

PROJET D'APPLICATION INFORMATIQUE

Résolveur de sudoku

Rapport

Auteurs :
FRÉTARD Loïc
HARDOUIN Paul
BOUCHER Laëtitia
BRUN Florian

Responsable :
M. GUESNET

Table des matières

Introduction	4
1 Vocabulaire utilisé	5
2 Fonctionnalités	6
2.1 Fonctionnalités obligatoires	6
2.2 Fonctionnalités facultatives	8
3 Implantation	8
3.1 SudokuModel	8
3.2 GridModel	9
3.3 CellModel	9
3.4 Command	10
3.5 History	10
3.6 RuleManager	10
3.7 Report	10
4 Heuristiques implantées	11
4.1 Un seul candidat : (only candidate)	11
4.2 Un candidat unique : (one candidate)	12
4.3 Des jumeaux/triplets : (pair/triplet)	12
4.4 Interactions entre régions	14
4.5 Candidats identiques	14
4.6 X-Wing	15
4.7 Groupes isolés	16
4.8 Groupes mélangés	16
5 Extension de l'application	18
5.1 Structure d'un fichier grilleXX.txt	18
5.2 Création d'une règle	19
6 Interface	20
6.1 Le composant principal - La grille	20

7	Jouabilité	23
8	Difficultés rencontrées	33
9	Répartition du travail	33
10	Conclusion	34
11	Annexes	34

Introduction

Objectif

Ce projet a pour objectif de réaliser une modélisation d'un résolveur de sudoku en langage Java.

L'origine du sudoku

Le sudoku a été inventé en 1979 par Howard Garns, un pigiste spécialisé dans les puzzles, et publié cette même année pour la première fois dans Dell Magazines sous le nom de Number Place. Après avoir été introduit au Japon, le nom devient Sudoku. En 2004, Le Times publie une première grille puis les autres journaux suivent. Depuis, le phénomène a fait le tour du monde et est arrivé en France. Inspiré du carré latin de Leonhard Euler, le but du jeu est que chaque ligne, colonne et région de 3x3 cases contienne chaque chiffre de 1 à 9 une seule fois.

Actuellement, il existe de nombreuses variantes pour le sudoku allant de la plus simple à la plus complexe. Comme exemple, nous avons : changer la taille de la grille et ne plus prendre systématiquement 3x3 cases mais 2x2 cases (idéal pour apprendre, se familiariser lorsque c'est la première fois que l'on joue), ou encore 10x10 cases si on aime les défis. Parmi ces variantes nous trouverons également la possibilité de remplacer les chiffres par des symboles ainsi, à la place de compter en base 10, nous pourrions compter en base 16, en hexadécimal, de 0 à F (pour des régions 4x4).

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

1 Vocabulaire utilisé

Cette section a pour but de répertorier les différents mots de vocabulaire que nous emploierons dans la suite.

Pour la partie code, nous utiliserons les équivalents de ces mots en anglais, ils seront noté ci-dessous entre parenthèses.

- Sudoku : terme désignant le jeu en lui-même.
- Grille (grid) : il s'agit de l'ensemble des régions et des cellules.
- Région (area) : ensemble contenant des cellules. La région est une unité du Sudoku, qui doit obligatoirement contenir une fois chaque symbole, mais une fois seulement.
- Cellule (cell) : ensemble pouvant contenir des candidats ou une valeur.
- Heuristiques : termes désignant les règles dont nous nous servirons pour créer nos algorithmes afin de résoudre une grille de sudoku.
- Ligne (row) : la ligne est une unité du Sudoku, qui doit obligatoirement contenir une fois chaque symbole, mais une fois seulement.
- Colonne (column) : la colonne est une unité du Sudoku, qui doit obligatoirement contenir une fois chaque symbole, mais une fois seulement.
- Unité (unit) : nous appelons "unités" les diverses parties de la grille de Sudoku : lignes, colonnes, et régions qui présentent des caractéristiques identiques en nombre de cases, de symboles et de règlements.

Il y a 27 unités dans une grille standard de Sudoku. Diverses méthodes expliquées dans ce guide, s'appliquent indifféremment à toutes les unités d'une grille, alors que d'autres pas. C'est à chaque fois précisé.

- Symbole (Symbol) : vous devez remplir chaque unité du Sudoku avec 9 symboles différents.

Dans une grille standard de Sudoku, on utilise les chiffres 1 2 3 4 5 6 7 8 et 9. Il est possible de trouver des grilles de Sudoku à remplir avec d'autres symboles.

- Case (cell) : chaque unité de la grille standard de Sudoku contient 9 cases, et chaque case un seul symbole. Il y a 81 cases dans une grille standard.
- Candidat (candidate) : les candidats sont les diverses possibilités de symboles pour une case donnée. Le principe du jeu est de réduire les nombres de candidats pour obtenir le symbole final à faire figurer dans la case.
- Valeur (Value) : une valeur est représentée par un unique symbole pour une case donnée. L'objectif étant de fixer une valeur dans chaque case et de trouver les bonnes valeurs.

2 Fonctionnalités

2.1 Fonctionnalités obligatoires

1. Fichier de chargement pour une nouvelle grille : l'utilisateur peut demander le chargement d'une grille afin de pouvoir effectuer l'une de deux actions suivantes : la compléter ou la créer afin d'avoir une nouvelle grille.
2. Sauvegarde/chargement d'une partie : afin de procéder à une sauvegarde, l'utilisateur peut enregistrer la grille actuelle dans un fichier et peut la récupérer ensuite grâce à une option de chargement.
3. Réinitialisation de la grille : option permettant de remettre la grille de sudoku telle qu'elle était à son chargement.
4. Modification d'une cellule : une cellule devra être modifiable afin de pouvoir ajouter/retirer des candidats.
5. Valeur définitive : dans une cellule, on pourra fixer une valeur définitive qui ne sera plus modifiable par la suite.
6. Élimination des possibilités pour la valeur sur la ligne, colonne et région sur lesquels se trouve la cellule : on pourra éliminer plusieurs possibilités d'une cellule par rapport à une région.
7. Possibilité : valeur supposé valide d'une cellule dans l'attente de la faire passer en valeur définitive.

8. Effacement : permet d'effacer une cellule, de la remettre à zéro.
9. Demander l'aide : bouton permettant à l'utilisateur de bénéficier d'une aide afin de résoudre la grille en donnant des indications permettant de ne pas rester bloqué.
10. Expliquer la règle : explication d'une règle de résolution.
11. Appliquer la règle : permet d'utiliser la règle choisie.
12. Résoudre la grille pas à pas : système permettant une résolution de grille de manière pas à pas, c'est-à-dire en résolvant cellule après cellule en appliquant différents algorithmes nécessitant l'utilisation de règles qui seront communiquées à l'utilisateur.
13. Résoudre complètement : solution complète de la grille de sudoku.
14. Si la grille ne peut pas être résolue : proposer une réinitialisation avant résolution en indiquant que la grille est irrésolvable.
15. Classement par difficultés : grille classée en fonction des heuristiques nécessaires permettant d'évaluer un système de niveaux.
16. Annuler : permet de défaire la dernière action action réalisée sur la grille.
17. Refaire : permet de défaire l'action annulé.
18. Raccourcis clavier : gestion des appuis simultanés de touches du clavier générant des possibilités inscrites dans le menu tel les sauvegardes/chargements de nouvelles grilles, l'annulation/réactivation d'une action etc.
19. Demande d'aide : système permettant à l'utilisateur de bénéficier d'une aide en cas de problème.
20. Tutoriel : quelques explications pour tous ceux qui désirent (ré)apprendre à résoudre un sudoku, quelles sont les règles, comment cela fonctionne etc.

21. Guide : quelques explications sur l'utilisation de l'application (gestion des boutons, comment jouer etc.).
22. Masquer la grille pendant la pause : lorsque l'utilisateur appuyera sur le bouton pause, ce dernier verrouillera la session de jeu et ainsi, il ne sera plus possible de jouer tant que l'utilisateur n'aura pas repris le jeu en appuyant sur le bouton reprendre.

2.2 Fonctionnalités facultatives

1. Prise en charge de grilles de dimensions variables : l'utilisateur peut demander à avoir une grille de n cases, ainsi, si n vaut 2, l'utilisateur pourra utiliser une grille de deux cases sur deux, offrant une grille de 4 cases au total.
2. Chronomètre : système permettant à l'utilisateur de connaître le temps qu'il a mis à résoudre une grille.
3. Éditeur/générateur : possibilités pour l'utilisateur de créer ses propres grilles selon des niveaux de difficultés.
4. Samourai : système de jeu impliquant 5 grilles de sudoku formant une grille géante disposé de telle sorte à ce qu'une grille centrale soit partagé par les 4 autres grilles disposés au coin de cette dernière.

3 Implantation

Pour implanter notre sudoku, nous avons choisi de fonctionner selon le modèle MVC, c'est-à-dire le système Modèle-Vue-Contrôleur. Afin de visualiser notre organisation, des diagrammes ont été réalisés et sont disponibles en annexe (annexe 18 : diagramme model, annexe 19 : diagramme history, annexe 20 : diagramme heuristic).

3.1 SudokuModel

Le sudoku possède deux grilles(GridModel) : celle du joueur et la solution. Le sudoku est aussi composé d'un historique. La grille solution est générée par application des heuristiques à l'aide d'un

gestionnaire d'heuristiques (RuleManager). La grille joueur est le résultat des interactions du joueur (ajouter/supprimer un candidat ou sélectionner/désélectionner une valeur).

L'historique répertorie les actions du joueur avec possibilité d'annuler ses actions.

Le joueur a gagné quand sa grille est finie, sans conflits de nombres (deux 6 sur la même ligne par exemple).

La partie est finie quand le joueur a rempli toutes les cases de sa grille d'une valeur. À tout moment le joueur peut demander de l'aide ou demander un indice (exécution d'une heuristique). On peut sauvegarder ou enregistrer sa partie ou encore générer une grille à partir d'un fichier texte (sur la première ligne doit se trouver la hauteur et la largeur de la grille puis sur les lignes qui suivent les valeurs des cases fixes du sudoku). Le joueur peut à tout moment réinitialiser sa partie, c'est à dire, mettre sa grille comme elle était avant toute interaction de sa part, seules les valeurs des cases fixes restent.

3.2 GridModel

Une grille possède une taille n (hauteur x largeur) fixe, interchangeable donnée en paramètre lors de l'initialisation et un tableau à double entrée (lignes, colonnes) de cellules (ModelCell). La grille connaît son nombre de valeurs/candidats possibles. Une coordonnée (ICoord) représente une des cases du tableau : elle commence de 0 et va jusqu'à $n - 1$. Elle peut donner la cellule par rapport à une coordonnée et inversement, mais aussi un ensemble de cellules d'une unité (colonne, ligne, région) d'une cellule en lui donnant soit sa coordonnée soit elle-même en paramètre. La grille possède un détecteur de validation de coordonnée pour vérifier que la coordonnée est toujours dans le tableau. De même que SudokuModel, on peut ajouter/supprimer un candidat de la grille ou ajouter/supprimer (si ce n'est pas une case fixe) une valeur.

3.3 CellModel

Une cellule possède un tableau de candidat. Une cellule peut être soit modifiable, c'est-à-dire qu'on peut modifier sa valeur, ou soit fixe, immuable : ce sont les cases du départ de la grille. Comme la grille, elle connaît son nombre de valeurs/candidats possibles qui est donné en paramètre. La cellule a soit une valeur, soit une liste de candidats possibles. On peut ajouter/supprimer un candidat de la cellule ou ajouter/supprimer (si ce n'est pas une case fixe) une valeur. On peut aussi demander à tout moment sa valeur, si une valeur est un candidat potentiel pour cette cellule ou même inverser ces candidats (si un nombre était un candidat potentiel ce n'est plus le cas et inversement).

3.4 Command

Une commande est un objet capable de modifier une grille selon certains critères (ajout/suppression candidat, ajout/suppression valeur). `AbstractCommand` est la classe fournissant un mécanisme général pour les commandes, elle est la super classe de `AddValue` (ajoute une valeur à la cellule), `AddCandidate` (ajoute un candidat à la cellule), `RemoveValue` (supprime une valeur à la cellule), `RemoveCandidate` (supprime un candidat à la cellule).

3.5 History

L'historique est une sorte de pile chronologique bornée. On peut toujours ajouter des éléments (`Command`) à un historique : lorsque l'historique est plein, rajouter un nouvel élément fait disparaître le plus ancien. On peut avancer et reculer le curseur repérant l'élément courant à loisir dans l'historique, mais si le curseur n'est pas sur l'élément le plus récent, ajouter un élément dans l'historique à cet instant fait disparaître les éléments postérieurs au curseur.

3.6 RuleManager

Deux comportements possibles existent pour une heuristique : soit on trouve la valeur à mettre dans la case (exemple : seul candidat), soit on peut supprimer des candidats dans les cases de la grille (exemple : jumeaux et triplet) ce qui réduit les possibilités de valeurs à chaque boucle d'heuristique. Le gestionnaire va parcourir les heuristiques de la plus simple à la plus complexe (qui sont classées dans une classe énumérative `Rule` où chaque algorithme de résolution est associé à une unique heuristique) pour trouver celle qui apporte une solution. Le gestionnaire peut demander à tout moment la description ou l'exécution de la dernière solution trouvée. L'heuristique choisie est un rapport (`Report`).

3.7 Report

Le rapport répertorie les cellules qui sont en lien avec l'action (suppression ou ajout de candidats/valeur) ou les cellules qui ont aidé à ce raisonnement. Elle possède une description qui lui est propre en rapport avec l'heuristique et une exécution des commandes (suppression candidats ou ajout valeur).

4 Heuristiques implantées

4.1 Un seul candidat : (only candidate)

C'est l'heuristique la plus simple. Si une case ne contient qu'un candidat c'est que ce candidat est la valeur finale de la case obligatoirement.

Exemple :

On peut voir ici que sur la troisième ligne et la sixième colonne se trouve le candidat 4. Comme étant le seul candidat dans cette case, de ce fait, on peut le sélectionner comme valeur.

FIGURE 1 – seul candidat

⁶ 8 9	5	³ 8 9	7	³ 4 9	2	⁶ 4	1	⁴ 8 9
² 9	³ 4 9	^{2 3} 9	6	^{1 3} 4 5 9	8	^{4 5} 4 5 7	^{4 5} 4 5 7	^{4 5} 4 5 7 9
1	^{4 6} 8 9	7	^{5 9} 4 5 9	^{4 5 9} 4 5 9	4	3	^{4 5 6} 8	2
² 7 9	^{7 9} 7 9	6	1	^{4 2} 7 9	5	8	^{2 3} 4 7	^{4 3} 4 7
3	^{1 2} 7 8 9	^{1 2} 8 9	² 9	^{2 4} 7 9	^{4 7} 4 7	^{1 2 4 5} 4 5 7	^{2 4 5} 4 5 7	6
^{2 5} 7	^{1 7} 7	4	8	^{2 6} 7	3	9	^{2 5 7} 5 7	⁵ 7
4	^{3 6} 8 9	5	^{2 6} 8	^{2 6} 8	⁶ 6	7	^{2 3 6} 8 6	1
⁶ 7 8	^{1 3 6} 7 8 6	^{1 3} 8	4	^{2 5 6} 7 8	9	^{2 5 6} 5 6 8	^{2 3 5 6} 5 6 8	^{5 3} 5 8
⁶ 7 8	2	⁸ 8	3	^{5 6} 7 8	1	^{4 5 6} 4 5 6	9	^{4 5 8} 4 5 8

4.2 Un candidat unique : (one candidate)

Prenons maintenant une ligne complète. Dans les cases vides, nous avons noté la liste des candidats potentiels. Chaque chiffre de 1 à 9 devant obligatoirement se trouver sur la ligne de façon unique, si dans les candidats de toutes les cases de la ligne, un candidat n'apparaît qu'une seule fois, alors c'est qu'il doit effectivement être placé dans cette case. Il est possible d'utiliser cette méthode dans toutes les unités de la grille (lignes, colonnes, régions).

Exemple : On peut voir ici que sur la première ligne, deuxième colonne, il y a un seul candidat dans la case. Donc il n'y a qu'une solution : 5.

FIGURE 2 – candidat unique

6	<small>4 5 8</small>	9	7	<small>1 2 5 8</small>	<small>2 5 8</small>	3	<small>1 2 8</small>	<small>1 2</small>
2	<small>7 8</small>	3	4	<small>1 8 9</small>	<small>8 9</small>	<small>1</small>	6	5
<small>1 5 7 8</small>	<small>5 7 8</small>	<small>5 8</small>	<small>1 2 5</small>	3	6	4	9	<small>1 2 7</small>
<small>5 7 8 9</small>	1	4	6	<small>2 5 8 9</small>	<small>2 5 7 8 9</small>	<small>5</small>	3	<small>9</small>
<small>5 7 8 9</small>	<small>5 7 8</small>	6	<small>5 9</small>	<small>3 4 5 8 9</small>	<small>5 7 8 9</small>	2	<small>1 4 5</small>	<small>1 9</small>
<small>5 9</small>	3	<small>2 5</small>	<small>2 5 9</small>	<small>4 5 9</small>	1	8	7	<small>6 9</small>
<small>4 5</small>	9	1	8	7	<small>2 5</small>	<small>5 6</small>	<small>2 5</small>	<small>2 3 6</small>
3	2	<small>5</small>	<small>1 5 9</small>	<small>1 5 6 9</small>	4	7	<small>1 5</small>	8
<small>5 8</small>	<small>5 6 8</small>	7	<small>1 2 5</small>	<small>1 2 5 6</small>	3	9	<small>1 2 5</small>	4

4.3 Des jumeaux/triplés : (pair/triplet)

Il n'est pas toujours possible de découvrir dès le début l'emplacement final et définitif d'un symbole. Cependant il est parfois possible de savoir dans quelle ligne ou colonne il ne se trouve pas, et donc d'en déduire dans quelle partie de la région il va finir par se trouver. Quand une unité contient deux cases avec une même paire de candidats (et eux seuls) alors ces candidats ne peuvent se trouver dans une autre case de l'unité.

Si ce n'est pas suffisant pour pouvoir le placer immédiatement. C'est cependant très utile pour supprimer les candidats de cette ligne. Les triplés fonctionnent exactement sur le même principe, mais avec 3 symboles libres

dans la même ligne ou colonne de la région.

Il est possible d'utiliser cette méthode dans toutes les régions alignées (horizontalement ou verticalement) et dans toutes les lignes ou colonnes. Si à l'intérieur d'un bloc, un chiffre n'est possible que dans une ligne/colonne, alors nous pouvons en déduire que ce chiffre ne peut pas être présent dans les cases de cette ligne/colonne n'appartenant pas au bloc.

Exemple :

On peut voir ici que sur la région centrale, les 1 sont uniquement présents et alignés sur la verticale. Si bien qu'on peut en déduire qu'un des deux 1 est la solution, donc on peut supprimer les autres de la colonne.

FIGURE 3 – jumeaux, triplés

<div>3 4 7</div>	<div>1 3 4 7 9</div>	<div>1 3 4 7 9</div>	<div>1 4 9</div>	<div>6</div>	<div>5</div>	<div>8</div>	<div>1 4</div>	<div>2</div>
<div>2 3 4 7 9</div>	<div>5</div>	<div>1 3 4 6 7 8 9</div>	<div>1 2 4 8 9</div>	<div>1 2 3 4 7 8 9</div>	<div>1 2 3 4 7 8</div>	<div>1 3 4 9</div>	<div>1 6 4 3 6 9</div>	
<div>2 3 4 9</div>	<div>1 2 3 4 6 8 9</div>	<div>1 3 4 6 8 9</div>	<div>1 2 4 8 9</div>	<div>1 2 3 4 8 9</div>	<div>1 2 3 4 8</div>	<div>7</div>	<div>1 6 4</div>	<div>5</div>
<div>2 4</div>	<div>1 2 4 8</div>	<div>1 4 8</div>	<div>5</div>	<div>1 2 4 8</div>	<div>9</div>	<div>6</div>	<div>3</div>	<div>7</div>
<div>6</div>	<div>2 3 4 7 8 9</div>	<div>3 4 7 8 9</div>	<div>2 4 8</div>	<div>2 3 4 8</div>	<div>2 3 4 8</div>	<div>2 4 9</div>	<div>5</div>	<div>1</div>
<div>5</div>	<div>1 2 3 4 8 9</div>	<div>1 3 4 8 9</div>	<div>7</div>	<div>1 2 3 4 8</div>	<div>6</div>	<div>2 4 9</div>	<div>2 4 8</div>	<div>4 8 9</div>
<div>4 7</div>	<div>4 6 7</div>	<div>4 6 7</div>	<div>3</div>	<div>1 2 4 7 8</div>	<div>1 2 4 7 8</div>	<div>5</div>	<div>9</div>	<div>4 6 8</div>
<div>8</div>	<div>3 4 6 7 9</div>	<div>2</div>	<div>1 4 6 9</div>	<div>5</div>	<div>1 4 7</div>	<div>1 3 4</div>	<div>1 6 4 7</div>	<div>3 6</div>
<div>1</div>	<div>3 4 6 7 9</div>	<div>5</div>	<div>2 4 6 8 9</div>	<div>2 4 7 8 9</div>	<div>2 4 7 8</div>	<div>2 3 4</div>	<div>2 6 4 7 8</div>	<div>3 6 8</div>

4.4 Interactions entre régions

Si à l'intérieur d'une ligne/colonne, un chiffre n'est possible que dans un bloc, alors nous pouvons en déduire que ce chiffre ne peut pas être présent dans les autres cases de ce bloc.

Exemple :

Dans cet exemple, on peut constater que dans R7 et R9, le 8 n'est pas présent sur L8. On peut donc supprimer les 8 dans la R8 à l'exception de ceux sur L8.

FIGURE 4 – Interactions entre régions

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

4.5 Candidats identiques

Cette méthode ne donne pas la possibilité de répondre à la question du choix, mais donne la possibilité de supprimer des candidats indésirables. Car, en effet, si deux cases ne contiennent que deux fois les mêmes candidats On peut être certain que ces deux candidats vont finalement terminer dans l'une et dans l'autre. Et donc supprimer ces candidats des autres cases.

Cette méthode s'applique dans les cas suivants :

- avec 2 candidats dans 2 cases
- avec 3 candidats dans 3 cases
- avec 4 candidats dans 4 cases

...

- avec N candidats dans N cases

Il est possible d'utiliser cette méthode dans toutes les unités de la grille.

4.6 X-Wing

Le nom X-Wing (ou aile en X) provient de la figure tracée par cette méthode.

En effet le principe est basé sur le choix à faire dans l'emplacement d'une valeur.

En effet si une valeur est placée dans un coin, la même valeur ne pourra être placée que dans le coin opposé, ce qui trace les diagonales des 2 possibilités.

Il est nécessaire de trouver 2 unités (lignes, colonnes ou régions) dans lesquelles on ne trouve que 2 candidats pour une même valeur. Et qu'en plus on retrouve cette correspondance dans 2 unités du même type, reliées par des unités communes aux unités de base. Pour pouvoir supprimer les autres candidats des unités communes.

Cette méthode s'applique dans les cas suivants, avec 2 candidats :

- dans 2 colonnes, en supprimant les candidats dans 2 lignes
- dans 2 colonnes, en supprimant les candidats dans 2 régions
- dans 2 lignes, en supprimant les candidats dans 2 colonnes
- dans 2 lignes, en supprimant les candidats dans 2 régions
- dans 2 régions, en supprimant les candidats dans 2 lignes
- dans 2 régions, en supprimant les candidats dans 2 colonnes

Il est possible d'utiliser cette méthode dans toutes les unités de la grille (régions, lignes, colonnes).

Exemple :

Dans cet exemple, il n'est possible de trouver le candidat 5 qu'à deux emplacements des lignes L2 et L8.

De plus, ces candidats font partie des colonnes communes C5 et C7.

Il est donc possible de supprimer tous les autres candidats 5 de ces 2 colonnes.

FIGURE 5 – XWing

<div>1 2 4 6 9</div>	<div>2 4 6 9</div>	3	<div>7 9</div>	<div>1 2 7 5 9</div>	<div>2 5 9</div>	8	<div>1 4 6 7 9</div>	<div>1 4 5 6 7</div>
7	8	<div>4 9</div>	3	<div>1 5 9</div>	6	<div>1 5 9</div>	<div>1 4 9</div>	2
<div>1 2 6 9</div>	<div>2 6 9</div>	5	4	<div>1 2 7 9</div>	8	3	<div>1 6 7 9</div>	<div>1 6 7</div>
<div>6 9</div>	3	8	1	<div>6 7 9</div>	4	2	5	<div>6 7</div>
<div>2 4 5 6 9</div>	<div>2 4 5 6 9</div>	<div>4 6 9</div>	<div>7 8 9</div>	<div>2 6 7 8 9</div>	<div>2 9</div>	<div>1 6</div>	<div>1 3 4 6 7</div>	<div>1 3 4 6 7</div>
<div>2 4 6</div>	1	7	5	<div>2 6</div>	3	9	8	<div>4 6</div>
<div>3 5 9</div>	<div>5 9</div>	2	6	<div>3 5 8 9</div>	7	4	<div>1 3 8</div>	<div>1 3 5 8 6</div>
8	7	<div>4 6</div>	2	<div>4 5 3</div>	1	<div>5 6 3</div>	<div>3 6</div>	9
<div>3 4 5 6 9</div>	<div>4 5 6 9</div>	1	<div>8 9</div>	<div>4 5 8 9</div>	<div>5 9</div>	7	2	<div>3 5 6 8</div>

4.7 Groupes isolés

Si 3 candidats se retrouvent seuls dans 3 cases, il est possible de savoir qu'ils finiront bien dans ces 3 cases. Même si les 3 candidats ne sont pas présents dans les 3 cases et donc supprimer ces candidats des autres cases. Cette méthode s'applique dans les cas suivants :

- avec 2 candidats dans 2 cases
- avec 3 candidats dans 3 cases
- avec 4 candidats dans 4 cases
- ...
- avec N candidats dans N cases

Il est possible d'utiliser cette méthode dans toutes les unités de la grille (régions, lignes, colonnes).

4.8 Groupes mélangés

Cette méthode ressemble beaucoup à celle expliquée sous "Groupes isolés". Mais son application n'est pas la même car elle ne concerne que les cases qui contiennent les candidats, et pas les autres.

Si on ne retrouve 3 candidats, que dans 3 cases, on est certain qu'ils

vont terminer dans ces 3 cases.

Il est donc possible de supprimer les autres candidats de ces dernières.

Cette méthode s'applique dans les cas suivants :

- avec 2 candidats dans 2 cases de X candidats
- avec 3 candidats dans 3 cases de X candidats
- avec 4 candidats dans 4 cases de X candidats
- ...
- avec N candidats dans N cases de X candidats

La méthode des "Groupes isolés" permettait de supprimer les candidats des autres cases, alors que celle-ci donne la possibilité de supprimer les autres candidats dans les mêmes cases.

De plus, avec cette méthode, il est nécessaire de ne trouver les candidats concernés que dans les cases utilisées (alors que c'est justement l'inverse qui est utile dans l'autre méthode).

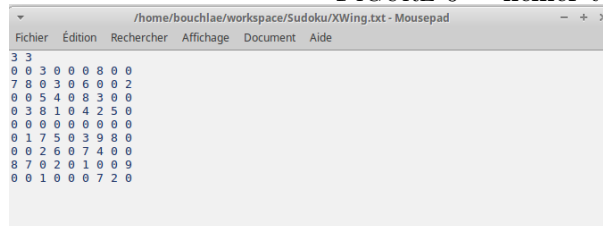
Il est possible d'utiliser cette méthode dans toutes les unités de la grille (régions, lignes, colonnes).

5 Extension de l'application

5.1 Structure d'un fichier grilleXX.txt

Ce fichier commence par les dimensions de la futur grille, d'abord le nombre de région par largeur, puis un espace, ensuite le nombre de région par hauteur, puis un saut de ligne. Le fichier continue par les valeurs des cases fixes espacées d'un espace entre chaque valeur. Les cases modifiables sont représentées par un zéro. Chaque ligne du fichier représente une ligne du sudoku comme suit la figure ci-dessous.

FIGURE 6 – fichier text



```

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

```

Le modèle transforme le fichier texte figure6 en cette grille figure7

FIGURE 7 – grille modélisé

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

5.2 Création d'une règle

On peut ajouter de nouvelle règle à l'application, pour cela il faut générer un rapport, c'est-à-dire être une sous-classe de ReportGenerator.

Une règle est constituée de sa description, d'une liste de valeur (ajout d'une valeur dans une case ou suppressions des candidats dans une ou plusieurs case(s)) et de quatre ensembles (pour la surbrillance de l'aide) :

- DELETION_CELL (rouge foncé) : ensemble des cellules concernées par la suppression d'un candidat
- DELETION_UNITS (rouge clair) : les unités des cellules de l'ensemble précédent
- DECISIVE_CELL (bleu foncé) : ensemble des cellules qui se base sur le raisonnement de la règle
- DECISIVE_UNITS (bleu clair) : les unités des cellules de l'ensemble précédent

Si DELETION_CELL n'est pas vide on peut en conclure que l'action de la règle est de supprimer les candidats de la liste dans cet ensemble de cellules. Au contraire si cet ensemble est vide c'est qu'on doit ajouter la valeur dans DECISIVE_CELL.

Si la règle n'aboutit à aucune avancée dans la grille (suppression candidat ou ajout valeur) elle renvoie null.

6 Interface

Notre application se présente de la manière suivante :
une barre de menu située au nord de la fenêtre vous permettra de choisir une action à réaliser parmi toutes les fonctionnalités qui vous sont proposées.

La grille de sudoku occupe le centre de notre application et se retrouve accompagnée d'une colonne, à l'est de la fenêtre, vous permettant de sélectionner parmi des boutons raccourcis, les fonctionnalités les plus utilisées telle défaire, refaire une action, demander un indice, réinitialiser la grille etc, puis, une zone d'aide, située en bas de la grille.

Il vous est également possible d'utiliser votre clavier afin de profiter des fonctionnalités que vous offre notre application.
Pour ce faire, il vous suffit de saisir la combinaison appropriée, combinaison que vous pourrez retrouver en consultant les listes des fonctionnalités disponibles dans le menu, au nord de l'application, où coïncide chaque fonctionnalité avec sa combinaison de la forme CTRL+LETTRE, qu'il vous suffit d'utiliser en appuyant simultanément sur la touche contrôle et la lettre correspondante à l'action souhaitée.

Pour ajouter/retirer des possibilités dans la grille, il vous suffit d'utiliser le bouton droit de la souris tandis que pour ajouter/retirer des candidats, il faudra utiliser le bouton gauche de votre souris.

6.1 Le composant principal - La grille

La grille est constituée de deux classes : Grid et Cell. Ces deux classes héritent de JPanel, conteneurs simples.

Cell La classe Cell est chargée d'afficher une cellule et possède donc un CellModel. Une Cell est constituée de plusieurs JPanel disposés sur un CardLayout (layout permettant de n'afficher qu'un seul composant à la fois). Ces JPanel sont destinés à accueillir des symboles (ici des JLabel) représentant chaque valeur possible, à l'exception du premier qui correspond à une cellule sans valeur. Ce premier JPanel contient donc d'autres JPanel, disposés sur un GridLayout (layout permettant de répartir les composant sur une grille occupant tout l'espace), chargés d'afficher les candidats.

Une Cell dispose d'une bordure pour bien la délimiter. Le GridLayout associé aux candidats est configuré pour les placer dans une grille carré de côté $n = \text{partie entière supérieur de la racine carré du nombre de valeur possible}$. En pratique, si les dernières lignes sont vides, la grille ne sera pas

carrée (ex : sudoku 6x6 -> candidats répartis sur une grille 2x3 au lieu de 3x3).

En ce qui concerne les contrôleurs, une Cell écoute les changements sur les propriétés liées à son modèle : VALUE pour afficher la bonne valeur (ou les candidats s'il n'y a pas de valeur), et CANDIDATE pour afficher, ou non, chaque candidat.

L'effet de changement de couleur de fond lors du survol à la souris est produit en écoutant les événements d'entrée et de sortie du composant liés à la souris, et ça pour chaque JPanel contenant un symbole (valeur ou candidat). Le traitement effectué est un simple affichage, ou non, du fond du JPanel, initialisé avec une couleur semi-transparente, pour laisser transparaître le fond de Cell (qui est un JPanel).

Tous ces contrôleurs sont assez directs. En revanche, le clique souris ne modifie pas directement le modèle. En effet, il nous était impossible de modifier directement le modèle étant donné que nous devions faire passer toutes les actions par l'historique. Nous avons donc choisi de lancer un événement de changement de propriété (VALUE ou CANDIDATE selon le clique et le composant) à destination principalement de Grid qui, lui possède tous les composants du modèle nécessaires.

Grid La classe Grid est chargée d'afficher une représentation d'un SudokuModel. L'affichage des cellules est délégué à un groupe de Cell qui sont regroupées en régions. C'est aussi cette classe qui s'occupe de récupérer les événements de clique souris depuis les Cell et, ainsi, crée les Command correspondantes et les envoie dans l'historique.

La disposition des cellules se fait grâce à un GridLayout contenant des JPanel (représentant les régions) disposant d'une bordure. Ces JPanel composés eux-mêmes de Cell dans des GridLayout.

En ce qui concerne l'aide, cette classe écoute les changements de la propriété liée LAST_REPORT de RuleManager (l'écouteur est envoyé au RuleManager du modèle (SudokuModel) par l'intermédiaire de ce dernier. Lorsque cette propriété change, la couleur de fond des cellules sont mise à jour. Si la nouvelle valeur de la propriété ne vaut pas null, les cellules correspondant aux coordonnées dans les ensembles du Report voient leur couleur de fond changer au profit de celle contenue dans le nom de son ensemble (CellSetName). Si la nouvelle valeur est null, alors les cellules liées à l'ancien Report reprennent leur couleur de fond initiale.

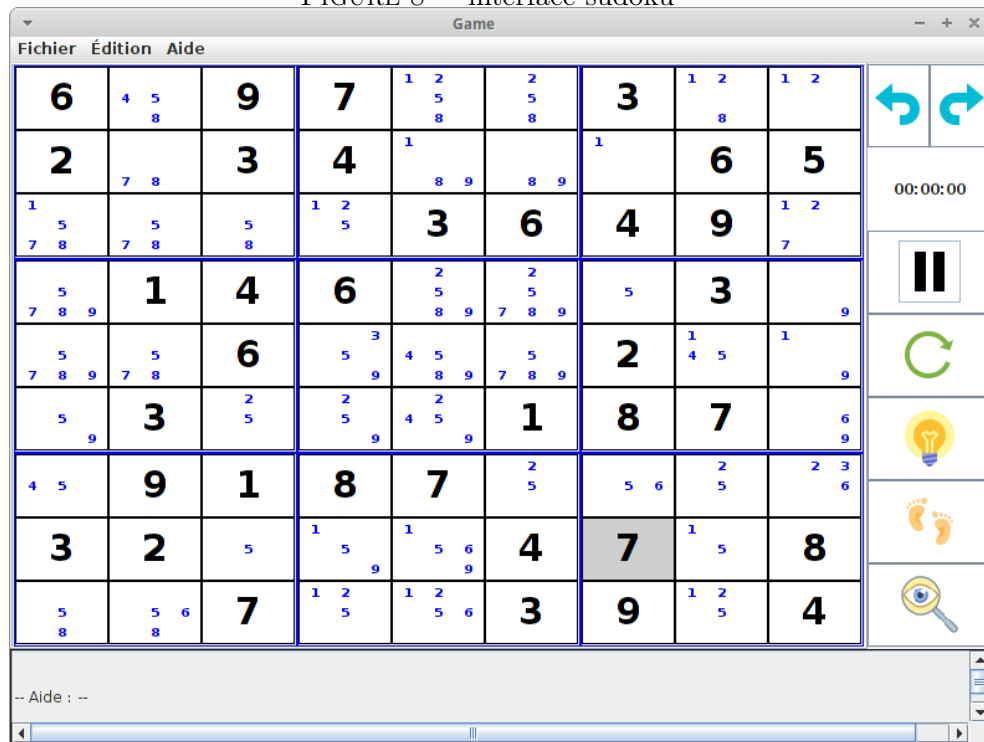
Le changement de grille est assez particulier puisque le chargement d'une sauvegarde implique seulement de changer un attribut (GridModel) dans le modèle tandis que le chargement d'une nouvelle grille implique de changer la référence du modèle (puisque l'on en crée un nouveau). Aussi, la grille chargée,

nouvelle ou sauvegardée, peut être de dimension différente de la précédente.

Pour régler le premier problème, Grid écoute tout changement de grille de son SudokuModel (propriété GRID) et utilise la méthode setModel() sur ce dernier (qui est le même objet) ainsi la méthode setModel() est appelée dans les deux cas.

Le deuxième problème est réglé dans la méthode setModel(). Si l'ancienne grille et la nouvelle sont de même taille, alors le modèle de chaque Cell est mis à jour. Sinon, tous les composant graphiques interne de Grid sont retirés et on recommence le processus de construction à partir de la création de la vue (createView())

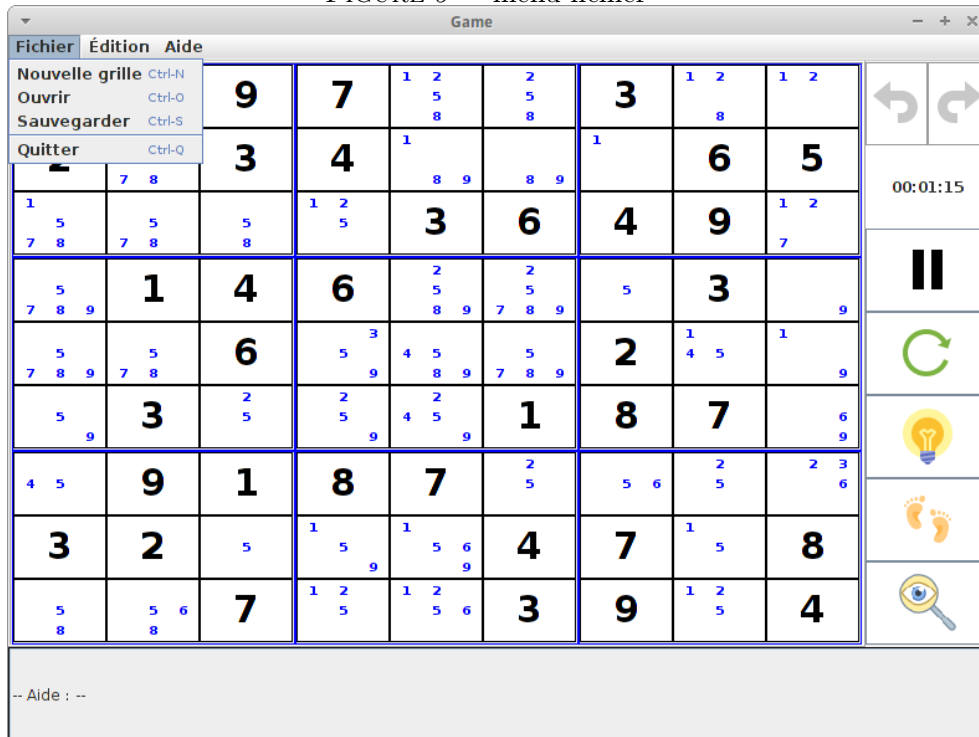
FIGURE 8 – interface sudoku



7 Jouabilité

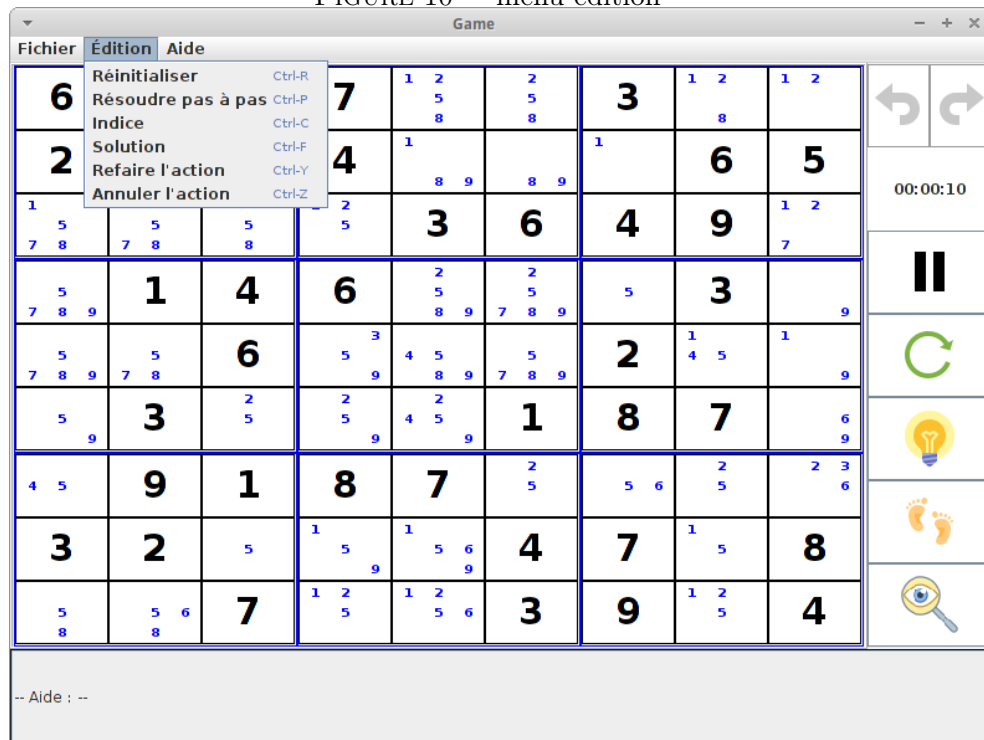
Dans le menu fichier, vous pourrez choisir entre choisir une nouvelle grille (par défaut, la grille numéro 2 est choisie), ouvrir un fichier de sauvegarde, sauvegarder la partie ou encore quitter la partie.

FIGURE 9 – menu fichier



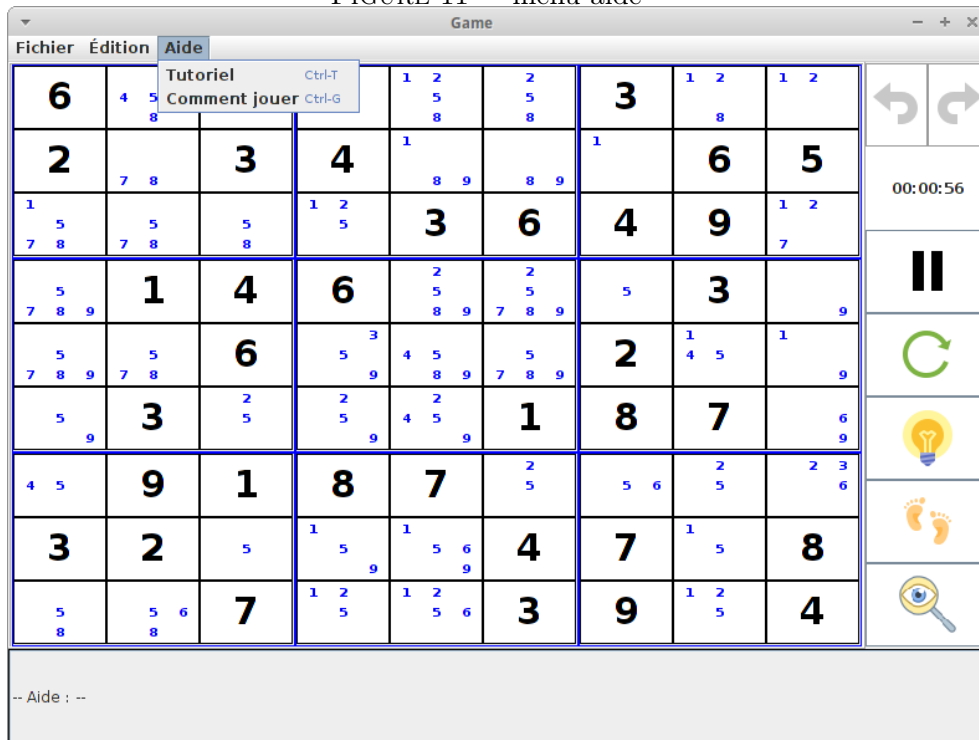
Dans le menu édition, vous aurez la possibilité de réinitialiser la grille, c'est-à-dire remettre votre grille choisie dans son état initial, résoudre pas à pas qui consiste à remplir les cases vides de la grille en expliquant comment procéder grâce à un petit texte apparaissant en bas de l'écran, l'indice, donne un message d'aide et surligne la ligne correspondant d'une couleur bleu sans donner la réponse, demander la solution complète de la grille, refaire l'action et défaire l'action précédente.

FIGURE 10 – menu édition



Dans le menu aide, vous aurez la possibilité de consulter les règles du jeu et un guide d'utilisation de notre application.

FIGURE 11 – menu aide

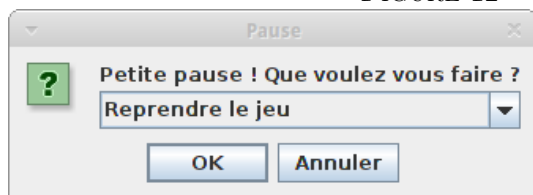


Toutes ces opérations sont également accessibles grâce aux raccourcis clavier suivant :

- CTRL+N : nouvelle grille
- CTRL+O : ouvrir un fichier de sauvegarde
- CTRL+S : sauvegarder sa partie
- CTRL+Q : quitter le jeu
- CTRL+R : réinitialiser
- CTRL+P : résoudre pas à pas
- CTRL+C : avoir un indice
- CTRL+F : solution complète
- CTRL+Y : refaire l'action
- CTRL+Z : annuler l'action
- CTRL+T : tutoriel
- CTRL+G : comment jouer

Lorsque vous appuyez sur le bouton pause, une fenêtre apparaît et vous propose de reprendre la partie ou de la quitter.

FIGURE 12 – pause



Si vous appuyez sur le bouton solution, la grille sera remplie entièrement.

FIGURE 13 – solution complète

The screenshot shows a game window titled "Game" with a menu bar containing "Fichier", "Édition", and "Aide". The main area displays a 9x9 grid of numbers, where the numbers 1 through 9 are highlighted in blue. The grid is as follows:

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

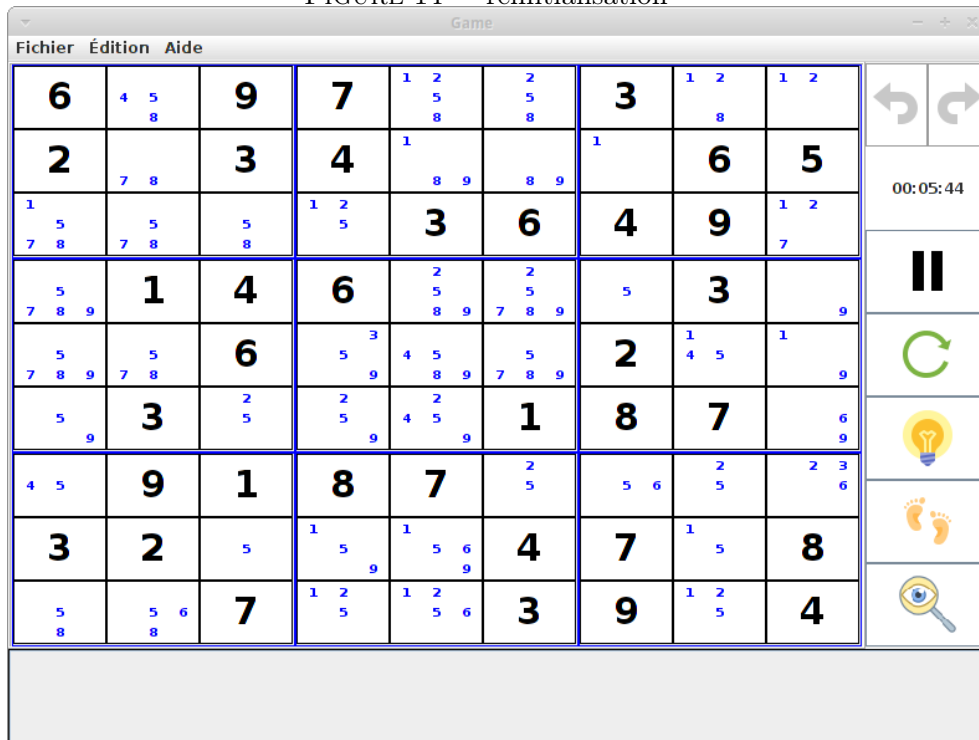
To the right of the grid is a sidebar with the following elements from top to bottom:

- Undo and redo arrows.
- A timer showing "00:04:49".
- A pause button (two vertical bars).
- A refresh button (circular arrow).
- A lightbulb icon (solution button).
- A footprint icon.
- A magnifying glass icon.

At the bottom of the window is a text area containing "-- Aide : --".

Vous avez la possibilité de réinitialiser la grille de sudoku et ainsi de recommencer la partie.

FIGURE 14 – réinitialisation



Si vous demandez un indice, vous pourrez avoir un texte d'aide et pourrez compléter la grille en vous aidant de la surbrillance des cases concernées.

FIGURE 15 – indice

Game

Fichier Édition Aide

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

↶ ↷

00:00:32

⏸

↻

💡

👣

🔍

Le candidat 4 n'est présent qu'une seule fois dans cette ligne.

En choisissant la méthode pas à pas, vous bénéficierez à la fois de conseils pour remplir la grille mais également de la réponse de la case concernée vous permettant de résoudre la grille de sudoku.

FIGURE 16 – pas à pas

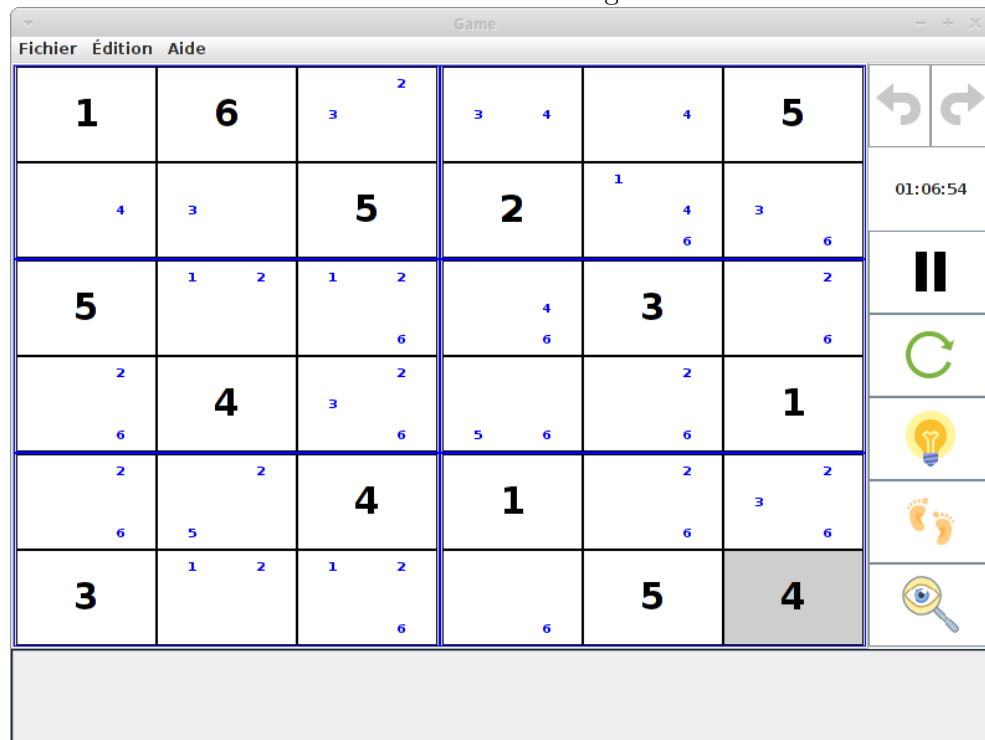
The screenshot shows a Sudoku game window titled "Game". The menu bar includes "Fichier", "Édition", and "Aide". The toolbar on the right contains icons for undo, redo, a timer showing "00:06:54", a pause button, a refresh button, a lightbulb icon for hints, footprints for search, and a magnifying glass for detailed search. The main area displays a 9x9 grid with numbers and candidates. The status bar at the bottom shows the message: "Le candidat 4 n'est présent qu'une seule fois dans cette ligne."

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

Le candidat 4 n'est présent qu'une seule fois dans cette ligne.

Il vous est possible de demander une nouvelle grille, en chargeant un fichier grilleXX.txt. L'application supporte différents formats de grilles.

FIGURE 17 – nouvelle grille



8 Difficultés rencontrées

Lors de la conception de notre projet, nous avons rencontrés quelques difficultés :

- gestion du temps, où chaque semaine, nous nous fixions des objectifs à atteindre,
- utilisation et prise en main du logiciel git et du site github.com,
- respect des demandes de la part du client,
- incompatibilité/réajustement lors de réunion de différents travaux des membres du groupe
- essais infructueux, sur deux semaines, d'une implantation de la partie vue de la grille avec des JTable. Certains fichiers sont encore dans le projet (`sudoku.view.test_jtable`) mais inutilisés.

9 Répartition du travail

Dans un premier temps, nous avons tous ensemble commencer à réfléchir à l'architecture du modèle. Puis au moment où nous avons eu une structure claire, Loïc, Paul et Laëtitia ont codé le modèle quant à Florian, il a commencé à concevoir et élaborer le design et les fonctionnements de l'application du jeu. Quant le modèle fut fini et la réflexion sur les structures des règles et du gestionnaire d'heuristiques achevée Laëtitia et Paul se sont mis à les coder, y compris l'historique. Quant à Loïc, il est parti avec Florian sur la partie graphique du projet mais plus centré sur les composants grille et cellule (surbrillance...) Pour finir, chacun a donné un coup de main dans toutes les parties du projet pour régler les derniers beugs et erreurs.

10 Conclusion

Ce projet nous a apporté une grande expérience car il s’agit de notre premier “gros” projet en équipe de plus de deux personnes.

Afin de réaliser un travail commun, nous avons opté pour un service de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git ainsi que le service web d’hébergement <https://github.com/>.

Ce dernier fut, pour nous, d’une grande aide, ce fut une toute nouvelle façon de procéder que nous a donné comme opportunité ce projet même si la prise en main fut assez compliquée.

Pour finir, ce projet fut, pour nous, très enrichissant car il nous a permis de réunir l’ensemble de nos connaissances au sein d’un même projet. Nous avons fait le choix de développer notre application dans le langage de programmation Java, qui est un langage très intéressant pour le développement d’applications graphiques.

Références

- [1] <https://www.mots-croises.ch/Sudoku.htm>
- [2] <http://www.le-sudoku.fr/>

Logiciels utilisés

- Eclipse (Java)
- ObjectAid (plugin Eclipse pour créer les diagrammes de classes)
- Git
- Kile (LaTeX)

11 Annexes

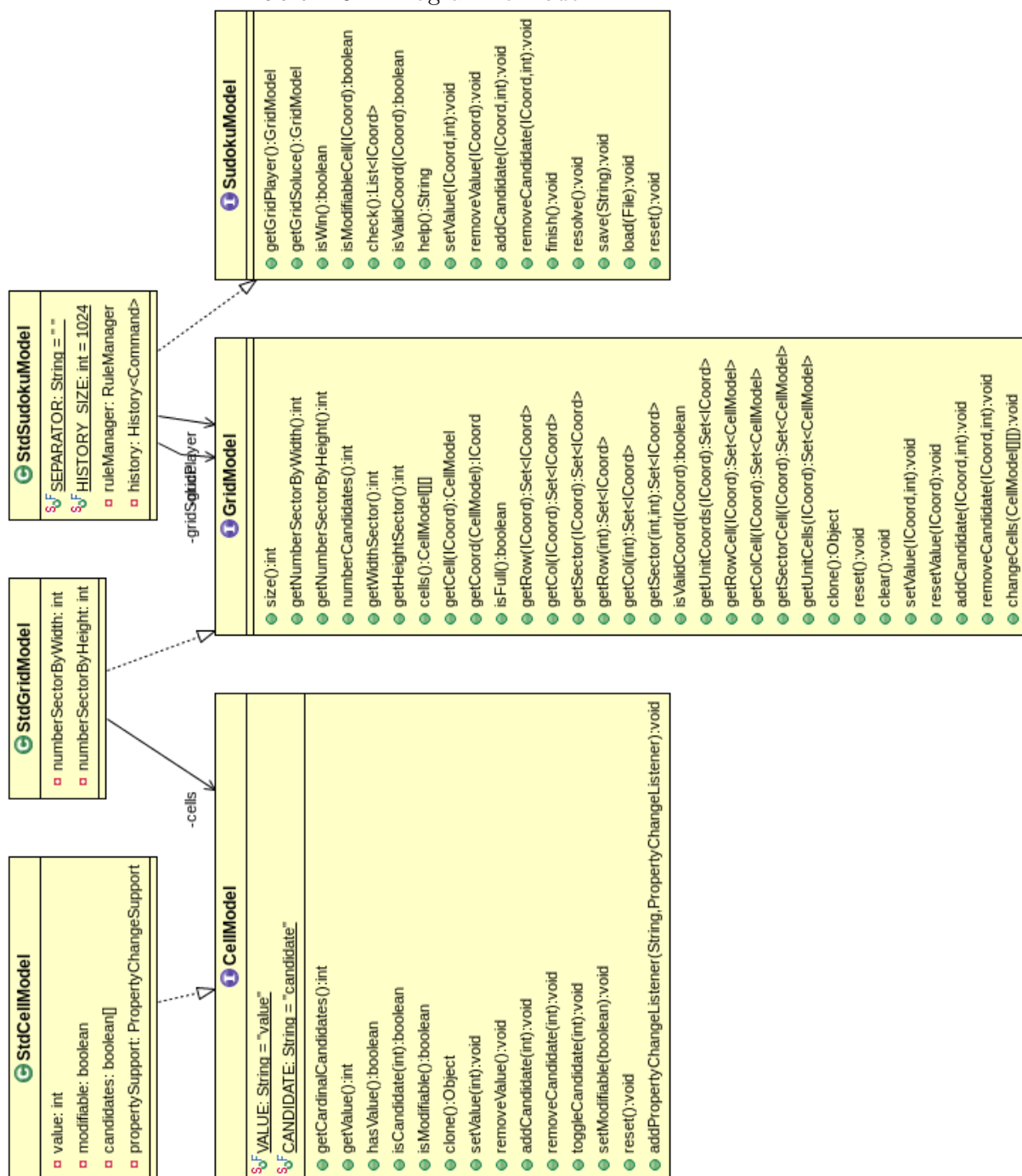


FIGURE 19 – Diagramme history

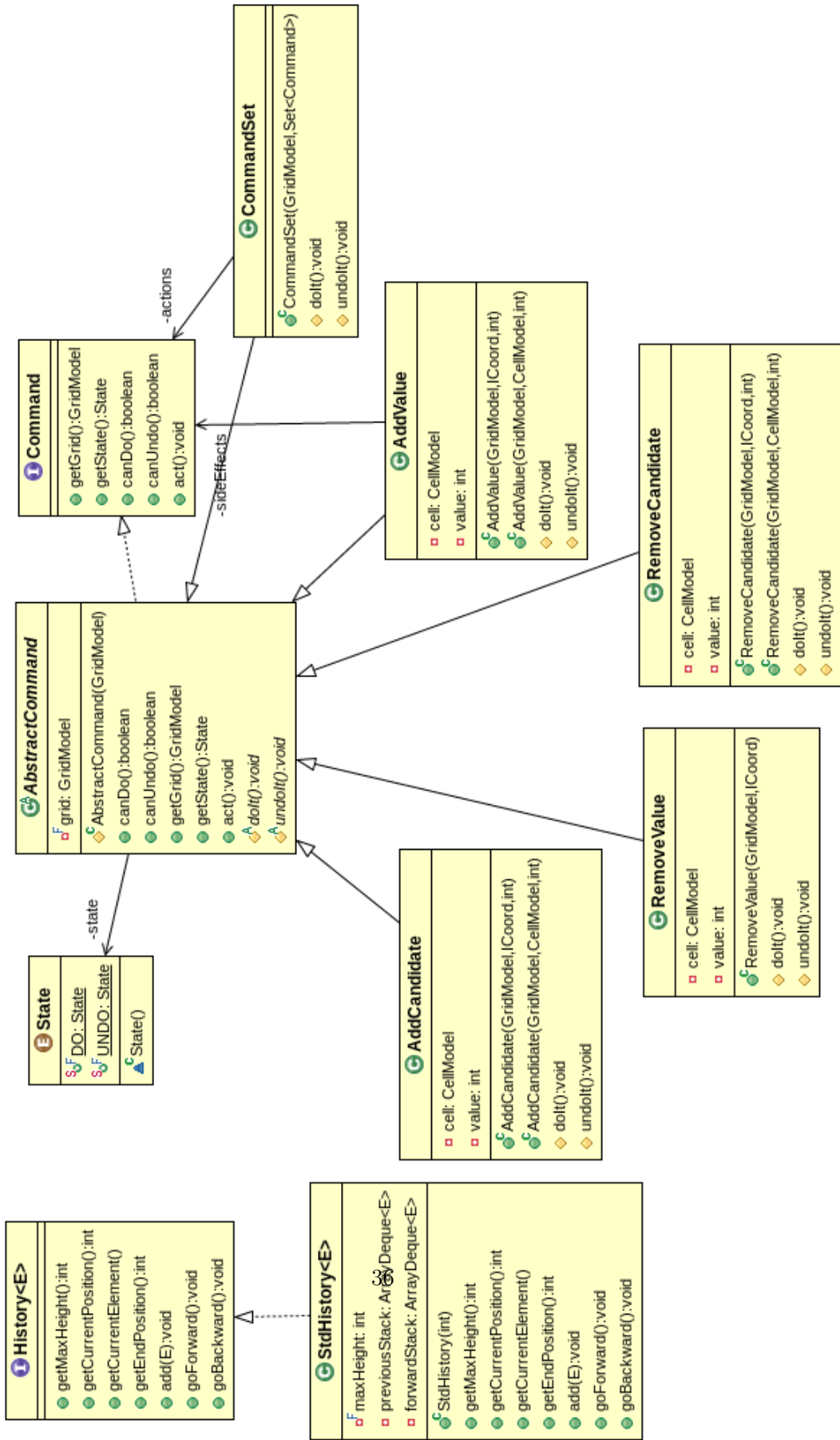


FIGURE 20 – Diagramme heuristic

