# Master Informatique 1

# Module BIA, TD 1

# Représentation et résolution de problème (1)

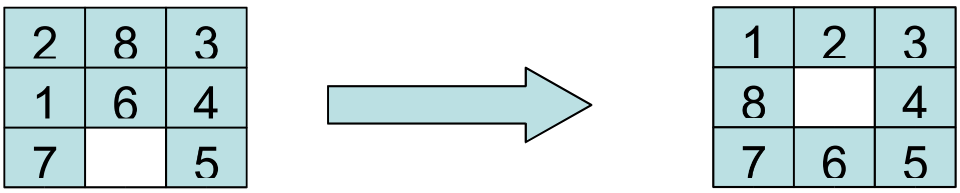
**Luisa Iturrioz, Alain Mille**

## 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 Recherche heuristique : mise en œuvre de l'algorithme A**

L'algorithme A* qui correspond à ce qui a été décrit en cours est le suivant : (A* est un algorithme de type A avec une heuristique h minorante, i.e. pour tout u, $h(u) \leq h^*(u)$, où $h^*(u)$ est le coût d'un chemin optimal - s'il existe, sinon $h^*(u) = +\infty$ - joignant l'état u à un état but)

---

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 $\neq \emptyset$ et u non terminal]

2.1 Supprimer u de OUVERTS et le mettre dans FERMES

2.2 Itérer sur les nœuds v successeurs de u

  Si [$v \notin$ (OUVERTS $\cup$ FERMES) ou $g(v) > g(u) + \text{coût}(u, v)$] Alors faire :

    $g(v) \leftarrow g(u) + \text{coût}(u, v)$

    $f(v) \leftarrow g(v) + h(v)$

    père(v) $\leftarrow$ u

    Ranger v dans OUVERTS, dans l'ordre f croissant, puis g décroissant

  Fin Itération 2.2

2.3 Si OUVERTS $\neq \emptyset$ Alors u $\leftarrow$ 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 de recherche aveugle permettant de faire une recherche dans un graphe ET/OU (hypergraphe particulier) **sans circuit** issu de la décomposition d'un problème est le suivant : (en absence de coût et pour borner l'espace exploré, on utilise un majorant sur le rang des états, noté rg(u) ; BSH retourne « Echec » si le rang est supérieur à un Seuil)

```
BSH(u) Backtrack Search dans un Hypergraphe
1.      Si u terminal Alors Retourner « Succès »
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 ∈ INSOLUBLES Alors Flag ← faux
                        Sinon faire :
                                rg(v) ← 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) ← i
                Retourner « Succès »
            Fin Itération 3
4.      Mettre u dans INSOLUBLES
5.      Retourner « Echec »
```

Dans cet algorithme, RESOLUS est l'ensemble 1) des états terminaux, et 2) des états u tels qu'il existe un connecteur $S_i(u)$ dont tous les successeurs v sont dans RESOLUS ; INSOLUBLES est l'ensemble 1) des états non terminaux sans successeur, et 2) des états u tels que pour tout connecteur $S_i(u)$ il existe un successeur v qui est dans INSOLUBLE.

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

| | |
|---|---|
| R1 : d → g, h<br>R2 : d → a, e, f<br>R3 : d → a, k<br>R4 : f → i<br>R5 : f → c, j<br>R6 : a → b, c<br>R7 : k → e, l | Les problèmes terminaux sont : b,c,e,l<br><br>Le problème à résoudre est : d<br><br>Considérer les règles dans l'ordre croissant et les sous-problèmes de « gauche à droite ».<br><br>Dessiner le graphe/arbre de décomposition. |

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