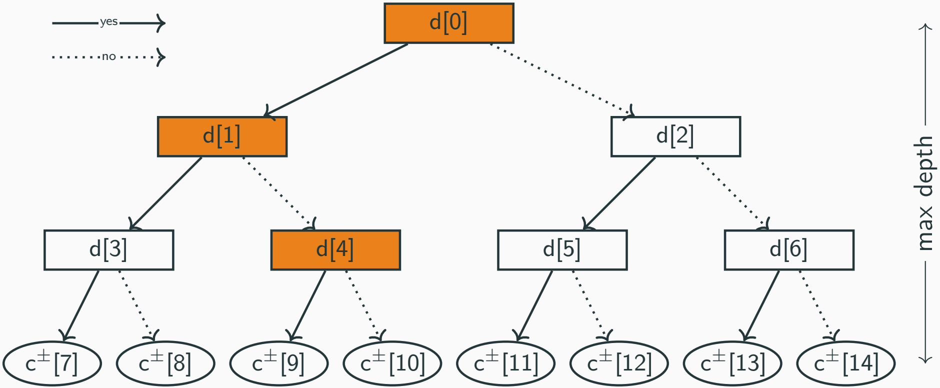
$dom(c[i]) = \{0, ..., N\}$

Model - AllDifferentExcept0



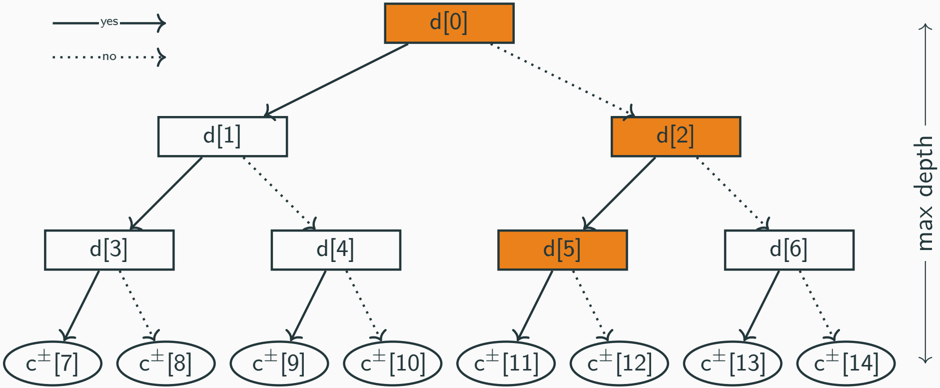
$dom(d[i]) = \{0, 1, ..., n\}$ $dom(c[i]) = \{0, ..., N\}$

Model - AllDifferentExcept0



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

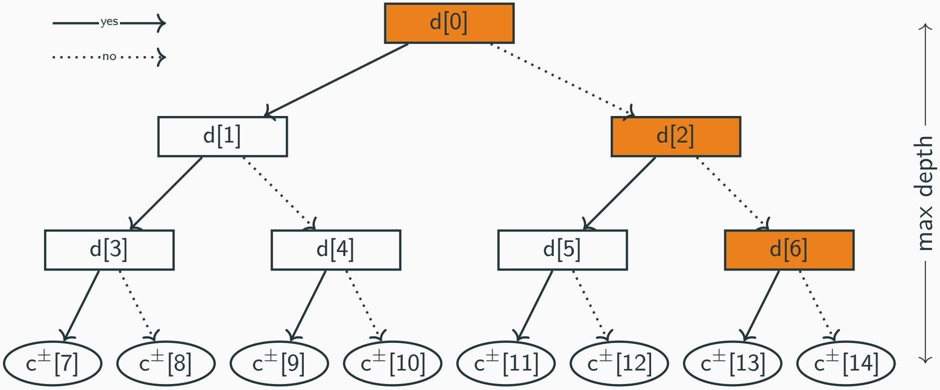
Model - AllDifferentExcept0



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

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

Model - AllDifferentExcept0



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

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

Model - Coversize

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.

Model - Coversize

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.

Model - Coversize

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.

Model - Coversize

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

Model - Coversize

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

Model - Coversize

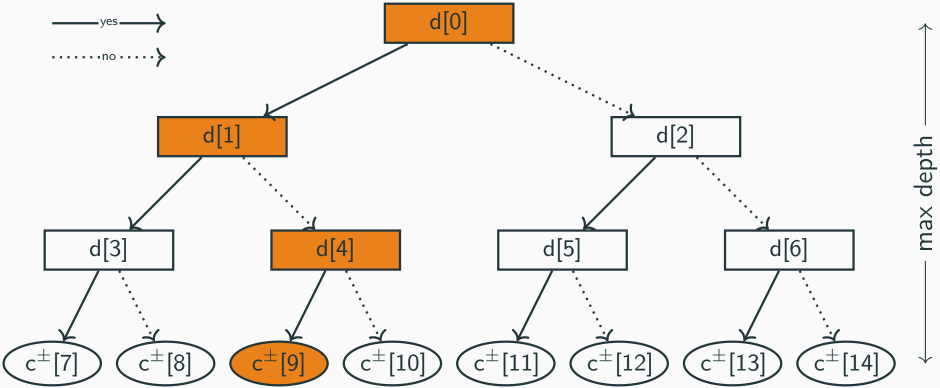
KU LEUVEN

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~~

Model - Coversize



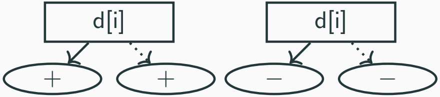
$Coversize(\{d[0], d[4]\}, \{d[1]\}, c^{+}[9])$ $\qquad$ $Coversize(\{d[0], d[4]\}, \{d[1]\}, c^{-}[9])$

Model - Other constraints and auxilliary variables

- constraints imposing minimum at leaf

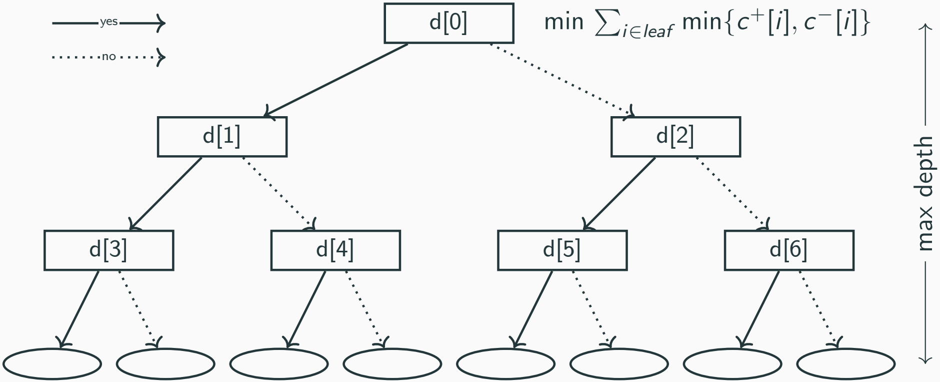
$$c^+[i] + c^-[i] \geq N_{min}$$

- constraints avoiding useless decisions



- redundant constraints improving speed

Model - Objective



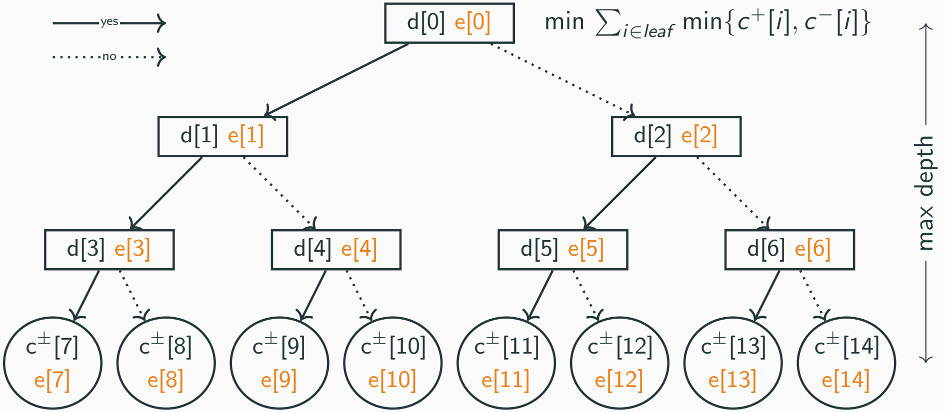
$$\min \sum_{i \in leaf} \min\{c^+[i], c^-[i]\}$$

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

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

Model - Objective

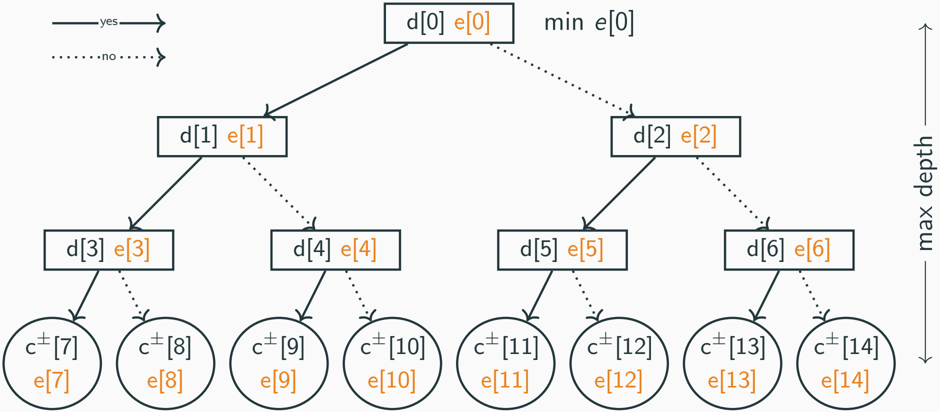
KU LEUVEN



$$\min \sum_{i \in leaf} \min\{c^+[i], c^-[i]\}$$

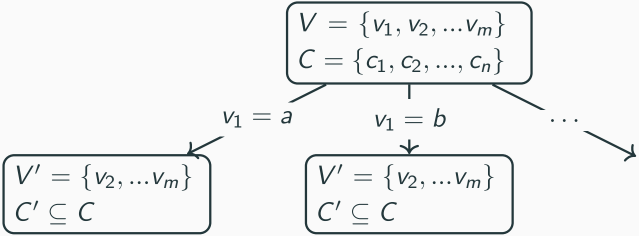
$dom(d[i]) = \{0, 1, ..., n\}$ $\quad dom(c[i]) = \{0, ..., N\}$ $\quad dom(e[i]) = \{0, ..., N\}$

Model - Objective



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

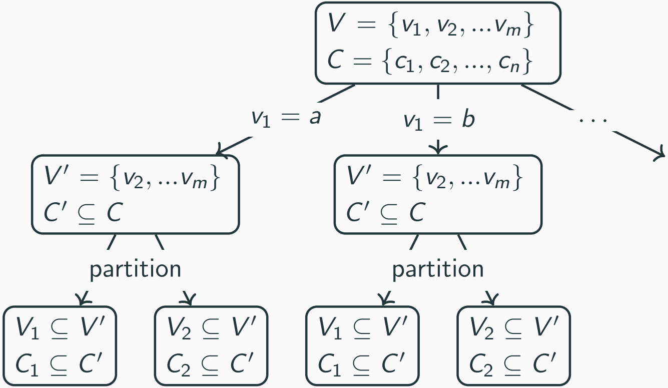
AND/OR search tree



OR nodes

$\mathrm{SOL} = \mathrm{SOL}_1$ or SOL_2 or $\ldots$

AND/OR search tree



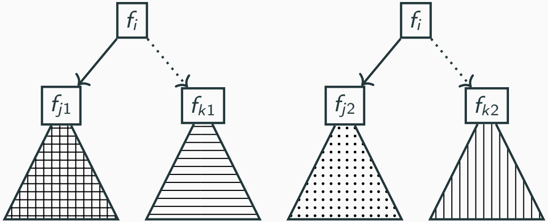
OR nodes

$\text{SOL} = \text{SOL}_1$ or SOL_2 or ...

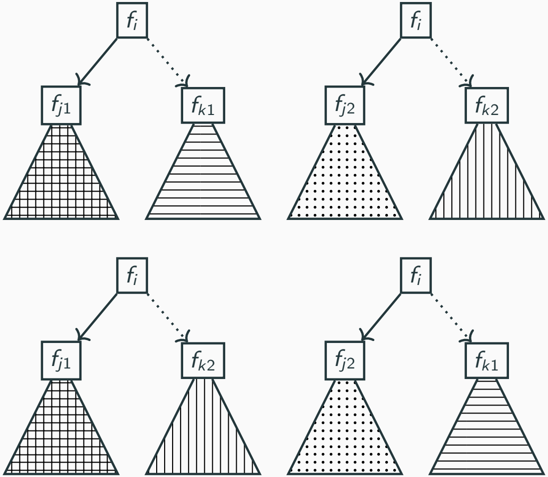
AND nodes

$\text{SOL} = \text{SOL}_1$ and SOL_2 and ...

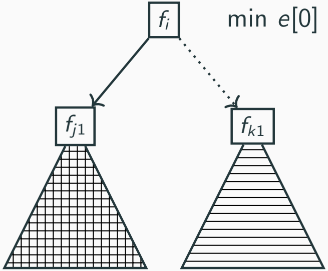
Search - AND/OR search tree



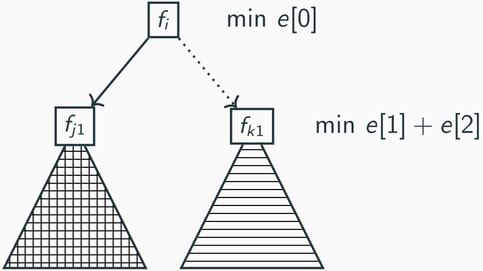
Search - AND/OR search tree



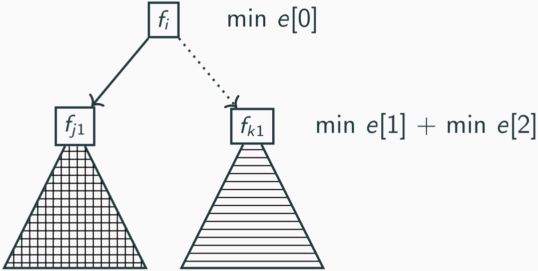
Search - AND/OR search tree



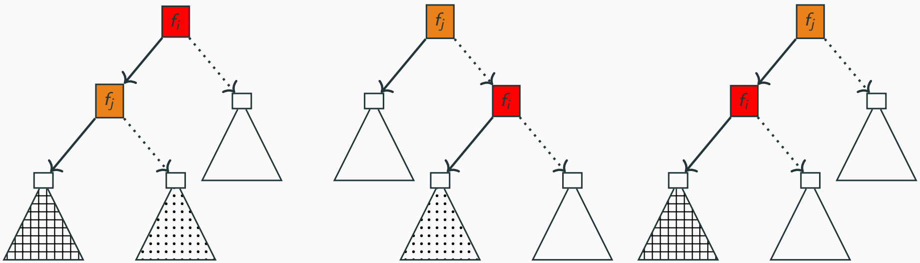
Search - AND/OR search tree



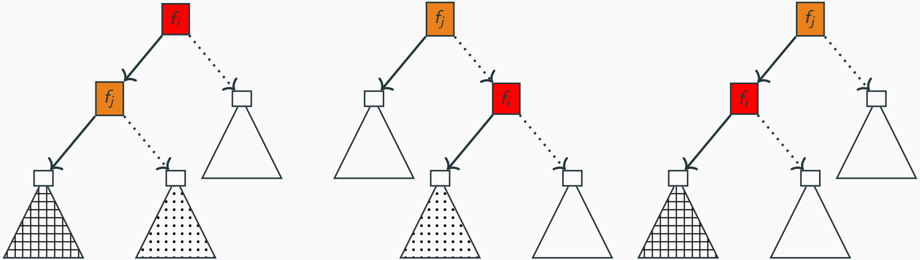
Search - AND/OR search tree

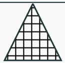
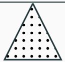


Search - Caching



Search - Caching



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

Experiments

	$N_{\min} = 1$			$N_{\min} = 5$			
	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

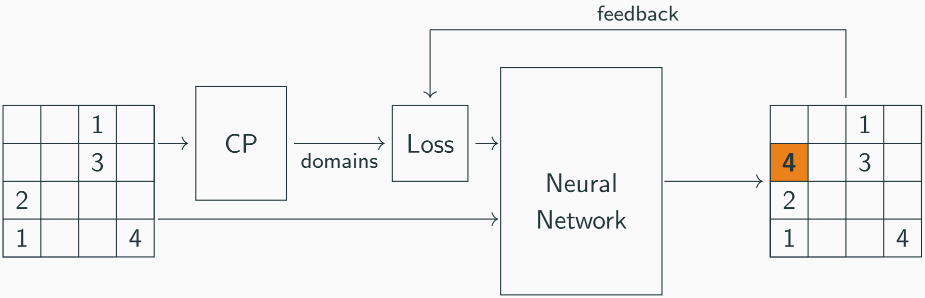
DL8: Dynamic programming approach using frequent itemsets mining

BinOCT: MIP-based approach running on CPLEX

[**Learning Optimal Decision Trees using Constraint Programming**, H.Verhaeghe, S.Nijssen, G.Pesant, CG.Quimper, P.Schaus, Constraint Journal, 2020]

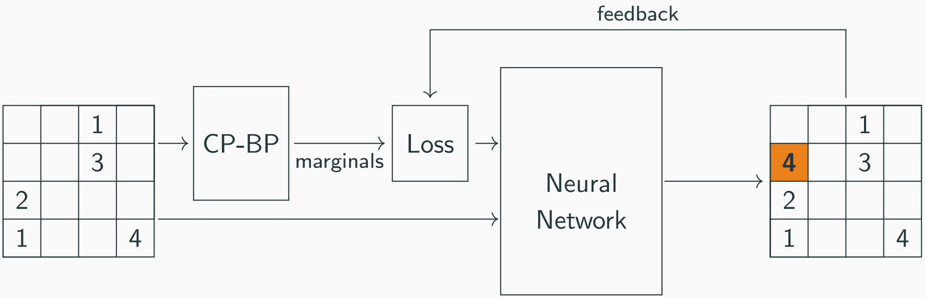
When CP helps ML: CP-BP for learning

The problem: Make NN output satisfy constraints

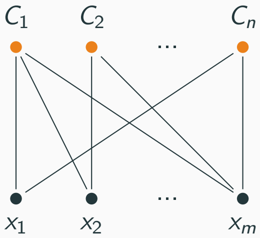


M. Silvestri, M. Lombardi, and M. Milano. "Injecting domain knowledge in neural networks: a controlled experiment on a constrained problem". In CPAIOR 2021.

The problem: Make NN output satisfy constraints



Belief propagation

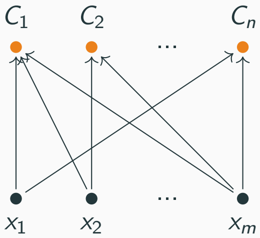


G. Pesant. "From support propagation to belief propagation in constraint programming". In JAIR 2019.

Belief propagation

KU LEUVEN

Message passing between variables and constraints

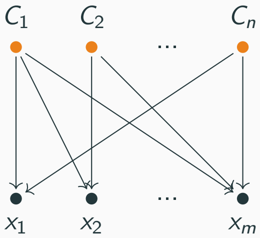


G. Pesant. "From support propagation to belief propagation in constraint programming". In JAIR 2019.

Belief propagation

KU LEUVEN

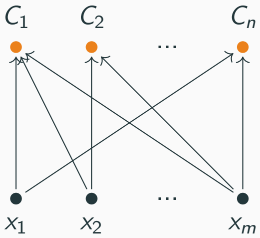
Message passing between variables and constraints



G. Pesant. "From support propagation to belief propagation in constraint programming". In JAIR 2019.

Belief propagation

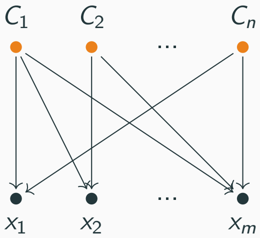
Message passing between variables and constraints



G. Pesant. "From support propagation to belief propagation in constraint programming". In JAIR 2019.

Belief propagation

Message passing between variables and constraints



G. Pesant. "From support propagation to belief propagation in constraint programming". In JAIR 2019.

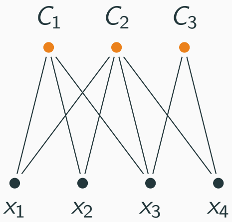
Belief propagation: Example

Variables: x_a, x_b, x_c, x_d

- $D_{x_a} = D_{x_a} = D_{x_a} = D_{x_a} = \{1, 2, 3, 4\}$

Constraints:

- $C_1 := AllDifferent(x_a, x_b, x_c)$
- $C_2 := x_a + x_b + x_c + x_d = 7$
- $C_3 := x_c \leq x_d$



True marginals (target)

	1	2	3	4
θ_{x_a}	0	.5	.5	0
θ_{x_b}	0	.5	.5	0
θ_{x_c}	1	0	0	0
θ_{x_d}	1	0	0	0

Two solutions: $(2, 3, 1, 1)$ and $(3, 2, 1, 1)$

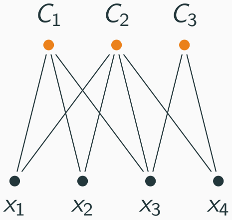
Belief propagation: Example

Variables: x_a, x_b, x_c, x_d

- $D_{x_a} = D_{x_a} = D_{x_a} = D_{x_a} = \{1, 2, 3, 4\}$

Constraints:

- $C_1 := AllDifferent(x_a, x_b, x_c)$
- $C_2 := x_a + x_b + x_c + x_d = 7$
- $C_3 := x_c \leq x_d$



Two solutions: $(2, 3, 1, 1)$ and $(3, 2, 1, 1)$

True marginals (target)

	1	2	3	4
θ_{x_a}	0	.5	.5	0
θ_{x_b}	0	.5	.5	0
θ_{x_c}	1	0	0	0
θ_{x_d}	1	0	0	0

Marginals at iteration 0

	1	2	3	4
$\hat{\theta}_{x_a}$	.25	.25	.25	.25
$\hat{\theta}_{x_b}$	.25	.25	.25	.25
$\hat{\theta}_{x_c}$	.25	.25	.25	.25
$\hat{\theta}_{x_d}$	.25	.25	.25	.25

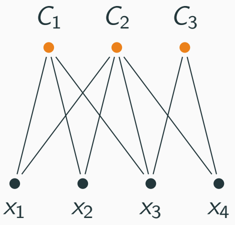
Belief propagation: Example

Variables: x_a, x_b, x_c, x_d

- $D_{x_a} = D_{x_a} = D_{x_a} = D_{x_a} = \{1, 2, 3, 4\}$

Constraints:

- $C_1 := AllDifferent(x_a, x_b, x_c)$
- $C_2 := x_a + x_b + x_c + x_d = 7$
- $C_3 := x_c \leq x_d$



Two solutions: $(2, 3, 1, 1)$ and $(3, 2, 1, 1)$

True marginals (target)

	1	2	3	4
θ_{x_a}	0	.5	.5	0
θ_{x_b}	0	.5	.5	0
θ_{x_c}	1	0	0	0
θ_{x_d}	1	0	0	0

Marginals at iteration 1

	1	2	3	4
$\hat{\theta}_{x_a}$	.50	.30	.15	.05
$\hat{\theta}_{x_b}$	.50	.30	.15	.05
$\hat{\theta}_{x_c}$	.62	.28	.09	.01
$\hat{\theta}_{x_d}$	.29	.34	.26	.11

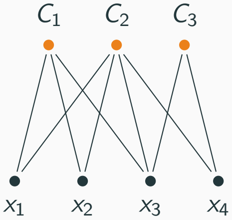
Belief propagation: Example

Variables: x_a, x_b, x_c, x_d

- $D_{x_a} = D_{x_a} = D_{x_a} = D_{x_a} = \{1, 2, 3, 4\}$

Constraints:

- $C_1 := AllDifferent(x_a, x_b, x_c)$
- $C_2 := x_a + x_b + x_c + x_d = 7$
- $C_3 := x_c \leq x_d$



Two solutions: $(2, 3, 1, 1)$ and $(3, 2, 1, 1)$

True marginals (target)

	1	2	3	4
θ_{x_a}	0	.5	.5	0
θ_{x_b}	0	.5	.5	0
θ_{x_c}	1	0	0	0
θ_{x_d}	1	0	0	0

Marginals at iteration 10

	1	2	3	4
$\hat{\theta}_{x_a}$	.01	.52	.46	.01
$\hat{\theta}_{x_b}$	.01	.52	.46	.01
$\hat{\theta}_{x_c}$	.98	.02	.00	.00
$\hat{\theta}_{x_d}$	.90	.10	.00	.00

Domains vs Marginals

Domains

	1	2	3	4
D_{x_a}	0	1	1	0
D_{x_b}	0	1	1	0
D_{x_c}	1	0	0	0
D_{x_d}	1	0	0	0

Marginals

	1	2	3	4
$\hat{\theta}_{x_a}$	.01	.52	.46	.01
$\hat{\theta}_{x_b}$	.01	.52	.46	.01
$\hat{\theta}_{x_c}$	.98	.02	.00	.00
$\hat{\theta}_{x_d}$	.90	.10	.00	.00

Loss function - supervised

$$Loss(x,y) = \underbrace{-\langle y, \log(\frac{1}{Z}\overbrace{f(x)}^{\hat{y}})\rangle}_{\texttt{cross entropy}} + \overbrace{\lambda}^{\texttt{weight}} \cdot \underbrace{t(x)}_{\texttt{CP feedback}}$$

Domains

$$t(x) = L_1(x, C) = \sum_k |C_k(x) - f_k(x)|$$

$$t(x) = L_2(x, C) = \sum_k (C_k(x) - f_k(x))^2$$

$$C_k(x) \in \{0, 1\}$$

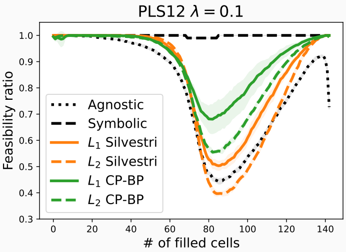
Marginals

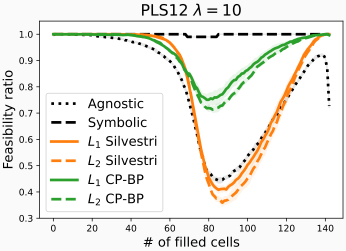
$$t(x) = L_1(x, \hat{\theta}) = \sum_k |\hat{\theta}_k(x) - f_k(x)|$$

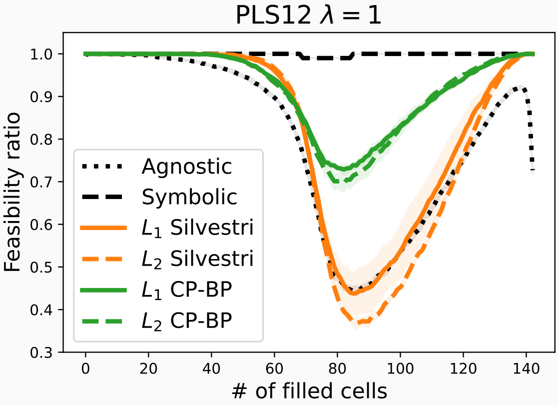
$$t(x) = L_2(x, \hat{\theta}) = \sum_k (\hat{\theta}_k(x) - f_k(x))^2$$

$$\hat{\theta}_k(x) \in [0, 1]$$

Results - supervised







Loss function - unsupervised

$$Loss(x,y) = \cancel{\underbrace{-\langle y, \log(\frac{1}{Z}\overbrace{f(x)}^{\hat{y}})\rangle}_{\texttt{cross entropy}}} + \cancel{\overbrace{\lambda}^{\texttt{weight}}} \cdot \underbrace{t(x)}_{\texttt{CP feedback}}$$

Domains

$$t(x) = L_1(x, C) = \sum_k |C_k(x) - f_k(x)|$$

$$t(x) = L_2(x, C) = \sum_k (C_k(x) - f_k(x))^2$$

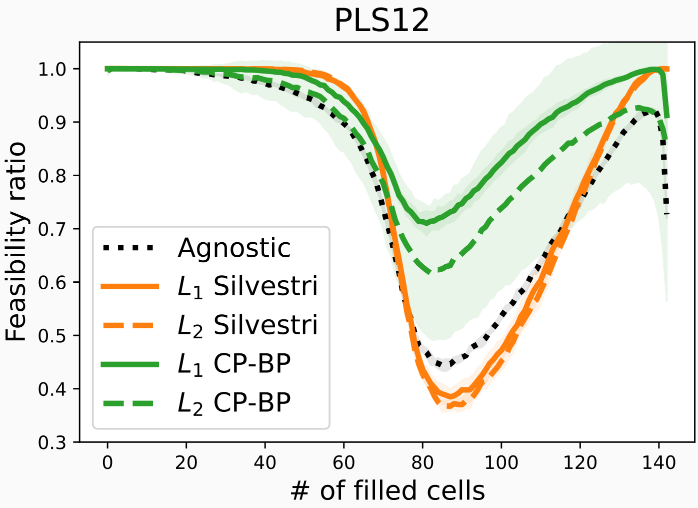
$$C_k(x) \in \{0, 1\}$$

Marginals

$$t(x) = L_1(x, \hat{\theta}) = \sum_k |\hat{\theta}_k(x) - f_k(x)|$$

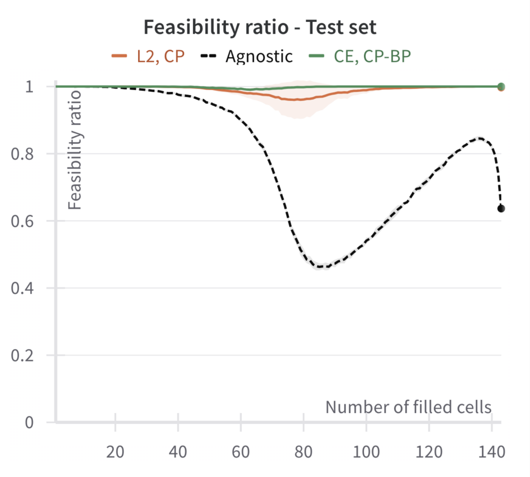
$$t(x) = L_2(x, \hat{\theta}) = \sum_k (\hat{\theta}_k(x) - f_k(x))^2$$

$$\hat{\theta}_k(x) \in [0, 1]$$


PLS12
Feasibility ratio
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
120
140
# of filled cells
Agnostic
L_1 Silvestri
L_2 Silvestri
L_1 CP-BP
L_2 CP-BP

Results - unsupervised - improved NN

KU LEUVEN

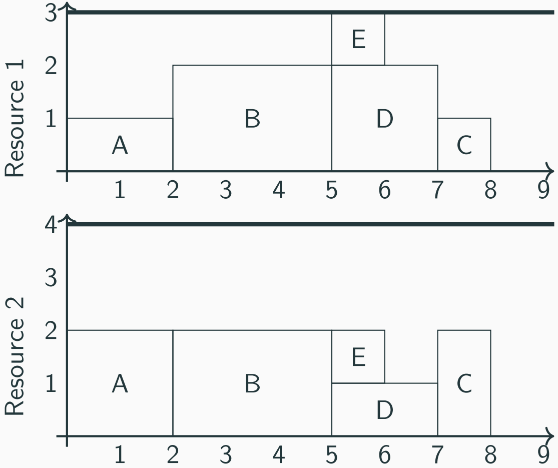


When ML helps CP: RCPSP using GNNs

The problem: improving RCPSP solving

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

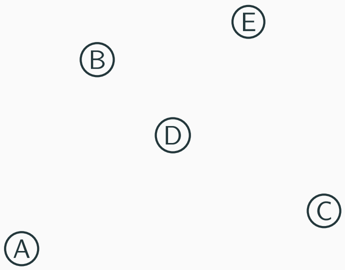
$C_{r_1} = 3$ and $C_{r_2} = 4$



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

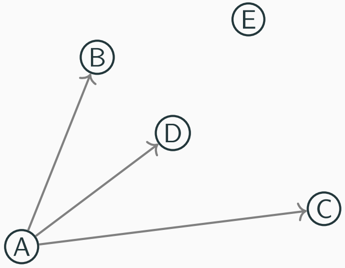
$C_{r_1} = 3$ and $C_{r_2} = 4$



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

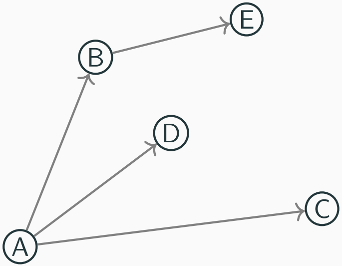
$C_{r_1} = 3$ and $C_{r_2} = 4$



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

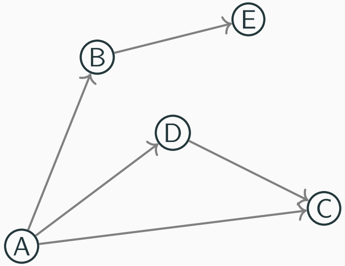
$C_{r_1} = 3$ and $C_{r_2} = 4$



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

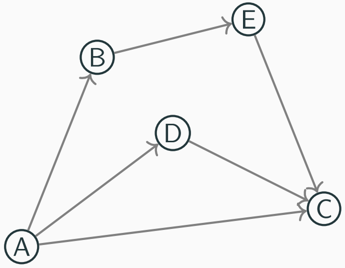
$C_{r_1} = 3$ and $C_{r_2} = 4$



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 $C_{r_2} = 4$

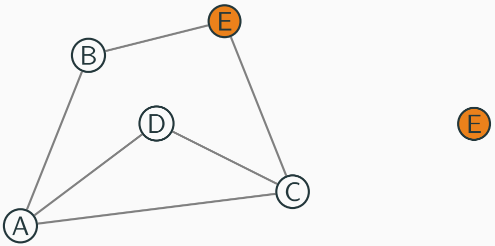


Main principle: for each node, creating an embedding of its neighborhood

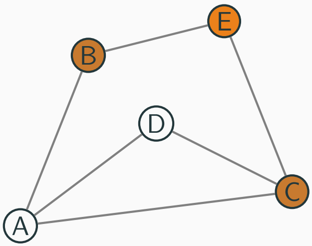
Tasks:

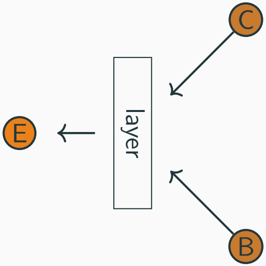
- Graph classification
- Node prediction
- Link prediction

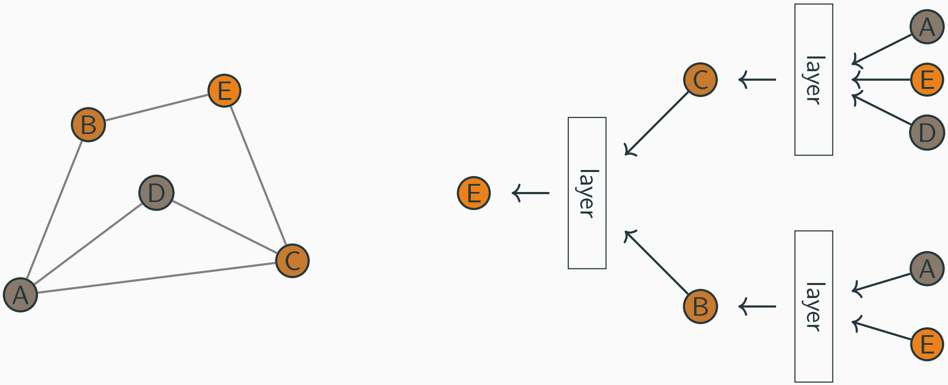
Graph Neural Networks



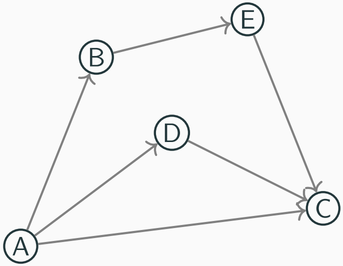
Graph Neural Networks

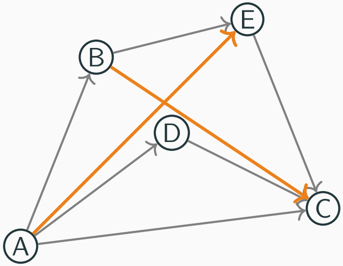




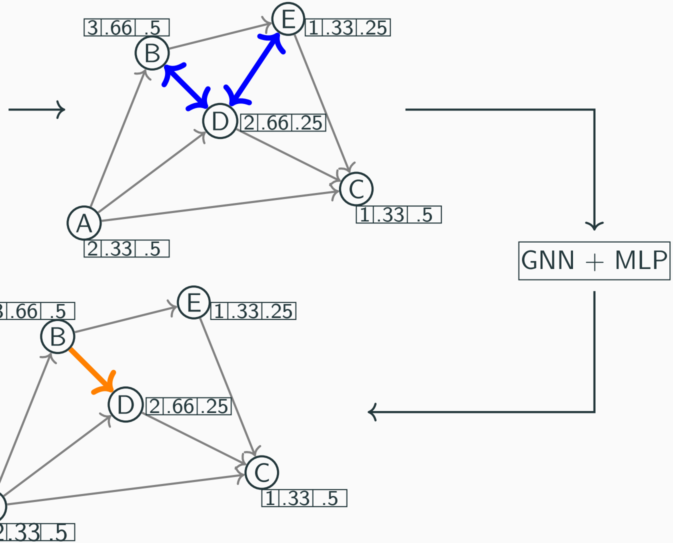

B
E
D
A
C
E
layer
C
B
layer
A
E
D
layer
A
E

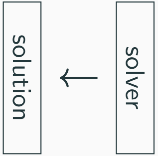
Transitive closure





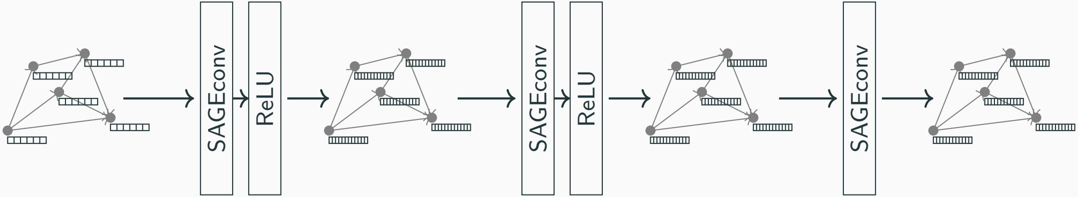
Task	p_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C





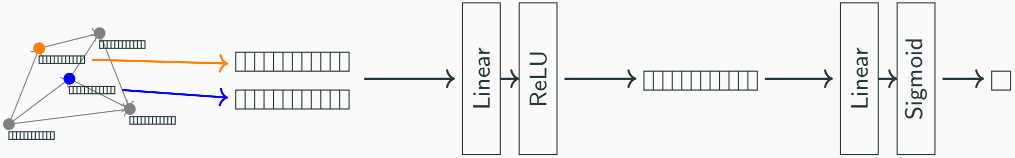
Our GNN

Goal: creates, for each node, embedding of the neighborhood



Our MLP

Goal: evaluate, given a candidate edge, its likeliness to exist



(preliminary) Results

Two usages of the learned precedences:

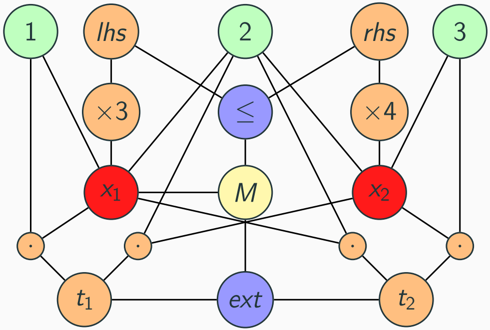
- additional constraints:
  - reduces search space
  - restriction of the problem
  - improve solution for a few instances
- task ordering:
  - preserve solutions
  - best first solution

When ML helps CP: Generic Graph Representation

Generic Graph Representation

KU LEUVEN

$$\begin{aligned} \max\ & x_1 \\ \text{s.t.}\ & 3x_1 \le 4x_2 \\ & \textsc{table}([x_1, x_2], [(1,2),(2,3)]) \\ & x_1 \in \{1,2\} \\ & x_2 \in \{2,3\} \end{aligned}$$



Results - Prediction accuracy

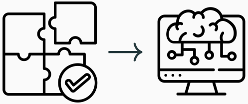
Task: Determine if it is sat or unsat

Architecture	SAT	TSP-Ext	TSP-Elem	Col	Knap
Problem-specific	94.3%	**96.3%**		77.0 %	**98.8%**
Tripartite [Marty et. al. 2023]	50.0%	50.0%		**84.6%**	50.0%
Ours	**94.4%**	84.5%	91.4%	84.4%	97.9%

[**Towards a Generic Representation of Combinatorial Problems for Learning-Based Approaches**, L.Boisvert, H.Verhaeghe, Q.Cappart, CPAIOR, 2024]

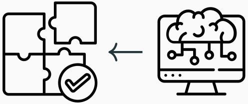
Conclusion

Conclusion - CP/ML synergies



CP helps ML

- Help with satisfying (hard) constraint



ML helps CP

- Deal with (big) data

ACP summer school 2023 - Recordings

KU LEUVEN

https://youtube.com/playlist?list=PLcByDTr7vRTYJ2s6DL-3bzjGwtQif33y3

Thank you for listening!

Any questions?

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