99999 NOM Prénom

MPI

Résolution du Rubik's Cube

Malgré le fait que toute position du Rubik's Cube peut se résoudre en moins de 20 mouvements, trouver une séquence de coups permettant d'y arriver n'est pas chose aisée.

Je me suis donc intéressé aux différents moyens d'y parvenir, afin de les comparer en termes de longueur de la solution ainsi qu'en temps de calcul requis.

Positionnement thématique (ÉTAPE 1) :

- INFORMATIQUE (Informatique pratique)
- INFORMATIQUE (Informatique Théorique)

Mots-clés (ÉTAPE 1) :

Mots-clés (en français)	**Mots-clés (en anglais)**
Parcours de graphe	*Graph traversal*
*Algorithme IDA**	*IDA* algorithm*
Heuristique	*Heuristic*
Algorithme de Thistlethwaite	*Thistlethwaite's algorithm*
Algorithme de Kociemba	*Kociemba's algorithm*

Bibliographie commentée

Le Rubik's Cube est un casse-tête inventé par Ernö Rubik en 1974 et popularisé dans les années 1980. Il a notamment obtenu le prix du meilleur jeu solitaire en 1980. Il consiste en un cube 3x3x3, composé de plus petits cubes. Les faces de chacun de ces petits cubes ont un autocollant de couleur, de telle sorte qu'il soit possible de résoudre le Rubik's Cube en faisant concorder les couleurs de chacun des petits cubes sur la face du grand cube **[1]**.

L'ensemble des positions possibles d'un Rubik's Cube crée un graphe. Ainsi, le problème de la résolution d'un Cube se ramène à la recherche d'un chemin entre l'état correspondant au cube mélangé et l'état correspondant au cube résolu dans ce graphe.

La solution la plus naturelle est un parcours en largeur, puisqu'il permet d'obtenir le plus court chemin. Cependant, avec plus de 43 x 10^18 positions possibles, ce serait extrêmement long.

Il est possible de réduire drastiquement le nombre de nœuds explorés en utilisant une heuristique, c'est-à-dire une fonction permettant de minorer le nombre de coups nécessaires à la résolution d'un cube. En effet, l'algorithme A* permet d'orienter la recherche en explorant en priorité les nœuds ayant une valeur d'heuristique plus faible, donc à priori plus proches du cube résolu. Cependant, un problème persiste avec l'utilisation de mémoire vive trop importante par l' algorithme A*. À chaque étape, chaque voisin n'ayant pas encore été exploré est ajouté à la liste des cubes à explorer. Cette liste devient donc rapidement trop longue pour être simplement stockée.

L'algorithme IDA* permet de réduire la taille de la liste en utilisant une profondeur de recherche maximale. Ainsi, si la somme de la profondeur actuelle du nœud et de la valeur de son heuristique est supérieure à une certaine valeur, on sait que la solution finale ne sera pas en dessous du seuil imposé, donc le nœud ne sera pas exploré, permettant de limiter le nombre d' éléments enregistrés comme à explorer **[2]**. Le seul point négatif est que si la recherche n' aboutit pas pour une certaine profondeur maximale, il faut l'augmenter avant de faire une nouvelle recherche. Cependant, la profondeur maximale d'un cube étant de 20 **[3]**, ce n'est pas un problème majeur.

L'algorithme de Korf utilise l'algorithme IDA* pour résoudre le Rubik's Cube. Il consiste à calculer à l'avance et à stocker dans une base de données 3 heuristiques : le nombre de coups nécessaires pour résoudre les coins uniquement, le nombre de coups pour résoudre la moitié des arêtes uniquement, et le nombre de coups pour résoudre l'autre moitié des arêtes uniquement **[4]** . Ainsi, prendre le maximum de ces 3 heuristiques permet d'obtenir une heuristique admissible (elle est bien une minoration du nombre de coups nécessaires) et cohérente. Cet algorithme permet donc d'obtenir le plus court chemin, mais a une durée d'exécution relativement élevée.

L'algorithme de Thistlethwaite, quant à lui, permet d'obtenir une solution bien plus rapidement, mais qui ne sera pas optimale en termes de nombre de coups. Il crée plusieurs groupes successifs de positions du cube, en fonction de si la position est atteignable en interdisant les quarts de tour de certaines faces **[5]**. En effet, il est possible de déterminer la présence d'une position dans un de ces groupes grâce à des propriétés de la position **[6]**. Ainsi, savoir qu'une position est dans un de ces groupes permet de se limiter dans le facteur d' embranchement de la recherche, puisque certains quarts de tour ne sont pas nécessaires à explorer. De plus, il est possible d'accélérer la transition entre chaque groupe en stockant le nombre de coups avant d'atteindre le groupe suivant dans une base de données calculée à l' avance, en utilisant le fait que de nombreuses positions sont équivalentes aux yeux d'un groupe **[7]**. Il existe également une variante de l'algorithme de Thistlethwaite, l'algorithme de Kociemba, utilisant moins de groupes intermédiaires **[8]**.

Problématique retenue

Quelles sont les différences entre les différents algorithmes de résolution du Rubik's Cube, en termes de temps de calcul utilisé et de longueur de la solution proposée ?

Objectifs du TIPE du candidat

1) Implémenter la structure du Rubik's Cube, d'abord en utilisant une approche naïve, puis une méthode plus poussée.

2) Implémenter différents algorithmes de résolutions du Rubik's Cube, en expérimentant notamment au niveau des heuristiques.

3) Comparer les performances obtenues.

Références bibliographiques (ÉTAPE 1)

[1] WIKIPEDIA : Rubik's Cube *: https://en.wikipedia.org/wiki/Rubik%27s_ Cube*

[2] WIKIPEDIA : Iterative deepening A* *: https://en.wikipedia.org/wiki /Iterative_ deepening_ A**

[3] ROKICKI TOMAS, KOCIEMBA HERBERT, DAVIDSON MORLEY, DETHRIDGE JOHN : Rubik's Cube God's Number is 20 *: http://www.cube20.org*

[4] KORF RICHARD : "Finding Optimal Solutions to Rubik's Cube Using Pattern Databases" *: Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on Innovative applications of artificial intelligence (AAAI/IAAI, 1997), pages 700-705*

[5] JAAP'S PUZZLE PAGE : Thistlethwaite's 52-move algorithm *: https://www.jaapsch.net /puzzles/thistle.htm*

[6] KAUR HARPREET : "Algorithms for solving the Rubik's cube" *: Degree project in computer science (KTH Royal Institute of Technology, Stockholm, Sweden, 2015)*

[7] HEIT JOREN : "Building and Solving Rubik's Cube in Mathworks® Matlab®." *: http://joren.ralphdesign.nl/projects/rubiks_ cube/cube.pdf (2011)*

[8] KOCIEMBA HERBERT : The Two-Phase-Algorithm *: http://kociemba.org/cube.htm*

DOT

[1] *: Août 2023 : Choix du sujet après un changement de thème. Premiers essais, algorithme naïf (parcours en largeur), cube 2x2 résolu, cube 3x3 beaucoup trop lent*

[2] *: Septembre - Octobre 2023 : Avancement des recherches, découverte des algorithmes A* et IDA*, de l'heurstique de Korf*

[3] *: Novembre 2023 : Calcul de l'heuristique trop lent, donc choix d'une heuristique moins précise, mais résultats décevants*

[4] *: Décembre 2023 : Analyse des performances, mène à la création d'une fonction de hachage personnalisée au lieu de celle utilisée par défaut*

[5] *: Janvier 2024 : Premiers résultats positifs, stockage plus efficace des valeurs précalculées d'heuristique (valeurs binaires au lieu d'écrire dans un fichier texte)*

[6] *: Février - Mars 2024 : Certains cubes ne peuvent toujours pas être résolus, retour aux recherches, découverte des algorithmes de Thistlethwaite et Kociemba*

[7] *: Avril 2024 : Changement d'implémentation du cube permettant de s'adapter aux algorithmes découverts en février et mars, implémentation des algorithmes : résultats au-delà de mes espérances*

[8] *: Mai 2024 : Réalistation des courbes servant à la présentation orale*