

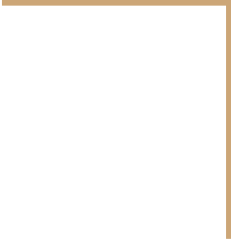


Présentation 3

PSAR


David TOTY
Maxime TRAN





Objectif:

Modéliser et résoudre le problème du Sudoku



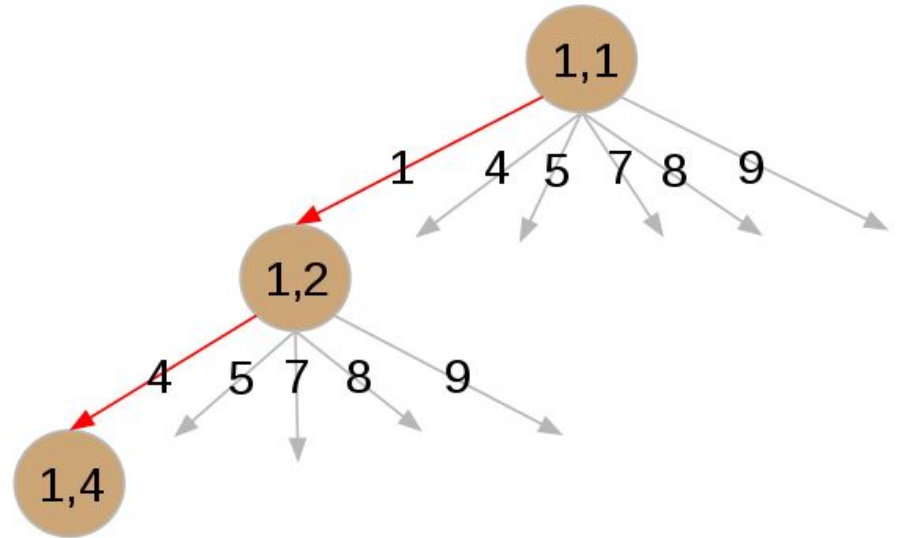
Rappel: Programmation par contraintes

En programmation par contraintes, un problème doit être formulé à l'aide des notions suivantes :

- Des **variables** prenant leurs valeurs dans des **domaines**.
- Des **contraintes** restreignant les valeurs possibles des variables dans leurs domaines en fonction des valeurs des autres variables.

Stratégie de recherche

- Parcours en profondeur
- Recherche adaptative



Modèle du problème de Sudoku sous MiniZinc

```
include "alldifferent.mzn";
```

```
% domaine: ensemble d'entier
```

```
set of int: domaine = 1..9;
```

```
% grille est un tableau à 2 dimensions de cases, indexé par le  
numéro de ligne et colonne, avec des variables de type integer
```

```
array [domaine, domaine] of int: grille;
```

```
% grille du sudoku, les 0 correspondant aux cases vides à remplir
```

```
grille =
```

```
[ | 0, 0, 0, 0, 0, 0, 0, 0, 0  
  | 0, 6, 8, 4, 0, 1, 0, 7, 0  
  | 0, 0, 0, 0, 8, 5, 0, 3, 0  
  | 0, 2, 6, 8, 0, 9, 0, 4, 0  
  | 0, 0, 7, 0, 0, 0, 9, 0, 0  
  | 0, 5, 0, 1, 0, 6, 3, 2, 0  
  | 0, 4, 0, 6, 1, 0, 0, 0, 0  
  | 0, 3, 0, 2, 0, 7, 6, 9, 0  
  | 0, 0, 0, 0, 0, 0, 0, 0, 0  
];
```

```
% puzzle: variable de décision
```

```
% domaine d'association : 1..9
```

```
array [domaine, domaine] of int: puzzle;
```

```
% Copie la grille dans un tableau de variables
```

```
constraint forall (i, j in domaine) (
```

```
    If grille [i, j] > 0 then puzzle [i, j] = grille [i, j] else true endif );
```

```
% ligne_diff (ligne) contraint les cases de la ligne ligne à être distinctes
```

```
predicate ligne_diff (int: ligne) =
```

```
    alldifferent (colonne in domaine) (puzzle[ligne, colonne]);
```

```
% colonne_diff(colonne) contraint les cases de la colonne colonne à être  
distinctes.
```

```
predicate colonne_diff (int: colonne) =
```

```
    alldifferent (ligne in domaine) (puzzle[ligne, colonne]);
```

Modèle du problème de Sudoku sous MiniZinc

% region_diff(ligne) de même pour les régions à être différentes

```
predicate region_diff(int: ligne, int: colonne) =  
    alldifferent (i, j in 0..2) (puzzle[ligne + i, colonne + j]);
```

% Utilisation des prédicats pour contraindre les valeurs de la grille

```
constraint forall (ligne in domaine)      (ligne_diff (ligne));  
constraint forall (colonne in domaine)    (colonne_diff (colonne));  
constraint forall (ligne, colonne in {1, 4, 7}) (region_diff (ligne, colonne));
```

% Problème de satisfaction : trouver les valeurs pour les variables de décisions pour satisfaire les contraintes

```
solve satisfy;
```

% Affichage

```
output [ if j = 1 then "\n" else " " endif ++  
        Show (puzzle[i,j]) | i,j in domaine] ++ ["\n"];
```

Résultat du problème

```
3000755@ppti-14-509-03:/users/nfs/Etu5/3000755/PSAR$ mzn-g12fd  sudoku.mzn
```

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

```
-----
```

Explication du code

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

```
predicate region_diff (int: ligne, int: colonne) =  
    alldifferent (i, j in 0..2) (grille[ligne + i, colonne + j]);  
  
constraint forall (ligne, colonne in {1, 4, 7}) (region_diff (ligne, colonne));
```


Explication du code

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

```
predicate region_diff (int: ligne, int: colonne) =  
    alldifferent (i, j in 0..2) (grille[ligne + i, colonne + j]);
```

% Les cases grisées

```
constraint forall (ligne, colonne in {1, 4, 7}) (region_diff (ligne, colonne));
```

Exemple :

region_diff (1, 4) -> région de la case rouge (1, 4)

Explication du code

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

% Les cases oranges par rapport à la case rouge (slide précédente)

```
predicate region_diff (int: ligne, int: colonne) =  
    alldifferent (i, j in 0..2) (grille[ligne + i, colonne + j]);
```

% Les cases grisées

```
constraint forall (ligne, colonne in {1, 4, 7}) (region_diff (ligne, colonne));
```

Exemple :

region_diff (1, 4) -> région de la case rouge (1, 4)
-> appel de la contrainte alldifferent -> région orange

Questions ?

Merci !
Pour nous contacter:

david.toty@etu.upmc.fr
maxime.tran@etu.upmc.fr

Encadrant du projet:

Responsable: Fabrice KORDON

Client: Tarek Menouer
