EPISODE 1: LA PROGRAMMATION PAR CONTRAINTES

Intelligence Artificielle: au delà de l'apprentissage automatique

---

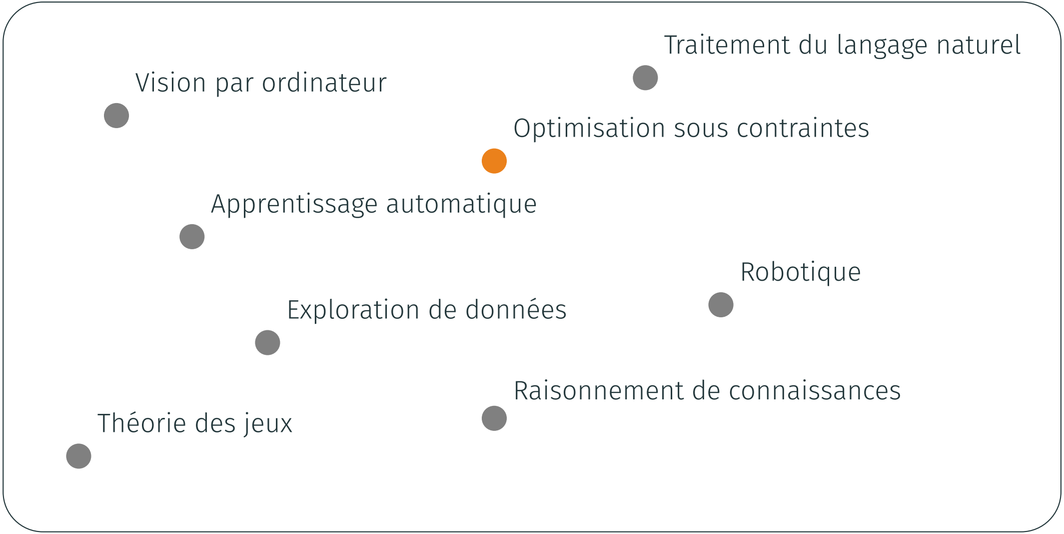
Hélène Verhaeghe

18 Novembre 2023

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

KU LEUVEN

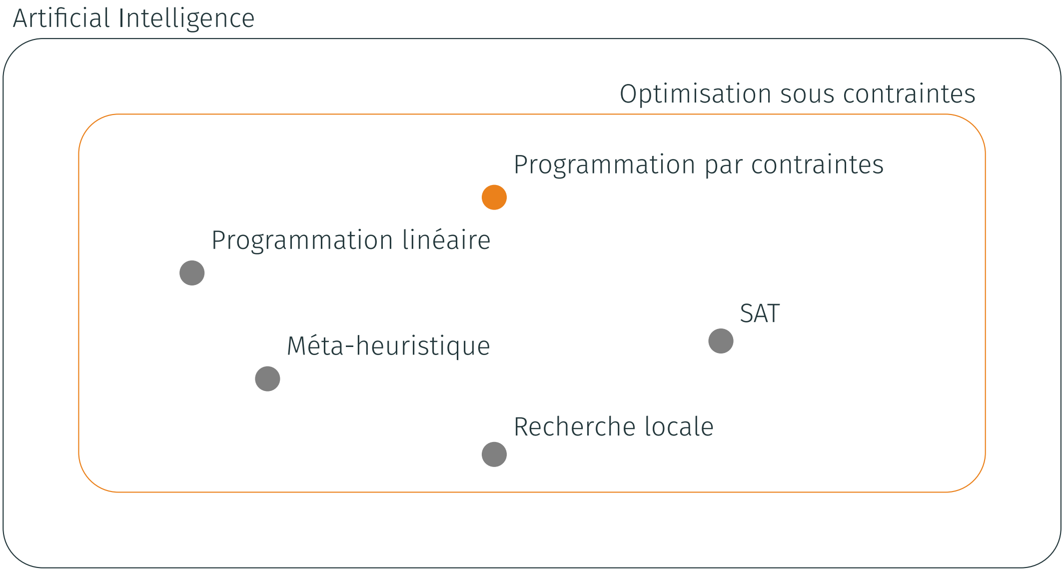
Artificial Intelligence



UTILITÉS DE L’OPTIMISATION SOUS CONTRAINTES

KU LEUVEN

PROGRAMMATION PAR CONTRAINTES PARMI L'IA



KU LEUVEN

Exploration complète	Exploration incomplète
· Séparation & évaluation (B&B)	· Recherche Locale
· **Programmation par contraintes**	· Recherche de voisinage large (LNS)
· Programmation entière	· Algorithme génétiques
· Résolution SAT	· Méta-heuristique
Avantage:	Avantage:
· Guaranties d'optimalité	· Rapide
Inconvénient:	Inconvénient:
· Prend du temps	· Pas de guaranties d'optimalité

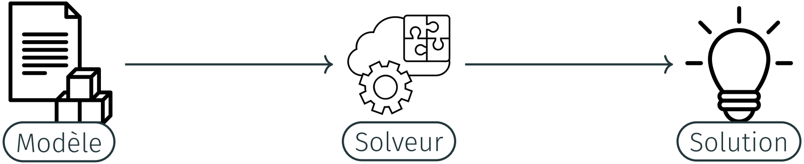
Quelle technique choisir? Dépend du but, obtenir la meilleure solution ou obtenir une bonne solution rapidement!

Qu’est-ce que la programmation par contraintes?

”En informatique, de toutes les approches en programmation, la programmation par contraintes se rapproche le plus de l’idéal : l’utilisateur décrit le problème, l’ordinateur le résout.” — E. Freuder

KU LEUVEN

En programmation par contraintes, on modélise de manière déclarative la solution souhaitée, l'ordinateur/le solveur trouve la solution.



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

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

KU LEUVEN

- Variables : $X, Y, Z, \ldots$
- Domaines : $\{1, 2\}$, $\{true, false\}$, $\{bleu, rouge, \ldots\}$
- Contraintes :
  - arithmétiques : $X + Y = Z$, $X \leq Y$
  - logiques : $A \wedge B$
  - globales : $AllDifferent(X, Y, Z)$, $Circuit(X_1, X_2, X_3)$

- Ecrit en Python
- Compatible NumPy
- Supporte une large variété de solveurs :
    - Or-Tools
    - Minizinc
    - PySDD
    - Z3
    - ...

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	

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

$$X_{i,j} = grid_{i,j} \quad \forall grid_{i,j} \neq 0$$

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	

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

$$X_{i,j} = grid_{i,j} \quad \forall grid_{i,j} \neq 0$$

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

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	

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

$$X_{i,j} = grid_{i,j} \quad \forall grid_{i,j} \neq 0$$

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

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

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	

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

$$X_{i,j} = grid_{i,j} \quad \forall grid_{i,j} \neq 0$$

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

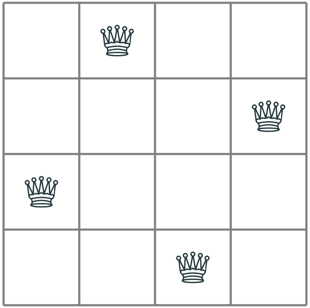
$$AllDifferent(X_{1,j}, \dots, X_{9,j}) \quad \forall 1 \leq j \leq 9$$

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

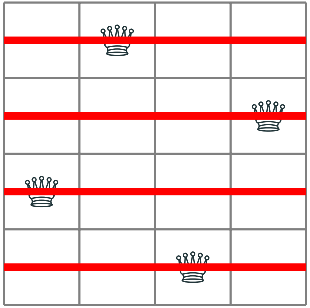
SUDOKU

	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	

```
grid = np.array([[0,2,0,5,0,1,0,9,0],[...

# Variables
puzzle = intvar(1,9, shape=(9,9), name="puzzle")

# Contraintes
m = Model()
m += [puzzle[grid!=0] == grid[grid!=0]]
m += [AllDifferent(row) for row in puzzle]
m += [AllDifferent(col) for col in puzzle.T]
for i in range(0,9,3):
    for j in range(0,9,3):
        m += AllDifferent(puzzle[i:i+3, j:j+3])

# Solution
m.solve()
```

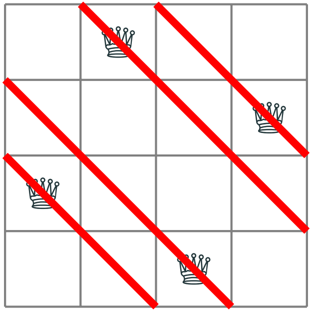


$$X_i \in \{1, \ldots, n\}$$



$$X_i \in \{1, ..., n\}$$

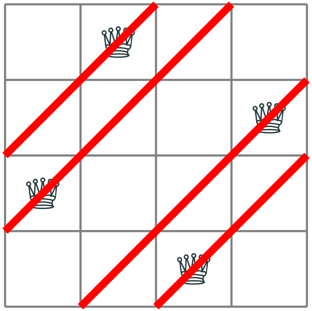
$$AllDifferent(X_1, ..., X_n)$$



$$X_i \in \{1, ..., n\}$$

$$AllDifferent(X_1, ..., X_n)$$

$$AllDifferent(X_1 + 0, ..., X_n + (n-1))$$

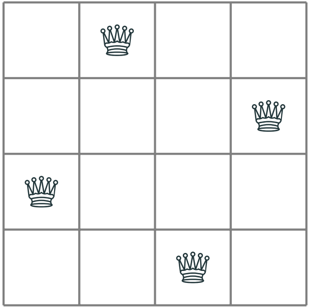


$$X_i \in \{1, ..., n\}$$

$$AllDifferent(X_1, ..., X_n)$$

$$AllDifferent(X_1 + 0, ..., X_n + (n-1))$$

$$AllDifferent(X_1 - 0, ..., X_n - (n-1))$$



```
# Variables
queens = intvar(1, N, shape=N, name="queens")

# Contraintes
m = Model()
m += [AllDifferent(queens)]
m += [AllDifferent([queens[i] + i for i in range(N)])]
m += [AllDifferent([queens[i] - i for i in range(N)])]

# Solution
m.solve()
```

$$succ_i \in \{1, ..., n\}$$

KU LEUVEN

$$succ_i \in \{1, ..., n\}$$

$$Circuit(succ)$$

VOYAGEUR DE COMMERCE

$$succ_i \in \{1, ..., n\}$$

$$Circuit(succ)$$

$$\min \sum distance[i][succ_i]$$

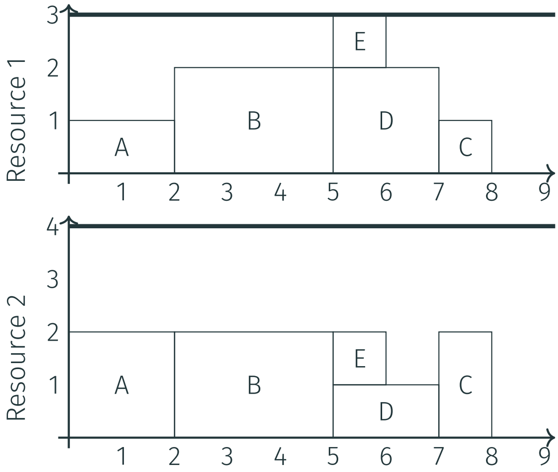
KU LEUVEN


```
distances = np.array([[1,3,5,6],[...

# Variables
succ = intvar(1,n, shape=N, name=”successeurs”)

# Contraintes
m = Model()
m += [Circuit(succ)]

# Objectif
m.minimize(sum(distances[i, x[i]] for i in range(n)))

# Solution
m.solve()
```

PROBLÈME DE GESTION DE PROJET À CONTRAINTES DE RESSOURCES (RCPSP)

Task	d_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 $C_{r_2} = 4$



KU LEUVEN

Task	d_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 $C_{r_2} = 4$

$$s_i \in \{0, ..., h\} \quad \forall i \in T$$

Task	d_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 $C_{r_2} = 4$

$$s_i \in \{0, ..., h\} \quad \forall i \in T$$

$$s_i + d_i \leq s_j \quad \forall i \prec j$$

Task	d_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 $C_{r_2} = 4$

$$s_i \in \{0, ..., h\} \quad \forall i \in T$$

$$s_i + d_i \leq s_j \quad \forall i \prec j$$

$$Cumulative(\{s_i | i \in T\}, \{d_i | i \in T\}, \{s_i + d_i | i \in T\}, \{c_{ir} | i \in T\}, C_r)$$

Task	d_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 $C_{r_2} = 4$

$$s_i \in \{0, ..., h\} \quad \forall i \in T$$

$$s_i + d_i \leq s_j \quad \forall i \prec j$$

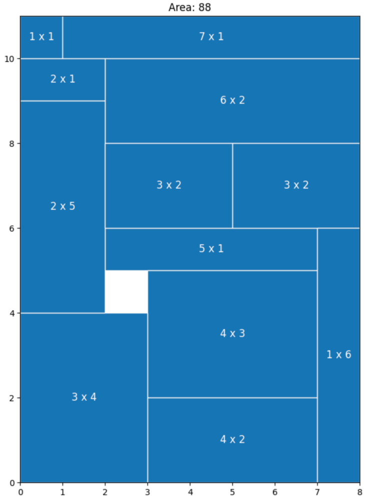
$$Cumulative(\{s_i | i \in T\}, \{d_i | i \in T\}, \{s_i + d_i | i \in T\}, \{c_{ir} | i \in T\}, C_r)$$

$$\min\{s_i + d_i | i \in T\}$$

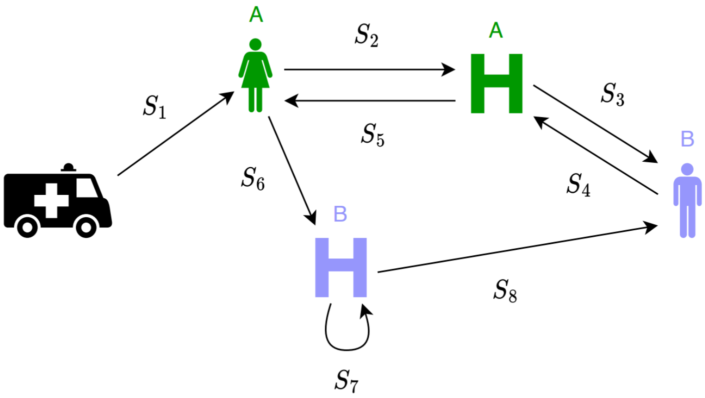
Task	d_i	c_{ir_1}	c_{ir_2}	succ
A	2	1	2	B C D
B	3	2	2	E
C	1	1	2	
D	2	2	1	C
E	1	1	1	C

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

```
d = np.array([2,3,1,2,1])
c = np.array([[1,2,1,2,1],[...
C = np.array([3,4])
h = 42 # horizon
prec= np.array([(0,1),(0,2),...

# Variables
s = intvar(0,h, shape=(9,9), name="puzzle")

# Contraintes
m = Model()
m += [s_i + d_i \leq s_j for (i,j) in prec]
m += [Cumulative(s,d,s+d,c[r],C[r]) for r in range(R)]

# Objectif
m.minimize(s+d)

# Solution
m.solve()
```




```
area_min_x, area_max_x = max(widths), sum(widths)
area_min_y, area_max_y = max(heights), sum(heights)

# Variables
pos_x = intvar(0, area_max_x, shape=n)
pos_y = intvar(0, area_max_y, shape=n)
total_x = intvar(area_min_x, area_max_x)
total_y = intvar(area_min_y, area_max_y)

# Contraintes
m = Model()
for i,j in all_pairs(range(n)):
    m += ((pos_x[i] + widths[i] <= pos_x[j]) |
          (pos_x[j] + widths[j] <= pos_x[i]) |
          (pos_y[i] + heights[i] <= pos_y[j]) |
          (pos_y[j] + heights[j] <= pos_y[i]))

m.minimize(total_x*total_y)

# Solution
m.solve()
```



Transport de patients vers les hôpitaux

"Insertion sequence variables for hybrid routing and scheduling problems", C. Thomas, R. Kameunier, P. Schaus, CPAIOR2020

Planification des tâches du robot Philae explorateur de comète, équipe du LAAS-CNRS (Toulouse)

KU LEUVEN

- Plusieurs modèles possibles, lequel choisir?
- Contraintes redondantes : aident-elles?
- Comment casser les symétries et réduire l’espace de recherche?
- Quelle technologie d’optimisation utiliser?
- ...

APERÇU DU SOLVEUR

KU LEUVEN

Le problème

Variables et domaines

$$X \rightarrow \left\{ \begin{matrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \end{matrix} \right\}$$

Contraintes

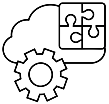
$$X + Y \geq 3$$

AllDifferent(X,Y,Z)

AtMost(2, [W,X,Y,Z], 4)

...

→ Solveur de Programmation par Contraintes →



Une solution

W=4

X=4

Y=2

Z=1

...

KU LEUVEN

Le problème

Variables et domaines

$$X \rightarrow \left\{ \begin{matrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \end{matrix} \right\}$$

Contraintes

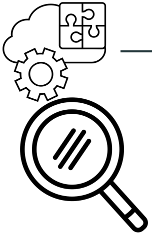
$$X + Y \geq 3$$

AllDifferent(X,Y,Z)

AtMost(2, [W,X,Y,Z], 4)

...

→ Solveur de Programmation par Contraintes →



Une solution

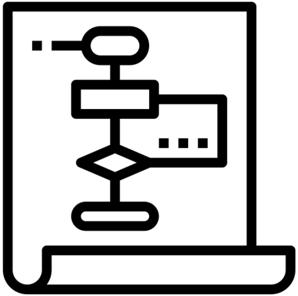
W=4

X=4

Y=2

Z=1

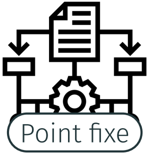
...



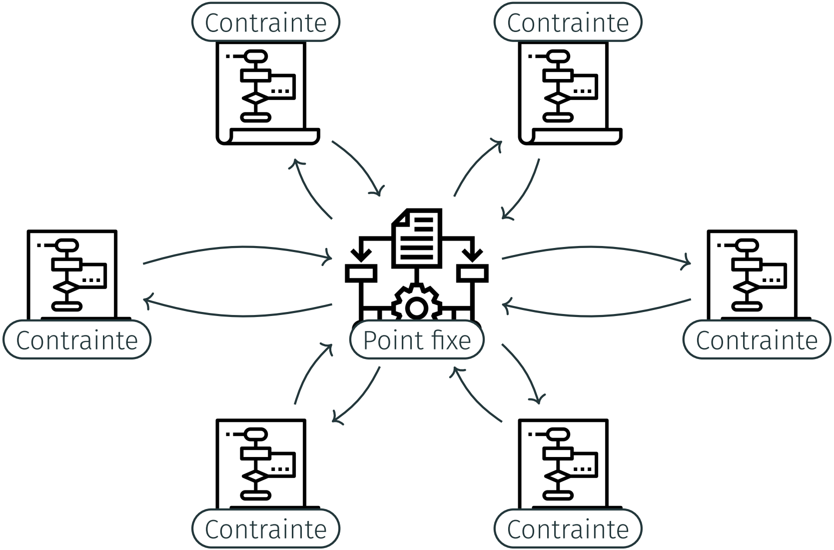
Entrée : état des domaines

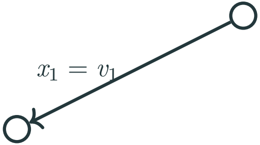
1. Mise à jour de la représentation interne
2. Filtrage de nouvelles valeurs

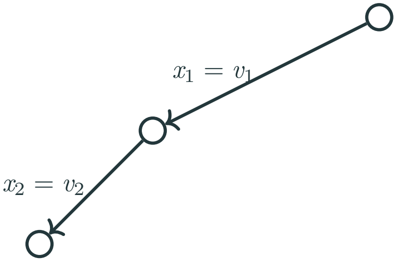
Sortie : état des domaines mis à jour


Point fixe

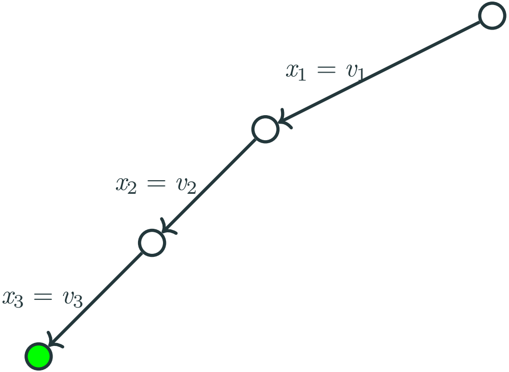
FONCTIONNEMENT DU SOLVER

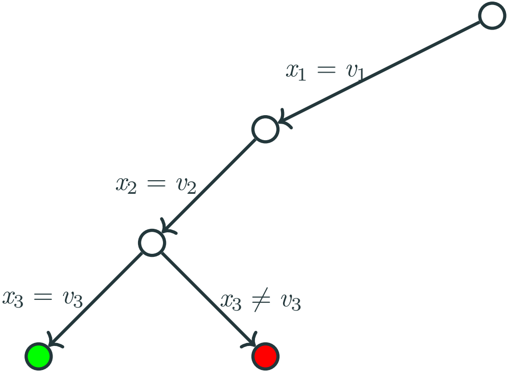



$x_1 = v_1$

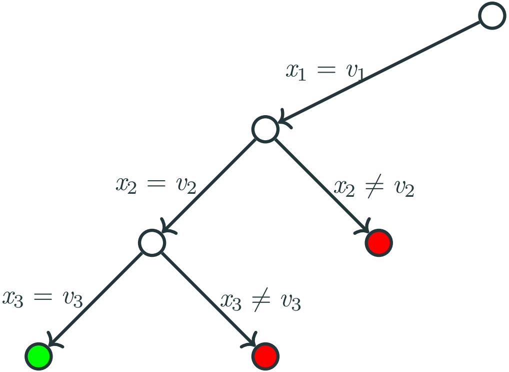


$x_1 = v_1$
$x_2 = v_2$

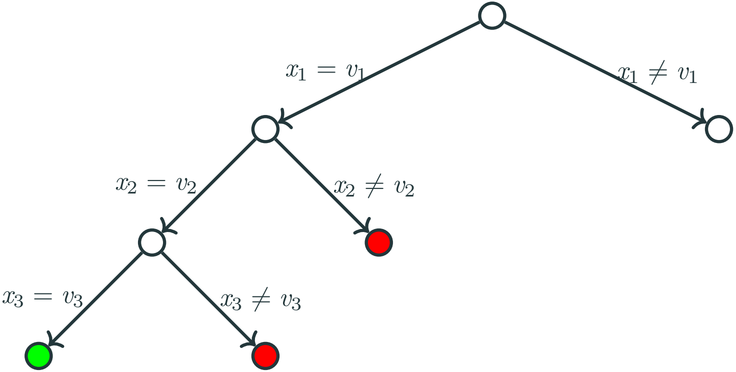

$x_1 = v_1$
$x_2 = v_2$
$x_3 = v_3$

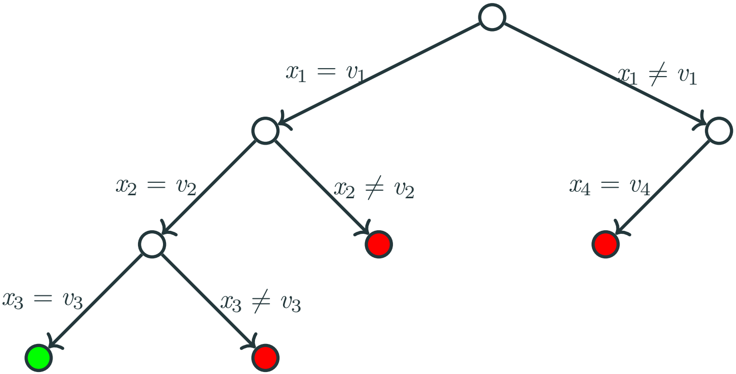


$x_1 = v_1$
$x_2 = v_2$
$x_3 = v_3$
$x_3 \neq v_3$

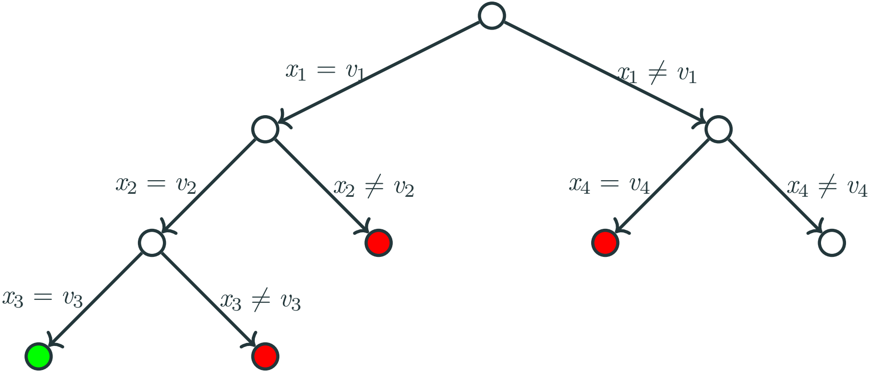

$x_1 = v_1$
$x_2 = v_2$
$x_2 \neq v_2$
$x_3 = v_3$
$x_3 \neq v_3$

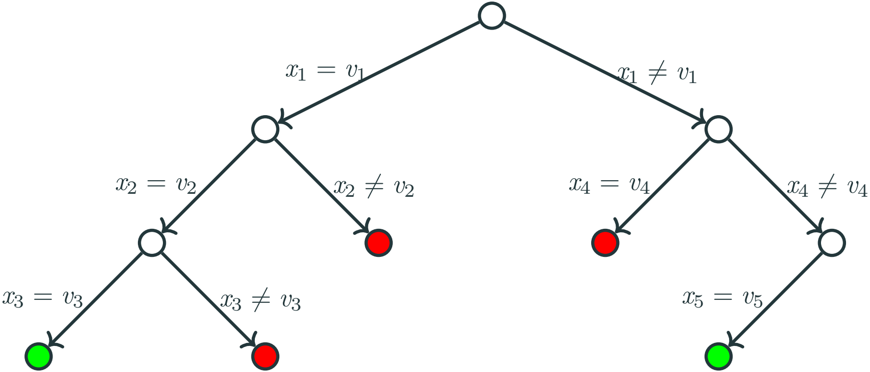
ARBRE DE RECHERCHE

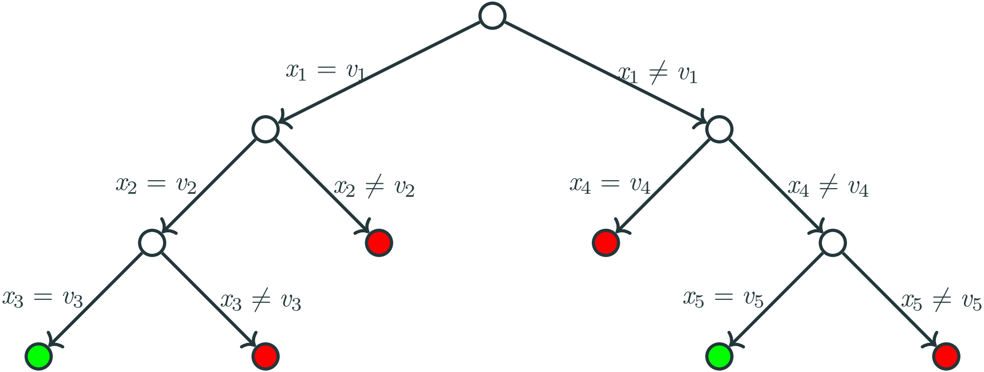



$x_1 = v_1$
$x_1 \neq v_1$
$x_2 = v_2$
$x_2 \neq v_2$
$x_4 = v_4$
$x_3 = v_3$
$x_3 \neq v_3$

KU LEUVEN




$x_1 = v_1$
$x_1 \neq v_1$
$x_2 = v_2$
$x_2 \neq v_2$
$x_4 = v_4$
$x_4 \neq v_4$
$x_3 = v_3$
$x_3 \neq v_3$
$x_5 = v_5$



$x_1 = v_1$
$x_1 \neq v_1$
$x_2 = v_2$
$x_2 \neq v_2$
$x_4 = v_4$
$x_4 \neq v_4$
$x_3 = v_3$
$x_3 \neq v_3$
$x_5 = v_5$
$x_5 \neq v_5$

Construit suivant une heuristique de sélection de variable/valeur.

But : Diminuer la taille de l'arbre le plus possible

Par exemple :

- sélectionner la variable avec le plus petit domaine
- sélectionner la variable qui est impliquée dans le plus de contraintes
- sélectionner le dernière variable ayant déclanché un retour en arrière
- ...

APPLICATION À L’INTELLIGENCE ARTIFICIELLE

				
	✔	✗	✔	✔
	✗	✗	✗	✔
	✔	✔	✔	✗
	✗	✔	✔	✗

KU LEUVEN

	✔	✗	✔	✔
	✗	✗	✗	✔
	✔	✔	✔	✗
	✗	✔	✔	✗

	✔	✗	✔	✔
	✗	✗	✗	✔
	✔	✔	✔	✗
	✗	✔	✔	✗

					
X	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	C= $\{0..4\}$
	✔	✗	✔	✔	$\{0,1\}$
	✗	✗	✗	✔	$\{0,1\}$
	✔	✔	✔	✗	$\{0,1\}$
	✗	✔	✔	✗	$\{0,1\}$

KU LEUVEN

					
X	$\{1\}$	$\{0, 1\}$	$\{0, 1\}$	$\{0, 1\}$	C= $\{0..2\}$
	✔	✗	✔	✔	$\{0, 1\}$
	✗	✗	✗	✔	$\{0\}$
	✔	✔	✔	✗	$\{0, 1\}$
	✗	✔	✔	✗	$\{0\}$

KU LEUVEN

					
X	$\{1\}$	$\{0\}$	$\{1\}$	$\{0,1\}$	C= $\{1..2\}$
	✔	✗	✔	✔	$\{1\}$
	✗	✗	✗	✔	$\{0\}$
	✔	✔	✔	✗	$\{0,1\}$
	✗	✔	✔	✗	$\{0\}$

					
X	{1}	{0}	{1}	{1}	C= {1}
	✔	✗	✔	✔	{1}
	✗	✗	✗	✔	{0}
	✔	✔	✔	✗	{0}
	✗	✔	✔	✗	{0}

Solution : Contrainte `CoverSize`

`CoverSize`(X, C, données)

- X: variables Booléennes par éléments
- C: compteur

$$\max C$$

$$CoverSize(\{X_i | i \in I\}, C, donnes)$$

$$X_i = \{0, 1\} \quad \forall i \in I$$

$$C = \{0, |I|\}$$

PROBLÈME D'IA: LA RECHERCHE D'ENSEMBLES D'ÉLÉMENTS FRÉQUENTS

KU LEUVEN

CoverSize: A Global Constraint for Frequency-based Itemset Mining

Pierre Schaus[1] and John O.R. Aoga[1,2](0000-0002-7213-146X) and Tias Guns[3]

[1]UCLouvain, ICTEAM (Belgium); [2]UAC, ED-SDI (Benin)
[3]VUB Brussels (Belgium) and KU Leuven (Belgium)
{john.aoga,pierre.schaus}@uclouvain.be; tias.guns@{vub.be,cs.kuleuven.be}

Abstract. Constraint Programming is becoming competitive for solving certain data-mining problems largely due to the development of global constraints. We introduce the CoverSize constraint for itemset mining problems, a global constraint for counting and constraining the number of transactions covered by the itemset decision variables. We show the relation of this constraint to the well-known table constraint, and our filtering algorithm internally uses the reversible sparse bitset data structure recently proposed for filtering table. Furthermore, we expose the size

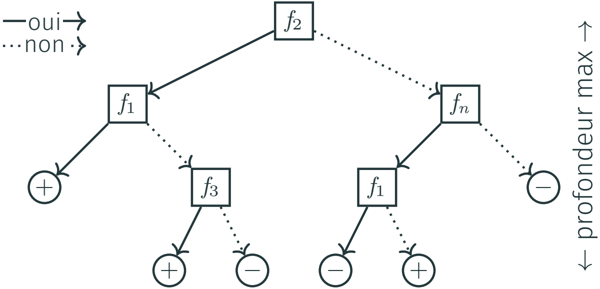
Base de données

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	$+$

KU LEUVEN

Base de données

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	$+$



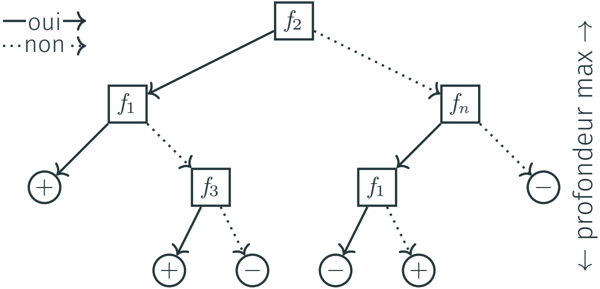
$$\min \sum (pred(i) - c(i))$$

Base de données

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	+

Nouvelle entrée

0	0	1	$\dots$	0	?



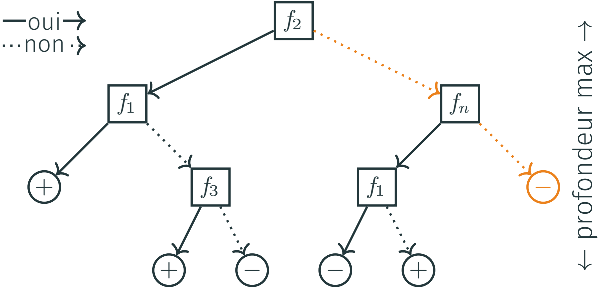
$$\min \sum (pred(i) - c(i))$$

Base de données

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	+

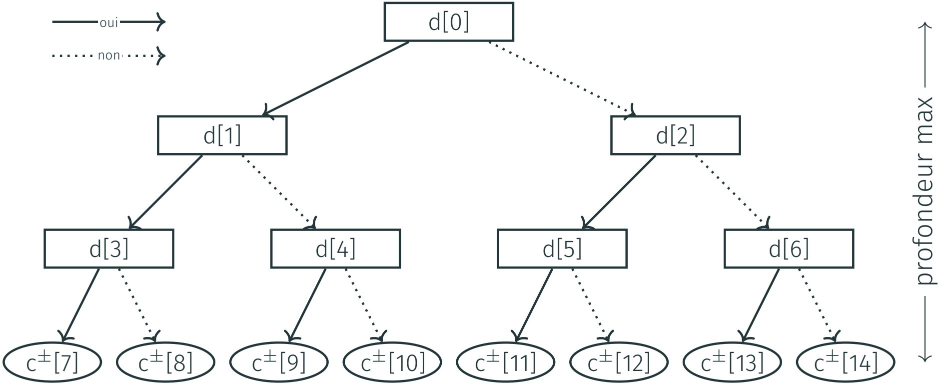
Nouvelle entrée

0	0	1	$\dots$	0	−



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

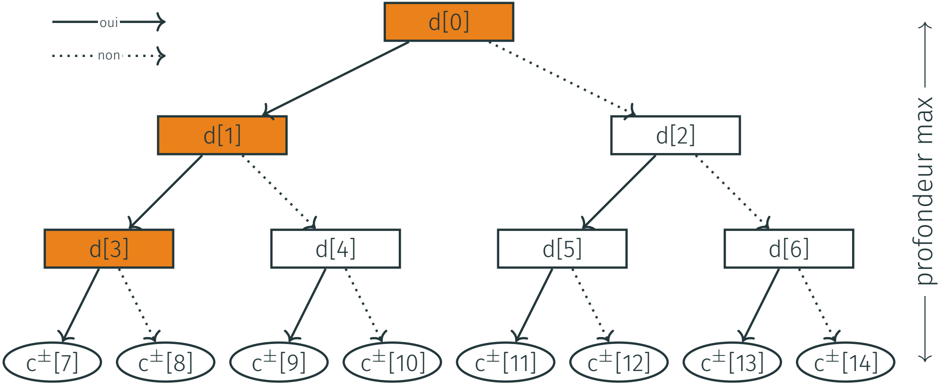
KU LEUVEN



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

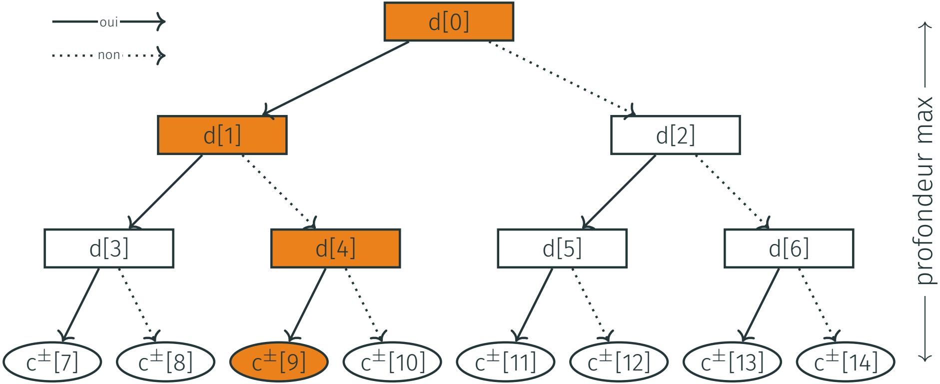
$dom(c[i]) = \{0, ..., N\}$

PROBLÈME D'IA: ARBRES DE DÉCISIONS OPTIMAUX - ALLDIFFERENTEXCEPT0



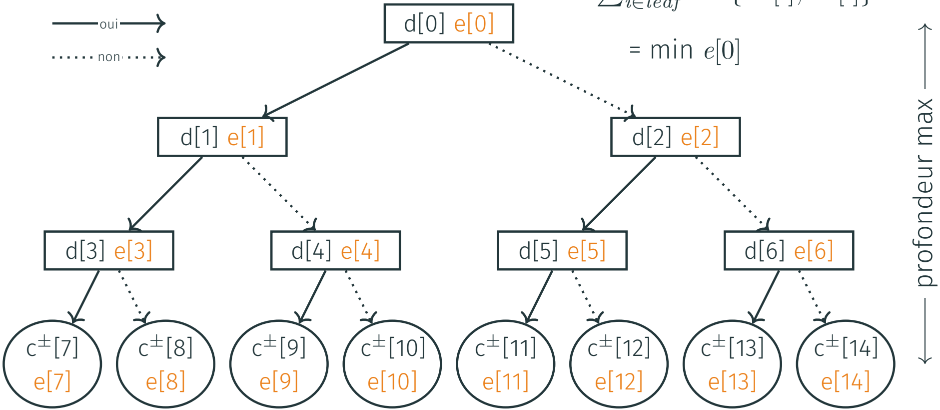
$dom(d[i]) = \{0, 1, ..., n\}$

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



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

PROBLÈME D'IA: ARBRES DE DÉCISIONS OPTIMAUX - OBJECTIF



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

$$= \min e[0]$$

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

KU LEUVEN

Constraints (2020) 25:226–250
https://doi.org/10.1007/s10601-020-09312-3

ORIGINAL RESEARCH

Check for updates

Learning optimal decision trees using constraint programming

Hélène Verhaeghe[1] · Siegfried Nijssen[1] · Gilles Pesant[2] · Claude-Guy Quimper[3] · Pierre Schaus[1]

Accepted: 29 September 2020 / Published online: 29 October 2020

Abstract

Decision trees are among the most popular classification models in machine learning. Tra-

CONCLUSION

Programmation par Contraintes = Modèle + Recherche

- Résolution de problème combinatoires
- Modélisation déclarative du problème
- Méthode exacte
- Modularité

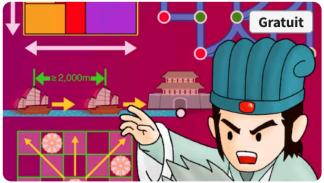
- Librairie CPMpy opensource sur Github :
  `https://github.com/CPMpy/cpmpy`

- MOOC des Prof. Jimmy Lee (The Chinese University of Hong Kong, Hong Kong) et Peter Stuckey (Monash University, Australie) sur la modélisation de problèmes combinatoires : `https://www.coursera.org/learn/basic-modeling?`

The University of Melbourne

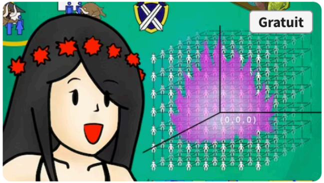
Basic Modeling for Discrete Optimization

Compétences que vous acquerrez:
Programmation informatique



The University of Melbourne

Advanced Modeling for Discrete Optimization

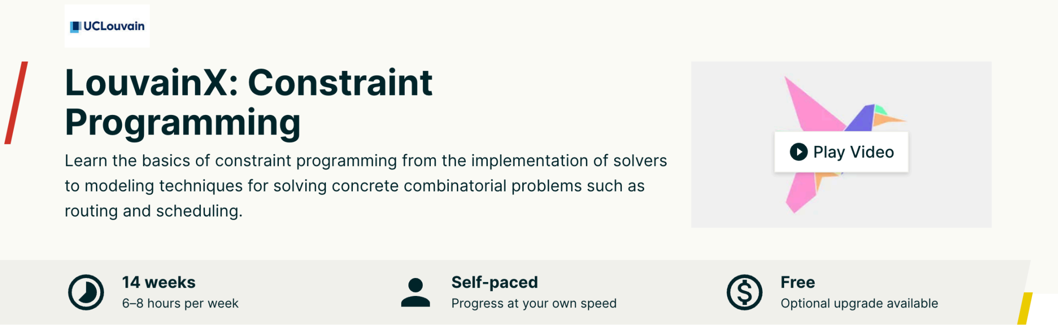


The University of Melbourne

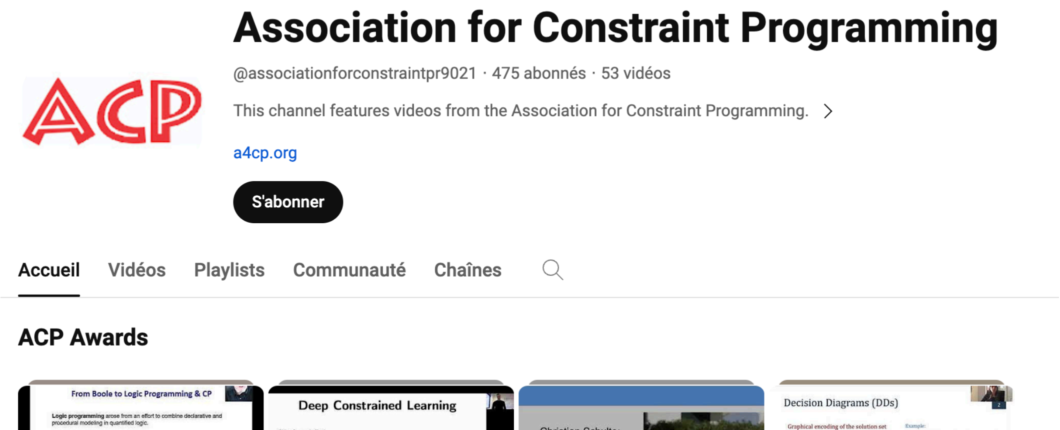
Solving Algorithms for Discrete Optimization

KU LEUVEN

- MOOC du Prof. Pierre Schaus (UCLouvain, Belgique) sur le fonctionnement des solveurs : `https://www.edx.org/learn/computer-programming/universite-catholique-de-louvain-constraint-programming`



- Chaine Youtube de l'association de programmation par contraintes : `https://www.youtube.com/@associationforconstraintpr9021`



Merci de votre attention!

Des questions?

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

Les icones viennent du Noun Project (`thenounproject.com`), graphistes : Alzam, Becris, Eucalyp, Handicon, HCP18, lastspark, Megan Day, Vectors Point