

LEARNING OPTIMAL DECISION TREES USING CONSTRAINT PROGRAMMING

CP2019

Hélène Verhaeghe¹, Siegfried Nijssen¹, Gilles Pesant², Claude-Guy Quimper³, and Pierre Schaus¹

4 September 2019

¹ICTEAM, UCLouvain, Place Sainte Barbe 2, 1348 Louvain-la-Neuve, Belgium, {*firstname.lastname*}@uclouvain.be

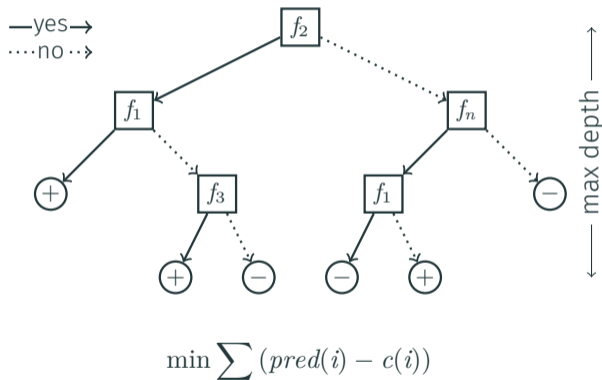
²Polytechnique Montréal, Montréal, Canada, *gilles.pesant@polymtl.ca*

³Université Laval, Québec, Canada, *claudio – guy.quimper@ift.ulaval.ca*



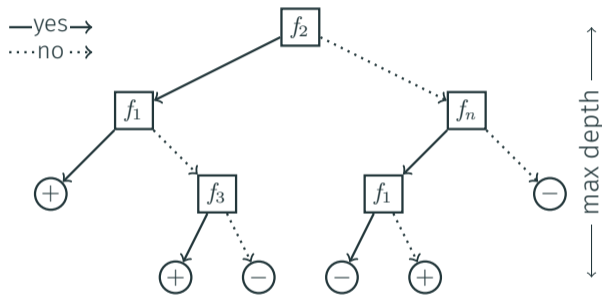
Database					
f_1	f_2	f_3	\dots	f_n	c
1	0	1	\dots	1	+
0	1	0	\dots	1	-
1	1	0	\dots	0	+
0	0	0	\dots	0	+
1	0	0	\dots	0	+
0	1	1	\dots	1	-
1	1	1	\dots	0	-
\vdots	\vdots	\vdots	\ddots	\vdots	\vdots
1	1	1	\dots	1	+

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	+



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	+

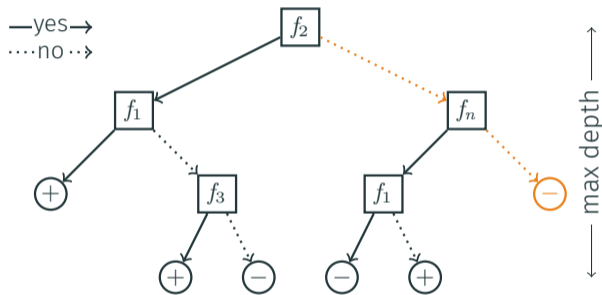
New sample					
0	0	1	...	0	?



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

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	-



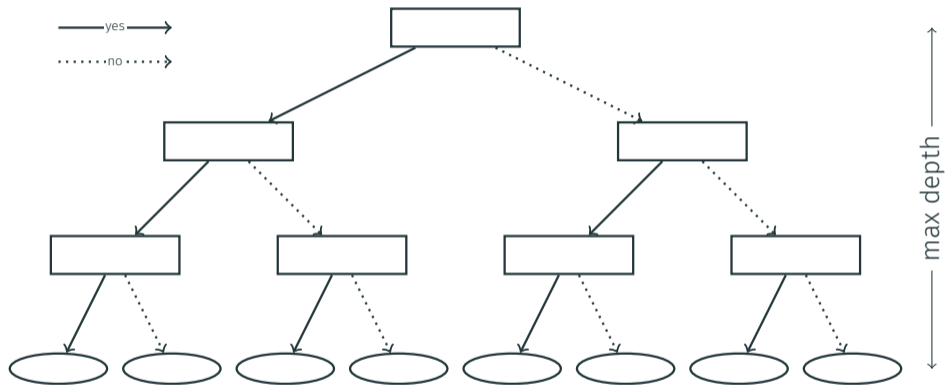
$$\min \sum (pred(i) - c(i))$$

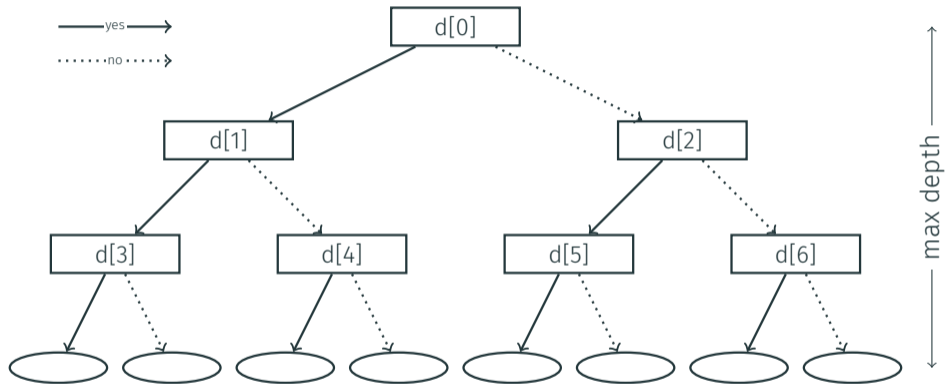
Greedy methods:

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

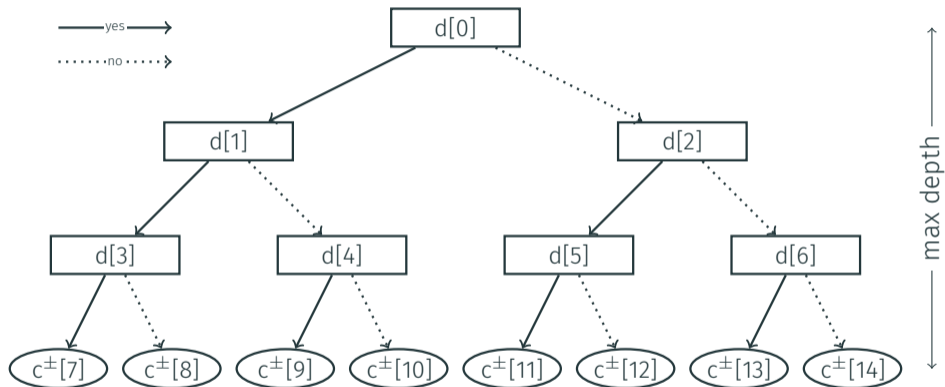
- Mining optimal decision trees from itemset lattices, Nijssen, S., Fromont, E., 2007
- Minimising decision tree size as combinatorial optimisation, Bessiere, C., Hebrard, E., O'Sullivan, B., 2009
- Optimal constraint-based decision tree induction from itemset lattices, Nijssen, S., Fromont, É., 2010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

CP MODEL



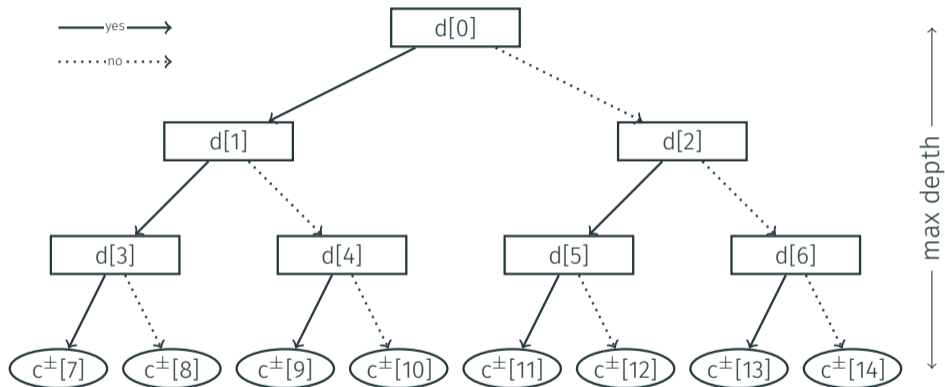


$$\text{dom}(d[i]) = \{1, \dots, n\}$$



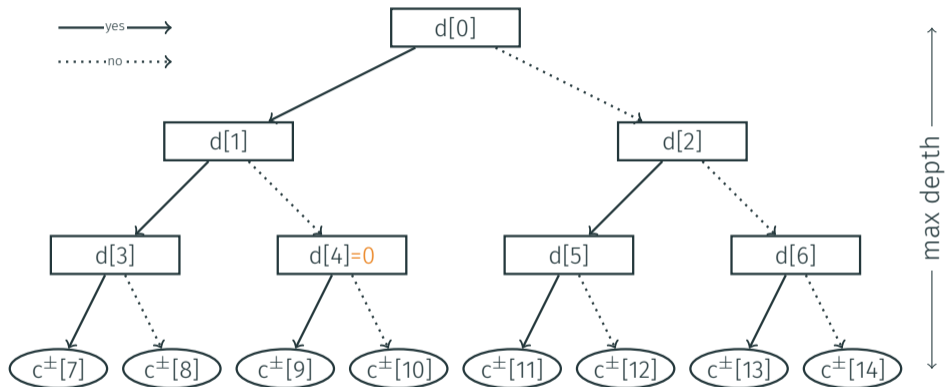
$$\text{dom}(d[i]) = \{1, \dots, n\}$$

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



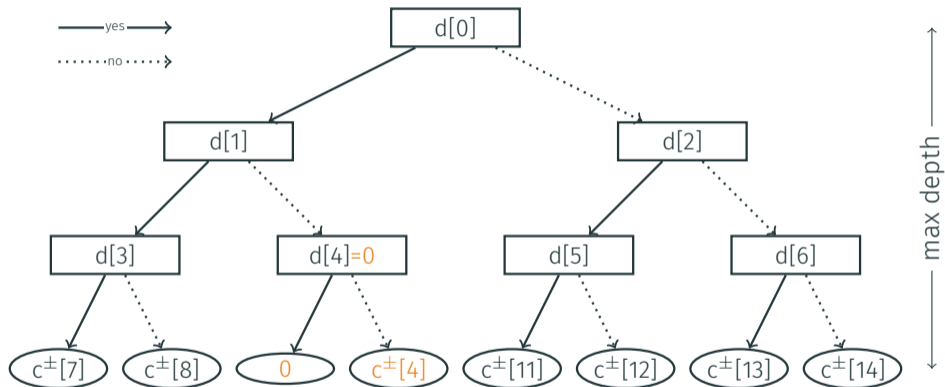
$$\text{dom}(d[i]) = \{0, 1, \dots, n\}$$

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



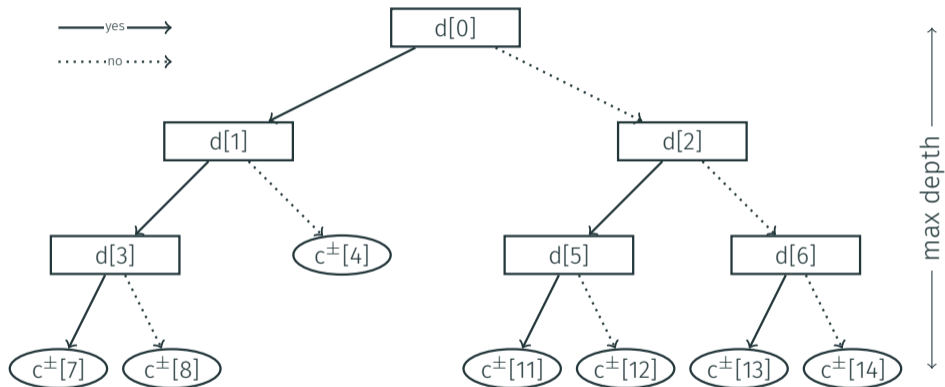
$$\text{dom}(d[i]) = \{0, 1, \dots, n\}$$

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



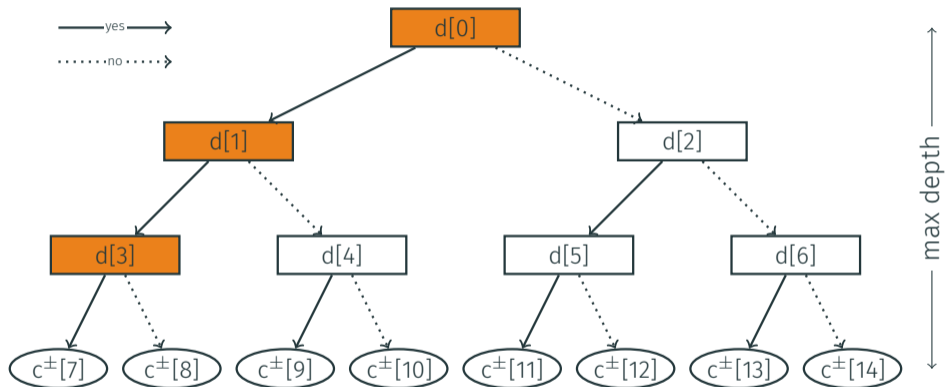
$$\text{dom}(d[i]) = \{0, 1, \dots, n\}$$

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



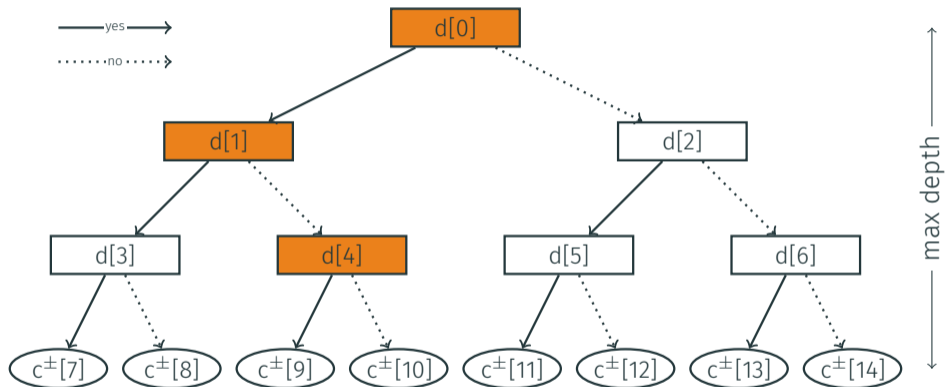
$$\text{dom}(d[i]) = \{0, 1, \dots, n\}$$

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



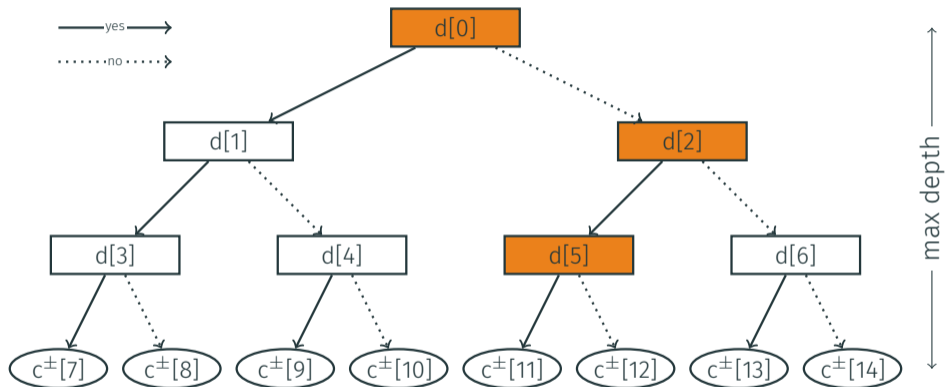
$$\text{dom}(d[i]) = \{0, 1, \dots, n\}$$

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



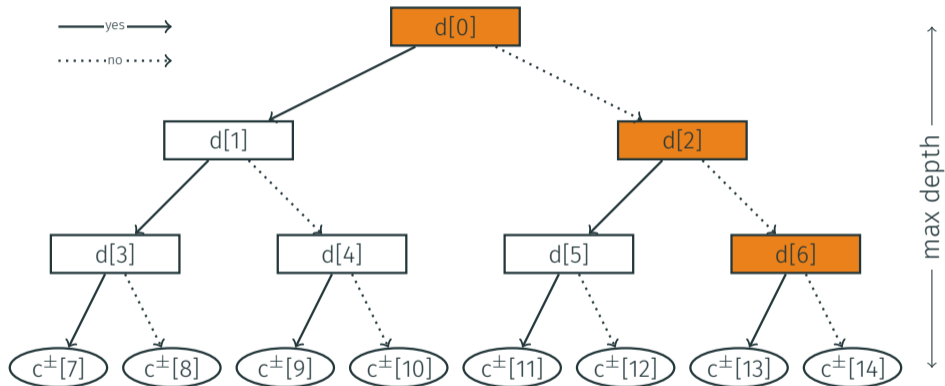
$$\text{dom}(d[i]) = \{0, 1, \dots, n\}$$

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



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

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



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

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

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	

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	

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	2

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	2

- 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

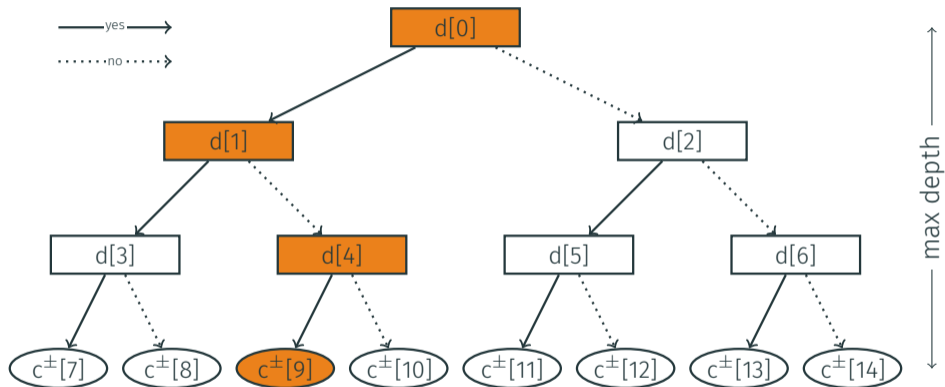
Features (Sparse)		Counter
y_1	y_2	
2	4	2

- 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



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

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

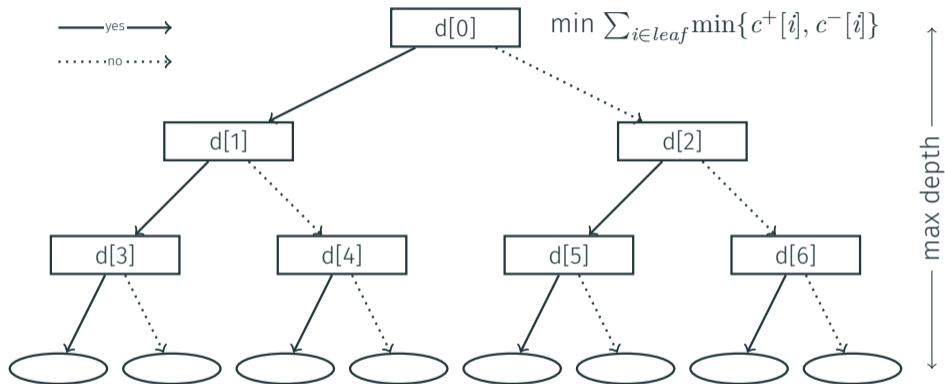
- constraints imposing minimum at leaf

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

- constraints avoiding useless decisions

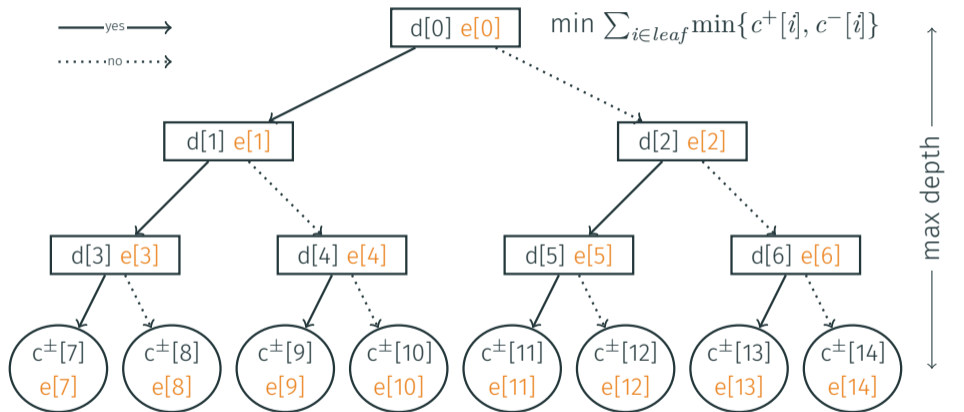


- redundant constraints improving speed



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

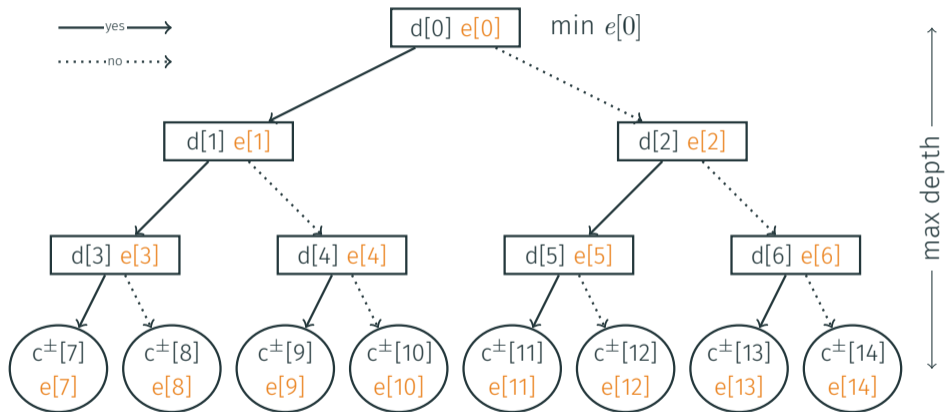
$$\text{dom}(c[i]) = \{0, \dots, N\}$$



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

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

$$\text{dom}(e[i]) = \{0, \dots, N\}$$



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

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

$$\text{dom}(e[i]) = \{0, \dots, N\}$$

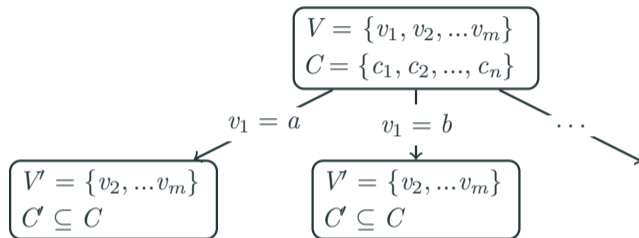
SEARCH

$$V = \{v_1, v_2, \dots, v_m\}$$

$$C = \{c_1, c_2, \dots, c_n\}$$

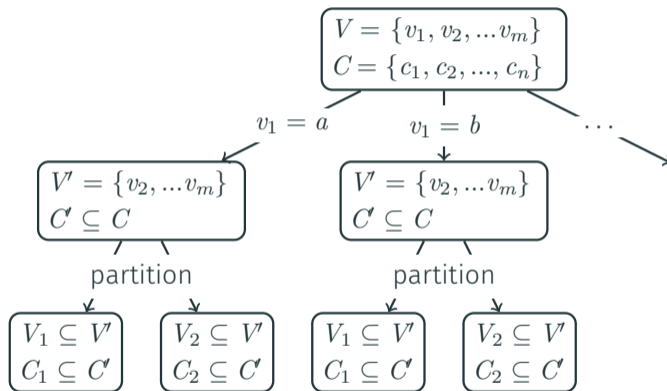
OR nodes

SOL = SOL₁ or SOL₂ or ...



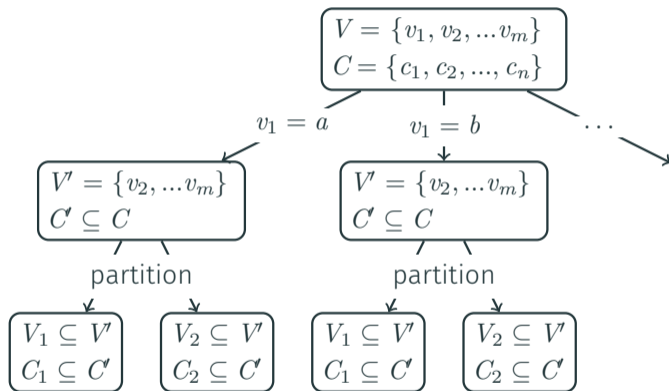
OR nodes

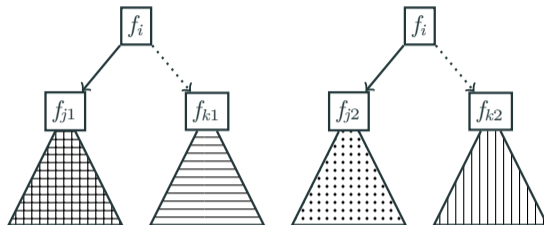
SOL = SOL₁ or SOL₂ or ...

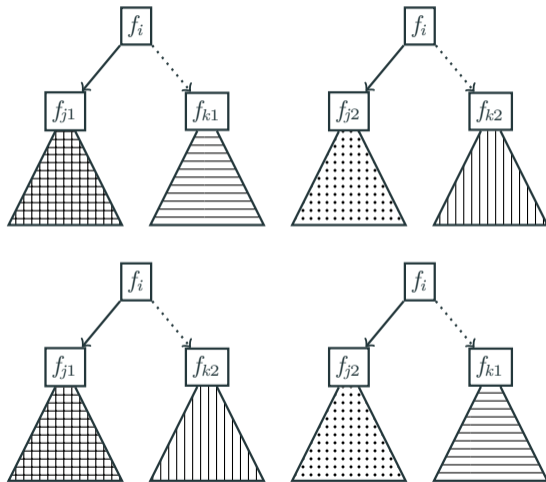


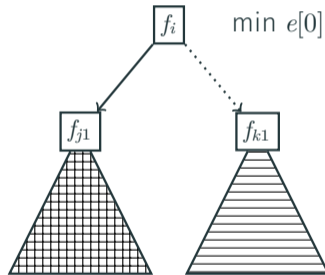
OR nodes

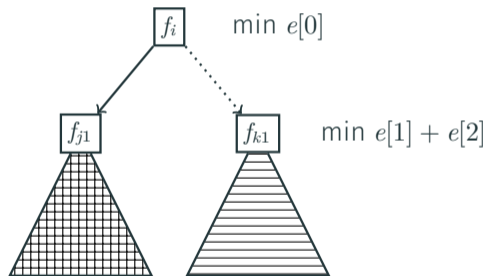
SOL = SOL₁ or SOL₂ or ...

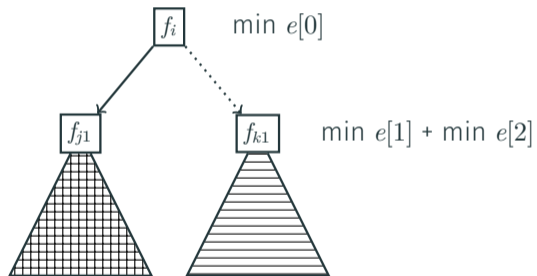
**OR nodes**SOL = SOL₁ or SOL₂ or ...**AND nodes**SOL = SOL₁ and SOL₂ and ...

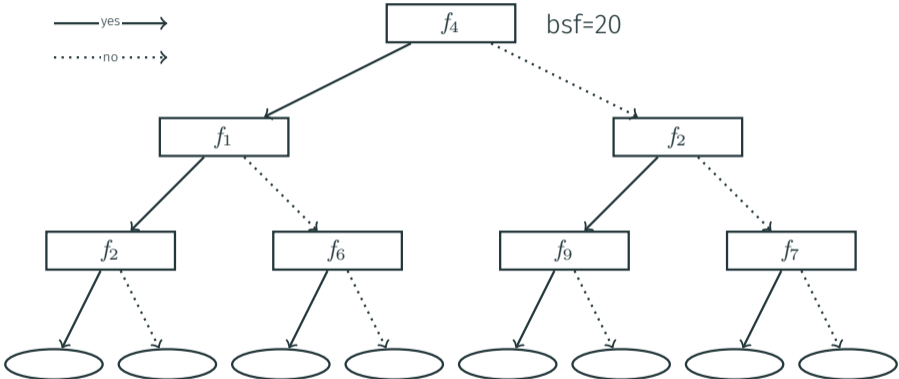


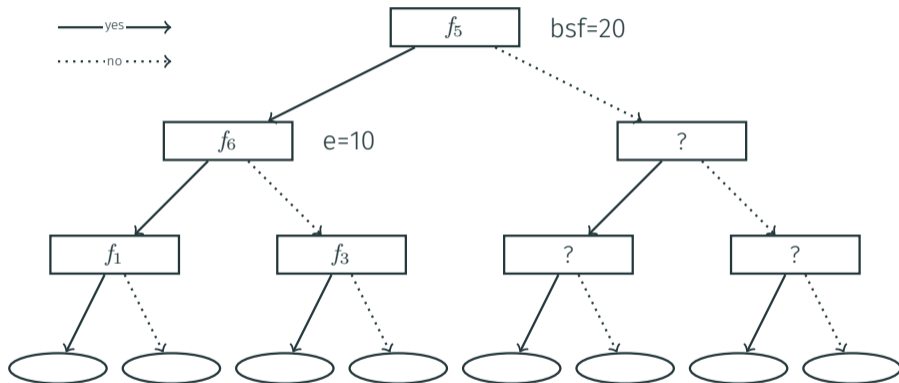


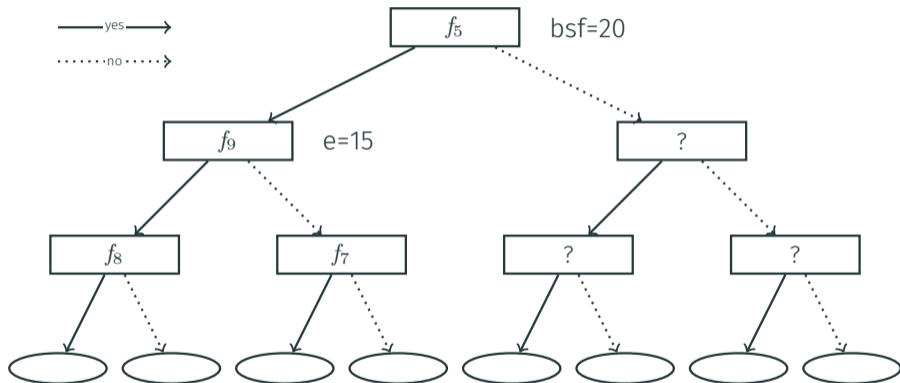


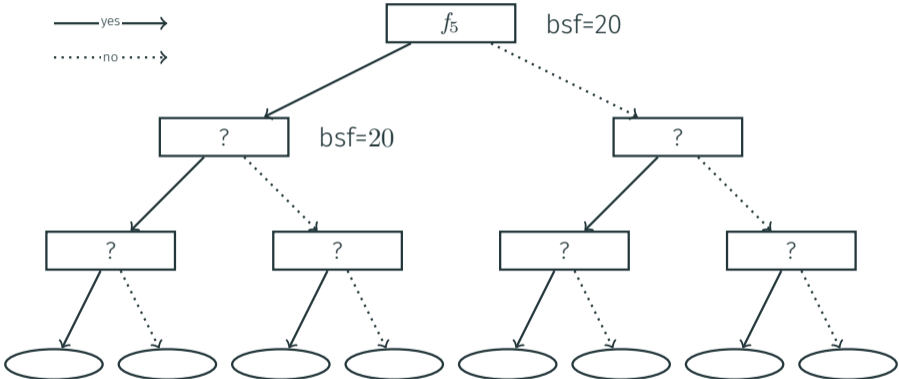


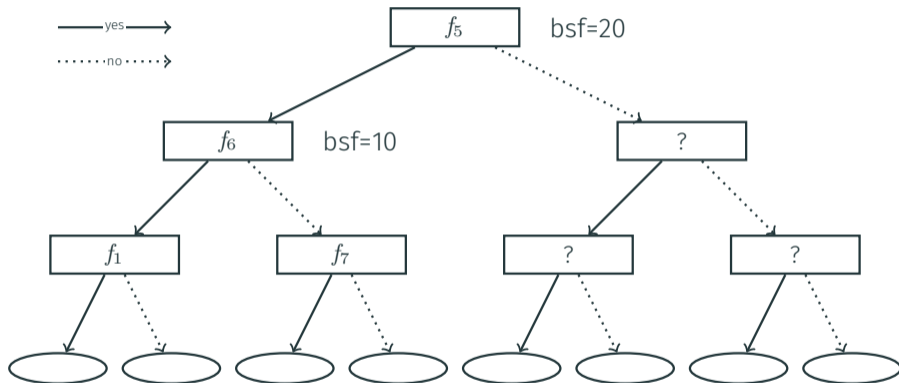


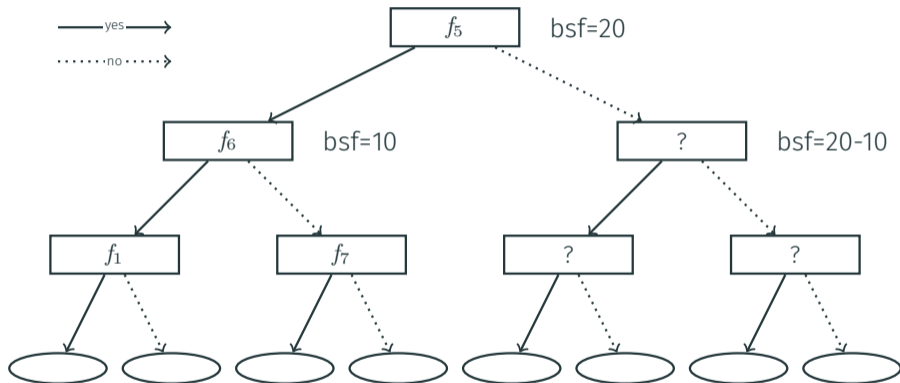


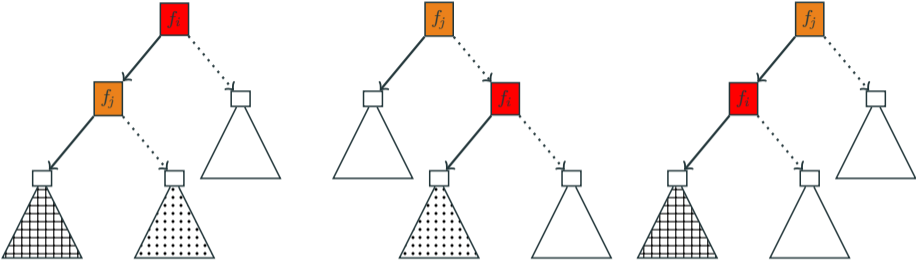












RESULTS

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

To summarize

- efficient method
- cp based
- exploits the structure of the problem
- anytime best solution

To go further

- multi-class decision trees
- continuous features through binarization
- other sum-based cost functions
- ...

Thank you for listening!

Any questions?