TP1 – Jeu de Sudoku

Travail présenté à

M. Jian-Yun Nie

Par

Émile Trottier (p1054384)

François Poitras (p)

IFT3335 – Introduction à l’intelligence artificielle

Université de Montréal

9 octobre 2015

1. Pour définir la résolution du jeu de Sudoku comme un problème de recherche dans l’espace d’états, il faut définir les éléments suivants :

* Un état correspond à un arrangement des chiffres dans la grille de Sudoku. On peut la voir comme une matrice 9x9 où chaque élément est un chiffre entre 0 et 9 inclusivement. Un zéro à la case (i,j) indique qu’elle est vide. Dans notre programme, nous utilisons la notion de nœud. Un nœud comprend un état, donc un tableau 9x9 indiquant une certaine disposition des chiffres dans une grille de Sudoku, ainsi que son coût, son nœud parent et une action. On peut voir le nœud comme une dérivation de son nœud parent lorsqu’on fait cette action.
* L’état de départ est la configuration initiale telle qu’elle nous est donnée dans le fichier 100sudoku.txt ou encore 1000sudoku.txt, mais évidemment sous forme de tableau 9x9 et non d’une ligne de 81 chiffres. Même si ces deux représentations ont le même sens, la première est beaucoup visuelle et beaucoup plus facile à lire et à comprendre.
* On ne peut pas définir l’état but puisque le problème est de le trouver (on ne le connaît pas au début). Par contre, on peut très bien définir une fonction de vérification de but. Il s’agit de vérifier que chaque ligne, chaque colonne et chaque boîte de la disposition X contienne chacun des chiffres entre 1 et 9 inclusivement. La méthode goal\_test(self, state) de la classe Problem de notre programme est une implantation de la fonction de vérification de but. Elle est trop longue pour être affichée dans le rapport malheureusement, il vous faudra regarder notre code.
* Nous avons déjà parlé en quelque sorte de la relation de successeur lorsque nous avons défini la notion de nœud. Une action se résume à transformer un zéro d’un état X en un certain chiffre entre 1 et 9 inclusivement pour se retrouver dans l’état Y. Dans cet exemple, le nœud correspondant à l’état Y stockerait un lien vers le nœud correspondant à l’état X (puisque c’est son parent) et l’action puisqu’elle est le lien entre ces deux nœuds. On dit donc que le nœud Y est un successeur du nœud X. L’ensemble de tous les nœuds B qu’on peut atteindre en une seule action à partir d’un nœud A, donc en changeant n’importe quel des zéros de la disposition A en un chiffre entre 1 et 9, est l’ensemble de tous les successeurs de A.
* Le coût d’étape est de 1 peu importe l’action qu’on fait. Cette caractéristique n’est pas très importante pour le problème de résolution du Sudoku vu qu’on sait d’avance combien nous coûtera la solution (ce sera le nombre de cases vides de l’état initial) et que nos décisions ne se baseront jamais là-dessus

2. Description de l’implantation

À faire lorsque le programme sera fini. Il faudra aussi s’assurer qu’après la remise, le programme roule bien.

2.1 Recherche en profondeur d'abord

La recherche en profondeur d'abord est un algorithme de backtracking classique. En effet, nous commençons par localiser une case vide (définie comme contenant un zéro) et nous la remplissons par le premier nombre qui ne cause pas de conflits dans cette case. Puis, récursivement, on recommence le processus. Si, au cours de l'exploration des nœuds, on découvre un état qui fait en sorte qu'aucun nombre n'est possible pour une case, on cesse l'exploration et on remonte dans l'arbre.

2.2 Recherche par Hill-Climbing

2.3 Recherche heuristique

3. Comparaison des algorithmes sur les 100 configurations (nombre de nœuds explorés, taux de succès avec limite de 500, 1000, 5000, 10000, …, complexité des algorithmes,…)