

# Jeu du taquin

L3 CILS, L3 BI, DL3

Le projet est à réaliser en binôme. Le travail sera évalué par 1) un rapport et 2) une soutenance. Pour les préparer, il est judicieux de prendre des notes pendant la conception, l'implémentation, les tests, et les expérimentations de vos programmes.

## Problème

Un taquin  $3 \times 3$  est un puzzle carré, composé de 8 nombres et d'un trou. Le jeu consiste à déplacer le trou au nord, au sud, à l'est ou à l'ouest à partir d'un état initial jusqu'à un état final donné, en limitant le nombre de coups.

1	3	X
5	7	6
4	2	0

Etat  $E$

0	1	2
3	4	5
6	7	X

Etat final

## Algorithme A\*

On part sur l'idée d'utiliser une heuristique. Soit  $E$  un état.

- On note  $g(E)$  la longueur du plus court chemin menant de l'état initial à  $E$ . C'est le plus petit nombre de déplacements du trou permettant de passer de l'état initial à  $E$ .
- L'heuristique  $h(E)$  est une estimation du nombre de déplacements nécessaires pour passer de  $E$  à l'état final. Plusieurs heuristiques seront définies ci-après, et l'un des objectifs du projet est de comparer leurs performances.
- La fonction d'évaluation est définie par  $f(E) = g(E) + h(E)$ .

Plusieurs choix d'implémentation de  $A^*$  doivent être faits.

Concernant les états, il faut une structure de données permettant de stocker la configuration du plateau, mais aussi le plus court chemin qui a permis de le découvrir et sa longueur. Une façon de faire est d'attribuer une adresse à chaque état, et de conserver pour chaque état l'adresse de son père ainsi que la longueur du chemin depuis l'état initial (= valeur de  $g(\cdot)$ ). Une autre façon de faire est de conserver pour chaque état le chemin sous la forme d'une chaîne de caractère, qui code les mouvements du trou pour obtenir l'état à partir de l'état initial (par exemple "ONN" pour ouest, puis nord, puis nord) ; dans ce cas, on a directement la longueur du chemin grâce à la longueur de la chaîne.

Concernant maintenant l'ensemble frontière, il faut une file de priorité d'états, ordonnée par valeur croissante de  $f(\cdot)$ . La question des états redondants au sein de cet ensemble peut se poser ; on rappelle qu'un même état peut être obtenu en suivant des chemins différents, et que

seul le plus court d'entre eux est vraiment intéressant. Choisir d'éliminer cette redondance des états permet de diminuer la taille de l'ensemble frontière, mais consomme du temps. Enfin, l'ensemble exploré permet de réduire la taille de l'ensemble frontière, en éliminant également une partie des états redondants. De nouveau, la question se pose de savoir s'il faut s'en servir : la gestion de cet ensemble nécessite du temps de calcul mais permet de gagner en mémoire. En termes de structures de données, il ne faut pas utiliser une simple liste pour l'ensemble exploré, mais un *arbre binaire de recherche* par exemple, ou une table de hachage.

## Les heuristiques

On introduit six heuristiques  $h_1(\cdot), \dots, h_6(\cdot)$ . Toutes sont définies comme des *distances de Manhattan pondérées réduites* entre l'état courant  $E$  et l'état final : pour  $1 \leq k \leq 6$ ,

$$h_k(E) = \left( \sum_{i=0}^8 \pi_k(i) \times \varepsilon_E(i) \right) \text{ div } \rho_k.$$

Concernant les différents éléments de cette définition :

1. La distance élémentaire  $\varepsilon_E(i)$  d'une tuile  $i$  d'un état  $E$  est le nombre de déplacements élémentaires qu'il faut faire pour la mettre à sa position dans l'état final. Ainsi, avec l'exemple dessiné en page 1, on a  $\varepsilon_E(1) = 1, \varepsilon_E(3) = 2, \varepsilon_E(6) = 3, \varepsilon_E(0) = 4, \dots$
2. Pour chaque heuristique, on donne un poids  $\pi_k$  plus important à certaines tuiles qu'à d'autres. On utilisera les jeux de poids suivants :

Tuile	0	1	2	3	4	5	6	7	X
$\pi_1$	36	12	12	4	1	1	4	1	0
$\pi_2 = \pi_3$	8	7	6	5	4	3	2	1	0
$\pi_4 = \pi_5$	8	7	6	5	3	2	4	1	0
$\pi_6$	1	1	1	1	1	1	1	1	0

3. On réduit l'impact du jeu de poids en utilisant des coefficients de normalisation ; on pose  $\rho_1 = \rho_3 = \rho_5 = 4$  et  $\rho_2 = \rho_4 = \rho_6 = 1$ . div dénote le quotient de la division entière.

## Travaux à réaliser, travaux d'expérimentation

Vos programmes doivent être en Python, votre rapport en L<sup>A</sup>T<sub>E</sub>X, votre soutenance accompagnée d'une série de slides en Beamer ou Powerpoint.

Il est intéressant d'étudier les performances de votre implantation en fonction de l'état initial choisi et de la distance utilisée. De bons critères de comparaison sont le nombre d'états sortis de la frontière, le nombre d'états explorés, la longueur de la solution, le temps CPU, etc.

Attention : comme pour le Rubik's Cube, seulement la moitié des états initiaux permettent d'atteindre l'état final. Il faut creuser cette question.

Au delà, il peut être intéressant d'étudier les taquins 4\*4, ou N\*N, ou bien faire une interface graphique ... vos idées sont les bienvenues ! Mais attention : mieux vaut un programme modeste qui fonctionne que pas de programme.