UFR des Sciences

CM 08 - Autre Paradigme
L2-INFO et L2-MATH

Lundi 07 Mars 2022

CM 08 - Recherches arborescentes : Résolution du taquin

1. Présentation du problème

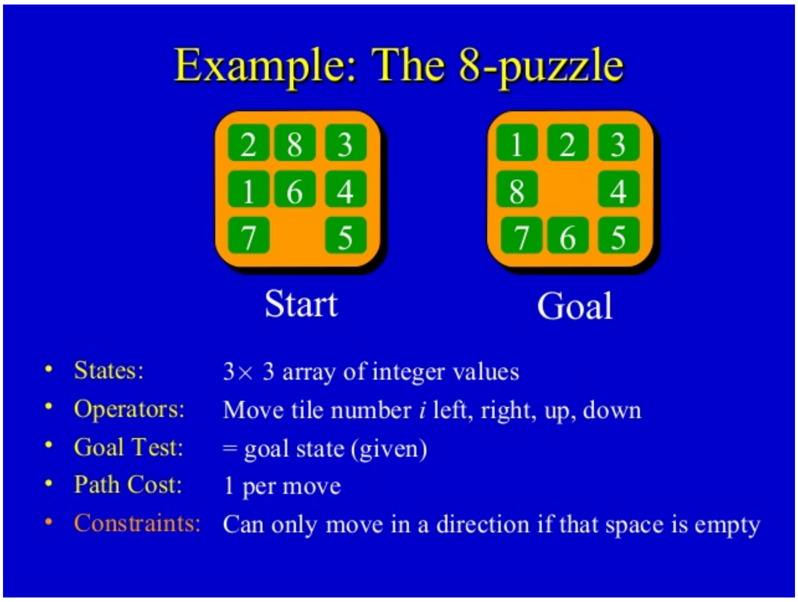

Figure 1

Exemple : succession de déplacements pour aller de l'état initial e (Start) à l'état final e_f (Goal) sur la Figure 1 ci-dessus. Pour cela, on effectue 5 déplacements :

2 8 3		2 8 3		2 _ 3		_ **2** 3		**1** 2 3		1 2 3
1 6 4	=>	1 _ 4	=>	1 **8** 4	=>	1 8 4	=>	_ 8 4	=>	**8** _ 4
7 _ 5		7 **6** 5		7 6 5		7 6 5		7 6 5		7 6 5

2. Modélisation du problème sous forme d'un ensemble d'états et de transitions entre états

Données : - un état initial : **e**

- un état terminal : **e_f**

- une fonction **successeurs**, qui à chaque état, associe l'ensemble de ses successeurs

Le passage d'un état à son successeur constitue une ***transition***. Le problème consiste à trouver un chemin (par applications successives de la fonction **successeurs**) allant de l'état initial **e** à l'état terminal **e_f**.

NB. Le problème consiste à déterminer un chemin, entre l'état initial **e** et l'état terminal **e_f**, mais ***pas forcément le chemin le plus court***, i.e. celui où on fait le minimum de transitions (déplacements).

3. Construction de l'arbre de recherche associé

- **racine** : l'état initial **e**
- **feuille** associée à un succès : l'état terminal **e_f**
- **nœuds intermédiaires** (associés aux états intermédiaires)
    - soit s un état ; **successeurs(s)** est l'ensemble des successeurs de l'état **s**
    - chaque nœud (état) s_i appartenant à successeurs (s) est un fils du nœud s

La **profondeur** de la racine sera 0. Si la profondeur d'un état s vaut i, alors la profondeur de chaque successeur de s vaudra (i+1).

Ci-dessous un extrait de l'arbre de recherche permettant d'aller de l'état initial **e** à l'état final **e_f**.

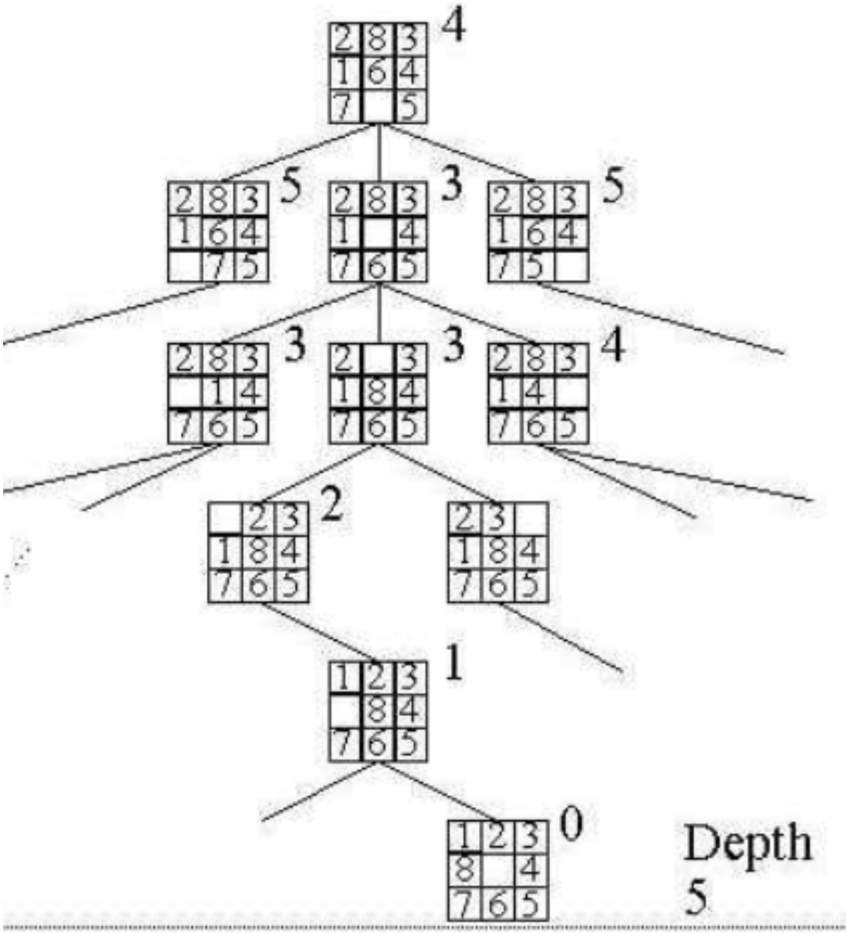

Figure 2

4. Différentes méthodes permettent de parcourir un arbre de recherche avec plus ou moins d'intelligence et d'efficacité.
Dans ce DM, on en considère 3 :

1. Parcours en largeur d'abord : le frère droit d'abord (ou encore parcours niveau par niveau)
2. Parcours en profondeur d'abord : le fils le plus à gauche d'abord (on descend dans l'arbre)
3. Parcours en meilleur d'abord : parcours plus intelligents où l'on essaie de déterminer, à chaque nœud, le(s) meilleur(s) fils à développer en premier

5. Arbre de recherche

Soit s un état ; on souhaite définir la fonction successeurs(s). Il y a 3 types de situations selon la place du carré vide dans l'état s :

a. le carré vide est au milieu → s possède 4 successeurs
b. le carré vide est dans un coin → s possède 2 successeurs
c. le carré vide est sur un bord, mais pas dans un coin → s possède 3 successeurs

On cherche à estimer *nb* qui est égal au nombre de nœuds de l'arbre de recherche à la profondeur *n* (i.e. après avoir effectué *n* déplacements). A la profondeur *n*, on a l'encadrement suivant : $2^n \leq nb \leq 4^n$
En effet, chaque nœud possède au moins 2 successeurs et au plus 4 successeurs.

→ On constate donc que la taille de l'arbre de recherche est exponentielle par rapport au nombre *n* de déplacements.

6. Parcours en meilleur d'abord

On souhaite définir des parcours plus intelligents en essayant de déterminer, à chaque nœud, le meilleur nœud à développer en premier. Pour cela, on essaye d'estimer le coût (fonction *h*) pour aller de l'état courant *s* à l'état terminal *e_f.* On traite alors les nœuds selon les ***valeurs croissantes*** de la fonction *h.*

Pour cela, on fait la ***somme sur tous les carrés (sauf la case vide)*** des distances entre la place courante du carré *i* et sa place finale (i.e. sa place dans l'état *e_f*).

Prenons pour exemple l'état *e*

```
e = 2 8 3      e_f = 1 2 3          les quatre carrés 2, 8, 1, 6 ne sont pas à leur place finale
    1 6 4            8   4
    7   5            7 6 5
```

Deux distances sont généralement utilisées :

- La distance de HAMMING qui vaut 1 si le carré i est bien placé, 0 sinon
- La distance MANHATTAN qui vaut le nombre de déplacements horizontaux + le nombre de déplacements verticaux qu'il faudrait faire pour amener le carré i à sa place finale

Exemple 1. Distances

En utilisant la distance de HAMMING, on a *h(e) = 4* car il y a 4 carrés mal placés.

En utilisant la distance MANHATTAN on a *h(e) = 5*
- pour amener le carré 2 à sa place, il faut 1 déplacement horizontal + 0 vertical → distance = 1+0 = 1
- pour amener le carré 8 à sa place, il faut 1 déplacement horizontal + 1 vertical → distance = 1+1 = 2
- pour amener le carré 1 à sa place, il faut 0 déplacement horizontal + 1 vertical → distance = 0+1 = 1
- pour amener le carré 6 à sa place, il faut 0 déplacement horizontal + 1 vertical → distance = 0+1 = 1

Donc *h(e) = 1+2+1+1 = 5*

Exemple 2. Parcours en meilleur d'abord

Sur la Figure 2, chaque nœud *s* de l'arbre est annoté par sa valeur *h(s)* utilisant la distance de HAMMING.

La Figure 2 illustre le parcours en meilleur d'abord.
→ on part de e qui est l'état initial (profondeur 0). On a *h(e)=4*
- puis on choisit son fils de coût le plus petit, i.e. de coût 3 (profondeur 1)
- puis on choisit l'un des 2 fils de coût 3 (en cas d'égalité sur le min, on en choisit arbitrairement un)
- puis on choisit son fils de coût le plus petit, i.e. de coût 2 (profondeur 3)
- puis on choisit son fils de coût le plus petit, i.e. de coût 1 (profondeur 4)
- puis on choisit son fils de coût le plus petit, i.e. de coût 0 (profondeur 5)

→ le parcours s'arrête car on a obtenu l'état final e_f