

CSI 4506

Introduction à l'Intelligence artificielle

Devoir n° 1

Pierre GUILMIN

Rémy TAO

1) Propriétés de l'environnement

L'environnement est totalement observable puisque l'on connaît la position de chacun des nombres à tout moment. Il est statique et déterministe puisque seuls les mouvements joués par le joueur le modifieront, de manière connue. Les états (positions de chaque nombre) sont finis et énumérables donc l'environnement est discret.

2) Performances des algorithmes

a) Sur un puzzle trivial

	Profondeur	Coût	Temps	Nœuds visités	Complet ?	Optimal ?
BFS	1	1	0.0001794179988792166s	1	Oui	Oui
DFS	1	1	0.0002400880039203912s	1	Oui	Oui
A* avec H1	1	1	0.0002638279984239489s	2	Oui	Oui
A* avec H2	1	1	0.00030552499811165035s	2	Oui	Oui

b) Sur un puzzle moyen

	Profondeur	Coût	Temps	Nœuds visités	Complet ?	Optimal ?
BFS	14	14	2.3598618030009675s	2674	Oui	Oui
DFS	/	/	/	/	Non	Non
A* avec H1	14	14	0.1358981860030326s	348	Oui	Oui
A* avec H2	14	14	0.013310654998349492s	76	Oui	Oui

c) Sur un puzzle difficile

	Profondeur	Coût	Temps	Nœuds visités	Complet ?	Optimal ?
BFS	19	19	181.6336577999973s	26491	Oui	Oui
DFS	/	/	/	/	Non	Non
A* avec H1	19	19	10.897252424001636s	4122	Oui	Oui
A* avec H2	19	19	0.46662688099604566s	755	Oui	Oui

3) Heuristique

Les deux heuristiques utilisées sont H1 et H2.

Pour l'heuristique H1, on regarde le nombre de cases qui ne sont pas au bon endroit, et ce nombre définit $h(n)$.

$$h(n) = \text{"nombre de cases au mauvais endroit"}$$

Pour H2, on regarde la distance Manhattan : pour chaque case, on regarde le nombre de fois qu'il faudrait la déplacer pour qu'elle arrive à l'endroit où elle doit être à l'état final, sachant qu'on fait abstraction de la présence des autres cases. On somme ce nombre de coups pour chaque case pour arriver à $h(n)$.

$$h(n) = \sum_{\text{cases de 1 à 8}} \text{"nombre de déplacements nécessaires pour arriver à la position finale"}$$

H1 est admissible car le nombre de coups nécessaires pour atteindre l'état final est supérieur ou égal au nombre de cases à la mauvaise place : pour chaque case à la mauvaise place on doit jouer au moins un coup pour qu'elle soit à la bonne place. H2 est admissible également car pour chaque case à la mauvaise place on jouera en réalité un nombre de coups supérieur ou égal au nombre de coups qu'il faudrait jouer en faisant abstraction des cases aux alentours.

H2 domine H1 car pour tout état n , $H_1(n) \leq H_2(n)$ puisque pour chaque nombre au mauvais endroit le nombre de cases à parcourir pour le mettre au mauvais endroit, même sans tenir compte des autres nombres, est supérieur ou égal à 1.

4) Conclusion

Pour conclure, l'algorithme BFS est globalement performant, c'est le meilleur dans ce problème pour faire de la recherche non informée car il est complet et optimal, et relativement efficace, TANT QUE la solution n'est pas trop profonde (jusqu'à une profondeur de 20-22 le temps de calcul est raisonnable).

Toutefois, plus la profondeur de la solution est élevée et plus les algorithmes à heuristique deviennent nécessaires, notamment pour le eight-puzzle difficile (le numéro 3) et davantage encore pour le puzzle numéro 7, pour lequel nous n'avons pas réussi à obtenir de solution avec BFS au bout de 2h de recherche, alors que A* avec l'heuristique 2 en trouve une (profondeur 24 et 6079 nœuds explorés) en 20.659913s. Le fait qu'un tree search soit implémentée au lieu d'un graph search économise également beaucoup de mémoire et de temps de calcul.

La différence entre nos deux heuristiques est bien visible pour les trois exemples, l'heuristique 2 est toujours plus efficace (attendu car H2 domine H1 comme démontré plus haut) avec une forte réduction du nombre de nœuds explorés.

DFS s'est montré peu efficace, en tournant pendant trop longtemps sur les puzzles moyens et difficile (nous avons arrêté l'algorithme dès qu'il atteignait une profondeur trop importante). Il se révèle ici incompetent, peut-être parce qu'il y a PEU de moyens d'arriver à la solution, notamment par une voie non optimale. Toutefois, sur le puzzle numéro 7 DFS trouve (miraculeusement), une solution en 1.59s. Solution non optimale avec une profondeur de 1728 (i.e 1728 coups) mais solution néanmoins, qui n'avait pas été trouvée par BFS en 2h de calcul.

L'algorithme à privilégier est finalement sans aucun doute, sur tous les plans, l'algorithme A* avec l'heuristique 2 : il est robuste, fiable et rapide.