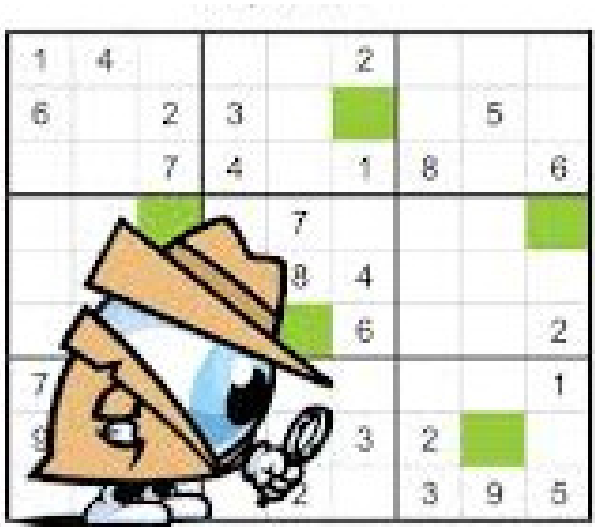
Implémentation d'un Système Tutoriel Intelligent de Sudoku

22 décembre 2013

Maxime Turnel TURM08128309
Jules Mozes MOZJ20059105
Ludovic Lefebvre LEFL16089009
Gilles Godefroid GODG26025804



INF7470 - Systèmes tutoriels intelligents
Automne 2013
Université du Québec à Montréal
Professeur : Roger Nkambou

Table des matières

Table des figures

Résumé

Le but de ce travail est de construire un coach de Sudoku permettant d'apprendre les règles du jeu et quelques stratégies de résolution de grilles de Sudoku. L'implémentation de ce système tutoriel a été faite en Java.

Notre choix a été de ne pas utiliser de jeux de Sudoku à notre disposition sur internet, et de concevoir de A à Z ce système tutoriel, ainsi que de ne pas utiliser les outils fournis par CTAT.

L'analyse des fichiers sources de logiciels disponibles, ainsi que l'apprentissage du fonctionnement de CTAT, nous sont apparus comme étant dispendieux en temps par rapport à la conception `de novo` d'un logiciel.

Enfin, nous avons fait le choix de nous limiter à 4 stratégies prises en compte par le tuteur. Cela nous permet de créer une base de système tutoriel sans pour autant prétendre maîtriser toutes stratégies complexes du sudoku.

1 Interface

Le défi de la conception de l'interface était de permettre d'afficher par l'utilisateur ses choix de possibilités pour une case, d'afficher un choix définitif , ainsi que de ne permettre aucuns changements sur les cases de la grille de départ.

Il fallait également pouvoir afficher les remarques et/ou les aides du tuteurs. Lorsque le tuteur parle, le joueur ne peux plus interagir avec le sudoku. C'est un moyen de "forcer" l'apprenant à tenir compte des remarques.

À compléter par un screenshot de l'interface.

2 Expert

Lors de la phase d'exploration avec l'outil CITRIX, nous avons étudié la possibilité de créer un tuteur cognitif qui utilise la librairie JESS pour modéliser les règles du jeu du sudoku.

Quand nous avons abandonné l'idée d'utiliser CITRIX, nous avons continué de croire que JESS était une alternative viable pour construire un expert en sudoku.

Lors de sa conception, nous avons identifié deux types d'informations que l'expert devait minimalement trouver et fournir aux autres modules du tuteur : la solution au puzzle et les prochaines actions possibles selon des stratégies de résolution préétablies.

2.1 La solution

L'algorithme que nous avons utilisé pour trouver la solution du puzzle et assez simple.

L'expert fournit à JESS toutes les possibilités de valeurs pour les cases blanches et les valeurs pour celle faisant partie du puzzle. Ensuite, une règle qui décrit quelles sont les cases qui ne peuvent être égales d'après les règles du jeu permet à JESS de parcourir toutes les possibilités et de trouver la bonne solution.

Comme JESS doit potentiellement parcourir tout l'espace problème, l'exécution de cette recherche est lente, mais nous avons jugé que cela n'avait peu d'impact, cette fonction n'étant exécutée qu'une seule fois, au début de la partie.

Cet algorithme aurait pu être grandement amélioré en appliquant de manière récursive les mêmes stratégies que celle que nous demandons à joueur d'acquérir.

2.2 Les prochaines actions

Le but du tuteur est de faire apprendre des stratégies de résolution aux joueurs. Mais pour savoir si l'apprenant applique bel et bien celles-ci, il faut identifier quelle stratégie a été utilisée pour effectuer chaque coup.

Alors l'expert analyse la grille de sudoku après chaque coup pour trouver tous les coups possibles d'après chaque stratégie et, lors du coup suivant, pouvoir déterminer la stratégie à laquelle il correspond. Cette méthode permet aussi de potentiellement déterminer si le joueur triche ou joue au hasard, car lors qu'il fournit une bonne réponse, mais qu'aucune stratégie ne prouve son coup, ce comportement est suspect.

3 Tuteur

3.1 Principe du tuteur

Le but du tuteur est de remplir le rôle de coach pour l'apprenant. Le tuteur accompagne l'apprenant tout au long de sa partie de Sudoku à travers des messages textuels.

Chaque action de l'apprenant est analysée puis le tuteur réagit en conséquence à travers ces feedback, de manière à le conduire vers l'acquisition de stratégies de résolution du Sudoku. Le tuteur s'adapte alors au niveau de l'apprenant, lui laissant de plus en plus d'autonomie une fois qu'une bonne expertise est atteinte, tout en insistant sur les concepts nouveaux qu'il introduit au fil de la progression.

3.2 Bienvenue !

À l'ouverture du logiciel, le tuteur présente le logiciel et son interface utilisateur. Si c'est la toute première fois que l'apprenant lance l'application, le tuteur procède directement par l'enseignement des principes de base du Sudoku, à savoir le but du jeu, la topologie de la grille de jeu et les règles de base.

3.3 Introduction des stratégies

L'introduction des stratégies est gérée par palier d'expertise. Au tout début d'apprentissage, le tuteur enseigne une stratégie initiale comme première directive de jeu. Tout au long de la progression, le tuteur explique à l'apprenant de nouvelles stratégies dès lors que les précédentes stratégies sont estimées maîtrisées par ce dernier. Elles ne sont expliquées qu'une seule fois à chaque fois. Elles peuvent être réintroduites intelligemment à travers l'utilisation des indices.

3.4 Application d'une stratégie avec succès

Dès lors que l'apprenant applique une stratégie avec succès, le tuteur le félicite et l'encourage, en explicitant la stratégie que l'apprenant vient d'utiliser. A la troisième félicitation du tuteur pour une stratégie donnée, ce dernier le félicite encore mais pour une ultime fois, estimant que l'utilisateur a été suffisamment flatté et maitrise désormais la stratégie en question.

3.5 Gestion des erreurs de l'apprenant

Les erreurs sont gérées par palier d'expertise.

Au début de l'apprentissage, le tuteur indiquera précisément les erreurs à l'apprenant immédiatement après les avoir commises.

Au fur et à mesure, le tuteur se fera plus discret sur cette détection des erreurs, obligeant l'apprenant à rester plus concentré sur sa partie et à devenir indépendant de la détection des erreurs du tuteur.

À partir d'un certain niveau d'expertise, le tuteur indiquera seulement que des erreurs sont présentes, donnant seulement un avertissement à l'apprenant. Pour le tuteur, les erreurs dues au hypothèses sont trois fois moins importantes que les erreurs dues aux valeurs définitives, il effectue donc un traitement pondéré globale des erreurs. Seules les erreurs d'attribution sont analysées, et non les erreurs d'omission.

3.6 Coups au hasard

Il est tout à fait possible pour un être humain de jouer au hasard. Le tuteur prend cela en compte en identifiant le hasard lors d'absence de stratégie applicable dans les coups de l'apprenant. S'il arrive à l'apprenant de jouer un coup au hasard menant à une valeur correcte, le tuteur le lui signale les trois première fois, de manière à ne pas restreindre ses libertés d'action.

Pour le cas où le coup mène à une valeur erronée, le tuteur considère cela comme erreur et la gestion des erreurs s'applique. Ainsi, aux stades plus avancés d'apprentissage, il sera difficile pour l'apprenant d'obtenir des renseignements sur des coups joués aléatoirement, ce qui renforce son autonomie, son jugement et sa prise de responsabilité vis à vis de ses coups.

3.7 Utilisation des indices

Si l'apprenant se retrouve bloqué et n'arrive pas à avancer, il peut décider à tout moment de demander un indice au tuteur. Comme premier indice, le tuteur rappelle à l'apprenant la stratégie la plus simple qu'il est possible d'utiliser afin qu'il progresse dans la résolution de la grille. Cela permet de lui remémorer une stratégie qu'il n'arrive pas à appliquer de manière procédurale en plus de focaliser son attention sur l'utilisation d'une stratégie en particulier.

Comme deuxième indice, le tuteur indique à l'apprenant où une valeur définitive peut être inscrite. Selon son niveau d'expertise, la position du coup à jouer est de moins en moins précise, de manière a améliorer la vision globale de l'apprenant quant à sa grille, même en cas de difficultés.

Lorsque l'utilisateur joue un coup après avoir demandé un indice, le compteur des indices retombe à zéro.

La structure générale du tuteur est décrite dans la figure 1

3.8 Possibles améliorations futures du tuteur

Tout d'abord, il n'y a aucune sauvegarde du modèle de l'apprenant. Par conséquent, si l'apprenant relance l'application, ses connaissances seront remise à zéro. La sauvegarde devrait être implémentée pour une utilisation plus poussées du logiciel.

En outre, l'évaluation de l'expertise de l'apprenant a été faite de manière subjective. il serait judicieux de procéder à une évaluation plus rigoureuse à travers des tests utilisateurs.

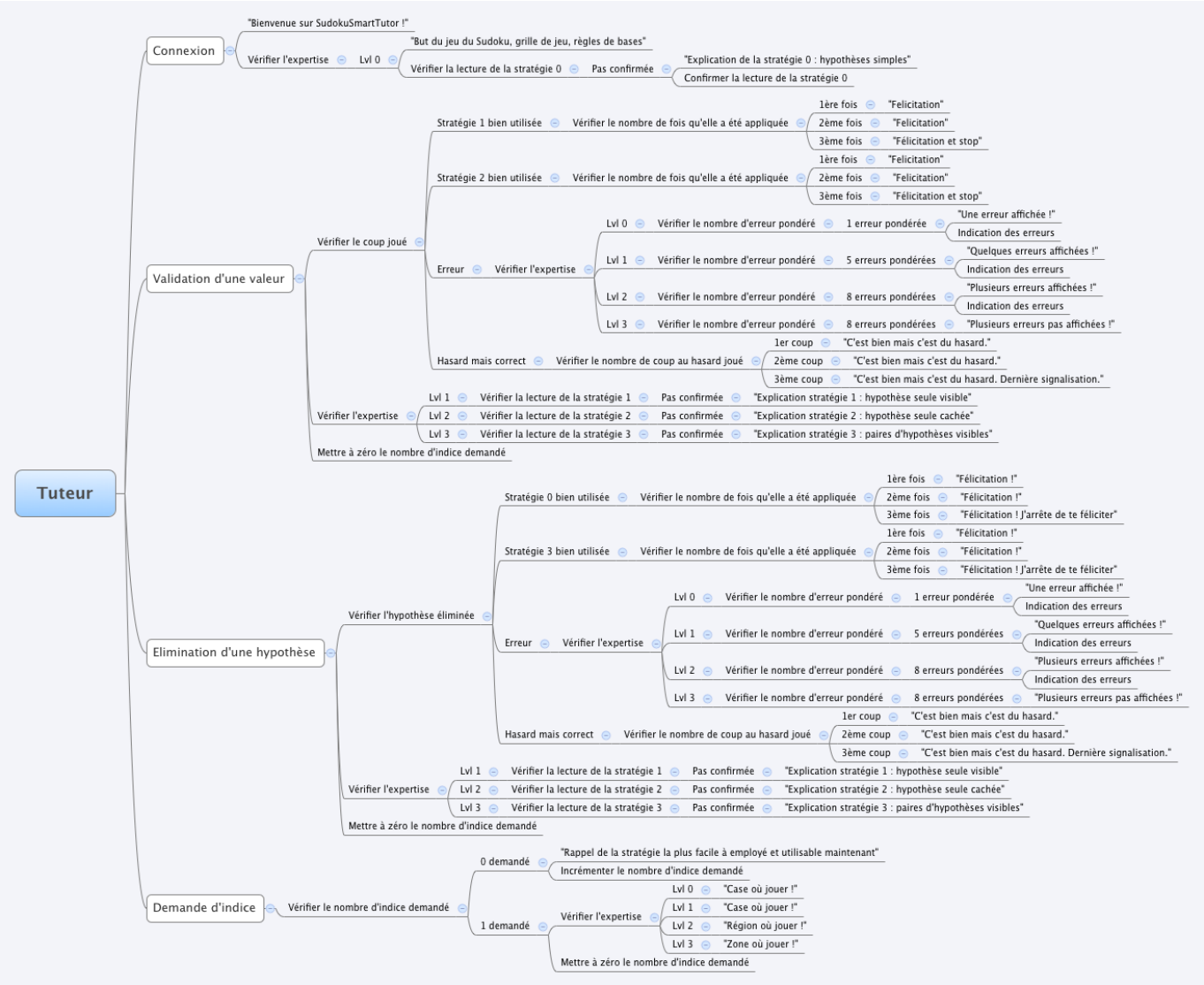


FIGURE 1 – Fonctionnalités du tuteur

D'autre part, les erreurs réalisées par l'apprenant ne sont pas analysées mais simplement signalées. amélioration possible serait de fournir des explications en rapport avec les erreurs commises, en répertoriant les erreurs courantes ainsi que leurs causes.

Enfin, l'apprenant pourrait être tenté d'exploiter le fonctionnement du tuteur pour arriver à résoudre des Sudoku plus facilement, notamment avec la gestion des erreurs couplée avec les coups au hasards.

Des tests utilisateurs devrait être mis en place afin d'ajuster la qualité, la quantité et la pertinence des interventions du tuteur de manière à améliorer son aspect pédagogique dans le but de parfaire les conditions d'apprentissage.

4 Le modèle de l'apprenant

4.1 Définition des connaissances

Comme précédemment expliqué, nous sommes partis sur une base de 4 stratégies (2 pour établir un chiffre dans une case et 2 pour supprimer des hypothèses dans une case) :

La classe se compose :

— D'un tableau de taille 4 comptabilisant le nombre de succès provenant de chaque stratégies,

— D'un tableau de taille 2 comptabilisant le nombre d'échec sur le placement d'un chiffre ou le retrait d'une hypothèse,

— De deux tableaux similaires aux deux précédant stockant une pondération pour déterminer le niveau.

— Un tableau représentant les seuils de passage des 4 niveaux,

— Un compteur de l'utilisation d'aléatoire,

— Un compteur du nombre de coups totaux joués.

Ainsi que décrit dans la figure 2

```
int KNOWLEDGE_COUNT = 4;

// 0 -> KNOWLEDGE_1, //Validation d'une valeur avec la stratégie de l'hypothèse seule explicite
// 1 -> KNOWLEDGE_2, //Validation d'une valeur avec la stratégie de l'hypothèse seule cachée
// 2 -> KNOWLEDGE_3, //Elimination d'une hypothèses avec les règles de bases
// 3 -> KNOWLEDGE_4, //Elimination d'une hypothèses avec la stratégie des paires d'hypothèses explicite
int knowledge_success[] = new int[KNOWLEDGE_COUNT];

// 0 -> FAIL_1, //Echec dans l'ajout d'un chiffre
// 1 -> FAIL_2, //Echec dans la suppression d'un hypothèse

int knowledge_fail[] = new int[KNOWLEDGE_COUNT];

int knowledge_sucess_ponderation[] = new int[KNOWLEDGE_COUNT];
int knowledge_fail_ponderation[] = new int[KNOWLEDGE_COUNT/2];
int knowledge_lvl[] = new int[4];

int numRandom;
int nombreDeCoupJoue;
```

FIGURE 2 – Composition de la classe apprenant

4.2 Gestion de l'apprentissage

Fonctionnement

En utilisant Jess, il est possible de savoir la liste des coups possibles pour chaque stratégie à un état donné. Ainsi, lorsque l'apprenant joue un coup, il est possible de

savoir si ce dernier est un succès ou un échec, et si il se base sur une stratégie ou s'il joue de manière aléatoire.

Succès

Un bon coup peut :

— Ne pas découler de l'utilisation de l'une des stratégies. On considère alors que c'est un coup aléatoire et on incrémente le compteur aléatoire dans le modèle de l'apprenant.

— Provenir de l'utilisation d'une stratégie. Dans ce cas on incrémente le compteur de connaissances de la stratégie associée.

Echec

Dans le cas d'un échec, il est difficile de déterminer les causes. C'est pour cela que nous avons choisi de séparer les échecs en deux catégories :

— La première lorsque un chiffre validé est faux.

— La deuxième quand une hypothèse bonne est enlevée.

Ces échecs vont nous permettre de mieux évaluer l'apprentissage des stratégies.

4.3 Niveau global du joueur

Pondération

Nous nous servons d'un système de pondération afin de donner plus ou moins d'importance à certaine stratégie plutôt qu'une autre. En effet plus la stratégie est complexe plus elle a un poids important dans le calcul global du niveau de l'apprenant.

Il y a également la possibilité de différencier l'impact des modification des hypothèse, qui est moindre, par rapport à celui des validations de chiffres (l'apprenant va effectuer beaucoup plus de manipulations sur les hypothèses que sur les placement de chiffre).

Calcul du niveau

Le calcul s'effectue en multipliant les réussites/échecs de l'application des connaissances par leur valeur de pondération et en les sommant (aussi bien pour les succès que les échecs). On renvoie au tuteur un niveau entre 1 et 4, à lui ensuite de communiquer avec l'étudiant de la manière la plus adapté.

4.4 Remarques

Les valeurs de pondération et les seuils de niveau ont été établis de manière arbitraire. Afin de les déterminer avec plus de pertinence, il aurait été bon de réaliser des tests sur un plus grand panel d'utilisateurs.

5 Stratégies

5.1 Vocabulaire

— Colonne : ensemble des 9 cases qui se trouvent à la verticale par rapport à une case. Il existe en tout 9 colonnes qui sont numérotés de 1 à 9, de la plus à gauche à la plus à droite.
— Ligne : ensemble des 9 cases qui se trouvent à l'horizontale par rapport à une case. Il existe en tout 9 lignes qui sont numérotés de 1 à 9, de la plus à gauche à la plus à droite.
— Région : ensemble des 9 cases qui se trouvent dans le même carré qu'une case. Il existe en tout 9 régions qui sont numérotés de 1 à 9, selon le sens de lecture de gauche à droite d'abord puis de haut en bas.
— Unité : colonne, ligne, ou région d'une case.

5.2 Stratégies par sélection

Stratégie de l'hypothèse

Pour un chiffre donné, pour une case donnée, le chiffre donné est valable selon la règle de la colonne, selon la règle de la ligne et selon la règle de la région. Alors le chiffre donné est un chiffre hypothétique.

Stratégie de l'hypothèse seule explicite

(anglais : naked single = célibataire nu) Si une case donnée ne contient qu'un seul chiffre hypothétique. Alors ce chiffre hypothétique est un chiffre inscrit.

5.3 Stratégies avancées par élimination

Stratégie de l'hypothèse seule implicite

(anglais : hidden single = célibataire caché) Pour une unité donnée, pour un chiffre donné, si ce chiffre est un chiffre hypothétique pour une et seule case de l'unité donnée,

alors ce chiffre est un chiffre inscrit.

Stratégie des l'hypothèses couplés explicite

(anglais : naked pair = pair nu) Commençons déjà avec celle du dessus ?

6 Guide d'utilisation

Pour exécuter le projet à partir de la ligne de commande, allez dans le dossier du projet et tapez la commande suivante :

java-jar "ProjetSTI.jar"

Lors du lancement du logiciel le tuteur donne toutes les indications de fonctionnement et les manoeuvres possibles.