UNIVERSITE DE ROUEN

30 mars 2017

L3 Informatique

Projet d'application informatique

# Résolveur de sudoku
# Rapport

*Auteurs :*
FRÉTARD Loïc
HARDOUIN Paul
BOUCHER Laëtitia
BRUN Florian

*Responsable :*
M. Guesnet

# Table des matières

# 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 Leon- hardt 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 pour 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 nécéssitant 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 annuler.

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 choisit 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 7 : diagramme model, annexe 8 : diagramme history, annexe 9 : diagramme heuristic).

### 3.1 SudokuModel

Le sudoku possède une grille pour le joueur (GridModel) et un historique. La grille du 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 toutes interactions de sa part, seules les valeurs des cases fixes restent.

### 3.2 GridModel

Une grille possède une taille n (hauteur x largeur) fixe, inchangeable donnée en paramètre lors de l'initialisation et un tableau à double entrée (lignes, colonnes) de cellules (ModelCell). La grille connait 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 connait 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). Abs-

tractCommand 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 possibiltés 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 énumerative 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'execution 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).

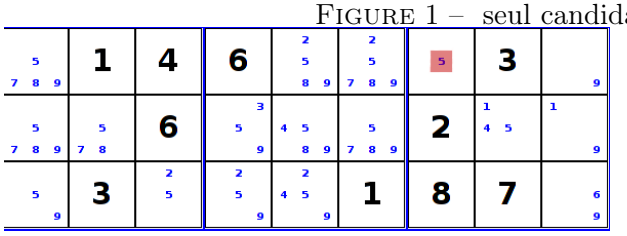
## 4 Heuristiques implantées

### 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 première ligne et la deuxième colonne se trouve le candidat 4. Comme étant le seul 4 dans la région, de ce fait, on peut le sélectionner comme valeur.

FIGURE 1 – seul candidat

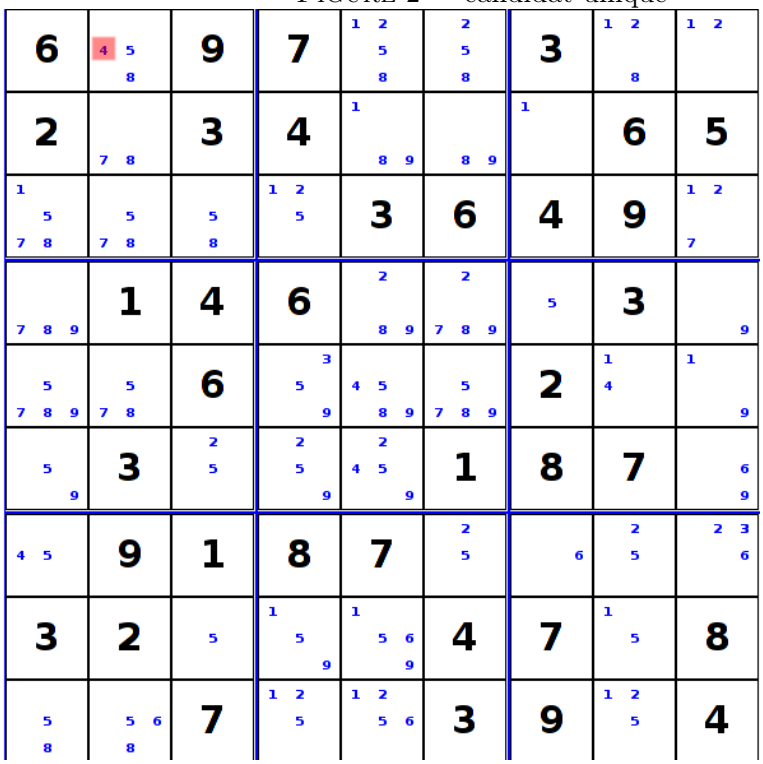


## 4.2 Un candidat unique : (one candidate)

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'apparait 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, septième colonne, il y a un seul candidat dans la case. Donc il n'y a qu'une solution : 5.

FIGURE 2 – candidat unique




## 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

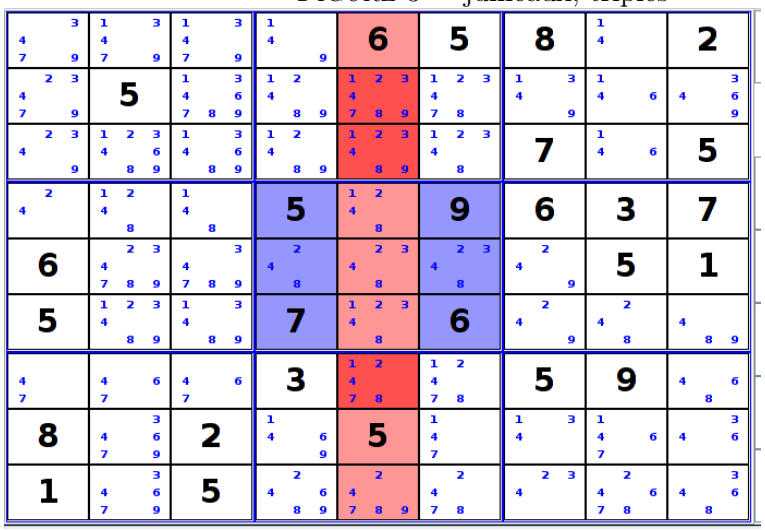
contient deux cases avec une même paire de candidats (et eux seuls) alors ces candidats ne peuvent se trouver dans une autre case de l'unité.

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

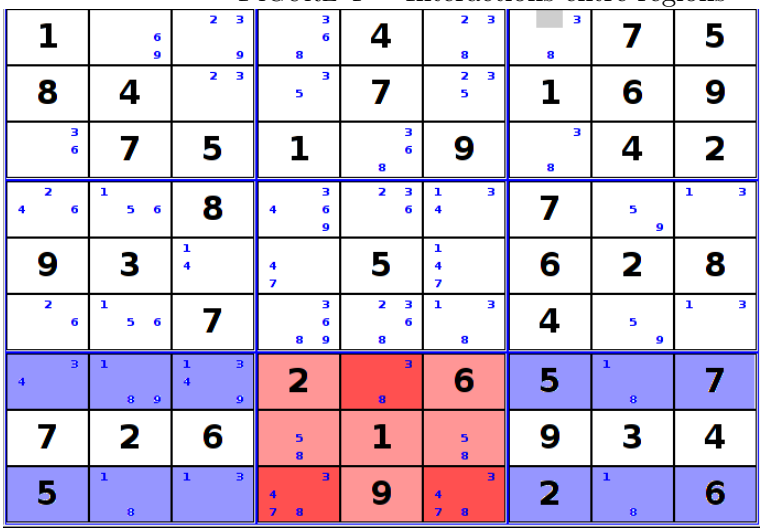


## 4.4 Interactions entre régions

Si à l'interieur 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 R7 et R9 que 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




## 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

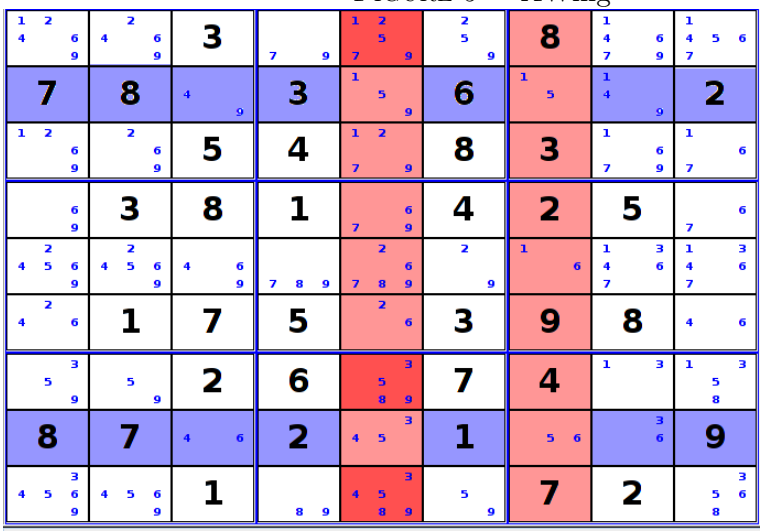
Le nom X-Wing (ou aile en X) provient de la figure tracée par cette méthode.
En effet le principe est basé sur le choix à faire dans l'emplacement d'une valeur.
En effet si une valeur est placée dans un coin, la même valeur ne pourra être placée que dans le coin opposé, ce qui trace les diagonales des 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 L5 et L7.
Il est donc possible de supprimer tous les autres candidats 5 de ces 2 colonnes.

FIGURE 5 – XWing



## 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 Interface

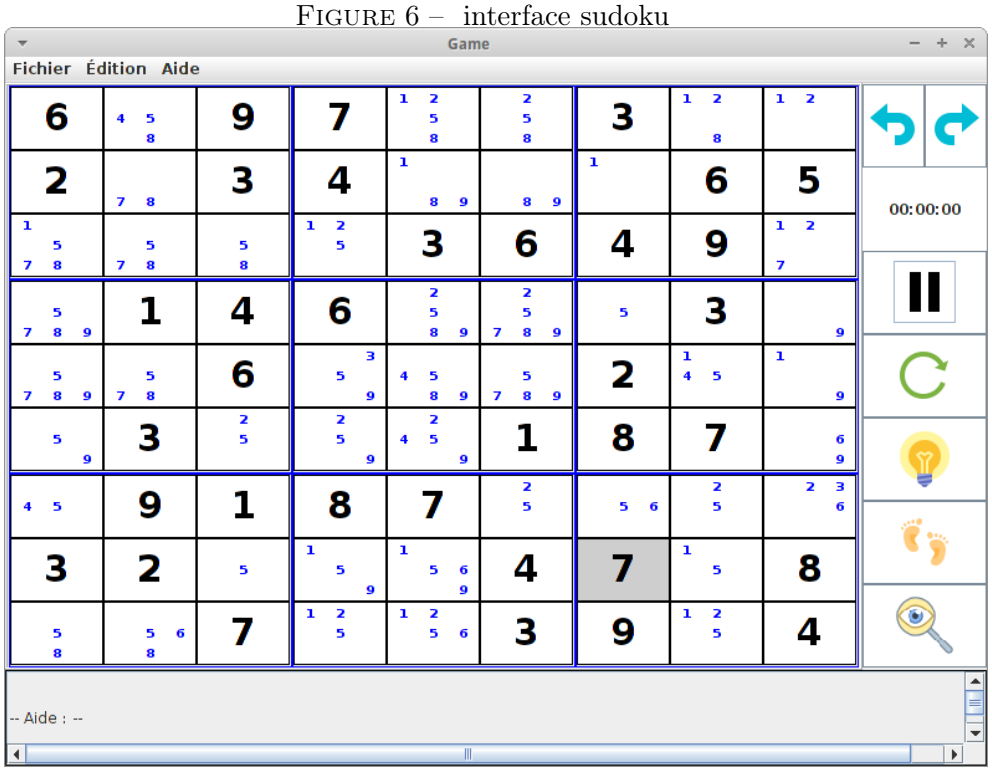
Notre application se présente de la manière suivante :
une barre de menu situé au nord de la fenêtre vous permettra de choisir une action à réaliser parmi toutes les fonctionnalités qui vous sont proposés.

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 utilisés tel défaire, refaire une action, demander un indice, réinitialiser la grille etc, puis, une zone d'aide, situé 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 souhaîtée.

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.

FIGURE 6 – interface sudoku




## 6 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

## 7 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 compliqué.

Pour finir, ce projet fut pour nous très enrichissant car il nous a permi 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 les développements d'applications graphiques.

# 8 Annexes

FIGURE 7 – Diagramme model

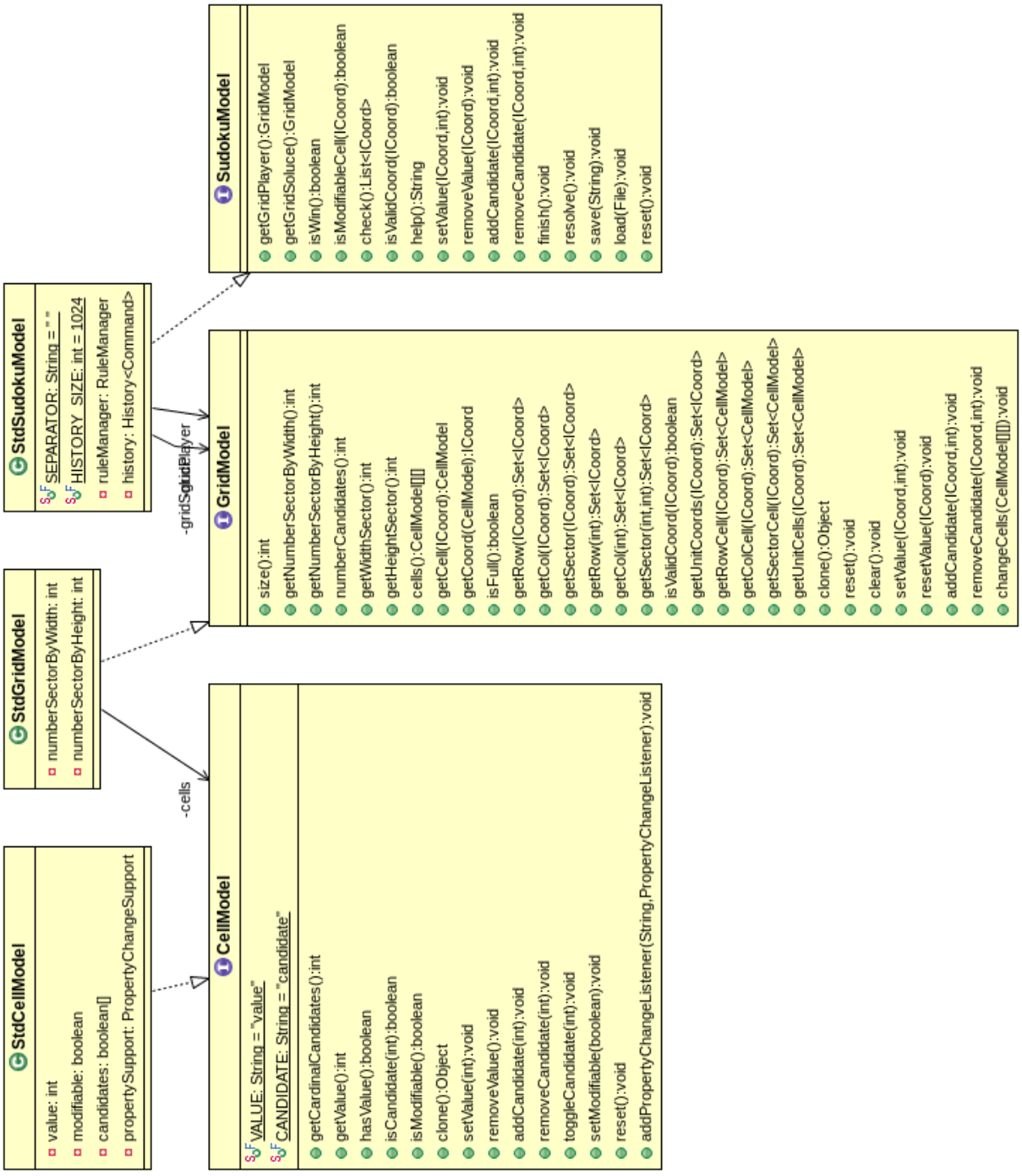


FIGURE 8 – Diagramme history

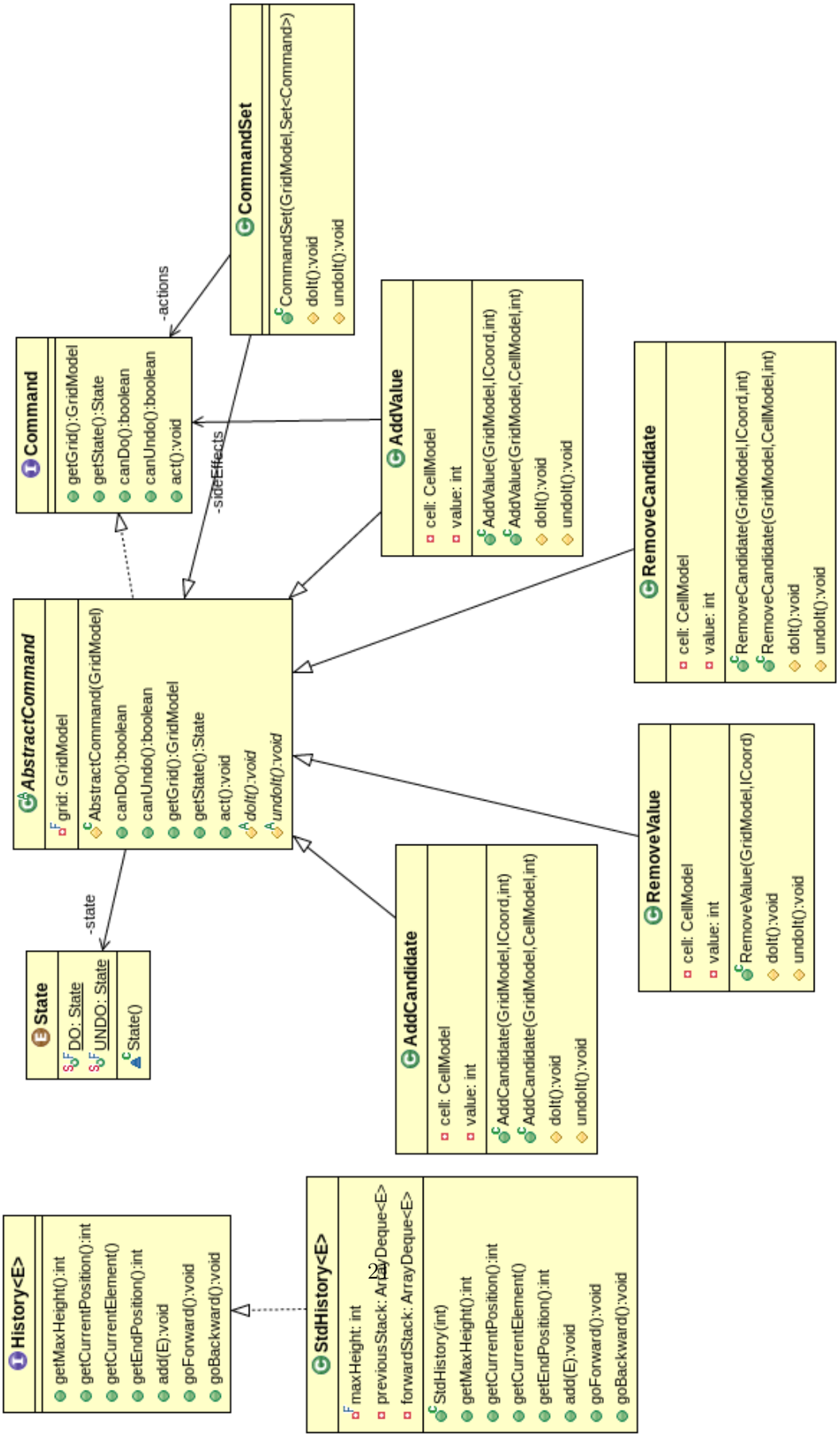


FIGURE 9 – Diagramme heuristic