Présentation 8
PSAR

David TOTY

Maxime TRAN

Objectif:
Modéliser et résoudre le problème du Sudoku

Qu’est ce qu’un Sudoku ?

Un sudoku est un jeu de puzzle sous la forme d’une grille de taille N^2 x N^2 composé de N^2 de lignes, N^2 de colonnes et N^2 de régions de taille N x N.

N représente le nombre de chiffre dans chaque région, chaque colonne, chaque ligne. Les valeurs des cases vont de 1 à N^2;

<u>Exemple:</u>

- Sudoku classique:
  - N=3 correspond à la taille des régions
  - 9 lignes, 9 colonnes et 9 régions
  - 81 cases

Différents types de Sudoku

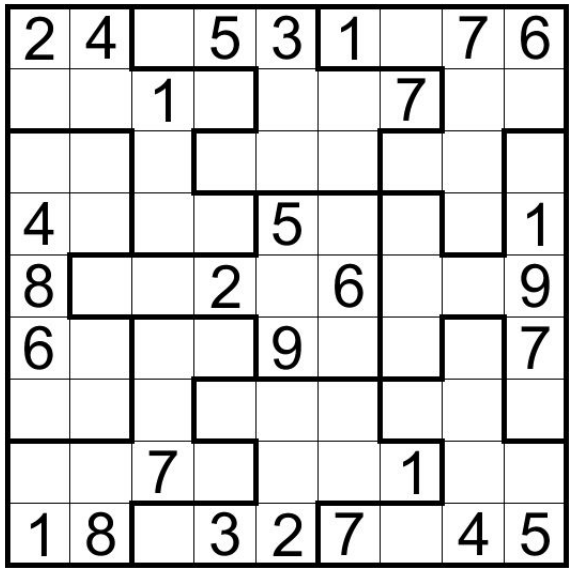
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Sudoku classique (N=3)

2			4
	4		3
	1	4	2
4			

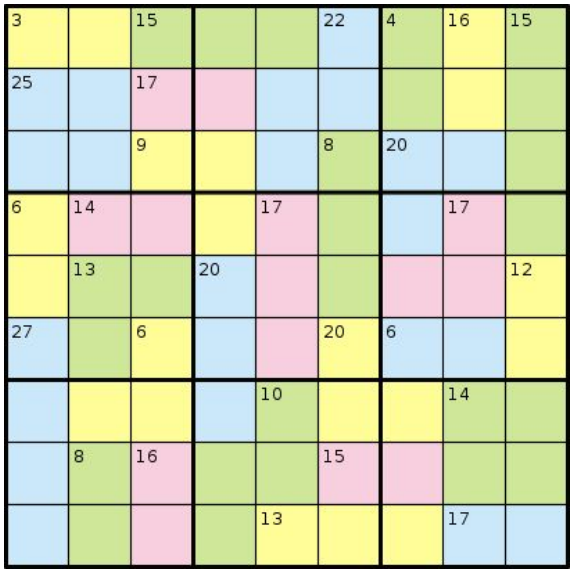
N=2

Différents types de Sudoku

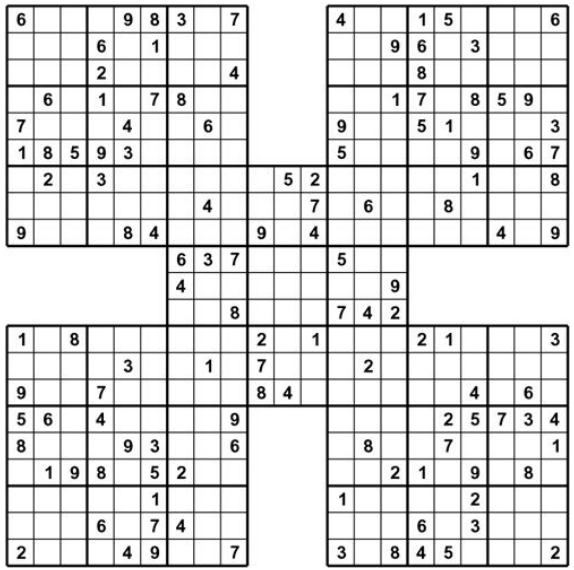


Chaos Sudoku

Différents types de Sudoku



Killer Sudoku



Samurai Sudoku

Comment résoudre un sudoku ?

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

(4,5) = ?

Valeur possible : Chiffre de 1 à 9.

Contrainte : 1 occurrence par ligne, colonne, région.

(4,5) = { ~~1~~, ~~2~~, 3, 4, 5, 6, ~~7~~, ~~8~~, ~~9~~ }

Rappel: Programmation par contraintes

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

- Des ***variables***, des ***domaines*** et des ***contraintes***.

Exemple:

Variables: Les 81 cases du Sudoku
Domaines: L'ensemble {1, 2, 3, 4, 5, 6, 7, 8, 9}
Contraintes: 1 seule occurrence de valeur sur chaque ligne, colonne et région

La résolution : Différents algorithmes.

La seule différence sera la manière de parcourir les solutions.

- Naif : Parcours selon l'ordre des colonnes puis ligne, selon la première case
- Adaptatif : Parcours selon un ordre croissant défini par le nombre de Solution. On appelle ces nombres le degré de liberté.
- Dynamique : idem que adaptatif mais les degré de liberté sont recalculés au fur et à mesure.

  Pour cela on utilisera une methode permettant de définir notre parcours selon les différents algorithmes.

Algorithme naïf

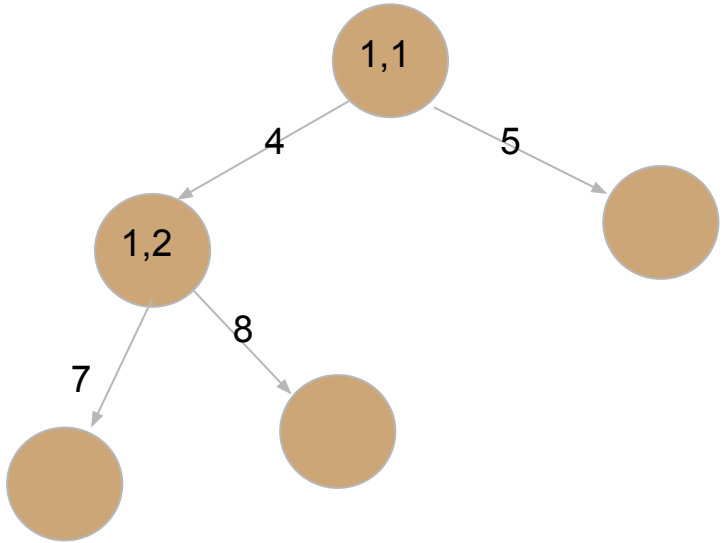
		3		2		6		
9			3		5			1
		1	8		6	4		
		8	1		2	9		
7								8
		6	7		8	2		
		2	6		9	5		
8			2		3			9
		5		1		3		

Parcours :

(1,1)

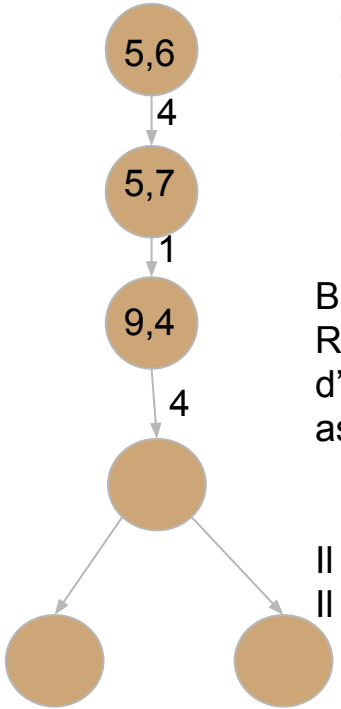
(1,2)

...



Algorithme adaptatif

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



Parcours :

(5,6)
(5,7)
(9,4)
...

Backtrack = Le fait de Revenir en arrière en cas d'incohérence : on restaure les assignations faites.

Il est moins couteux.
Il demande moins d'assignation.

Les différents algorithmes de recherche

Codé sous Java:

- SolveNaif ();
  Inspiré d'un model récursif trouvé sur internet.
  Adapter ici pour une utilisation récursive.
- SolveAdaptatif ();
- SolveDynamique ();

Algorithme adaptatif

Tableau du Sudoku:

0 6 0 0 5 0 0 2 0
0 0 0 3 0 0 0 9 0
7 0 0 6 0 0 0 1 0
0 0 6 0 3 0 4 0 0
0 0 4 0 7 0 1 0 0
0 0 5 0 9 0 8 0 0
0 4 0 0 0 1 0 0 6
0 3 0 0 0 8 0 0 0
0 2 0 0 4 0 0 5 0

Tableau de parcours:

(nombre de degré de liberté, ligne, colonne)

1 : 3 : 7
1 : 6 : 4
1 : 8 : 4
2 : 0 : 4
2 : 0 : 6
2 : 2 : 4
2 : 2 : 6
2 : 3 : 5
2 : 4 : 1
2 : 4 : 7
...

Algorithme adaptatif

public void initParcours();

Mise en place du système de backtrack.

Tableaux:

- PossibilitéSuivante
- Hauteur

Algorithme adaptatif

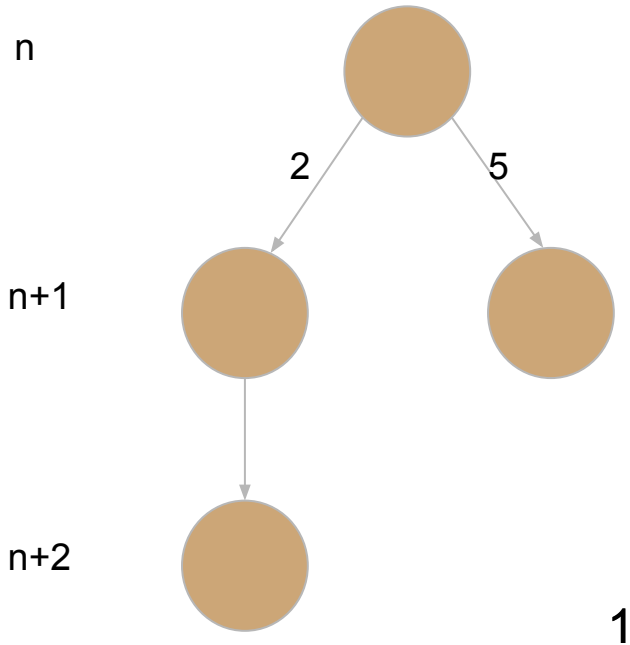
public void initParcours();

Mise en place du système de backtrack.

Tableaux:
- PossibilitéSuivante
- Hauteur

Tableau de hauteur stocke n.

Tableau de possibilitéSuivante stocke la valeur 5.



Références - Points de mesure

Génération de sudokus : Générateur paramétrable trouvé sur internet permettant de définir la taille et le taux de remplissage du sudoku.

Il sera donc intéressant de comparer les temps selon ces 2 paramètres.

Pour chaques test, nous allons générer 1000 sudokus aléatoires, les solvers travailleront sur les mêmes sudokus.

Références - Points de mesure

Nous comparons les temps de résolution par rapport *aux différents solvers utilisés*, a*ux tailles des Sudokus* et *aux taux de remplissage*.

On s’intérésse surtout aux ***temps moyens***, ***médians***. Ces temps sont mesurés en nanoseconde mais seront convertis si nécéssaire.

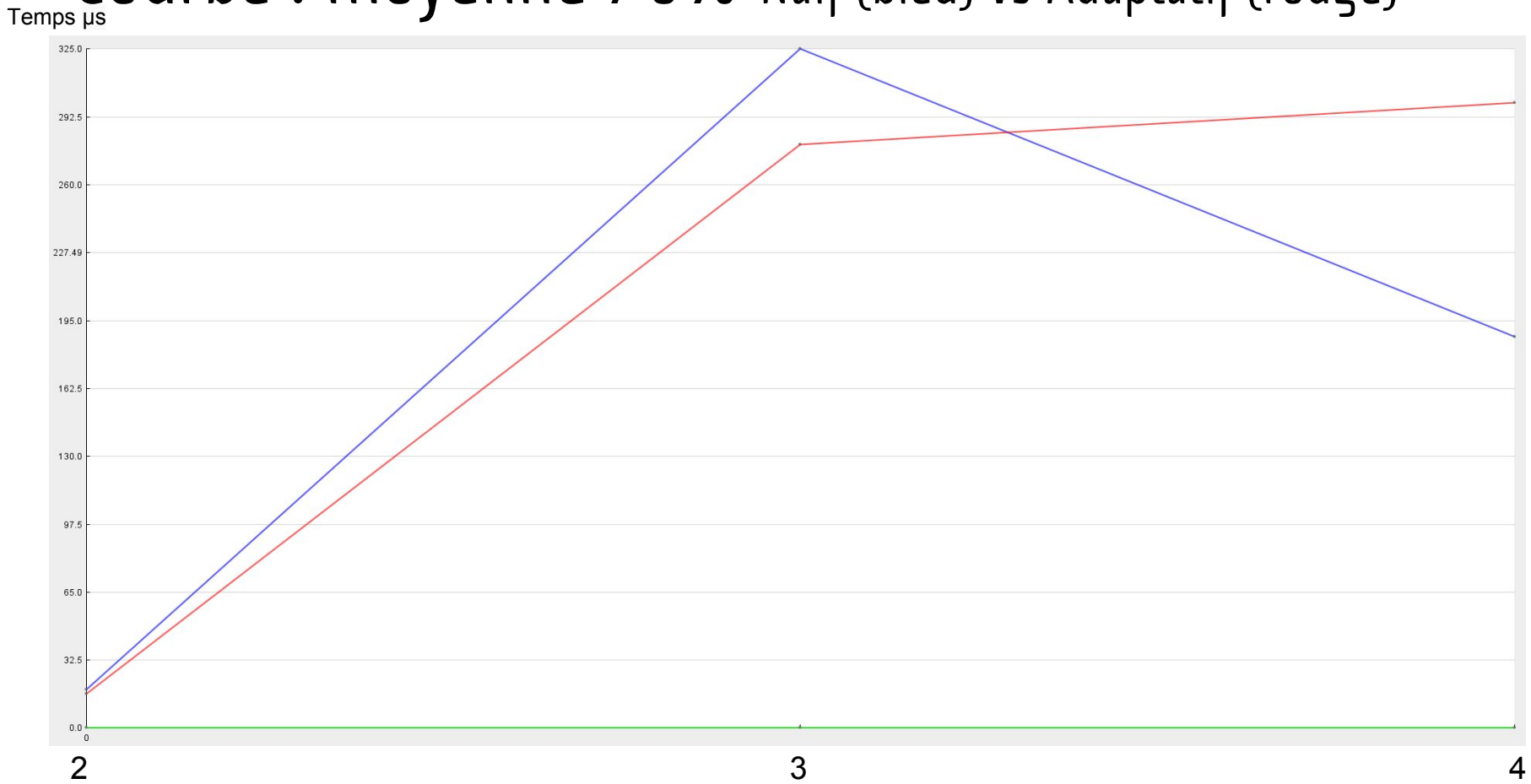
Résultats & tests.

	SolverNaïf			
	Temps moyen en µs		Temps médian en µs	
	Taux 40%	Taux 70%	Taux 40%	Taux 70%
Sudoku N=2	24	18	21	15
Sudoku N=3	369	325	119	89
Sudoku N=4	1574520	187	121488	67

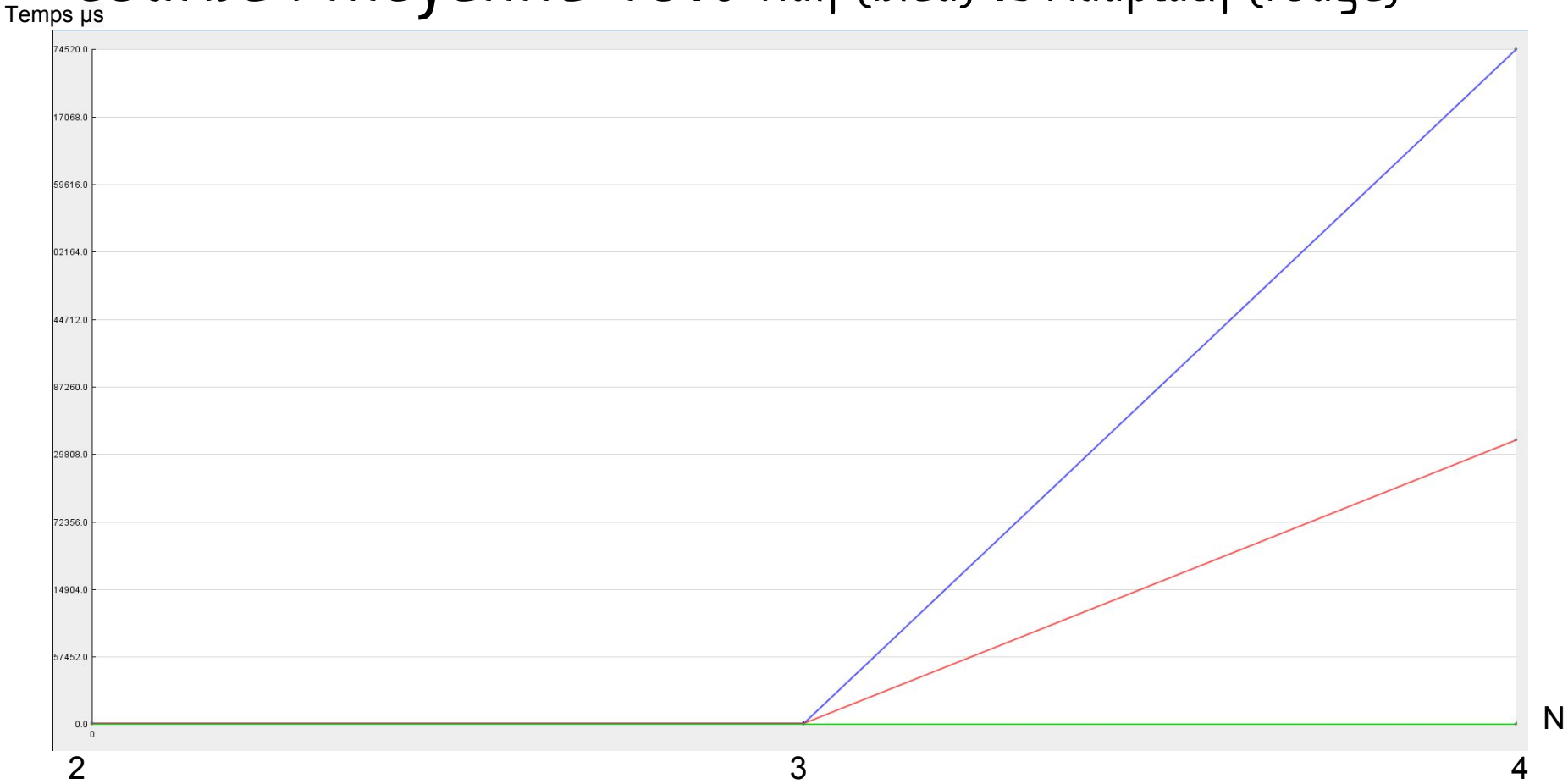
Résultats & tests.

	SolverAdaptatif			
	Temps moyen en µs		Temps médian en µs	
	Taux 40%	Taux 70%	Taux 40%	Taux 70%
Sudoku N=2	19	16	18	11
Sudoku N=3	293	279	90	62
Sudoku N=4	662634	299	110259	138

Courbe : moyenne 70% Naif (bleu) vs Adaptatif (rouge)



Courbe : moyenne 40% Naif (bleu) vs Adaptatif (rouge)



Autres tests : Naif vs Adaptatif.

1000 Sudokus aléatoire.
Taux de remplissage de 50%.
Taille N=3.
Temps en µs.

	Temps total	Temps moyen	Temps max	Temps min	Variance	ecart type	Mediane
Naïf	52172	52	1736	6	1229273	35	21
Adaptatif	40340	40	555	6	139759	11	14

Algorithme adaptatif appliqué au Chaos Sudoku

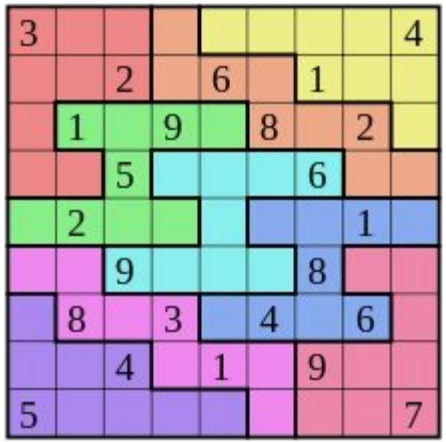


Tableau du Sudoku:

```
{{3, 0, 0, 0, 0, 0, 0, 0, 4},
 {0, 0, 2, 0, 6, 0, 1, 0, 0},
 {0, 1, 0, 9, 0, 8, 0, 2, 0},
 {0, 0, 5, 0, 0, 0, 6, 0, 0},
 {0, 2, 0, 0, 0, 0, 0, 1, 0},
 {0, 0, 9, 0, 0, 0, 8, 0, 0},
 {0, 8, 0, 3, 0, 4, 0, 6, 0},
 {0, 0, 4, 0, 1, 0, 9, 0, 0},
 {5, 0, 0, 0, 0, 0, 0, 0, 7}};
```

Tableau des régions:

```
{{1, 1, 1, 2, 3, 3, 3, 3, 3},
 {1, 1, 1, 2, 2, 2, 3, 3, 3},
 {1, 4, 4, 4, 4, 2, 2, 2, 3},
 {1, 1, 4, 5, 5, 5, 5, 2, 2},
 {4, 4, 4, 4, 5, 6, 6, 6, 6},
 {7, 7, 5, 5, 5, 5, 6, 9, 9},
 {8, 7, 7, 7, 6, 6, 6, 6, 9},
 {8, 8, 8, 7, 7, 7, 9, 9, 9},
 {8, 8, 8, 8, 8, 7, 9, 9, 9}};
```

Algorithme adaptatif appliqué au Chaos Sudoku

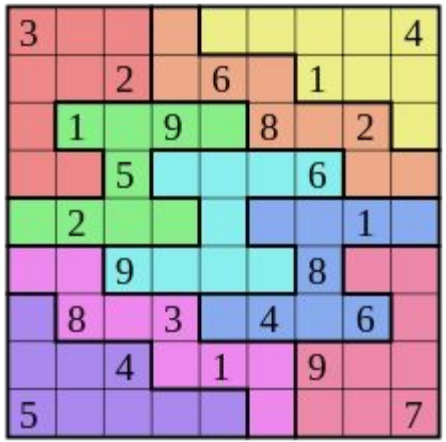


Tableau du Sudoku:

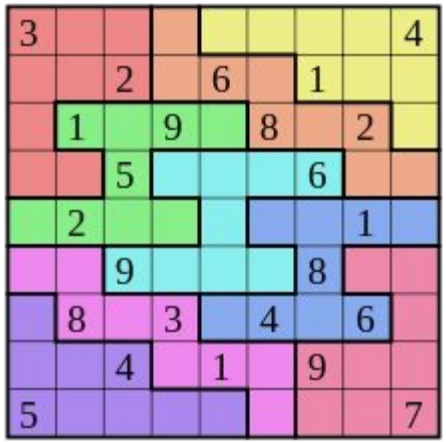
```
{{3, 0, 0, 0, 0, 0, 0, 0, 4},
 {0, 0, 2, 0, 6, 0, 1, 0, 0},
 {0, 1, 0, 9, 0, 8, 0, 2, 0},
 {0, 0, 5, 0, 0, 0, 6, 0, 0},
 {0, 2, 0, 0, 0, 0, 0, 1, 0},
 {0, 0, 9, 0, 0, 0, 8, 0, 0},
 {0, 8, 0, 3, 0, 4, 0, 6, 0},
 {0, 0, 4, 0, 1, 0, 9, 0, 0},
 {5, 0, 0, 0, 0, 0, 0, 0, 7}};
```

Tableau des régions:

```
{{1, 1, 1, 2, 3, 3, 3, 3, 3},
 {1, 1, 1, 2, 2, 2, 3, 3, 3},
 {1, 4, 4, 4, 4, 2, 2, 2, 3},
 {1, 1, 4, 5, 5, 5, 5, 2, 2},
 {4, 4, 4, 4, 5, 6, 6, 6, 6},
 {7, 7, 5, 5, 5, 5, 6, 9, 9},
 {8, 7, 7, 7, 6, 6, 6, 6, 9},
 {8, 8, 8, 7, 7, 7, 9, 9, 9},
 {8, 8, 8, 8, 8, 7, 9, 9, 9}};
```

```
protected boolean checkBox(int[ ][ ] region, int row, int col, int num);
```

Algorithme adaptatif appliqué au Chaos Sudoku



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

Résultat:

3 5 8 1 9 6 2 7 4
4 9 2 5 6 7 1 3 8
6 1 3 9 7 8 4 2 5
1 7 5 8 4 2 6 9 3
8 2 6 4 5 3 7 1 9
2 4 9 7 3 0 8 5 6
9 8 7 3 2 4 5 6 1
7 3 4 6 1 5 9 8 2
5 6 1 2 8 9 3 4 7

30509 µs

Conclusion

- Les différences entre solveurs se définissent par le parcours de solutions qu'ils empruntent.
- On utilise des tableaux pour sauvegarder la hauteur des différentes possibilités pour procéder au backtrack en cas d'incohérence.
- Les temps de résolution dépendent principalement du taux de remplissage du sudoku.
- Nous sommes dans des cas heuristiques : les temps des solveurs sont globalement proches sauf pour certains sudokus où le solveur adaptatif est bien meilleur.
- Le solveur naïf est moins stable en matière de dispersion du temps.
- Le chaosudoku est solvable facilement (comparaison de tableau) mais le temps est relativement important

Questions ?

Merci !
Pour nous contacter:

david.toty@etu.upmc.fr

maxime.tran@etu.upmc.fr

Encadrant du projet:

Responsable: Fabrice KORDON
Client: Tarek Menouer