

# Introduction

## à l'intelligence artificielle

### (recherche dans les graphes)

Antoine Cornuéjols

INAPG

antoine.cornuejols@agroparistech.fr

<http://www.lri.fr/~antoine>

Cours IA

## Plan

- 1- Introduction à l'IA
- 2- Méthodes de résolution de problèmes**
- 3- Le raisonnement
- 4- La représentation des connaissances
- 5- L'IA distribuée
- 6- Apprentissage artificiel : introduction, EV, réseaux de neurones
- 7- AA : arbres de décision, validation, boosting
- 8- AA : sélection d'attributs, apprentissage non supervisé
- 9- AA : apprentissage par renforcement

Cours IA (A. Cornuéjols)

3/88

## Ouvrages conseillés

- o S. Russell & P. Norvig : **Artificial Intelligence: A modern approach** (2nd ed.). Prentice Hall, 2003 (Trad. française : « Intelligence artificielle », Pearson Education, 2006, 1184 p.)
- o Nilsson N. (98) : **Artificial Intelligence : A new synthesis**. Morgan Kaufmann, 1998.
- o E. Rich & K. Knight : **Artificial Intelligence**. McGraw-Hill, 1991
- o I. Millington : **Artificial Intelligence for Games**. Morgan Kaufmann, 2006.

Cours IA (A. Cornuéjols)

2/88

## 2. Recherche dans les graphes : plan

- o **Quels problèmes ? Quelles résolutions ?**
- o **Notion de graphe de recherche**
- o **Techniques de recherche non informée**
  - o En largeur d'abord
  - o En profondeur d'abord
  - o En profondeur itérative
- o **Techniques de recherche informée**
  - o En meilleur d'abord
  - o **A\***
- o **Graphes ET/OU**
- o **Techniques par satisfaction de contraintes**

Cours IA (A. Cornuéjols)

4/88

## 2. Spécification de problèmes (1)

### o Quels problèmes ?

- Problèmes de classes primaires
- Démonstration de théorèmes
- Sortir avec son petit ami et réviser pour le contrôle du lendemain
- Convaincre un interlocuteur
- Choix d'un circuit optimal pour le prochain voyage en Indonésie
- Reconnaître à l'aéroport quelqu'un que l'on n'a jamais vu
- ...

### o Démarche générale

1. Passage d'un énoncé informel à une spécification précise
2. Recherche d'une solution à l'intérieur du cadre défini par ces spécifications

## 2. Spécification de problèmes : types d'énoncés (2)

### ▫ Énoncés de type combinatoire

Trouver dans un ensemble (espace)  $X$  donné, les éléments (points)  $x$  satisfaisant un ensemble de contraintes  $K(x)$

Ex : Pb des 8 reines, cryptarithmique (SEND + MORE = MONEY)

### ▫ Énoncés avec opérateurs de changement d'états

À partir d'un état initial donné, d'un critère objectif et d'un ensemble d'opérateurs de changement d'états, trouver une suite d'opérateurs permettant de passer de l'état initial à un état objectif

Ex : Pb des missionnaires et des cannibales, les tours de Hanoï, le jeu du taquin

### ▫ Énoncés avec opérateurs de décomposition de pbs en ss-pbs

Étant donné un problème (ou but), des opérateurs de décomposition du pb en ss-pbs, des problèmes dits primitifs (dont on connaît immédiatement la solution), trouver des opérateurs à appliquer pour décomposer le problème initial en un ensemble de ss-pbs primitifs

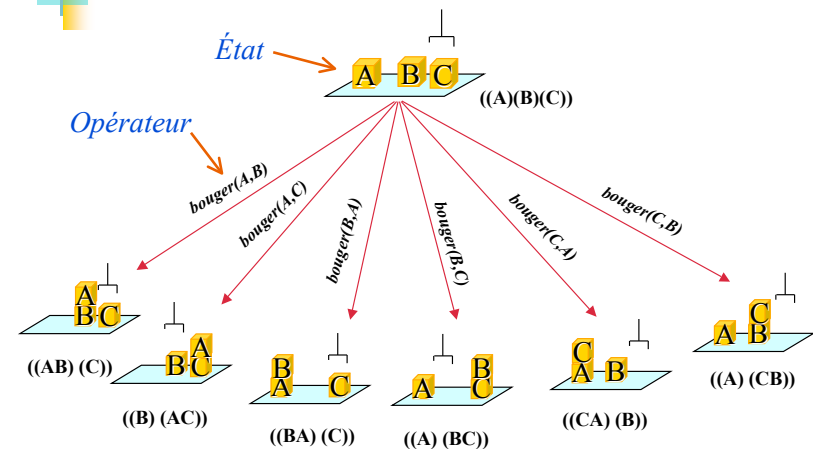
Ex : Problèmes de planification, Tours de Hanoï, intégration symbolique

## 2. Résolution de problèmes

### Démarche générale :

1. Trouver une bonne **représentation** du problème
2. Trouver des **opérateurs** pour manipuler cette représentation
3. Effectuer un **contrôle de stratégie**

## 2. Notion de graphe de recherche (1)



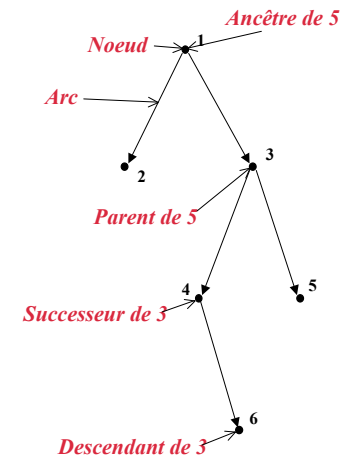
## 2. Notion de graphe de recherche (2)

On suppose que :

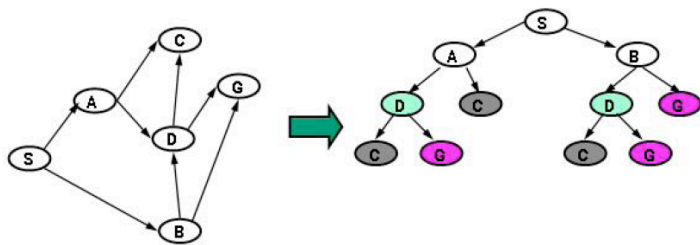
- o Les actions sont déterministes
- o Les actions ont des effets « discrets » sur le monde

## 2. Notion de graphe de recherche (3)

- o Graphe de résolution ou graphe d'états [Nilsson, p.125]
- o Graphe de recherche
  - Nœud
  - Arc (et coût)
  - Parents / ancêtres / successeurs
  - Critère objectif
  - Développement d'un nœud
- o Stratégie de contrôle
  - Fonction déterminant le choix du nœud à développer
  - Recherche aveugle ou *non informée*
  - Recherche heuristique ou *informée*

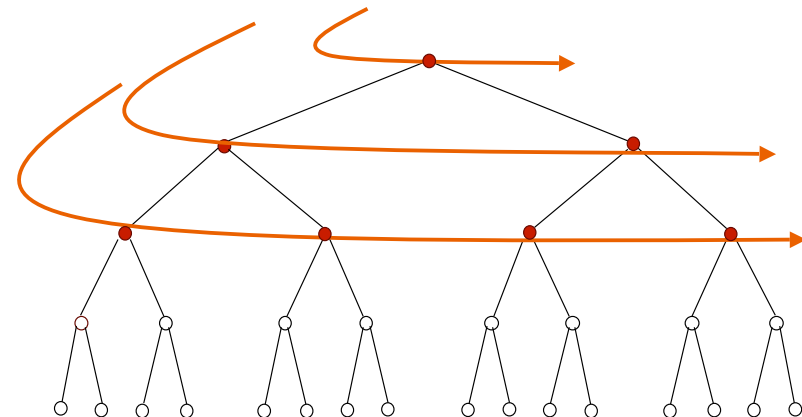


## 2. Graphe de recherche et arbre de recherche



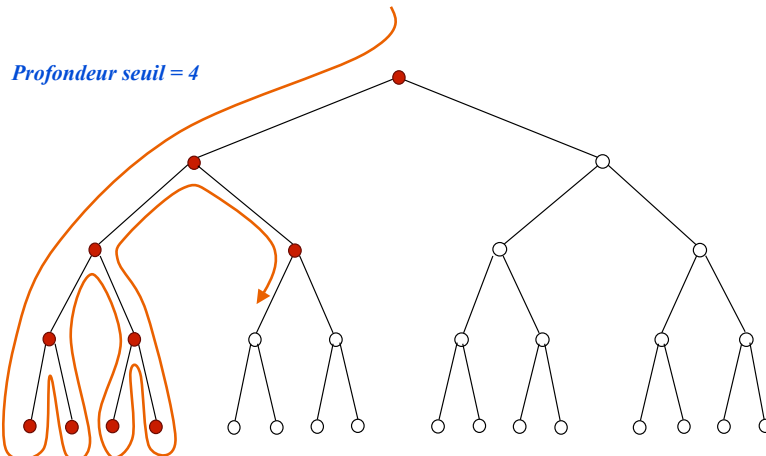
## 2. Recherche aveugle : en largeur d'abord

Stratégie systématique : niveau par niveau



## 2. Recherche aveugle : en profondeur d'abord

Stratégie systématique : avec retour arrière



## 2. Propriétés des stratégies de recherche

- **Complétude**  
La stratégie parvient-elle nécessairement à une solution si il en existe une ?
- **Complexité en temps**  
Nombre de nœuds développés (ou évalués) durant la recherche
- **Complexité en espace**  
Nombre maximal de nœuds en mémoire lors de la recherche
- **Optimalité**  
La solution retournée est-elle optimale en coût ?

Facteurs :

- $b$  : facteur de branchement
- $d$  : profondeur
- $m$  : profondeur maximale de l'espace d'états

## 2. Largeur d'abord : propriétés

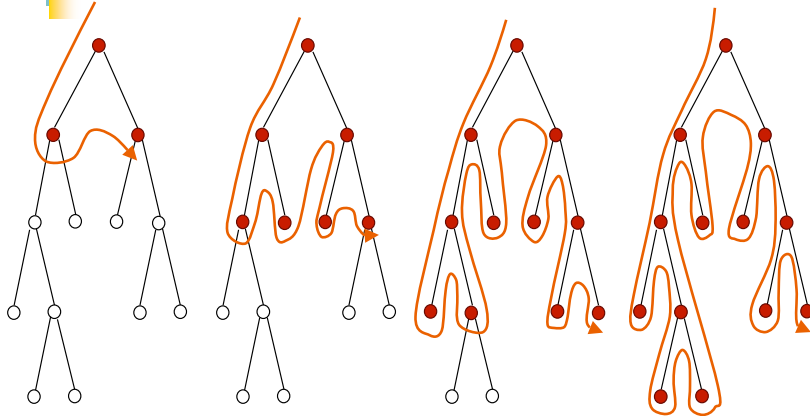
- **Complétude ?**
  - Oui (si  $b$  est fini)
- **Complexité en temps ?**
  - Exponentiel en  $d$  :  $1 + b + b^2 + b^3 + \dots + b^d = O(b^d)$
- **Complexité en espace ?**
  - $O(b^d)$
- **Optimalité ?**
  - Oui (si coût de chaque arc = 1)

➔ **La complexité en espace est le gros problème**

## 2. Profondeur d'abord : propriétés

- **Complétude ?**
  - Non : échoue si profondeur infinie ou si boucles
- **Complexité en temps ?**
  - Exponentiel en  $m$  :  $O(b^m)$       Dramatique si  $m \gg d$
  - Mais si les solutions sont denses dans le graphe, peut-être plus rapide que "largeur d'abord"
- **Complexité en espace ?**
  - $O(bm)$       cad **espace mémoire linéaire !!**
- **Optimalité ?**
  - Non

## 2. Profondeur itérative (iterative deepening)



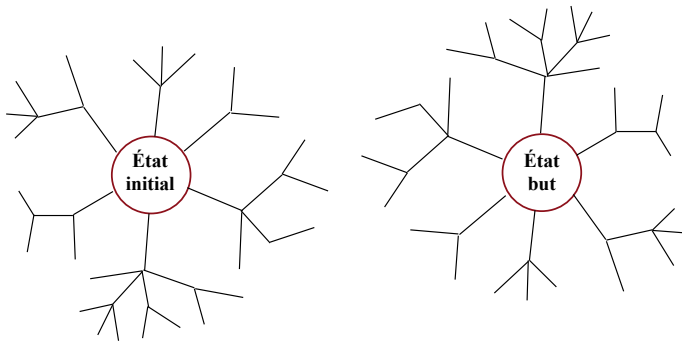
Profondeur seuil = 1   Profondeur seuil = 2   Profondeur seuil = 3   Profondeur seuil = 4

## 2. Profondeur itérative : propriétés

- Complétude ?
  - Oui
- Complexité en temps ?
  - $(d+1)b^0 + db^1 + (d-1)b^2 + \dots + b^d = O(b^d)$
  - A peine plus que largeur d'abord
- Complexité en espace ?
  - $O(bd)$     cad **espace mémoire linéaire !!**
- Optimalité ?
  - Oui (si le coût des arcs = 1)

➔ **Méthode préférée** quand grand espace de recherche et profondeur de la solution inconnue

## 2. Recherche bidirectionnelle



## 2. Coût uniforme

## 2. Les méthodes en meilleur d'abord

OUVERT  $\leftarrow$  {état initial} ; FERME  $\leftarrow$   $\emptyset$  ; Succès  $\leftarrow$  Faux

Tant que OUVERT  $\neq$   $\emptyset$  et Succès = Faux

**Etape 1 : Choix** d'un noeud  $n$  dans OUVERT

Si  $n$  est un état terminal alors Succès  $\leftarrow$  Vrai  
et retourner le chemin solution trouvé

**Sinon 1**

**Etape 2 : développement de  $n$**

Supprimer  $n$  de OUVERT

L'ajouter à FERME

Pour chaque successeur  $s$  de  $n$

Si  $s$  n'appartient ni à OUVERT ni à FERME

ajouter  $s$  à OUVERT

père( $s$ )  $\leftarrow$   $n$

Sinon mise à jour éventuelle du père de  $s$

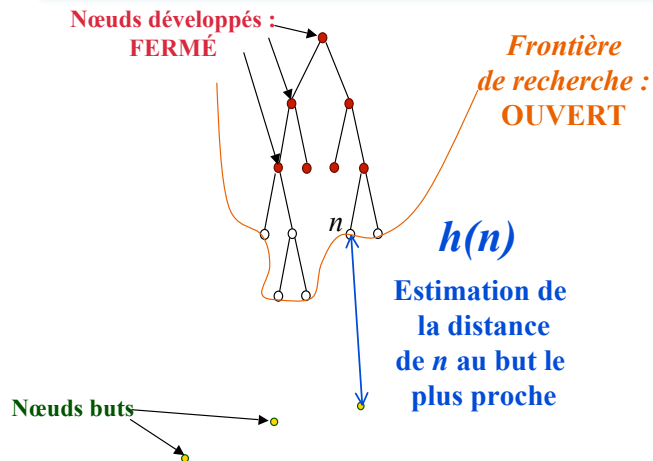
**Echec** si OUVERT =  $\emptyset$

## 2. En meilleur d'abord : la fonction d'évaluation

- Le choix du nœud à développer dépend d'une **fonction d'évaluation**  $f(n)$  estimant le mérite de  $n$
- Les nœuds  $n$  sur la frontière de recherche sont ordonnés dans une liste **OUVERT** par ordre croissant

## 2. Les méthodes informées

Disposent d'une information sur la *proximité au but*



## 2. La méthode A ( $A^*$ )

Fonction d'évaluation :

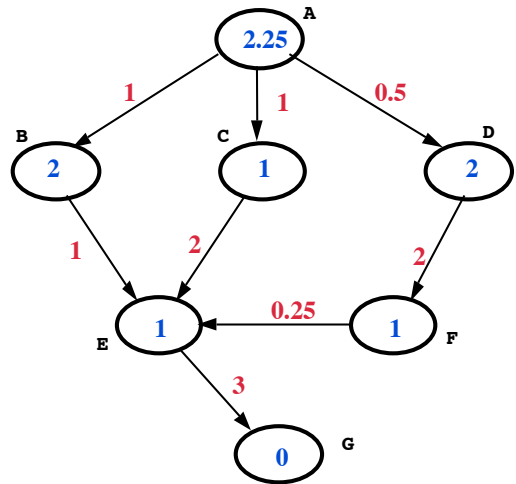
$$f(n) = g(n) + h(n)$$

Coût estimé d'un chemin de la racine à un objectif passant par  $n$

Coût du chemin le plus court connu actuellement pour aller de la racine au nœud  $n$  :  
fonction dynamique

Estimation du coût optimal pour atteindre un objectif à partir du nœud  $n$  :  
fonction statique

## 2. La méthode A (A\*) : illustration (1/4)



## 2. La méthode A (A\*) : illustration (2/4)

FERMÉ / OUVERT	Développement de l'arborescence
FERMÉ : A(2.25) OUVERT : C(2), D(2.5), B(3)	
FERMÉ : C(2), A(2.25) OUVERT : D(2.5), B(3), E(4)	

## 2. La méthode A (A\*) : illustration (3/4)

FERMÉ : C(2), A(2.25), D(2.5) OUVERT : B(3), E(3.5), F(4)	
FERMÉ : C(2), A(2.25), D(2.5), B(3) OUVERT : E(3) par B, F(3.5), E(4) par C est éliminé	

## 2. La méthode A (A\*) : illustration (4/4)

FERMÉ : C(2), A(2.25), D(2.5), B(3), E(3) par B OUVERT : F(3.5), G(5)	
FERMÉ : C(2), A(2.25), D(2.5), B(3), E(3) par B, F(3.5) OUVERT : E(3.75) par E, mais pas meilleur que E(3) par B de FERMÉ, donc pas de ré-actualisation de E ni donc de G(5). Et comme G(5) est le seul à rester dans OUVERT et que c'est le but, Succès par le chemin A-B-E-G	

## 2. L'optimalité de A\*

Propriété (optimalité ou admissibilité de A\*) :

Si la fonction d'estimation de la distance au but  $h$  est systématiquement **optimiste**, l'algorithme A s'appelle A\* et retourne nécessairement une solution optimale.

Preuve :

## 2. Algorithme plus ou moins informé

Définition :

Si deux versions  $A_1^*$  et  $A_2^*$  d'un algorithme A\* ne diffèrent que par leur fonction d'estimation  $h$  avec  $n$  (non terminal)  $h_1(n) < h_2(n)$ , alors on dit que  $A_2^*$  est mieux informé que  $A_1^*$ .

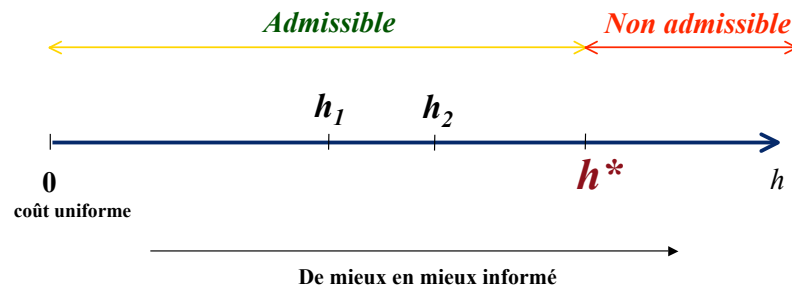
Théorème :

Si  $A_2^*$  est mieux informé que  $A_1^*$ , alors lorsque leurs recherches sont terminées, tous les nœuds développés par  $A_2^*$  l'ont également été par  $A_1^*$  (et peut-être d'autres).

→  $A_2^*$  est plus efficace que  $A_1^*$

## 2. Algorithme plus ou moins informé

$$\forall n, h_1(n) \leq h_2(n)$$



## 2. Algorithme plus ou moins informé : illustration

o Taquin

## 2. Algorithme plus ou moins informé : tableau comparatif

Profondeur	Nombre moyen de nœuds développés			Facteur de branchement effectif		
	IDS	A*(h <sub>1</sub> )	A*(h <sub>2</sub> )	IDS	A*(h <sub>1</sub> )	A*(h <sub>2</sub> )
2	10	6	6	2,45	1,79	1,79
4	112	13	12	2,87	1,48	1,45
6	680	20	18	2,73	1,34	1,30
8	6384	39	25	2,80	1,33	1,24
10	47127	93	39	2,79	1,38	1,22
12	364404	227	73	2,78	1,42	1,24
14	3473941	539	113	2,83	1,44	1,23
16		1301	211		1,45	1,25
18		3056	363		1,46	1,26
20		7276	676		1,47	1,27
22		18094	1219		1,48	1,28
24		39135	1641		1,48	1,26
Moyenne				2,8	1,43	1,27

## 2. Algorithme plus ou moins informé : tableau comparatif

- **Taquin 4x4 :**
  - ~ 6 jours pour trouver une solution en moyenne sans heuristique
  - ~ 50s avec heuristique h<sub>1</sub>
  - ~ 0.1s avec heuristique h<sub>2</sub>

Korf, R. (2000) « *Recent progress in the design and analysis of admissible heuristic functions* », Proc. of SARA-2000, Springer-Verlag.

## 2. A\* itératif (iterative-deepening A\*: IDA\*)

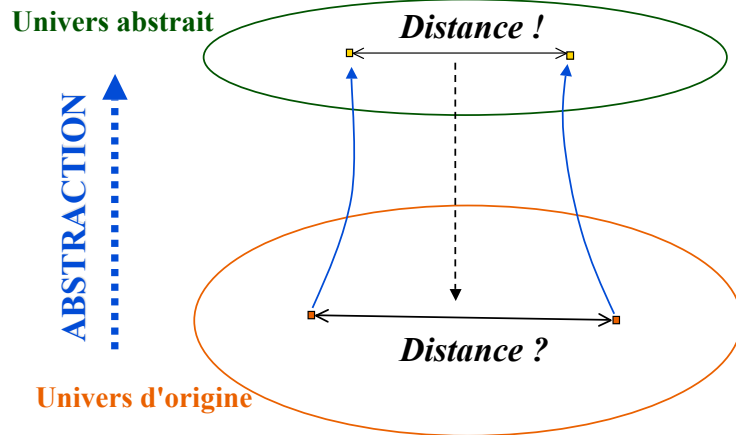
- [Nilsson, p.153]

## 2. La découverte de fonction heuristique

- **Approches possibles**
  - Par abstraction
  - Par essais et erreurs
  - Par apprentissage

## 2. La découverte de fonction heuristique par abstraction

Principe :



## 2. La découverte de fonction heuristique par abstraction

### Illustration : le jeu du taquin [Pearl, 1984]

- Soient les *prédicats de description* :
  - $\text{sur}(X, Y)$  : la tuile  $X$  est sur la case  $Y$
  - $\text{vide}(Y)$  : il n'y a pas de tuile sur la case  $Y$
  - $\text{adj}(Y, Z)$  : les cases  $Y$  et  $Z$  sont adjacentes
- Soit l'*opérateur de déplacement*  $\text{bouger}(X, Y, Z)$  défini par :
  - Préconditions :  $\text{sur}(X, Y) \ \& \ \text{vide}(Z) \ \& \ \text{adj}(Y, Z)$
  - Ajouts :  $\text{sur}(X, Z) \ \& \ \text{vide}(Y)$
  - Retraits :  $\text{sur}(X, Y) \ \& \ \text{vide}(Z)$
- **Problème 1** : Trouver une séquence d'opérateurs  $\text{bouger}(X, Y, Z)$  instanciés pour aller de l'état initial à l'état final.
- **Problème 2** : Estimer la distance au but, cad le nombre d'opérations nécessaires

## 2. La découverte de fonction heuristique par abstraction

Opérateur de déplacement  $\text{bouger}(X, Y, Z)$  défini par :

- Préconditions :  $\text{sur}(X, Y) \ \& \ \text{vide}(Z) \ \& \ \text{adj}(Y, Z)$
- Ajouts :  $\text{sur}(X, Z) \ \& \ \text{vide}(Y)$
- Retraits :  $\text{sur}(X, Y) \ \& \ \text{vide}(Z)$

## 2. La découverte de fonction heuristique par abstraction

Opérateur de déplacement  $\text{bouger}(X, Y, Z)$  défini par :

- Préconditions :  $\text{sur}(X, Y) \ \& \ \text{vide}(Z) \ \& \ \text{adj}(Y, Z)$
- Ajouts :  $\text{sur}(X, Z) \ \& \ \text{vide}(Y)$
- Retraits :  $\text{sur}(X, Y) \ \& \ \text{vide}(Z)$

### Abstraction par relaxation (suppression) des préconditions

- Suppression des préconditions  $\text{vide}(Z) \ \& \ \text{adj}(Y, Z)$  :
  - Chaque carreau mal placé peut être directement mis sur la case objectif :  $h_1$
- Suppression de la précondition  $\text{vide}(Z)$  :
  - Chaque carreau mal placé peut être déplacé sur une case adjacente :  $h_2$
- Suppression de la précondition  $\text{adj}(Y, Z)$  :
  - Chaque carreau mal placé peut être directement mis sur la case vide :  $h_3$

## 2. La découverte de fonction heuristique par essais / erreurs

### Principes :

1. Sur un petit problème dont une solution optimale est connue : chercher une fonction  $h$  telle que  $f(n) = g(n) + h(n)$

2. Si plusieurs heuristiques admissibles  $h_i$  sont connues, prendre

$$h(n) = \arg \max_i h_i(n)$$

3. Sélectionner des attributs de description de l'état qui semblent jouer un rôle dans l'estimation de la distance au but et les combiner dans une fonction d'évaluation (.... éventuellement par apprentissage)

## 2. La découverte de fonction heuristique : état de l'art

o **EURISKO ? [Lenat, 1982]** : recherche par transformation syntaxique dans l'espace des heuristiques

### Par abstraction

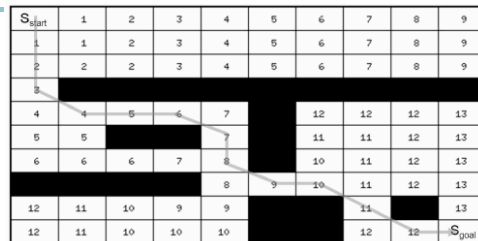
o ABSOLVER [Priedetis, 1993]

- o Par la méthode "relaxed problem" : relâche les restrictions sur les opérateurs
- o Découverte de l'heuristique la plus efficace pour le taquin
- o Découverte d'heuristiques pour le Rubik's cube, .... (13 pbs)

### Par apprentissage

- o Apprentissage par renforcement (cf cours n°4)
- o Apprentissage par recherche d'abstraction (Thèse J-D Zucker (1996))

## LifeLong A\*



Changed Eight-Connected Gridworld

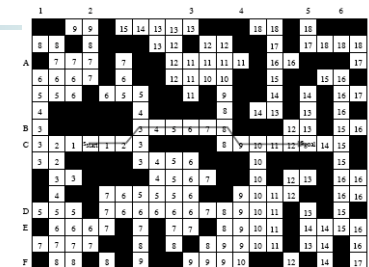


## LifeLong A\*

Comment réutiliser au mieux les connaissances issues des explorations antérieures du monde ?

[Koenig, Likhachev & Furcy, AIj (2003)]

[Fedor & Cornuéjols, 2008]



Changed Eight-Connected Gridworld

