UQAC

Cours 8INF847 - Intelligence artificielle

Kévin Bouchard, Ph.D.

TP 2 - Création CSP pour le jeu du Sudoku

Claire Mielcarek

Esther Prudhon-Delagrange

Orann Weber

8 octobre 2018

1 Introduction

Ce projet a pour objectif de modéliser un agent solveur de sudokus, présentés sous la forme de problèmes à résolution de contraintes. Nous souhaitons être capables de résoudre un grand nombre de sudokus en un temps réduit.

2 Modélisation

Nous modélisons le problème du sudoku tel un problème à satisfaction de contraintes. Chaque case du sudoku va être soumise à un certain nombre de contraintes, répertoriées dans son tableau *contraints*. Chacune des contraintes va lier la cellule concernée à une cellule "voisine" (dans la même ligne, colonne ou le même carré), et va être représentée de manière binaire, par un booléen, définissant la compatibilité des valeurs des deux cases.

Afin de résoudre ce problème, nous utilisons la méthode du backtracking, par le biais de la fonction *backtrakingSearch* dans notre main. Nous propageons les contraintes en utilisant l'algorithme AC3 dans notre classe *Sudoku* et non pas un forward checking classique. Nous utilisons AC3 après l'assignement. Cet algorithme permet de détecter les cas impossibles plus rapidement et ainsi de réduire le temps de traitement.

Les cellules du sudokus sont sélectionnées et traitées dans un ordre particulier afin d'optimiser le programme. Pour cela, nous faisons tout d'abord appel à l'algorithme Most Remaining Value (fonction *mrvSelection()*), que nous couplons à l'algorithme Degree Heurisitc afin de départager les cellules égales (fonction *degreeHeuristicSelection()*).

Les valeurs sont également sélectionnées dans un ordre particulier, afin d'optimiser encore plus l'algorithme. Pour cela, nous avons la fonction *leastConstrainingValues()* qui va ordonner les valeurs du domaine de façon à ce que les valeurs les moins comprises dans les domaines des contraintes de la celulle soient testées en priorité.

3 Execution du programme de résolution

Afin d'exécuter le programme, il suffit d'ouvrir un terminal et de lancer l'exécution du .jar fourni avec la commmande : *java -jar "work2.jar*. Le programme va alors résoudre et afficher les 100 sudokus que l'on fournit dans le fichier sudokus.txt dans le même dossier. Afin de lui faire résoudre un nouveau sudoku, il suffit de l'ajouter dans le fichier sudokus.txt en respectant la syntaxe (les lignes du sudokus sont écrites à la suite), il n'est pas nécessaire de modifier le code du programme.

Le code de notre application est disponible ici : `https://github.com/Orann/ia_work2`.