

# Résolution de problèmes à l'aide de graphes d'états

## PARTIE 1 - MODELISATION D'UN PROBLEME DE PLANIFICATION

### TAQUIN

Le taquin est une sorte de puzzle où il faut remettre les nombres dans l'ordre. Il est formé d'une grille de  $N \times N$  cases où sont placées  $N^2-1$  tuiles étiquetées par les nombres 1 à  $N^2-1$ , une des cases restant vide. Une des tuiles situées à côté de la case vide peut être déplacée vers cette case. Les déplacements se font donc verticalement ou horizontalement, mais pas en diagonale.

Le taquin est souvent utilisé pour tester les algorithmes de recherche. En augmentant la taille de la grille, les problèmes deviennent de plus en plus complexes. Les algorithmes d'aujourd'hui arrivent à résoudre les taquins  $3 \times 3$  et  $4 \times 4$  (qui ont des espaces d'états respectivement de taille 181 440 et d'environ 1,3 milliards), mais les instances du taquin  $5 \times 5$  (avec un espace d'états de taille  $10^{25}$ ) restent difficiles...

Nous souhaitons résoudre un taquin de taille  $3 \times 3$ , où les tuiles sont numérotées de 1 à 8. Les états initial et final sont donnés sur les figures ci-dessous.

2	8	3
1	6	4
7		5

État initial

1	2	3
8		4
7	6	5

État final

**Q1 : Modélisez ce problème selon le canevas présenté en cours :**

Pour rappel, le canevas de modélisation d'un problème est :

{ États / État initial / Opérateurs (action, condition d'application et fonction de successeur) / Test de but / Fonction de coût (simple ou avec heuristique) }

**Q2 : Dessinez les 3 premiers niveaux du graphe d'états.**

**Q3 : Proposez une heuristique permettant de guider la recherche.**

==== Indices de correction =====

**Modélisation n°1 de ce problème :**

**États :** Les états sont des configurations des huit tuiles numérotées de 1 à 8 et de la tuile vide dans les neuf cases de la grille  $3 \times 3$ .

**État initial :** N'importe quel état pourrait être choisi comme l'état initial, on prend celui de l'énoncé.

Opérateur : Il y a 4 actions possibles correspondant aux quatre façons de **changer la position du carré vide** : haut, bas, gauche, droite. Dans certaines configurations, il n'y aura que 2 ou 3 actions possibles (définir les conditions d'application correspondantes).

Fonction de successeur : Cette fonction spécifie les états résultants des différentes actions. Par exemple, la fonction va nous dire que l'exécution de l'action droite dans l'état de la figure gauche ci-dessous produira l'état de la figure droite ci-dessous.

7		4
5	2	6
8	3	1

7	4	
5	2	6
8	3	1

FIG. 1 - Deux états du jeu de taquin.

Test de but : L'état but est unique et fixé au début du jeu (n'importe quel état peut être choisi comme état but, même si en pratique il s'agit de remettre les nombres dans l'ordre).

Coût des actions : Chaque déplacement d'une tuile a coût de 1 afin de trouver une solution avec le moins de déplacements.

**Cette première modélisation n'est pas assez formalisée pour pouvoir être implémentée sans ambiguïté... il faut donc reprendre tout cela et être plus formel et plus précis comme dans la modélisation n°2.**

### Modélisation n°2 de ce problème :

États : 8 variables (une par élément du taquin) avec pour valeur le couple (i,j) de ses coordonnées sur le tableau et une variable pour le trou (VE), où  $i \in [1 ; 3]$  et  $j \in [1 ; 3]$

	1	2	3
1	2	8	3
2	1	6	4
3	7		5

Sur le cas présenté dans l'énoncé :

État initial :  $V1=(1,2)$ ,  $V2=(1,1)$ ,  $V3=(3,1)$ ,  $V4=(3,2)$ ,  $V5=(3,3)$ ,  $V6=(2,2)$ ,  $V7=(1,3)$ ,  $V8=(2,1)$ ,  $VE=(2,3)$

État but :  $V1=(1,1)$ ,  $V2=(2,1)$ ,  $V3=(3,1)$ ,  $V4=(3,2)$ ,  $V5=(3,3)$ ,  $V6=(2,3)$ ,  $V7=(1,3)$ ,  $V8=(2,3)$ ,  $VE=(2,2)$

Opérateurs : (il existe d'autres possibilités, l'important est que les étudiants aient compris la notion d'opérateurs)

Les opérateurs les plus simples consistent à **déplacer le « trou »**

H : déplacer vers le haut le trou. Condition : le trou ne doit pas être sur la ligne 1. Dans l'état, c'est la variable représentant la case juste au-dessus qui change de valeur. Plus formellement :

Condition :  $VE=(C,L)$  avec  $L > 1$  ( $C =$  colonne,  $L =$  ligne)

Modification de l'état :

-  $VE$  [qui était égal à  $(C,L)$ ]  $\leftarrow VE=(C,L-1)$  [le « vide » monte d'une ligne, ne change pas de colonne] ;

-  $Vk$  [qui est la pièce qui descend, avait pour valeur  $(C,L-1)$ ]  $\leftarrow Vk=(C,L)$  [elle descend d'une ligne, ne change pas de colonne]

(les autres variables ne changent pas)

B : déplacer vers le bas le trou : ....

D : déplacer vers la droite le trou : ....

G : déplacer vers la gauche le trou : .....

Sans heuristique, il faut explorer tout l'espace des états

A chaque état, on peut (ou non) appliquer un opérateur et on recommence avec l'état obtenu. Si l'état final est obtenu, on a satisfait le but et la solution est le chemin constitué par la succession des opérateurs à partir de l'état initial. Si on veut trouver la solution optimale, il faut continuer à explorer l'espace des états et c'est le chemin le plus court qui est optimal.

Différentes heuristiques :

### **Heuristique W : nombre de cases non encore à la bonne place**

Cette heuristique est minorante et monotone. Elle consiste à prendre  $h(e)$  le nombre de cases qui ne sont pas, dans l'état  $e$ , à la place exigée par l'état but  $b$ . Cette heuristique est simple mais elle ne tient pas compte du nombre de coups pour amener une case pleine à sa destination dans l'état but  $b$  selon qu'elle en est plus ou moins éloignée.

### **Heuristique P : somme des distances « en pâté de maison »**

On utilise donc la distance « pâté de maison » (en anglais « city-block » ou « Manhattan » distance). Cette heuristique est également minorante et monotone ; elle n'est pas formellement mieux informée que  $W$  mais l'expérience confirme qu'elle est beaucoup plus efficace. Si  $Le(c)$  et  $Ce(c)$  (resp.  $Lb(c)$  et  $Cb(c)$ ) désignent les numéros de la ligne et de la colonne de la case  $c$  dans l'état  $e$  (resp. dans l'état but), on définit :  $d(c, n, b) = |Le(c) - Lb(c)| + |Ce(c) - Cb(c)|$

Alors  $h(n)$  est égal à la somme des distance  $d(x, e, b)$  des cases pleines.  $P$  est plus fine que  $W$ , mais ne tient pas compte de la difficulté à inverser deux cases voisines.

### **Heuristique P + 3S**

Contrairement à  $W$  et  $P$ , cette heuristique ne peut s'utiliser que dans le cas du taquin  $3 \times 3$ . Pour un état  $e$ , on définit d'abord  $S$  de la façon suivante :

- On choisit un sens de parcours des cases non centrales autour de la case centrale. On ne considère que les cases non centrales différentes de la case vide. A chacune d'entre elles, on attribue :

- Un poids de 0 si elle est aussi non centrale dans l'état but  $b$  et si dans les deux états ( $e$  et  $b$ ) la case suivante (dans le sens de parcours choisi) est la même.

- 2 sinon.

- A la case centrale, si elle est pleine dans l'état  $e$ , on attribue :

- 0 si elle est bien placée par rapport à  $b$  ; 1 sinon.

On a donc  $S(e)$  = somme des poids des cases pleines dans l'état  $e$  et  $h(e) = P(e) + 3 * S(e)$ .

---

---