Résoudre de manière *optimale* le Rubik's cube

Frédéric Magniez

Difficultée : difficile

http://www.enseignement.polytechnique.fr/profs/informatique/Frederic.Magniez/index.php?n=Main.PI15

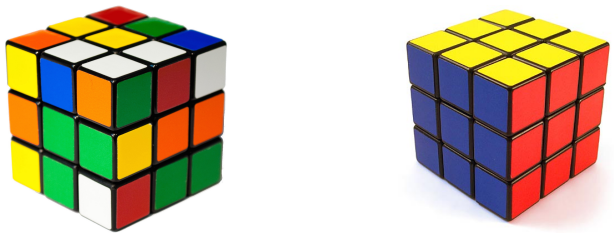

Figure 1: Le Rubik's cube mélangé, à gauche; et résolu, à droite.

1 Objectifs

Le but de ce projet est de recomposer un Rubik's cube mélangé en utilisant le nombre minimal de coups, c'est-à-dire de quart de tours ou de demi-tours. Il ne s'agit pas d'appliquer telle ou telle méthode, mais d'effectuer une recherche, parmi les solutions possibles, d'une solution utilisant un nombre minimal de coups.

Le Rubik's cube est un casse-tête inventé par Erno Rubik à la fin des années 70. La version considérée dans ce projet consiste en un cube $3 \times 3 \times 3$ (voir Figure 1). Ses 6 faces comportent donc chacune 9 facettes et chaque facette est soit de couleur blanche, bleue, jaune, orange, rouge, verte. Dans sa position d'origine, toutes les facettes sont monochromes.

La résolution du Rubik's cube s'effectue par une suite de quart de tours (dans n'importe quel sens), ou de demi-tours comme le montre la Figure 2.

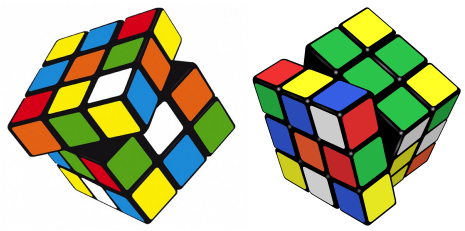

Figure 2: Rotations de face du Rubik's cube

2 Configuration

Une configuration du Rubik's cube peut soit être vue comme une matrice à trois dimensions $3 \times 3 \times 3$ dont les entrées codent les couleurs de la pièce correspondante (0 pour l'intérieur, 1 pour le centre, 2 pour les arêtes, 3 pour les coins), sachant qu'il y a 6 couleurs possibles : blanc, bleu, jaune, orange, rouge, vert.

Question 1. *Ecrire une fonction permettant de saisir une configuration, face par face, et une autre permettant d'afficher une configuration. Un affichage simple ressemblant à celui de la Figure 3 conviendra.*

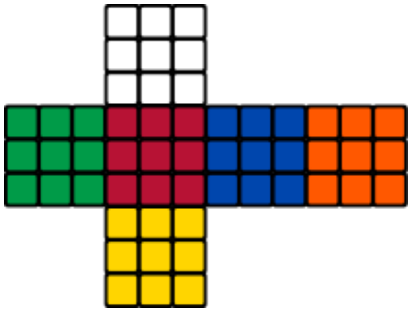

Figure 3: Un affichage simple d'une configuration de Rubik's cube

La première étape consiste à coder les opérations de quart et de demi tours sur une configuration donnée. Chaque opération sera codée par la face à tourner, ainsi que part le tour à effectuer (1/4 dans un des deux sens ou 1/2).

Question 2. *Ecrire une telle fonction.*

3 Recherche d'une solution optimale

L'espace des configuration du Rubik's cube peut être vu comme un graphe où chaque sommet est une configuration, et chaque arête relie deux configurations pouvant être obtenues l'une de l'autre par un quart ou un demi tour. Chaque sommet a donc 18 voisins, et le nombre total de sommets est $8! \times 37 \times 12! \times 210 = 43252003274489856000$.

Question 3. *Justifier ces nombres.*

Une solution minimisant le nombre de courps pourrait donc être en théorie utilisée par une recherche de plus courts chemins. Cependant le graphe est tellement grant qu'il ne peut être représenté dans la mémoire de l'ordinateur. Une stratégie différente va donc être utilisée. Elle est potentiellement plus lente, mais n'utilise que peu d'espace mémoire. Il s'agit de l'algorithme IDA* (iterative-deepening-A*) introduit par R. Korf en 1985.

Pour trouver un chemin entre un sommet s et à un autre sommet t dans un graphe dirigé G, cet algorithme réalise (en première approximation) un parcours en profondeur bornée par une profondeur donnée, qui peut potentiellement revisiter plusieurs fois certains nœuds. La réalisation de cet algorithme repose sur les principes suivants :

1. Tous les chemins (sans boucle) partant de s et de longueurs successives sont étudiés, en considérant d'abord tous les chemins de longueur 1, puis ceux de longueur 2, et ainsi de suite, jusqu'à atteindre une taille où un chemin rejoignant t existe.

2. Le parcours en longueur des chemins de longueur ℓ est effectué en ne retenant que le chemin courant, c'est-à-dire la séquence des arêtes conduisant de t au sommet courant.

3. Une estimation du nombre de coups restants à effectuer est utilisée pour abandonner certaines recherches, lorsque ce nombre est trop grand.

Question 4. *Ecrire une procédure résolvant le Rubik's cube en ne considérant que les deux premiers principes de l'algorithme IDA*. Tester le programme sur des cubes peu mélangés, puis augmenter peu à peu le mélange. Que constatez-vous ?*

Question 5. *Montrer qu'une condition nécessaire pour utiliser une fonction estimant le nombre de coups restants est que cette fonction soit un minorant de ce dernier.*

Question 6. *Ecrire une fonction qui calcule le nombre minimal de coups pour bien placer une pièce donnée, puis une autre qui renvoie le maximum de cette fonction pour toutes les pièces possibles d'une configuration donnée.*

Question 7. *Utiliser la fonction précédente pour écrire une nouvelle procédure résolvant le Rubik's cube en abandonnant certaines recherche lorsqu'une configuration trop éloignée est atteinte. Arrivez-vous à résoudre des cubes plus mélangés ?*

Il y a plusieurs méthodes pour améliorer cette estimation du nombre de coups restant à effectuer. Une méthode originale (encore introduite par R. Korf dans son article *Finding Optimal Solutions to Rubik's Cube Using Pattern Databases*) et performante consiste à utiliser en sous-routine encore l'algorithme IDA* mais pour un problème plus simple. Dans notre cas ce problème consiste à ne mettre que certaines pièces en place.

Question 8. *Ecrire une fonction qui calcule le nombre minimal de coups pour bien placer uniquement certaines pièces :*

1. *Cas 1 : les 8 coins (à 3 faces).*

2. *Cas 2 : 6 arêtes (à 2 faces) parmi les 12 (à choisir)*

3. *Cas 3 : les 6 autres arêtes.*

Créer trois fichiers contenant ces valeurs, afin ne pas avoir à les recalculer à chaque fois.

Question 9. *Pour tronquer votre recherche, utiliser maintenant le maximum des trois valeurs pré-calculées à la question précédente. Que constatez-vous ?*