Université Denis Diderot - Paris VII

Projet De Programmation
Résolution de grilles sudoku

Réalisé par :

- Mechouar Fella
- Su Li-Fang
- Hadj Mohand Amayas
- Berat Stephane

Encadré par :

- François Laroussinie
- Benjamin Lipp

Table des Matières

I. Introduction : 2
II. Installation et utilisation : 2
A. Guide d'installation : 2
B. Guide d'utilisation : 3
III. Etude Conceptuelle : 6
A. Architecture à base de Booléens : 6
B. Architecture principale du projet : 6
IV. Détails algorithmiques des Fonctionnalités du programme : 7
A. Générateur : 7
B. Backtracking : 8
C. Déduction : 8
D. Stratégies : 10
1. Stratégie Chiffres Exclusifs : 11
2. Stratégie Chiffre exclusif Région : 12
3. Stratégie Paire Exclusive : 13
4. Hidden Double : 14
5. Locked Candidates 14
6. Stratégie Naked Uplets 15
7. Stratégie X-Wing : 16
8. Stratégie Swordfish : 17
9. Stratégie Coloring : 18
10. Stratégie XYWing : 18
11. Stratégie XYZWing : 19
E. Limites : 20
V. Démonstration : 21
VI. Conclusion : 22
VII. Bibliographie : 23

I. Introduction :

Le sudoku , est un jeu en forme de grille défini en 1979 par l'Américain Howard Garns, mais inspiré du carré latin, ainsi que du problème des 36 officiers du mathématicien suisse Leonhard Euler.

Le but du jeu est de remplir la grille avec une série de chiffres (ou de lettres ou de symboles) tous différents, qui ne se trouvent jamais plus d'une fois sur une même ligne, dans une même colonne ou dans une même région (également appelée « bloc », « groupe », « secteur » ou « sous-grille »). La plupart du temps, les symboles sont des chiffres allant de 1 à 9, les régions étant alors des carrés de 3 × 3. Quelques symboles sont déjà disposés dans la grille, ce qui autorise une résolution progressive du problème complet. « Wikipédia »

A travers notre projet du module du semestre 4 intitulé " UE PROJET ", on s'intéresse à la façon dont les grilles sudoku peuvent être résolus, ce qui nous mènent à mettre en œuvre un ensemble complet de stratégies, qui permettent à une machine d'acquérir un certain niveau d'intelligence lui permettant de résoudre des grilles sudoku de différents niveaux de difficultés.

Pour découvrir beaucoup plus de détails techniques sur notre processus de travail, on vous invite à lire la suite de ce document.

II. Installation et utilisation :

A. Guide d'installation :

Pour installer notre programme veuillez suivre les instructions suivantes :

- Téléchargez le fichier (UE_PROJET_SUDOKU.jar) qui se trouve dans la branche master.
- Ensuite, je vous propose deux méthodes pour lancer le programme :

Méthode 1 :

- Téléchargez le Makefile qui se trouve sur la branche master
- Placez le fichier (.jar) et le Makefile dans le même dossier.
- Sur la ligne de commandes exécutez la commande :

make run

```
fella@fella-K401LB:~$ make run
```

Méthode 2 :

- Sur la ligne de commandes, placez-vous dans le dossier qui contient le fichier (.jar)

Exécutez la commande :

java -jar UE_PROJET_SUDOKU.jar

```
fella@fella-K401LB:~$ java -jar UE_PROJET_SUDOKU.jar
```

- Le programme va se lancer. Pour vous en servir, on vous invite à lire le guide d'utilisation.

B. Guide d'utilisation :

- Sur la ligne de commandes, exécutez la commande :

java -jar UE_PROJET_SUDOKU.jar

- Un menu s'affiche et vous demande de choisir entre :

```
-----------------------------------------------------------------------------------------------
Bienvenue dans le sudoku solver !
-----------------------------------------------------------------------------------------------
1 - Charger une nouvelle grille d'un fichier
2 - Generer une nouvelle grille
3 - Sortir du programme
```

- Charger une grille à partir d'un fichier : Vous devez introduire le chemin (relatif ou absolu) de votre fichier texte.

```
Veuillez introduire le chemin du fichier contenant la grille :
./9x9Diabolique.txt
Grille chargee depuis : ./9x9Diabolique.txt
22/81 Chiffres initiaux !

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|_  _  9  |1  _  _  |_  _  5  |
|_  _  _  |_  _  5  |_  _  _  |
|_  _  6  |_  8  _  |_  4  3  |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|_  9  _  |_  _  _  |_  _  _  |
|_  8  _  |_  2  _  |_  _  _  |
|_  2  4  |_  _  3  |9  _  _  |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|_  _  _  |_  _  6  |_  _  _  |
|_  _  _  |_  _  _  |8  1  _  |
|5  3  _  |8  _  2  |_  _  _  |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

- **Note :** Le fichier texte doit avoir la structure suivante :(capture du format du fichier)

```
3 3

0, 0, 9, 1, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 6, 0, 8, 0, 0, 4,

3, 0, 9, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 2, 0, 0, 0, 0, 0, 2, 4, 0, 0, 3, 9,

0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 1, 0, 5, 3, 0, 8, 0, 2,

0, 0, 0,
```

- Générer une grille : dans cette étape vous aurez à introduire les tailles de la sous-grille Le nombre de lignes et le nombre de colonnes, ainsi que le pourcentage de vidage qui est un entier entre 1 et 100 et votre grille sera prête.

```
-----------------------------------------------------------------------------------------------
Bienvenue dans le sudoku solver !
-----------------------------------------------------------------------------------------------
1 - Charger une nouvelle grille d'un fichier
2 - Generer une nouvelle grille
3 - Sortir du programme
2
Generateur Sudoku
Veuillez saisir la largeur de la petite grille
4
Veuillez saisir la longueur de la petite grille
4
Veuillez saisir le pourcentage de cases a supprimer
30
```

- Après le choix de grille, un autre menu s'affiche en vous donnant le choix entre :

```
 Voulez vous :
1 - Resoudre la grille par Deduction
2 - Resoudre la grille par Backtracking avec application prealable des strategies
3 - Resoudre la grille directement par Backtracking
4 - Sauvegarder la grille actuelle
5 - Charger une autre grille
6 - Sortir du programme
```

- ❖ Résoudre la grille par déduction : cette option vous permettra de résoudre votre grille en utilisant l'algorithme de déduction, vous trouverez les détails techniques dans les chapitres suivants.
- ❖ Résoudre la grille par application des stratégies et le backtracking : cette option vous permettra de résoudre votre grille en combinant l'ensemble des stratégies avec l'algorithme de backtracking, vous trouverez les détails techniques dans les chapitres suivants.
- ❖ Résoudre la grille par le backtracking : cette option vous permettra de résoudre votre grille en utilisant l'algorithme de backtracking, vous trouverez les détails techniques dans les chapitres suivants.
- ❖ Sauvegarder la grille actuelle : cette option vous permet de sauvegarder votre grille dans un fichier texte pour la réutiliser ultérieurement.

```
(78/81)  Deduction : UNIQUE POSSIBLITY => 6 * 7 : 3 write on the Sudoku
(79/81)  Deduction : UNIQUE NUMBER IN BOX => 1 * 3 : 6 write on the Sudoku
(80/81)  Deduction : UNIQUE POSSIBLITY => 1 * 4 : 4 write on the Sudoku
(81/81)  Deduction : UNIQUE POSSIBLITY => 3 * 3 : 4 write on the Sudoku

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|8  4  9  |1  3  7  |6  2  5  |
|2  7  3  |6  4  5  |1  9  8  |
|1  5  6  |2  8  9  |7  4  3  |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|3  9  1  |4  7  8  |2  5  6  |
|6  8  5  |9  2  1  |3  7  4  |
|7  2  4  |5  6  3  |9  8  1  |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|4  1  8  |7  9  6  |5  3  2  |
|9  6  2  |3  5  4  |8  1  7  |
|5  3  7  |8  1  2  |4  6  9  |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Resolu par Deduction en 0.179 secondes
--------------------------------------------------------------------------------------------------
```

- Après la résolution de votre grille, le menu principal se réaffiche et vous donne le choix entre continuer à s'amuser dans le programme ou bien Quitter le programme. On espère que vous vous êtes bien amusés !

```
--------------------------------------------------------------------------------------------------
Bienvenue dans le sudoku solver !
--------------------------------------------------------------------------------------------------
1 - Charger une nouvelle grille d'un fichier
2 - Generer une nouvelle grille
3 - Sortir du programme
3
--------------------------------------------------------------------------------------------------
                                 À très bientôt  !!
--------------------------------------------------------------------------------------------------
Sudoku - UE PROJET S4            Année Universitaire :  2019 - 2020     Université de Paris ©
```

III. Etude Conceptuelle :

A. Architecture à base de Booléens :

Cette partie est en quelque sorte un bonus du projet que nous n'avons pas gardé. Elle avait pour but de comparer les deux méthodes (celle de booléen et « ArrayList »).
Bien sûr nous ne pouvions pas faire le même travail qu'avec les «ArrayList» pour une raison évidente de manque de temps cependant nous avons comparé pour les méthodes classiques et basiques qui permettent de résoudre des grilles de niveaux faciles et nous avons trouvé que de manière général, les «ArrayList» étaient une méthode plus optimal car elle a besoin de faire des parcours de boucle moins important (bien que nous ne les ayons pas comparé avec un algorithme dédié mais en terme de rapidité de résolution) et cela semblait coïncider avec nos prévisions .

Je conclurais en disant que notre méthode actuelle semble plus performante mais que nous nous somme restreint à la partie simple du projet cela n'est donc pas une vérité générale et peut être que la méthode booléenne aurait été meilleur de manière général.

B. Architecture principale du projet :

Pour la structuration générale du projet, on a choisi une approche orientée objet qui s'adapte parfaitement à nos besoins pour exprimer notre solution.

Si on part dans les détails intérieurs de l'architecture, vous remarquerez que la classe "Grille" est la classe principale de la structure, dont toutes les autres classes dépendent car c'est la classe qui contient les informations principales de la grille sudoku sur lesquelles repose le fonctionnement du projet.

D'autre part, on note que la structure de données principale utilisée dans le projet était les « ArrayList ». Ce choix revient aux avantages que représente cette structure, vu que c'est une structure dynamique ainsi que les opérations élémentaires implémentées avec cette structure tels que, la suppression, la recherche, l'ajout des éléments.

Pour vous donner un aperçu général sur notre architecture on vous représente ce diagramme qui contient toutes les classes du projet ainsi que les relations entre elles.

Diagramme de Classes Projet Sudoku

UE PROJET | 2019 - 2020

Dépendance «use»

Aggregation

Composition

Couple
- x : int
- y : int

Save

Launcher
- sc : Scanner

Parseur

Case
- val : int
- Possibility : ArrayList<Integer>

Generateur

StrategyHiddenDouble

Backtracking
- compteur : int

StrategyChiffreExclusifRegion

Grille
- t : case [][]
- nbGarder; : int
- VALEUR_VIDE : int
- TAILLE_COTE_PLATEAU : int
- TAILLE_SOUSZONE_LIGNE : int
- TAILLE_SOUSZONE_COL : int
- nbCases : int

Deduction

StrategyLockedCandidates

StrategyChiffresExclusifs

StrategyPairesExclusives

StrategySwordfish

StrategyPairesExclusives

StrategyXYZWing

StrategyColoring
- sauvegarde : Grille
- tsav : Case[][]
- level : int

StrategyUplets
- uplettest : ArrayList<Integer>
- complementaire : ArrayList<Integer>

StrategyXWing

StrategyXYWing

XYZindex
- x : int
- y : int
- z : int
- couple : Couple

IV. Détails algorithmiques des Fonctionnalités du programme :

A. Générateur :

Comme son nom l'indique le générateur consiste à générer des grilles sudoku de la taille qu'on veut notamment 2*3, 3*3, 4*4, 5*5...etc. , le but c'est de pouvoir générer des grilles aléatoires résoluble avec les deux techniques implémentaient dans le projet, notamment en utilisant les différentes réglés de déductions ou la méthode du Backtracking , la génération d'une grille n*k sudoku passe par 6 étapes :

- Demander à l'utilisateur d'introduire la largeur et la longueur des petites box ainsi qu'un pourcentage qui représente le nombre de cases à supprimer dans la grille
- La création d'un tableau de tableaux vide de n*k lignes et n*k colonnes
- La création d'une «ArrayList» qui comporte les chiffres de 1 jusqu'à n*k
- Remplir la première ligne de la grille vide aléatoirement à partir de cette «ArrayList»
- Remplir le reste de la grille en déplaçant la première ligne vers la gauche ou la droite tout en respectant les réglés principaux du sudoku pour ne pas avoir le même chiffre dans la même colonne ou la même ligne ou la même box.
- Vider aléatoirement la grille remplit tout en s'assurant qu'après chaque suppression la solution existe et qu'elle est unique tel que le nombre de suppression correspond au pourcentage introduit par l'utilisateur au début de la génération.

Notant que la notion du pourcentage a été introduite récemment , car avant cela on utilisait la notion du Level qui correspondait au niveau de difficulté de la résolution de la grille à générer , mais après avoir fait plusieurs tests, on s'est rendu compte que Level n'était pas très optimal pour notre programme car chaque level correspondait à un pourcentage bien précis de cases à supprimer et que ce pourcentage comme il pourra être très pratique pour les petites grilles mais il pourra entraîner des bugs et des arrêts d'exécution de notre programme pour les grilles grandes de tailles , alors on a décidé de laisser une certaine liberté pour l'utilisateur afin qu'il puisse introduire le pourcentage qui souhaite tout en respectant l'indication affichée qui vont l'encadrer et aider à choisir un pourcentage pratique et correcte.

Notant aussi que la notion du pourcentage est 'safe' car au cas où l'utilisateur n'a pas respecter l'indication on a introduit des cas d'arrêts du programme notamment si le pourcentage est beaucoup plus important en vue de la taille de la grille, le programme ne va pas essayer de vider la totalité des cases qui représente ce pourcentage.

Après avoir appliqué toutes les méthodes qui correspondent aux étapes décrites ci-dessus , on obtient une grille sudoku de n*k résoluble avec nos deux méthodes de résolution.

B. Backtracking :

Le principe du backtracking est résoudre un sudoku par force brute, en testant toutes les combinaisons possibles. Cela revient à utiliser la puissance CPU ainsi que la mémoire de l'ordinateur afin de parcourir un arbre avec diverses obligations répondant aux règles du Sudoku sur chacun des nœuds, jusqu'à trouver une solution.

Lorsqu'aucune solution répondant aux règles n'est possible avec les chiffres nouvellement placés par backtracking, il faut revenir en arrière et essayer d'autres chiffres en amont de l'arbre.

Ainsi, le nombre de nœuds possibles sur cet arbre correspond aux nombres de possibilités de chaque case multipliée entre elle, ce qui peut devenir excessivement long dans le cas de grandes grilles. Un compteur de format BigInteger à été mis en place pour cela afin de visualiser le nombre de tests maximums nécessaires au backtracking.

Afin d'optimiser l'algorithme dans notre projet, nous avons décidé de commencer par les cases ayant un plus petit nombre de possibilités, ce qui permet de "couper » de plus grandes branches de l'arbre en cas d'impossibilité, et donc de réduire le nombre de tests.

Aussi, nous permettons à l'utilisateur d'appeler une série de stratégies, ou des déductions multiples avant d'appeler le backtracking, afin de réduire le nombre de possibilités initiales.

Cet algorithme permet aussi de calculer le nombre de solutions d'une grille, ce qui est utilisé dans le générateur de grilles : si une grille générée a plus d'une unique solution elle sera alors considérée comme invalide.

C. Déduction :

Les déductions consistent à écrire un chiffre sur la grille de Sudoku, dont on est sûr à 100% de sa présence.

Cela correspond à 4 cas précis correspondant aux règles du sudoku :

- Un chiffre donné n'a qu'une seule position possible sur une ligne
- Un chiffre donné n'a qu'une seule position possible sur une colonne
- Un chiffre donné n'a qu'une seule position possible sur une box

- Une case n'a qu'une unique possibilité, à la suite de l'application des stratégies que nous détaillerons plus tard dans le rapport, ces stratégies ayant éliminés une ou plusieurs possibilités sur cette case

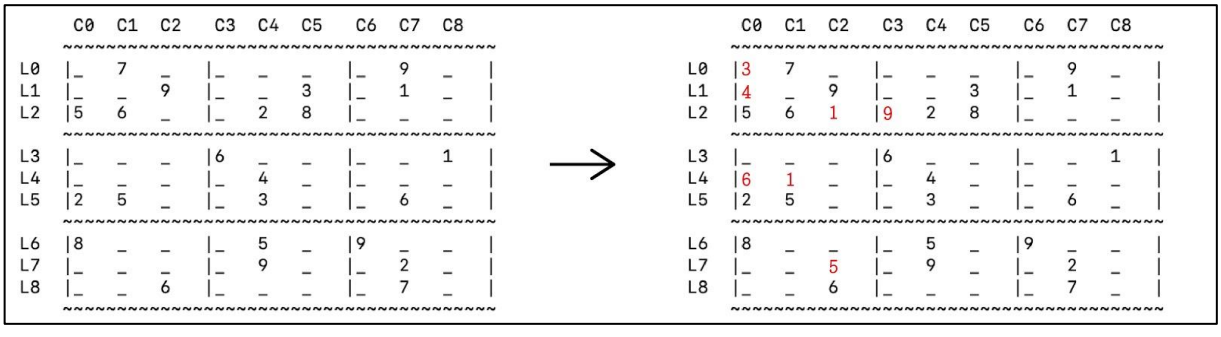
Dans ces 4 cas, on écrit le chiffre avec une fonction spécifique, qui met à jour les possibilités des cases de la ligne, colonne, box, en supprimant de leur possibilités la valeur du chiffre que l'on vient d'écrire (et l'ensemble des possibilités pour la case ou l'on vient d'écrire la valeur).

Les déductions sont ce qu'il y a de moins coûteux niveau algorithmique car il permet ainsi d'enlever de nombreuses possibilités si les conditions sont réunies. Nous allons donc les prioriser par rapport à l'applications des stratégies, qui sont beaucoup plus coûteuses.

Algorithme :

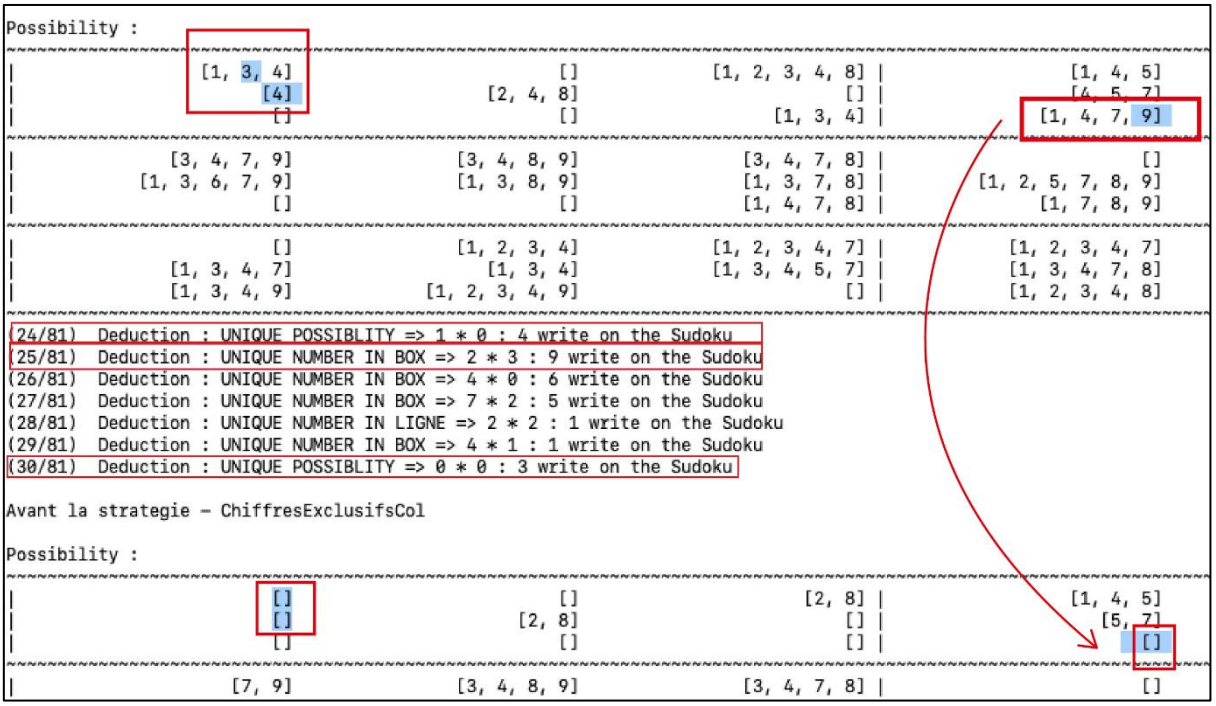
```
while(true) {
        On essaye de déduire une nouvelle valeur répondant à un des 4 cas.
            ❖ Si cela n'est pas possible, on applique les stratégies de la moins
              coûteuse à la plus coûteuse jusqu'à en trouver une qui fonctionne et
              supprime au moins une nouvelle possibilité.
            ❖ Si toutes les stratégies ont été testées et sont en échec, on propose le
              backtracking.
            ❖ Si tous les chiffres ont été trouvés, on renvoie la grille résolue.
}
```

L'exemple suivant concernant nous montre une série de 7 déductions sur la grille de test ex1. Nous partons de la grille suivante :



Ces ajouts mettent à jour l'ensemble des possibilités des autres cases.

Exemple sur la grille ex1 :



Ainsi, l'écriture du 4, ligne 1*0 permet d'effacer le 4 des possibilités des cases 1*1 et 1*3.

L'écriture du 9 en case 2*3 permet d'effacer l'ensemble des possibilités de cette case.

L'écriture du 3 en 0*0, du 4 en 0*1 et du 1 en 2*2 permet de réduire les possibilités de la case 0*2 de [1, 2, 3, 4, 8] à [2, 8].

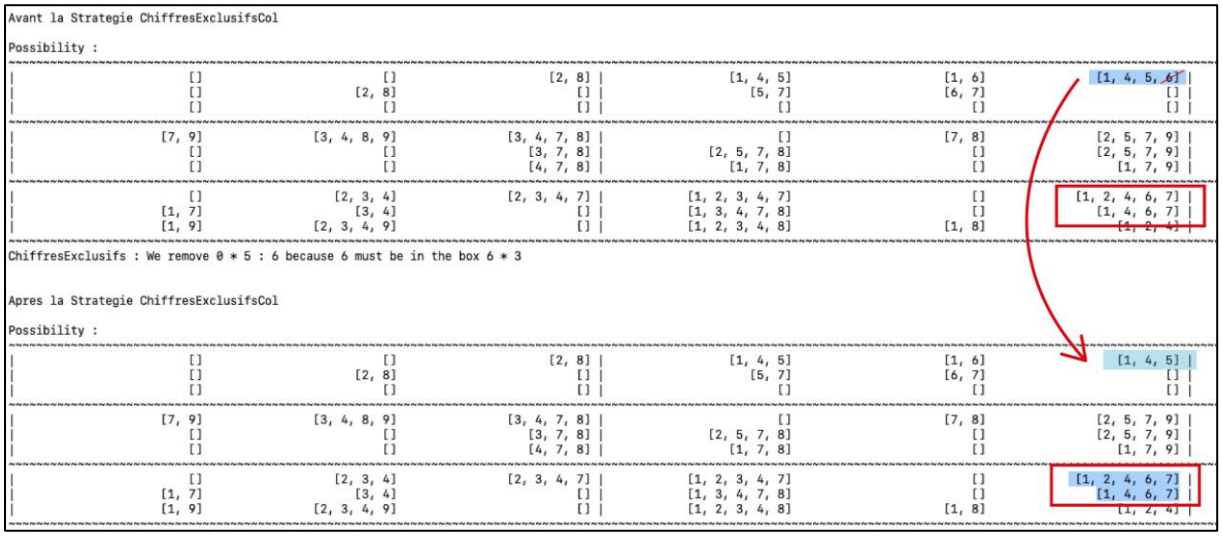
D. Stratégies :

` Les stratégies sont des moyens permettant d'éliminer des possibilités sur certaines cases répondant à des conditions précises. Dans le cadre de ce projet, nous avons implémenté plusieurs stratégies, permettant de résoudre des grilles de taille et de difficulté variable.

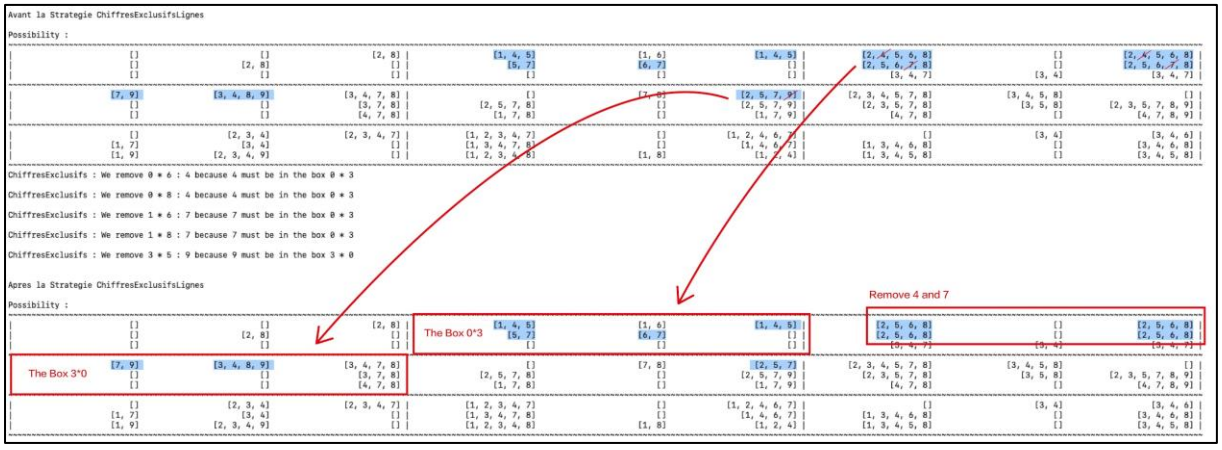
1. Stratégie Chiffres Exclusifs :

Si, à l'intérieur d'une box, deux ou trois chiffres identiques figurent dans une Ligne et qu'ils ne figurent pas ailleurs dans la box, on peut alors supprimer sans autre ce chiffre se trouvant dans les autres box sur la même Ligne. Cette stratégie s'applique également pour une Colonne.

Dans cet exemple sur la grille ex1, sur la box 6x3 (milieu bas), le 6 est obligatoirement sur la colonne 5. On peut donc supprimer le 6 des autres possibilités de la colonne.



Dans le même exemple, on peut voir un raisonnement similaire peut avoir lieu sur les lignes. Le 4 et 7 sont obligatoirement sur la box 0x3. On peut donc supprimer le 4 et 7 des autres possibilités de la ligne. Ainsi que le 9 qui est obligatoirement sur la box 3x0, donc on supprime le 9 des autres possibilités de la ligne.



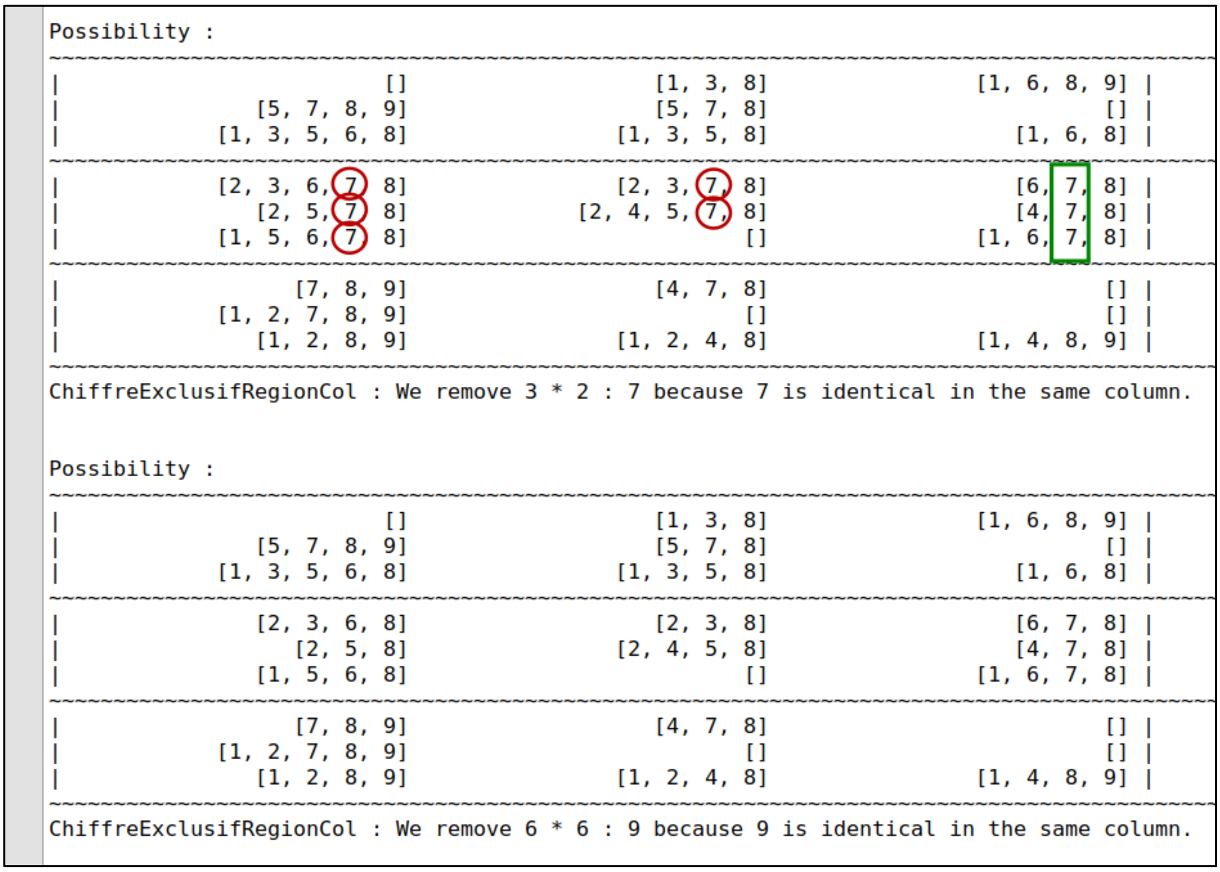
2. Stratégie Chiffre exclusif Région :

Cette stratégie consiste à voir, si :

- Dans une ligne de la région la possibilité d'avoir le chiffre se répète deux ou trois fois et nulle part sur la même ligne pour les régions à gauche et à droite.
  OU
- Dans une Colonne de la région la possibilité d'avoir le chiffre se répète deux ou trois fois et nulle part sur la même ligne pour les régions en haut et en bas.

Alors,
On peut éliminer toutes les possibilités d'un nombre donné dans la région sauf dans la ligne ou la colonne concernée

Dans l'exemple suivant vous pouvez voir, que c'est un cas de Stratégie chiffre Exclusif Région colonne, car on remarque sur la troisième colonne de la deuxième région en partant du haut, le nombre 7 apparait 3 fois (triangle vert) et il n'apparait pas sur la même colonne dans première région et la troisième en partant du haut. Ce qui forme le schéma de la Stratégie chiffre Exclusif Région colonne, donc on peut supprimer toutes les possibilités du chiffre 7 dans la deuxième région (cercles en rouge) sauf celles de la troisième colonne (triangle vert)



```
Possibility :
--------------------------------------------------------------------------------
|              []          [1, 3, 8]     [1, 6, 8, 9] |
|    [5, 7, 8, 9]          [5, 7, 8]               [] |
| [1, 3, 5, 6, 8]       [1, 3, 5, 8]        [1, 6, 8] |
--------------------------------------------------------------------------------
| [2, 3, 6, 7, 8]       [2, 3, 7, 8]        [6, 7, 8] |
|    [2, 5, 7, 8]    [2, 4, 5, 7, 8]        [4, 7, 8] |
|    [1, 5, 6, 7, 8]              []     [1, 6, 7, 8] |
--------------------------------------------------------------------------------
|       [7, 8, 9]          [4, 7, 8]               [] |
| [1, 2, 7, 8, 9]                 []               [] |
|    [1, 2, 8, 9]       [1, 2, 4, 8]     [1, 4, 8, 9] |
--------------------------------------------------------------------------------
ChiffreExclusifRegionCol : We remove 3 * 2 : 7 because 7 is identical in the same column.

Possibility :
--------------------------------------------------------------------------------
|              []          [1, 3, 8]     [1, 6, 8, 9] |
|    [5, 7, 8, 9]          [5, 7, 8]               [] |
| [1, 3, 5, 6, 8]       [1, 3, 5, 8]        [1, 6, 8] |
--------------------------------------------------------------------------------
|    [2, 3, 6, 8]          [2, 3, 8]        [6, 7, 8] |
|       [2, 5, 8]       [2, 4, 5, 8]        [4, 7, 8] |
|    [1, 5, 6, 8]                 []     [1, 6, 7, 8] |
--------------------------------------------------------------------------------
|       [7, 8, 9]          [4, 7, 8]               [] |
| [1, 2, 7, 8, 9]                 []               [] |
|    [1, 2, 8, 9]       [1, 2, 4, 8]     [1, 4, 8, 9] |
--------------------------------------------------------------------------------
ChiffreExclusifRegionCol : We remove 6 * 6 : 9 because 9 is identical in the same column.
```

3. Stratégie Paire Exclusive :

Si dans une box se trouve deux Cases dans lesquelles les deux mêmes uniques possibilités figurent, alors on peut supprimer ces deux chiffres des autres cases de la box.

Cet exemple nous démontre l'usage de cette stratégie sur la grille ex3.

Comme on peut le voir, les cases 0x1 et 1x0 se voient retirer de leur liste de possibilité les valeurs 1 et 8 de part de l'exclusion des chiffres 1 et 8 étant obligatoirement sur la ligne 2.

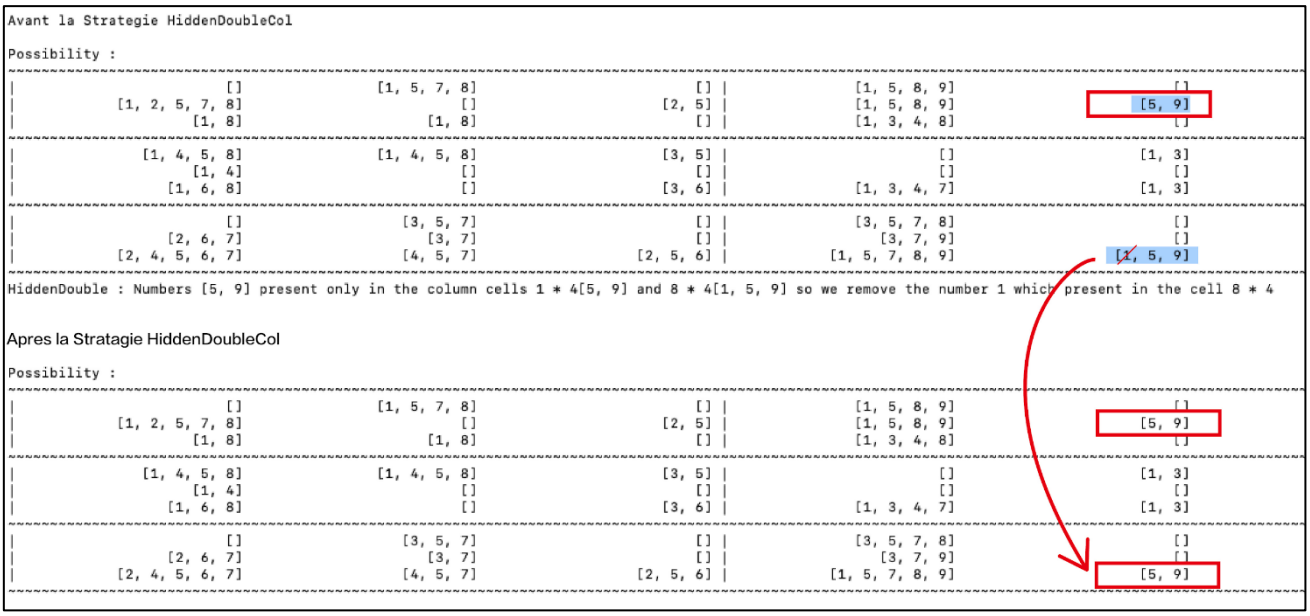
```
Aavant la strategie - PairesExclusives

Possibility :
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|                     []              [1, 5, 7, 8]                  [] |
|          [1, 2, 5, 7, 8]                      []              [2, 5] |
|                 [1, 8]                   [1, 8]                  [] |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|           [1, 4, 5, 8]            [1, 4, 5, 8]              [3, 5] |
|                 [1, 4]                      []                  [] |
|              [1, 6, 8]                      []              [3, 6] |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|                     []               [3, 5, 7]                  [] |
|              [2, 6, 7]                  [3, 7]                  [] |
|        [2, 4, 5, 6, 7]               [4, 5, 7]           [2, 5, 6] |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Paires exclusives we remove 0 * 1 : 1 because it present in cell  2 * 0 [1, 8]   2 * 1 [1, 8]

Paires exclusives we remove 1 * 0 : 1 because it present in cell  2 * 0 [1, 8]   2 * 1 [1, 8]

Paires exclusives we remove 0 * 1 : 8 because it present in cell  2 * 0 [1, 8]   2 * 1 [1, 8]

Paires exclusives we remove 1 * 0 : 8 because it present in cell  2 * 0 [1, 8]   2 * 1 [1, 8]


Apres la strategie - PairesExclusives

Possibility :
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|                     []                  [5, 7]                  [] |
|              [2, 5, 7]                      []              [2, 5] |
|                 [1, 8]                  [1, 8]                  [] |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|           [1, 4, 5, 8]            [1, 4, 5, 8]              [3, 5] |
|                 [1, 4]                      []                  [] |
|              [1, 6, 8]                      []              [3, 6] |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
|                     []               [3, 5, 7]                  [] |
|              [2, 6, 7]                  [3, 7]                  [] |
|        [2, 4, 5, 6, 7]               [4, 5, 7]           [2, 5, 6] |
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Cette stratégie a été adaptée à N valeurs, car on peut aussi imaginer que 3 cases présentant les 3 mêmes possibilités sont aussi exclusives pour les autres cases de la box.

4. Hidden Double :

Si 2 chiffres sont candidats pour 2 cases d'une même ligne ou colonne, mais pour aucune autre case de cette ligne ou colonne, on peut éliminer les autres possibilités pour ces 2 cases.

Dans l'exemple de la grille ex2, les chiffres 6 et 9 sont candidats dans la colonne 4 uniquement sur les cases 1*4 et 8*4. Ces 2 cases portent donc forcément soit le 6 ou le 9, on peut donc supprimer les autres possibilités, dans ce cas le 1 de la case 8*4.



Cette stratégie a elle aussi été adaptée à N valeurs, car on peut aussi avoir N chiffres uniquement sur N cases, permettant aussi d'exclure les autres possibilités sur ces N cases.

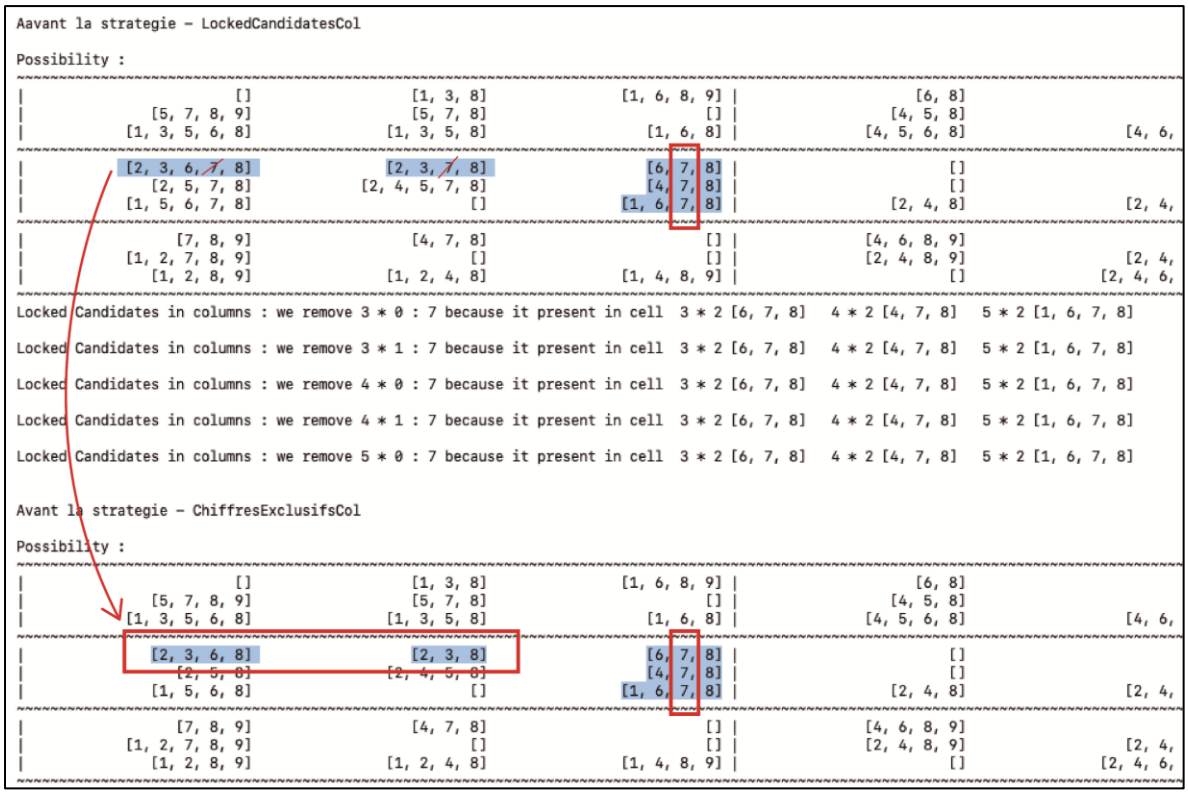
5. Locked Candidates

Dans une box, on peut identifier un chiffre comme étant bloqué sur une ligne ou colonne spécifique si l'ensemble des cases des autres box de cette même ligne / colonne ne proposent pas cette valeur en tant que candidat.

On peut alors affirmer que le chiffre est bloqué sur la ligne ou colonne de la box initiale, et supprimer cette valeur des possibilités autres cases de cette même box.

Dans cet exemple de la grille ex3, concernant la colonne 2, le chiffre 7 est obligatoirement dans les cases 3x3 ou 3x4 ou 3x5.

On peut alors éliminer cette valeur des cases 3x0 et 3x1.

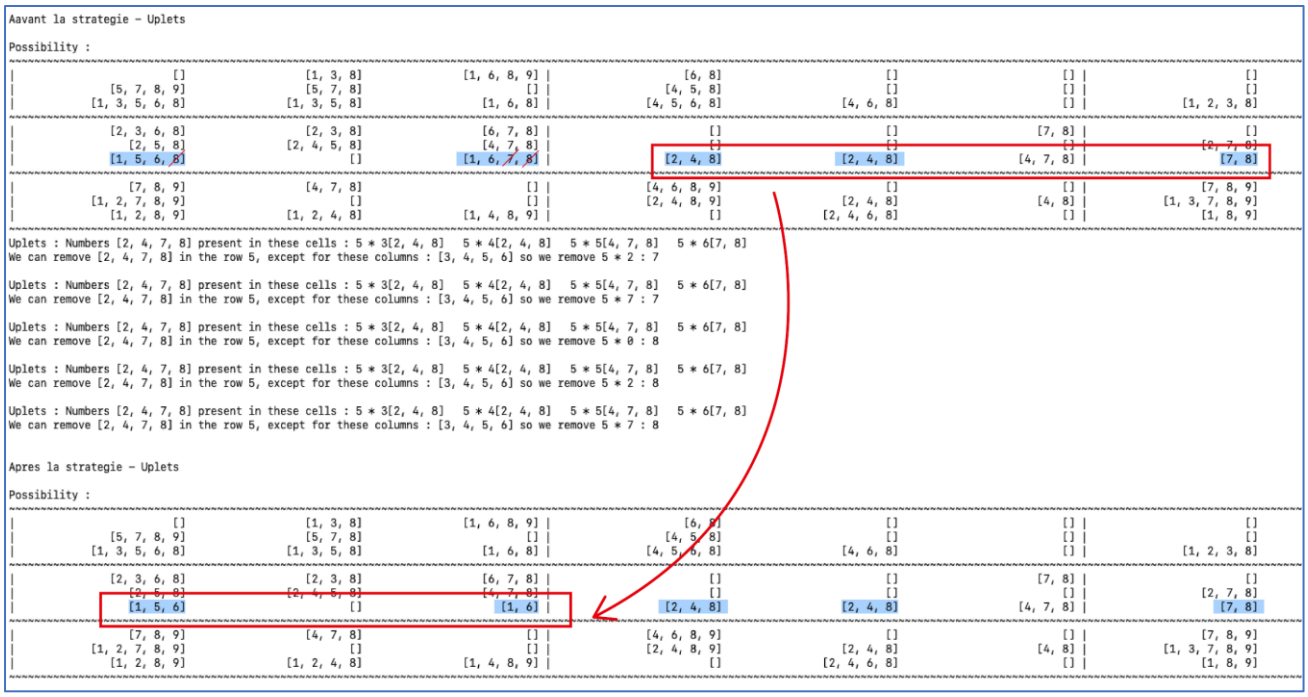


6. Stratégie Naked Uplets

Cette technique avancée suppose que si N cases d'une ligne ou colonne contiennent des valeurs identiques toutes incluses entre N valeurs similaires, on peut alors supprimer ces valeurs des autres cases de la même ligne ou colonne impliquée.

Cet exemple sur la grille ex3 nous montre un Naked Quadruplet : les cases 5x3,5x4,5x5,5x6 ont toutes des candidats strictement inclus dans les 4 valeurs 2,4,7,8.

Ces 4 cases contiennent donc obligatoirement ces 4 valeurs. On peut alors affirmer que les cases 5x0 et 5x2 ne peuvent les contenir, et supprimer ces possibilités des candidats.



7. Stratégie X-Wing :

Cette stratégie consiste à voir, si :

- Dans deux lignes différentes pour un nombre donné il existe exactement deux possibilités qui se trouvent sur les mêmes colonnes.
  OU
- Dans deux Colonnes différentes pour un nombre donné il existe exactement deux possibilités qui se trouvent sur les mêmes colonnes.

Alors,
On peut éliminer toutes les possibilités d'avoir ce nombre sur les colonnes (Resp les lignes) sauf pour les deux lignes (Resp colonnes) identifiées pour établir le schéma d'un X-Wing.

Dans, l'exemple suivant on a trouve dans deux lignes 44 et 48, la possibilité du nombre 38 identique dans les deux colonnes 5 et 18 c'est pour cela on supprime les possibilités du nombre 38 des colonnes 5 et 18 en laissant seulement celles qui se trouvent sur les lignes 5 et 18.

```
Strategie X-Wing colonne : We found on : lignes  44 , 48 cols : 5 , 18 we remove 44 * 35 : 38
Strategie X-Wing colonne : We found on : lignes  44 , 48 cols : 5 , 18 we remove 44 * 36 : 38
Strategie X-Wing colonne : We found on : lignes  44 , 48 cols : 5 , 18 we remove 44 * 37 : 38
Strategie X-Wing colonne : We found on : lignes  44 , 48 cols : 5 , 18 we remove 44 * 38 : 38
Strategie X-Wing colonne : We found on : lignes  44 , 48 cols : 5 , 18 we remove 44 * 44 : 38
Strategie X-Wing colonne : We found on : lignes  44 , 48 cols : 5 , 18 we remove 44 * 46 : 38
Strategie X-Wing colonne : We found on : lignes  44 , 48 cols : 5 , 18 we remove 44 * 47 : 38
Strategie X-Wing colonne : We found on : lignes  44 , 48 cols : 5 , 18 we remove 48 * 35 : 38
Strategie X-Wing colonne : We found on : lignes  44 , 48 cols : 5 , 18 we remove 48 * 36 : 38
Strategie X-Wing colonne : We found on : lignes  44 , 48 cols : 5 , 18 we remove 48 * 40 : 38
Strategie X-Wing colonne : We found on : lignes  44 , 48 cols : 5 , 18 we remove 48 * 47 : 38
```

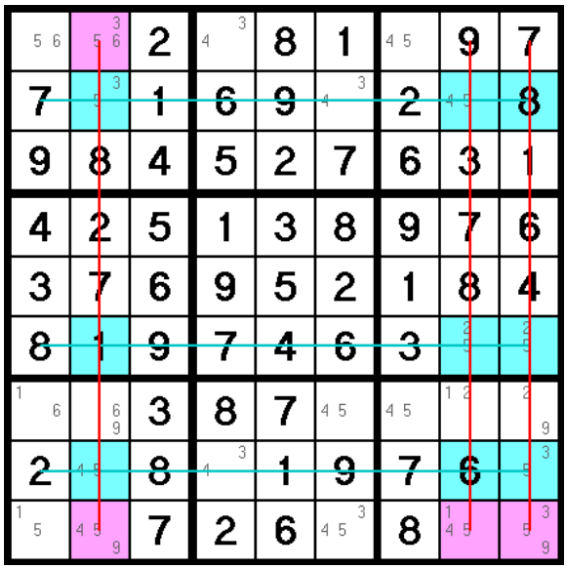
8. Stratégie Swordfish :

Cette stratégie vient comme une généralisation pour la stratégie X-Wing, elle consiste à voir si :

- Dans trois lignes différentes pour un nombre donné il existe deux ou trois possibilités au plus qui se trouvent sur les mêmes colonnes.
  OU
- Dans trois colonnes différentes pour un nombre donné il existe deux ou trois possibilités au plus qui se trouvent sur les mêmes lignes.

Alors,
On peut supprimer les possibilités d'avoir ce nombre dans les colonnes (Resp les lignes) sélectionnées sauf dans les lignes (Resp les colonnes) qui ont été sélectionnées pour établir le schéma d'un Swordfish.

Vu la rareté de cette stratégie dans les tests du programme, on prendra une illustration générale pour cette méthode.
On remarque dans cet exemple, un schéma swordfish des lignes. Dans les lignes 2,6, et 8, la possibilité du nombre 5 est pressentes dans trois colonnes identiques 2 ,8 et 9. C'est pour cela, on peut supprimer la possibilité d'avoir le nombre 5 dans les colonnes 2,8 et 9 sauf pour les lignes 2,6 et 8.



9. Stratégie Coloring :

Le coloring est une technique permettant, en cas d'échec des autres stratégies de tester un chiffre sur les cases ayant un faible nombre de possibilité. Pour cela un premier appel à cette stratégie réalise une sauvegarde de la grille, et pour une case ayant le moins de possibilité possible (idéalement 2), va écrire le premier chiffre sur la grille.

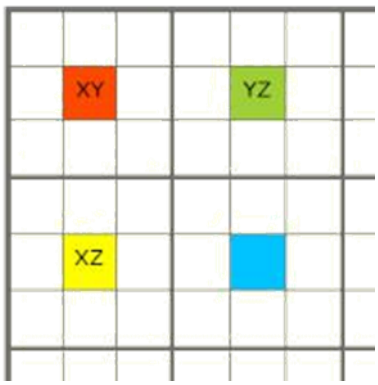
Dès lors le but est de relancer une suite de déduction et de stratégies. Si cela mène à une grille insolvable, on pourra donc affirmer qu'il s'agissait là d'un échec. La sauvegarde sera alors restaurée et le 2e chiffre sera écrit. On reprendra alors les déductions à ce niveau. Si cela mène à une grille solvable, on pourra affirmer que la supposition était juste.

10. Stratégie XYWing :

Il existe deux cas ou appliquer cette stratégie :

Premier cas :

- quelque part dans notre grille on a une case qui a exactement deux possibilités qu'on nomme 'xy' et sur la même ligne on a une autre case contenant 2 possibilités 'yz' et sur la même colonne une autre case contenant 2 possibilités 'xz
- 'Dans ce cas, on prend l'abscisse de la case 'yz' et l'ordonné de la case 'xz' et on regarde si cette case contienne 'z' comme possibilité, si c'est le cas, on le supprime



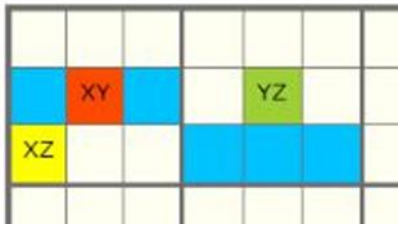
- Si la case rouge était 'x' alors le jaune ne pourrait pas être 'x' alors elle sera forcément 'z' et donc on ne pourra pas avoir 'z' sur la même ligne que la case jaune
- Si la case rouge était 'y' alors la case verte aurait pour valeur 'z' et donc on ne pourra pas avoir 'z' sur la même colonne que la case verte
- En prenant en compte les deux raisonnements précédents, on déduit qu'il est impossible d'avoir la possibilité 'z' dans la case bleu

Deuxième cas :

Ce qui diffère du premier cas, c'est que l'une des deux cases peut partager la même n*k box avec la case 'xy', on distingue deux autres cas :

- Le cas où la case 'xy' partage la même box avec 'xz' et la même ligne avec 'yz' :

- Dans ce cas on supprime les occurrences de ‘z’ dans la ligne qui a le même indice avec ‘xy’ dans la même box



Et

- Les occurrences de ‘z’ dans la ligne de la box ou on trouve ‘yz’ dont l’indice de la ligne est le même avec ‘xz’

  - Si la case rouge était ‘x’ alors la jaune sera forcément ‘z’ et donc on élimine toutes les possibilités de ‘z’ sur la même ligne

  - Si la case rouge était ‘y’ alors la verte sera forcément ‘z’ et donc on élimine toutes les possibilités de ‘z’ dans la même box,

  - En prenant en compte les deux raisonnements précédents, on déduit qu’il est impossible d’avoir la possibilité ‘z’ dans les cases bleus

- Le cas ou la case ‘xy’ partage la même box avec ‘yz’ et la même colonne avec ‘xz’ :

- Dans ce cas, on supprime les occurrences de ‘z’ dans la colonne qui a le même indice avec ‘xy’

Et

- Les occurrences de ‘z’ dans la colonne de la box ou on trouve ‘xz’ dont l’indice de la colonne est le même avec ‘yz’

  - Si la case rouge était ‘x’ alors la jaune sera forcément ‘z’ et donc on élimine toutes les possibilités de ‘z’ sur la même colonne

  - Si la case rouge était ‘y’ alors la verte sera forcément ‘z’ et donc on élimine toutes les possibilités de ‘z’ dans la même box,

  - En prenant en compte les deux raisonnements précédents, on déduit qu’il est impossible d’avoir la possibilité ‘z’ dans les cases bleus

11. Stratégie XYZWing :

Cette stratégie consiste à :

- Trouver dans la grille une case contenant 3 possibilités ‘xyz’
- Chercher dans la même box une case contenant 2 possibilités ‘xz’

Ensuite on a deux cas :

Premier cas :

- Chercher une case qui contienne 2 possibilités ‘yz’ sur la même ligne que ‘xyz’
- Supprimer la possibilité ‘z’ dans toutes les cases de la ligne de la box de ‘xyz’

Deuxième cas :

- Chercher une case qui contienne 2 possibilités ‘yz’ sur la même colonne que ‘xyz’ - Supprimer la possibilité ‘z’ dans toutes les cases de la colonne de la box de ‘xyz’

- Si la case rouge était ‘x’ alors la case jaune sera ‘z’ et donc on élimine toutes les possibilités de ‘z’ dans la box
- Si la case rouge était ‘y’ alors la case verte sera ‘z’ et donc on élimine toutes les possibilités de ‘z’ dans la ligne
- Si la case rouge était ‘z’ alors là on élimine toutes les possibilités de ‘z’ dans la box

- Dans les 3 cas on ne peut pas avoir ‘z’ dans la même ligne de box de la case rouge , alors on supprime toutes les possibilités de ‘z’ dans les cases bleu

E. Limites :

Certaines grilles tel que la grille la plus difficile au monde [3] ne sont résolues par déduction. Cela s'explique par le besoin de multiples tests de type Coloring, car aucune stratégie habituelle ne fonctionne sur ce type de grilles. Cela qui revient à un backtracking complet de la grille.

La génération de grilles avec un fort taux de suppression de cases est lui aussi une limite. Une publication datant de 2012 [5] a démontré qu'une grille 9x9 peut n'avoir qu'une unique solution avec un minimum de 17 chiffres. Les chercheurs ont à l'époque fait appel à des supercalculateurs afin de les générer.

La génération de ce type de grille est très complexe car la suppression aléatoire des chiffres peut facilement nous mener dans une impasse (boucle infinie). Afin de limiter le nombre d'essais lors de la demande d'un taux de suppression de cases important, nous avons ajouté un cas d'arrêt, après un nombre de rollback conséquent dans le cas d'une demande de suppression trop élevée.

Ainsi la demande d'une grille avec n=k=3 avec 70% de taux de suppression finit généralement par nous délivrer une grille avec 62-65% de suppression.
L’application des stratégies sur les grilles larges et difficiles peut devenir très coûteux en temps, comme pour trouver un quintuplet convenant à 5 cases d’une ligne de 100 cases.

Le Backtracking fonctionne sur l’ensemble des grilles. Il a été grandement optimisé, mais peut toujours prendre un temps conséquent dans le cas de grilles larges ou ayant des nombreuses cases avec de multiples possibilités.

L'application des stratégies ainsi que du backtracking dans ces cas extrêmes sont néanmoins possibles avec notre programme, car il peut aussi s'exécuter sur des ordinateurs beaucoup plus puissants afin de révéler pleinement son potentiel.

L'ensemble des stratégies ayant été codées en fonction des tailles des sous box, qui peuvent parfois être différentes (n != k) , ou de taille importante, cela nous a permis de résoudre la majorité des grilles soumises avec des ordinateurs assez peu puissants.

V. Démonstration :

Vous trouverez dans cette partie un lien vers la vidéo démonstrative qui contient plusieurs tests des fonctionnalités du programme :

https://youtu.be/Y33UCq064rY

VI. Conclusion :

Ce projet était en premier lieu une véritable expérience dans laquelle on a appris la gestion du travail, la communication et la collaboration au sein d'une équipe. Grâce au bon encadrement par les professeurs responsables, notre expérience est devenue similaire à celle d'un vrai projet en entreprise. Ce qui nous a permis en quelque sorte d'avoir un aperçu sur le monde professionnel,

Techniquement, à travers ce projet on a pu acquérir des notions relatives à l'intelligence artificielle. Et ce, en rendant possible la résolution des grilles sudoku par une machine en quelques secondes grâce à l'identification des « patterns » des stratégies de résolution, chose qui demande des heures et des heures de réflexion pour un être humain.

Pour conclure, on veut adresser un petit mot de remerciement à nos encadreurs qui étaient toujours à l'écoute et qui nous ont permis de réussir notre projet.

VII. Bibliographie :

[1] TOUVET David. davidtouvet.com/blog | *Sudoku : une méthode de résolution simple et infaillible!* [en ligne]. TOUVET David. Publié le 02/08/2005. Disponible sur : http://www.davidtouvet.com/blog/archives/2005/08/02/sudoku-une-methode-de-resolution-simple-et-infaillible/[Consulté depuis février 2020]

[2] SUDOKU9X9.COM. *Sudoku9x9* [en ligne]. Publié depuis 2009. Disponible sur https://sudoku9x9.com [Consulté depuis février 2020]

[3] COLLINS Nick. *Worlds hardest sudoku and discussions* [en ligne]. The Telegraphe. Publié le 28/06/2012. Disponible sur : https://www.telegraph.co.uk/news/science/science-news/9359579/Worlds-hardest-sudoku-can-you-crack-it.html [Consulté depuis février 2020]

[4] INKALA Arto. *AI Sudoku Top 10* [en ligne]. AI SUDOKU. Publié le 11/11/2006. Disponible sur : http://aisudoku.com/index_en.html [Consulté depuis mars 2020]

[5] REICH Eugenie Samuel. *Mathematician claims breakthrough in Sudoku puzzle* [en ligne]. Publié le 06/01/2012. Disponible sur : https://www.nature.com/news/mathematician-claims-breakthrough-in-sudoku-puzzle-1.9751 [Consulté depuis mars 2020]

[6] *Learn Sudoku!* [en ligne]. Publié en 2008. Disponible sur : https://www.learn-sudoku.com/index.html
[Consulté depuis mars 2020]

[7] Home Page Rentals, LLC.*Sudoku What's Your Favorite Puzzle?* [en ligne]. Publié en 2008. Disponible sur http://www.sudokuessentials.com/ [Consulté depuis avril 2020]

[8] SWAMI Sudoku. *Swordfish Examples & Tips* [en ligne]. Publié le 11/01/2018. Disponible sur https://www.youtube.com/watch?v=Be8YyH_xhgs [Consulté en avril 2020] 1 vidéo, 37 mins.

[9] THONKY | *Swordfish Strategy* [en ligne]. THONKY. Publié en 2001. Disponible sur : https://www.thonky.com/sudoku/sword-fish [Consulté en avril 2020]