Projet APO
Sudoku (et variantes) : résolution et génération

Mathieu Lefort

11 décembre 2024

1 Modalités pratiques

Ce projet est à faire par groupe de 3 et est à rendre pour le 2 février 2025.

2 Évaluation

Vous serez évalués sur les points suivants :
— Méthodologie (1 point)
— Conception (8 points)
— Code (8 points)
— Documentation (3 points)
— Extensions optionnelles (? points suivant vos choix)

2.1 Méthodologie

Vous expliquerez votre méthodologie de travail (articulation entre conception et codage, déroulé temporel du projet, ...) et la répartition des tâches entre les différents membres du groupe.

2.2 Conception

Vous utiliserez tous les différents types de diagrammes vus en TDs à bon escient (i.e. permettant à quelqu'un d'extérieur au projet de le comprendre et de pouvoir le coder). Pour rappel, pour le diagramme de cas d'utilisations, chaque cas d'utilisation doit être décrit par un texte.
Votre conception devra être modulaire et donc permettre une intégration sans trop d'efforts des différentes extensions possibles. Il ne vous est pas demandé de modéliser les extensions non faites, mais d'intégrer leur éventuel rajout dans votre réflexion lors de votre conception.
De plus, les différents choix (majeurs) de votre conception devront être justifiés.

2.3 Code

Le code devra être clair (indentation, noms de variables pertinentes, ...), utiliser la généricité et être (raisonnablement) efficace. Il devra également et surtout être en accord avec la conception. Pensez à bien vérifier et traiter (avec des exceptions) les cas d'erreurs dans vos méthodes.
Pensez à utiliser au maximum les packages pour structurer vos fichiers/classes.

2.4 Documentation

Vous générerez la javadoc de votre code (documentation des classes, méthodes, paramètres, ...). Vous penserez également à commenter votre code là où vous le jugez utile.

2.5 Extensions

Plusieurs extensions optionnelles vous sont proposées. À vous de voir celles qui peuvent vous intéresser. Il est tout à fait possible également de proposer (et d'implémenter) vos propres extensions. Si vous décidez de faire des extensions, votre note maximale potentielle peut dépasser 20. L'idée des extensions n'est pas vraiment de "rattraper" des points mais plutôt une opportunité de pousser l'exploration de Java plus loin si vous le souhaitez.

3 Sujet

L'objectif du projet est d'implémenter la résolution [1] et la génération de sudokus [2] (et leurs variantes [3]).

3.1 Sudoku

Un sudoku dispose d'une grille de taille $N \times N$, constituée de N lignes découpant la grille, N colonnes découpant (différemment) la grille et N blocs découpant (encore différemment) la grille. Il y a également N symboles différents qui doivent être disposés (chacun N fois) de façon à ce que chacun.e des lignes/colonnes/blocs contienne les N symboles. Il existe donc une grande diversité de grilles de sudoku (voir figures 1, 2 et 3).

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

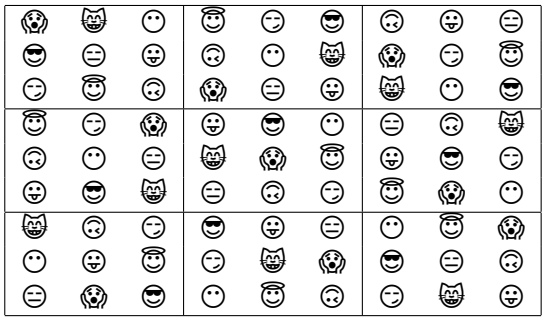

FIGURE 1 – Une grille "classique" 9×9 constituée de blocs 3×3 remplie avec des chiffres entre 0 et 9 (à gauche) ou n'importe quel autre symbole (à droite).

E	4	6	F	9	7	2	1	3	5	D	A	G	B	C	8
C	5	9	7	4	G	8	3	E	B	1	2	6	F	D	A
B	2	1	8	A	D	C	F	4	6	G	7	3	5	E	9
3	A	G	D	5	6	B	E	C	9	8	F	7	4	1	2
2	B	7	E	C	3	1	4	9	G	5	6	A	D	8	F
9	C	A	4	F	2	7	6	1	D	3	8	B	E	5	G
1	6	D	5	B	8	E	G	F	A	2	4	9	3	7	C
G	8	F	3	D	5	9	A	B	E	7	C	2	1	6	4
8	E	B	1	6	A	D	2	G	7	4	9	5	C	F	3
D	G	C	6	7	9	4	5	8	F	A	3	E	2	B	1
4	7	3	9	G	1	F	B	5	2	C	E	D	8	A	6
A	F	5	2	8	E	3	C	D	1	6	B	4	9	G	7
7	9	E	C	3	F	G	8	6	4	B	5	1	A	2	D
6	D	4	A	E	C	5	7	2	8	9	1	F	G	3	B
5	3	2	G	1	B	A	9	7	C	F	D	8	6	4	E
F	1	8	B	2	4	6	D	A	3	E	G	C	7	9	5

1	2	6	4	5	3
3	4	5	1	6	2
6	5	3	2	4	1
4	6	1	3	2	5
5	1	2	6	3	4
2	3	4	5	1	6

FIGURE 2 – On peut aussi modifier la taille. À gauche, du 16×16 constitué de blocs 4×4 remplis avec des chiffres hexadécimaux (i.e. entre 0 et 9 ou entre A et F). À droite, du 6×6 constitué de blocs alors rectangulaires 3×2 remplis avec des chiffres entre 0 et 6.

Le but du jeu est de retrouver la grille complète à partir d'une grille partiellement remplie (voir figure 4). Cette grille partiellement remplie ne doit pouvoir amener qu'à une unique solution valide, i.e. respectant les contraintes (voir figure 5).

3.2 Multidoku

Une des extensions possibles du sudoku et d'en avoir plusieurs qui partagent des parties, que cela soit des lignes et/ou des colonnes et/ou des blocs (voir figures 6 et 7). Les règles sont alors les mêmes dans chacun des sudokus, à ceci près que les parties partagées doivent évidemment être identiques.

1. https ://fr.wikipedia.org/wiki/Algorithmes_de_résolution_des_sudokus
2. Pour information il y a des championnats du monde de sudoku.
3. Vous pouvez voir des exemples et les essayer ici : https ://sudoku-puzzles.net/fr/

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

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

FIGURE 3 – Mais on peut aussi changer la forme des blocs, qui peuvent également ne pas être convexes mais simplement connexes (à droite, la couleur des chiffres indique les blocs).

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

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

FIGURE 4 – À gauche une grille à compléter, à droite son unique solution.

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

FIGURE 5 – Cette grille n'est pas valide car les remplissages 2/7/7/2 et 7/2/2/7 sont possibles et la solution n'est donc pas unique.

3.3 Résolution

Le sudoku peut être considéré comme un problème de satisfaction de contraintes. En pratique, la résolution peut s'appuyer sur 2 mécanismes [4] : l'application de règles de déduction et le retour sur trace. Ces méthodes vont être expliquées sur le sudokus, mais sont évidemment directement adaptables au multidokus.

3.3.1 Règles de déduction

Plusieurs règles déductives peuvent être formalisées pour contraindre les valeurs des chiffres dans les différentes cases. De manière directe, comme il ne doit y avoir qu'une seule fois chaque symbole par ligne/colonne/bloc, si une case a un certain symbole s alors toutes les autres cases de la même ligne/colonne/bloc ne peuvent pas contenir s. De la même manière, si un symbole s n'est possible que dans une seule case d'un.e ligne/colonne/bloc, alors s doit être affecté à cette case. Mais on peut aussi dériver d'autres contraintes. Par exemple si 2 cases d'une même ligne (ou colonne ou bloc) ne peuvent avoir que les symboles s_1 et s_2, alors les autres cases de la même ligne (ou colonne ou bloc) ne peuvent pas contenir s_1 ou s_2.

Bien qu'un sudoku ne puisse avoir qu'une solution et qu'il y a donc forcément toujours une contrainte qui permette de forcer la valeur d'(au moins) une case, la détermination exhaustive de toutes les règles déductives est trop compliquée (et/ou peu efficace) et les solveurs se limitent donc à un ensemble restreint de règles (simples) comme celles mentionnées au-dessus (même si la liste n'est pas exhaustive). Bien que cela soit suffisant pour

4. https ://www.ams.org/notices/200904/rtx090400460p.pdf

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

FIGURE 6 – Un multidoku constitué de 5 sudokus 3×3, avec les chiffres en rouges dans les blocs partagés.

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

FIGURE 7 – Un multidoku constitué de 5 sudokus 3×3, avec les chiffres en rouges dans les lignes/colonnes partagées.

résoudre la plupart des sudokus, il peut arriver que certaines configurations (pour les sudokus (très) difficiles)) ne puissent être résolues avec seulement ces règles simples (voir par exemple figure 8).

3.3.2 Retour sur trace

Pour surmonter cette limitation, le solveur va alors choisir aléatoirement un symbole (possible) pour une des cases encore non déterminée, puis appliquer de nouveau les règles de déduction jusqu'à

— ce qu'aucune règle ne permet de déterminer la valeur d'aucune autre case. On se retrouve alors dans le cas précédent et l'algorithme va de nouveau faire un choix aléatoire pour une des cases indéterminées et on réitère.

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

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

FIGURE 8 – À gauche, une grille où l'application de règles déductives (simples) ne permet pas de trouver de nouvelles valeurs de cases. Il existe pourtant bel et bien une (unique) solution (à droite).

— une incompatibilité, i.e. une case qui ne puisse se faire attribuer aucune valeur. Dans ce cas c'est que le dernier choix aléatoire était mauvais et on va donc faire un autre choix pour la case considérée. Si aucune autre possibilité n'est faisable, c'est le choix précédent qu'il faut remettre en cause et ainsi de suite. i.e. on fait un retour sur la trace explorée pour trouver la solution possible.
— une solution, qui par définition est unique et on peut donc arrêter l'exploration.

Si l'on reprend par exemple la grille de la figure 8, l'application de cette méthode peut nous donner le déroulement montré sur la figure 9. Évidemment le choix des cases où le choix est fait doit être déterminé de manière efficace pour limiter le temps de résolution de la grille.

Cet algorithme correspond en fait à un parcours en profondeur de l'arbre des possibles (voir figure 10), les branches étant créées à chaque fois qu'un choix doit être fait parce que les règles de déduction ne permettent plus de fixer la valeur d'autres cases. On peut aussi imaginer utiliser un ensemble de règles vide, i.e. le choix se fera pour chaque case au hasard. Ce n'est évidemment pas la méthode la plus efficace, mais il est garanti de trouver une solution ... possiblement au bout d'un temps assez long puisqu'en pratique plusieurs centaines de milliers de combinaison risquent de devoir être évaluées.

3.4 Génération

On va d'abord générer une grille complète qui respecte les règles du sudoku. Pour cela on peut utiliser l'algorithme de résolution à partir d'une grille vide. Il n'y aura évidemment pas qu'une [5] seule solution dans ce cas, mais on veut "juste" en trouver une. Si l'on veut mettre un peu de variabilité dans les grilles générées, il faudrait évidemment mettre de l'aléatoire dans le choix des cases où une valeur sera testée ainsi que dans la valeur testée parmi celles possibles.

Une façon de générer une grille (à jouer) de sudoku à partir d'une grille complète est de supprimer itérativement des chiffres de la grille jusqu'à ce que celle-ci ne soit plus résolvable [6], i.e. qu'elle amène à plusieurs solutions possibles. Si cela arrive, il faut remettre en place le chiffre supprimé et tester la suppression d'un autre chiffre. Comme pour la partie de la résolution par retour sur trace, un arbre des possibilités est un bon moyen de structurer cet algorithme, et le choix des chiffres à enlever va aussi impacter la rapidité de génération des grilles.

On peut générer plusieurs grilles incomplètes à partir d'une même grille complète initiale :
— On n'est obligé de supprimer tous les chiffres jusqu'à ne plus pouvoir en enlever sans rendre la grille invalide. On peut choisir un nombre prédéterminé de chiffres que l'on veut supprimer, ce qui permet par exemple de régler le niveau de difficulté du jeu (globalement plus il y a de chiffres restants, plus la grille est facilement résolvable).
— À partir d'une grille complète, on peut généner d'autres grilles qui respectent les contraintes du sudoku en permutant les colonnes ou les lignes dans un bloc, en permutant des lignes ou colonnes de blocs ou en permutant les symboles. On peut donc aussi le faire sur la grille incomplète, sans remettre en cause le fait qu'il s'agisse de grilles valides.

5. Sur une grille classique 3×3 il y en a 6 670 903 752 021 072 936 960.
6. Pour le sudoku classique 3×3 on sait qu'il doit y avoir au moins 17 indices https ://arxiv.org/pdf/1201.0749.

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

(a) On va par exemple choisir la valeur de la case avec un ?, qui, de par les contraintes/règles de déduction, ne peut prendre que les valeurs 4 ou 6

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

(b) Si on choisit un 6 pour ?, l'application des règles de déduction nous amène jusqu'à ce point. On va alors faire un choix pour la case ? qui, d'après les contraintes peut être un 1 ou 4.

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

(c) Si on choisit un 1 pour ?, l'application des règles de déduction nous indiquent que la case X n'a plus de solution possible.

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

(d) On va alors choisir l'autre solution, i.e. 4, pour ?. Mais les règles de déduction nous indiquent là aussi un problème, vu que les cases X n'ont pas de solution possible.

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

(e) Comme il n'y a plus d'autre choix possible pour ?, on revient donc à l'autre choix possible pour ? qui est 4. On applique alors les règles de déduction qui nous amènent à la solution de la grille.

FIGURE 9 – Exemple d'application du retour sur trace sur la grille de la figure 8.

4 Tronc commun

Vous devez proposer une conception permettant d'implémenter n'importe quel sudoku ou multidoku, de procéder à sa résolution, et de générer de nouvelles grilles. En particulier votre programme permettra via un menu textuel :

- de rentrer une grille à résoudre
- de choisir la méthode de résolution : que des règles avec un ensemble à préciser (quite à ne pas réussir à résoudre), que du retour sur trace ou mixe des 2
- d'afficher [7] le sudoku/multidoku résolu (ou de dire si il n'est pas résolvable)
- d'afficher/loguer la suite d'"opérations" effectuées pour résoudre la grille (ou pourquoi elle n'est pas résolvable)
- de générer une ou plusieurs grilles à résoudre à partir d'une grille complète et d'un niveau de difficulté

7. Si vous voulez faire des affichages avec couleur en Java, regarder les codes ANSI.

FIGURE 10 – Représentation de l'algorithme de retour sur trace (figure 9) sous forme d'arbre des possibles. On explore d'abord le sous-arbre gauche, puis son sous-sous-arbre gauche, ... (i.e on l'explore en profondeur), jusqu'à arriver à un problème auquel cas on remonte pour explorer un autre sous(-sous...) arbre) ou à la solution.

5 Extensions possibles

5.1 Environnement de travail (0.5 point)

Vous utiliserez un git (dont l'adresse sera à fournir dans le rapport) pour faire le suivi de versions tout au long de votre projet.

5.2 Tests unitaires (2 points)

Vous implémenterez des tests unitaires des méthodes (principales) de votre programme (regardez `https://junit.org/junit5/`).

5.3 Interface graphique (4 points)

En plus d'une interface textuelle, vous proposerez une interface graphique pour votre programme. Vous respecterez au mieux le *design pattern* MVC (Modèle Vue Contrôleur), à savoir a minima avoir les classes d'affichages séparées de celles du modèle.

5.4 Fichier de configuration et sauvegarde (1 point)

La grille pourra être fournie dans un fichier ainsi que le choix de l'algorithme de résolution (règles de décisions utilisées, utilisation ou non du retour sur trace). Les logs de la résolution ainsi que les différents états de la grille pourront être sauvegardés (et être rechargés pour une future résolution) dans un format humainement lisible et éditable.

5.5 Rajout de règles de déduction (1 point)

On veut que l'utilisateur puisse rajouter (sans modifier le code Java) des règles de déductions à celles fournies par votre programme.

5.6 Grilles avec multiples solutions (0.5 point)

On autorise maintenant d'avoir plusieurs solutions à une grille et on veut que le programme trouve toutes les solutions possibles. Évidemment on suppose que la grille n'a pas un trop grand nombre de solutions (par exemple pas la grille vide donc) pour que le programme converge rapidement.

5.7 Rajout de contraintes (1 point)

On veut proposer d'autres variantes en rajoutant des contraintes à la grille de sudoku. Par exemple on peut forcer que les diagonales d'un sudoku possèdent les N symboles, que certaines cases aient une valeur paire ou impaire, ou donner les comparaisons ($<$ ou $>$) entre les cases adjacentes d'un même bloc, ...

5.8 Résolution par l'humain (1 point)

On veut rajouter la possibilité à l'humain de résoudre lui-même la grille de sudoku. On utilisera alors l'algorithme de résolution comme une aide (l'utilisateur pourra par exemple demander quelles valeurs sont possibles dans une case, quelle case peut être remplie, avec quelle valeur, ...).