Rapport du TP1

# Cours IFT3335-A-A22 - - Intelligence artificielle : introduction

# Mahmoud Labidi : 20176755

# Yacine Mkhinini : 20170474

Notre expérience durant ce TP était assez spéciale. En effet, à la suite de certaines contraintes de temps et d’autre obligations, nous avons été obligés de faire l’intégralité du TP en seulement quelques jours.

Afin de pouvoir comparer l’efficacité des différents algorithmes utilisé dans notre programme nous avons utilisé 3 fichiers différents avec un nombre de combinaison de départ différents et une difficulté différente :

Voyons d’abord les résultats donnés par l’algorithme de Norvig de base qui choisit le careau avec le plus petit nombre de candidats (3eme heuristique) :

Solved 95 of 95 95sudoku puzzles (avg 0.03 secs (34 Hz), max 0.16 secs).

Solved 100 of 100 hard puzzles (avg 0.01 secs (118 Hz), max 0.03 secs).

Solved 1000 of 1000 hardest puzzles (avg 0.01 secs (138 Hz), max 0.03 secs)

* On a donc un taux de réussite de 100% et une moyenne de 0.02 secs environ pour résoudre chaque sudoku de n’importe quelle difficulté.

Si on met désactiver le 3ième critère en choisissant un carreau aux hasard (fonction solveRandom dans notre code) :

On obtient les résultats suivants :

Solved 95 of 95 95sudoku puzzles (avg 0.03 secs (31 Hz), max 0.19 secs).

Solved 100 of 100 hard puzzles (avg 0.01 secs (130 Hz), max 0.02 secs).

Solved 1000 of 1000 hardest puzzles (avg 0.01 secs (124 Hz), max 0.03 secs).

On a donc un taux de réussite de 100% et une moyenne de 0.02 secs environ pour résoudre chaque sudoku de n’importe quelle difficulté.

* On remarque que les résultats ont légèrement diminuer et notre algorithme n’est plus autant efficace puisque l’arbre de recherche n’est plus ordonné

L’heuristique de la question 3 qu’on a choisi c’est Hidden Singles de l’article Solving sudoku de Angus Johnson. En effet, c’est l’algorithme qui nous a parus le plus intéressant à implémenter. La logique est la suivante :

Cette heuristique consiste à regarder toutes les unités (lignes, colonne et bloc) et chercher une case qui contient un candidat unique dans son unité. Ceci signifie que ce candidat ne peut être que dans cette case puisque l’unité doit nécessairement contenir ce chiffre.

Cela nous permet d’assigner ce candidat unique a cette case.

Pour son fonctionnement nous avons défini la méthode : Hidden\_Singles (unit, values)

En guise de comparaison, voilà les résultats obtenus avec les mêmes fichiers pour cette heuristique :

Solved 95 of 95 95sudoku puzzles (avg 0.03 secs (32 Hz), max 0.17 secs).

Solved 100 of 100 hard puzzles (avg 0.01 secs (136 Hz), max 0.02 secs).

Solved 1000 of 1000 hardest puzzles (avg 0.01 secs (134 Hz), max 0.03 secs).

On a donc un taux de réussite de 100% et une moyenne de 0.02 secs environ pour résoudre chaque sudoku de n’importe quelle difficulté.

* On remarque que les résultats ont légèrement augmenter et notre algorithme est plus autant efficace puisque cette heuristique élimine des candidats non nécessaires et donc l’arbre de recherche est plus cours

Pour le dernier algorithme de la question 4 ; le Hill climbing, nous avons implémenté la méthode Hill\_climbing(values). Cette méthode reçois en paramètre un dictionnaire (des carreau, candidats) et applique l’algorithme tel que décris par la page Wikipedia.

On remplit chaque bloc par des valeurs aléatoire sans prendre en considérations les colonnes et les lignes. On génère la liste des swaps possible dans chaque bloc et on calcule le swap optimal de cet état en se servant de la méthode conflict\_count qui compte le nombre de conflits dans une table de sudoku

Voici les résultats trouvés :

Solved 0 of 95 95sudoku puzzles (avg 0.45 secs (2 Hz), max 0.81 secs).

Solved 44 of 100 hard puzzles (avg 0.13 secs (7 Hz), max 0.58 secs).

Solved 425 of 1000 hardest puzzles (avg 0.13 secs (7 Hz), max 0.53 secs).

* Naturellement cet algorithme de nature glouton n’arrive pas à résoudre tous les cas mais occasionnellement arrive à résoudre quelques cas de test.

En guise de conclusion, je dirais que faire ce TP était une expérience assez spéciale qui était aussi éprouvante qu’amusante et divertissantes pour deux amis passionnés par l’informatique, et certainement c’était le travail le plus informative qu’on eu dans notre programme jusqu’à maintenant.