# **Master Informatique 1**

#### Module BIA

#### **TD 1**

# Représentation et résolution de problème (1)

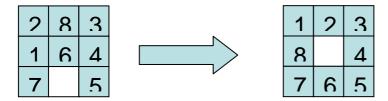
### Luisa Iturrioz, Alain Mille

#### 1 Introduction

Le TD est prévu sur 3 tiers-temps avec des passages d'étudiants au tableau. Ce document ne donne donc que le sujet.

# 2 Résolution de problème par exploration d'espace d'états

### 2.1 Représentation du problème du Taquin



### Etat initial

Etat But

Proposer une représentation d'un état du taquin. Proposer des opérateurs pour passer d'un état à un autre. Voyez-vous une heuristique permettant de ne pas explorer l'ensemble des états ?

# 2.2 Mise en œuvre de l'algorithme A\*

L'algorithme A\* qui correspond à ce qui a été décrit en cours est le suivant~ :

```
Algorithme A*
1.
        Initialisation : OUVERTS \leftarrow \{u_0\}; FERMES \leftarrow \emptyset; g(u_0) \leftarrow 0; u \leftarrow u_0
2.
        Itérer tant que [OUVERTS Ø et u non terminal]
2.1
                Supprimer u de OUVERTS et le mettre dans FERMES
22
                Itérer sur les nœuds v successeurs de u
                        Si [v \notin (OUVERTS \cup FERMES) \text{ ou } g(v) > g(u) + coût(u,v)] Alors faire :
                               g(v) \leftarrow g(u) + coût(u, v)
                               f(v) \leftarrow g(v) + h(v)
                               père(v) \leftarrow u
                               Ranger v dans OUVERTS, dans l'ordre f croissant, puis g décroissant
                Fin Itération 2.2
2.3.
                Si OUVERTS Ø Alors u ← tête(OUVERTS)
                Fin itération 2
3.
        Si OUVERTS = \emptyset
                               Alors le problème n'admet pas de solution
                               Sinon fournir la solution chemin(u)
```

Comprendre et appliquer cet algorithme au problème du taquin tel qu'il est posé dans ce document avec les différentes heuristiques. Il s'agit donc de faire « tourner à la main » l'algorithme en traçant les différentes structures et variables utilisées.

### 2.3 Résolution par décomposition de problèmes

Rappel : la décomposition d'un problème en sous-problèmes plus simples est un principe applicable à des problèmes modélisables de manière récursive, mais pas seulement !

Un algorithme permettant de faire une recherche dans un arbre ET/OU (aussi appelé hypergraphe dans la littérature) issu de la décomposition d'un problème est le suivant :

```
BSH(u) Backtrack Search dans un Hypergraphe
       Si u terminal Alors Retourner « Succès »
1
2.
       Si aucune règle de décomposition n'est applicable en u ou si rg(u) > Seuil
       Alors Retourner « Echec »
3
       Itérer sur les règles de décomposition i applicables en u
3.1.
          Flag ← vrai
3.2.
          Tant que Flag, itérer sur les nœuds v, successeurs de u en lesquels la règle i décompose u
               Si v ∉ RESOLUS Alors faire :
                      Si v \in INSOLUBLES Alors Flag \leftarrow faux
                      Sinon faire:
                             rg(v) \leftarrow rg(u) + 1
                             Si BSH(v) = « Echec » Alors faire :
                                    Mettre v dans INSOLUBLES
                                    Flag ← faux
                             Sinon mettre v dans RESOLUS
              Fin Itération 3.2
3.3
          Si Flag Alors faire:
              Mettre u dans RESOLUS
              règle(u) \leftarrow i
              Retourner « Succès »
          Fin Itération 3
4.
       Mettre u dans INSOLUBLES
5.
       Retourner « Echec »
```

Comprendre et appliquer cet algorithme à un problème qui serait décomposé selon les règles de décomposition suivantes :

```
R1: d \rightarrow g, h

R2: d \rightarrow a, e, f

R3: d \rightarrow a, k

R4: f \rightarrow i

R5: f \rightarrow c, j

R6: a \rightarrow b, c

R7: k \rightarrow e, 1

Les problèmes terminaux sont: b, c, e, l

Le problème à résoudre est: d

Considérer les règles dans l'ordre croissant et les sous-problèmes de « gauche à droite ».
```

Faire « tourner à la main » l'algorithme en traçant les structures et variables importantes.