

Mini-projet d'Intelligence artificielle

José MARTINEZ

2023-2024

A ► LE CUBE DE M. RUBIK ET AUTRES CASSE-TÊTE



Le *Rubik's Cube* est un célèbre jeu des années 1970–1980 à nos jours.¹ Un cube est composé de $3 \times 3 \times 3$ petits cubes aux facettes colorées, comme illustré en figure 1. Il peut être mélangé en faisant pivoter aléatoirement des faces du cube selon ses deux fois trois axes de rotation. (Sur chaque axe, x , y et z , chaque face peut pivoter indépendamment de son opposée. Les « tranches » centrales ne pivotent pas. En effet, les six petits cubes centraux peuvent être considérés comme fixes.) Il convient alors de remettre le cube dans l'unique bon ordre, celui où les six faces du cube présentent une couleur unie, soit comme lorsqu'il a été acheté !

1. Cf. [fr.wikipedia.org/wiki/Rubik's_Cube](https://fr.wikipedia.org/wiki/Rubik%27s_Cube)

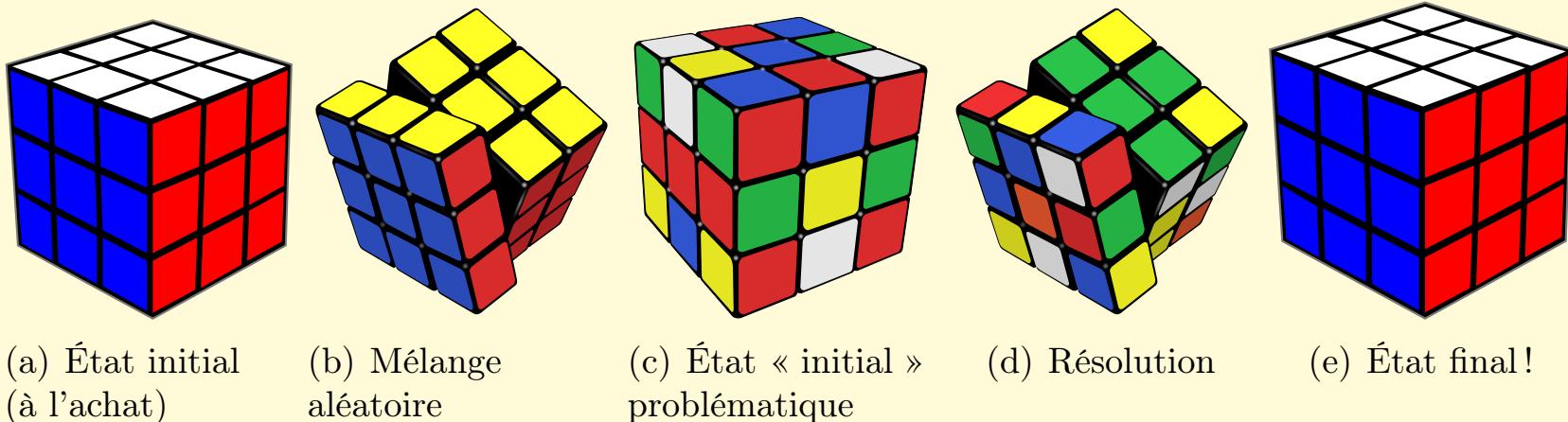


FIGURE 1 – Étapes du jeu

Ce jeu a connu d'innombrables recherches, *aussi bien mathématiques qu'informatiques*, afin de l'étudier et de le résoudre. Un résultat important est qu'aucune résolution ne nécessite plus de vingt rotations² alors que l'espace de recherche est de $4,33 \times 10^{19}$ positions.

D'un point de vue pratique, de nombreux algorithmes ont été mis au point afin de le résoudre manuellement, depuis les méthodes pour débutants, comme la résolution par niveau ou par croix, à des méthodes pour joueurs de compétition.³

2. Tomas ROKICKI, Herbert KOCIEMBA, Morley DAVIDSON, John DETHRIDGE, *The Diameter of the Rubik's Cube Group Is Twenty*, SIAM Journal on Discrete Mathematics, Vol. 27, No. 2, pp. 1082–1105, 2013

3. Le record en date du 12 juin 2023 est de 3,134 s...

Cf. [fr.wikipedia.org/wiki/Liste_des_records_du_monde_en_Rubik's_Cube](https://fr.wikipedia.org/wiki/Liste_des_records_du_monde_en_Rubik%27s_Cube) pour une liste de records pour différents types de cubes et la vidéo du record de cette année pour le cube standard www.youtube.com/watch?app=desktop&v=gh8HX4itF_w.

Ce jeu a connu d'innombrables variations aussi, tout à la fois sur la taille et sur la forme.⁴ Certaines variations ne sont que cosmétiques, d'autres génèrent des difficultés supplémentaires.

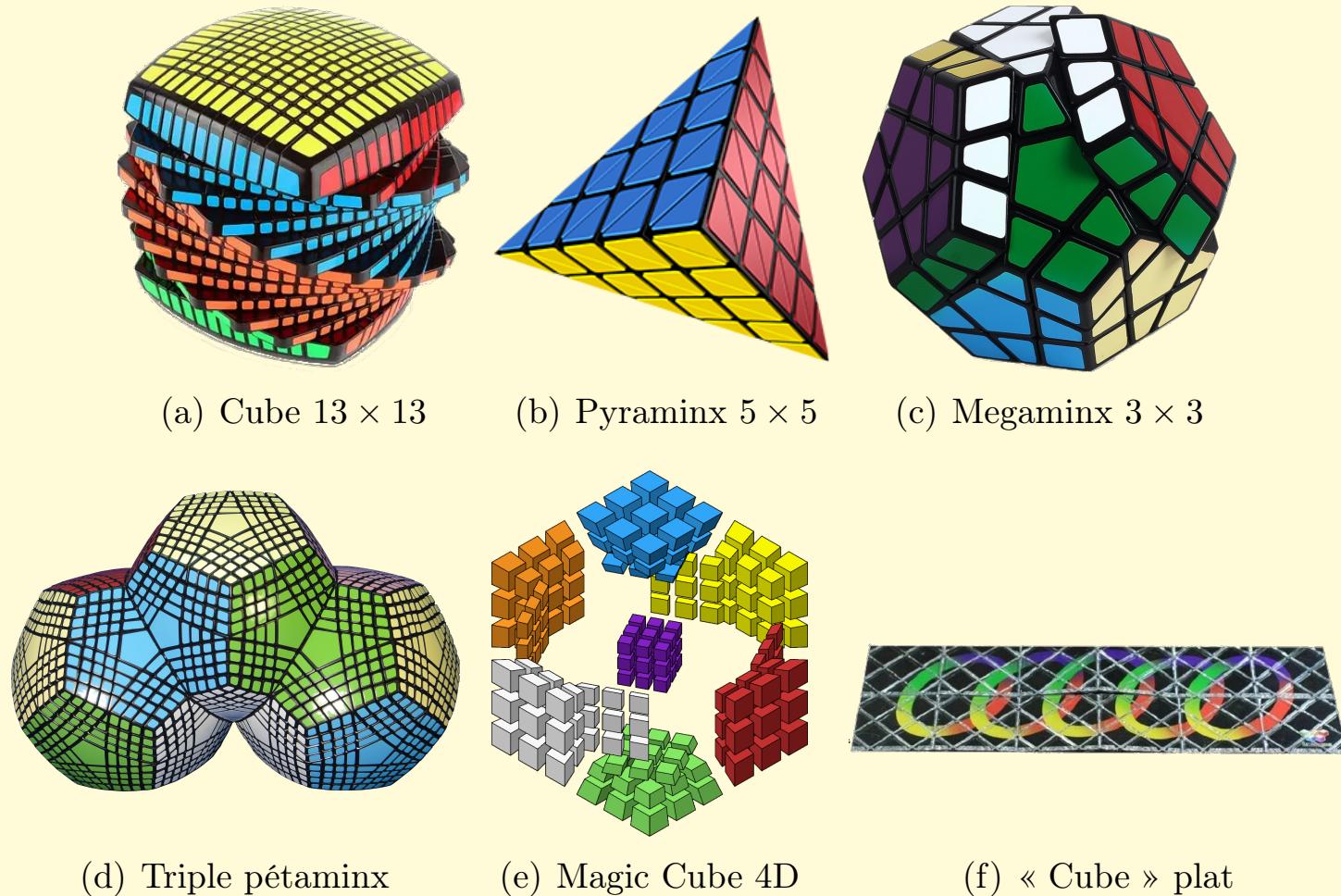


FIGURE 2 – Quelques variations autour du *Rubik's Cube*

4. La version quadri-dimensionnelle peut être pratiquée sur le site hypercubing.xyz/hyperspeedcube/.

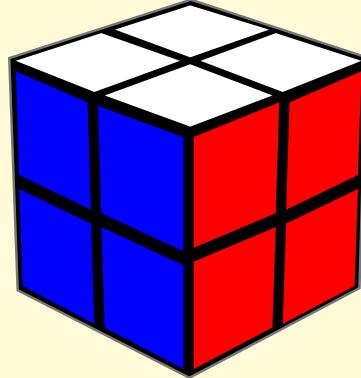


FIGURE 3 – Le plus petit *Rubik’s Cube*, au-delà du gadget 1×1

Partie I

- 1 ► Modélez ce *type* de jeu avec l’approche (E, i, F, O, P, C) .

Remarque. Modéliser au-delà du cube standard permet finalement de *simplifier* le modèle.

- 2 ► Dérivez-en une version particularisée pour le *Rubik’s Cube*, dans sa version 2×2 pour simplifier le problème.

- 3 ► Étudiez la structure du graphe de recherche :

(i) Quelle est sa taille ?

(ii) Est-il symétrique ?

Si oui, quels problèmes cela pose-t-il ?

Si oui, quels avantages cela apporte-t-il ?

Éventuellement, comment peut-on remédier aux problèmes ?

- (iii) Quel est son facteur de branchement ?

- (iv) Est-il dense ?
 - (v) Avec le facteur de branchement, quelle est l'implication sur un *arbre* de recherche ?
- 4 ► Au vu de l'étude de la question 3, choisissez un solveur adapté parmi ceux vus en cours, depuis la version récursive naïve par retours-arrière jusqu'à A^* . Expliquez votre choix en détaillant les avantages et inconvénients de chaque approche ou famille d'approches. Discutez les difficultés restantes. Éventuellement, proposez une « nouvelle » variante.
- 5 ► Y a-t-il des heuristiques applicables à ce (type de) problème, aussi bien en théorie qu'en pratique au vu de leur complexité de mise en œuvre, pour mémoire :
- (i) ordonnancement ;
 - (ii) contraintes ;
 - (iii) équivalences ;
 - (iv) évaluation.

Partie II

- 6 ► Traduisez la modélisation des questions 1 et 2 en code.

Remarque. Pour cette partie et la suite, il est (fortement) conseillé d'utiliser OCaml plutôt que Python :

- Le code compilé fonctionnel est un ordre de magnitude plus rapide que Python.
- La détection d'erreurs de typage est bien plus fiable qu'avec le typage progressif de *typing* et *mypy*.
- Le réusinage de code est bien plus aisé.

Remarque. Dans les deux langages, cela ne représente que quelques dizaines de lignes de code utiles.

Remarque. Vous pourriez alors résoudre des problèmes, mais ne perdez pas de temps à cela.
Ce n'est pas le but de ce mini-projet !

7 ► Proposez une *méthode générale et directe* de résolution de ce type de puzzle.

Remarque. Cet algorithme ne doit pas être l'algorithme parfait. Il doit s'agir « seulement » d'un algorithme qui ne nécessite *aucun* retour-arrière.

8 ► Faites *découvrir* les détails combinatoires à l'ordinateur.

Remarque. Le programme construit à la question 6 va devoir être adapté pour chercher des solutions à de nouveaux sous-problèmes plutôt qu'au problème complet de résolution d'un cube.

9 ► En combinant les réponses des questions 7 et 8, vous avez pour partie écrit, pour partie *appris* un algorithme *direct* de résolution du *Rubik's Cube* 2×2 . Bravo ! C'est lui que vous allez maintenant utiliser pour résoudre n'importe quel problème « instantanément ».

Évaluation

Comme tout problème relevant de l'Intelligence artificielle, ce mini-projet est complexe. Il peut donner lieu à des développements extrêmement poussés, y compris jusqu'au niveau d'un problème de recherche. Il n'est pas demandé de redécouvrir toutes les techniques et algorithmes qui ont été mis au point durant des décennies pour l'analyser et le résoudre. Il est donc nécessaire de procéder par étapes en évaluant les niveaux de complexité de solutions de plus en plus générales, en sélectionnant le niveau adéquat en fonction de vos compétences

et du temps alloué, et en répartissant de manière adéquate et équilibrée la charge de travail entre les deux membres d'un binôme. Il est également conseillé d'anticiper les modifications ultérieures afin de minimiser les changements dans le code de l'étape précédente, et donc les erreurs.

L'ordre correspond à un développement raisonné et progressif depuis l'analyse du problème, l'étude critique et éclairée d'approches de résolution, jusqu'à la mise en œuvre d'une solution adaptée.

Éléments d'appréciation

Les codes sources ne suffisent pas. Les éléments d'appréciation sont fonction de chaque partie :

1. La définition formalisée du problème est évaluée sur :
 - (a) la cohérence et la correction formelle de vos propositions ;
 - (b) la clarté dans les explications et les justifications.
2. La résolution du problème est évaluée sur :
 - (a) les mêmes éléments que précédemment pour la proposition d'algorithme direct ;
 - (b) la clarté de la traduction des algorithmes en code (documentation et tests) ;
 - (c) les performances observées.

Compte-rendus

Le travail à rendre comportera trois parties :

1. Une première partie fournira *une analyse et résolution théorique* du problème (questions 1, 2, 3, 4 et 5 de la partie I). Cette partie est à rendre après la deuxième séance de TP.

Important. Le dépôt sur Madoc devra avoir été effectué au plus tard le
dimanche 15 octobre 2023 à 23 h 59.

Au delà, la note de cette partie, qui compte pour moitié, sera **zéro**.

Cela se justifie par le fait que des indications de solution seront ouvertes sur Madoc pour poursuivre le projet.

2. Le rapport final y ajoutera les extensions et détails de réalisation de la partie II en décrivant les difficultés rencontrées, les solutions proposées et le bon fonctionnement du système.
3. Enfin, les codes sources et des exemples d'utilisation seront aussi déposés sur Madoc. *Les seuls langages de programmation autorisés sont Python et OCaml, voire Haskell.* Chaque fonction devra contenir sa **documentation** : commentaires, précondition et post-condition, et exemples d'utilisation si possible ou nécessaire.
Le code contiendra les **tests automatisés** permettant de vérifier le bon fonctionnement de l'ensemble aussi bien sous Linux que Windows ou encore MacOS.