

# *Introduction*

à

## *l'intelligence artificielle*

### *(recherche dans les graphes)*

*Plan*

- 1- *Introduction à l'IA*
- 2- *Résolution de problèmes par exploration de graphes*
- 3- *Résolution de problème en environnement non déterministe*
- 4- *Apprentissage par renforcement*
- 5- *Planification*

*Antoine Cornuéjols*

*AgroParisTech*

[antoine.cornuejols@agroparistech.fr](mailto:antoine.cornuejols@agroparistech.fr)

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

Cours IA

Cours IA (A. Cornuéjols)

2/61

## *Résolution de problème par Exploration de graphes*

*2. Recherche dans les graphes : plan*

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

3/61

Cours IA (A. Cornuéjols)

4/61

## 2. Spécification de problèmes (1)

### 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
- ...

### 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, cryptarithmétique (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

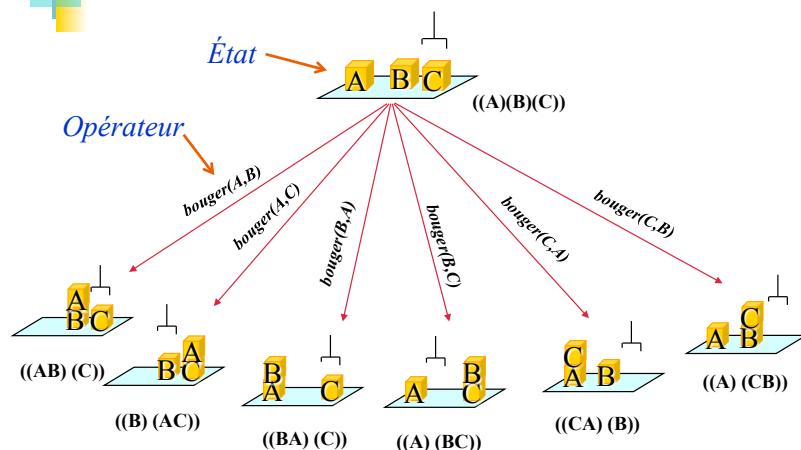
Ex : Problèmes de planification, Tours de Hanoï, intégration symbolique

## 2. Résolution de problèmes

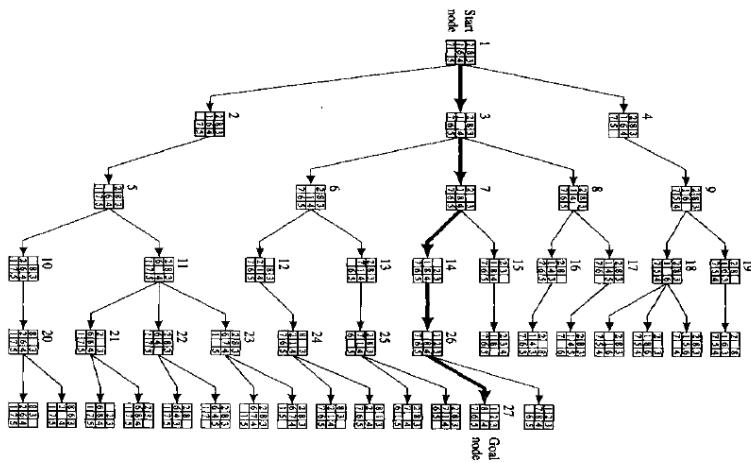
### Démarche générale :

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 (1)



Cours IA (A. Cornuéjols)

9/61

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

On suppose que :

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

Cours IA (A. Cornuéjols)

10/61

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

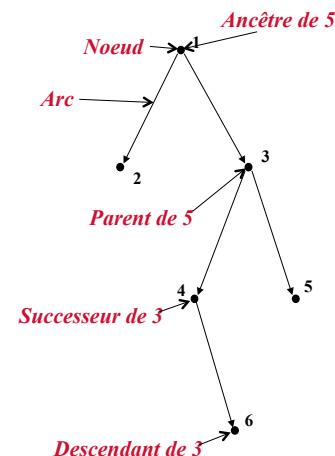
### ○ Graphe de résolution ou graphe d'états

### ○ Graphe de recherche

- Nœud
- Arc (et coût)
- Parents / ancêtres / successeurs
- Critère objectif
- Développement d'un nœud

### ○ Stratégie de contrôle

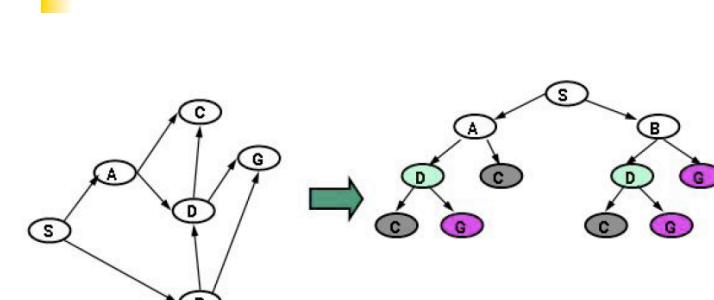
- Fonction déterminant le choix du nœud à développer
- Recherche aveugle ou *non informée*
- Recherche heuristique ou *informée*



Cours IA (A. Cornuéjols)

11/61

## 2. Graphe de recherche et arbre de recherche

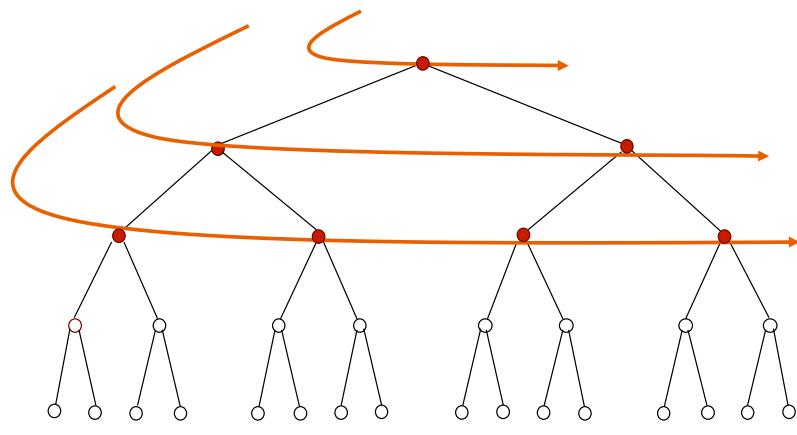


Cours IA (A. Cornuéjols)

12/61

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

Stratégie systématique : niveau par niveau

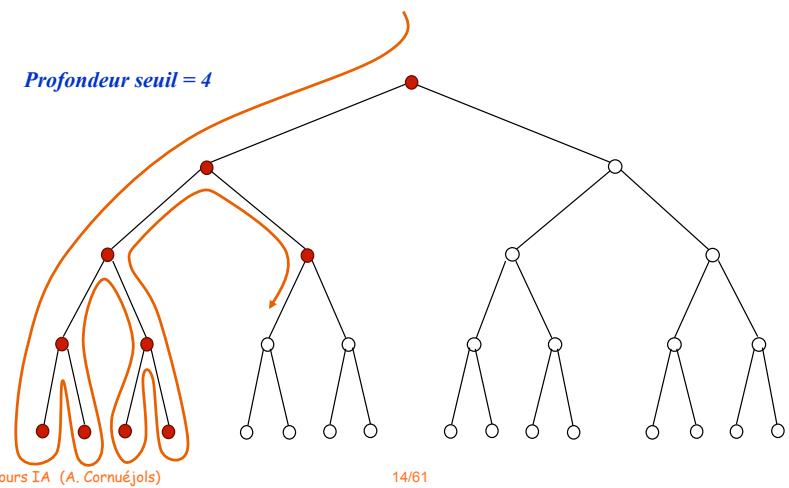


Cours IA (A. Cornuéjols)

13/61

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

Stratégie systématique : avec retour arrière



Cours IA (A. Cornuéjols)

14/61

## 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 sur une branche 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

Cours IA (A. Cornuéjols)

17/61

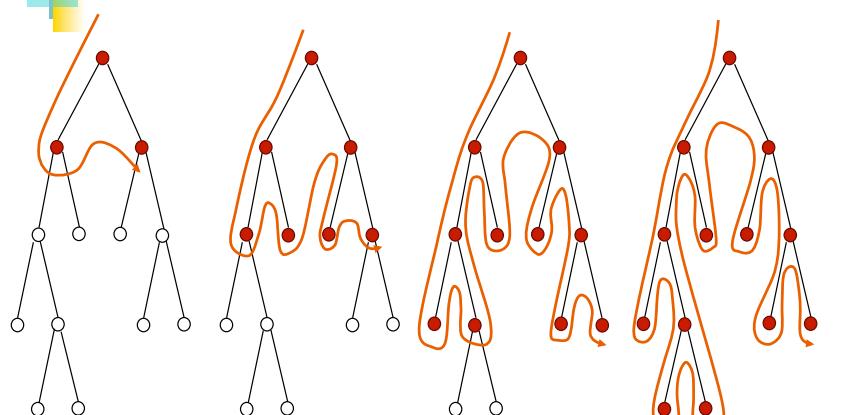
## Space: the final frontier

- In many (most) search problems, **memory is often the limiting factor**
  - Let's consider a searching tree with **branching factor 8** and **depth 10**. Suppose that a node requires 8 bytes of storage (very little).
  - Breadth-first search may require up to  $(2^3)^{10} \times 2^3 = 2^{33}$  bytes = 8,000 Mbytes = 8 Gbytes
  - One strategy is to **trade time for memory**.
  - For instance, we can emulate breadth-first search by repeated application of depth-first search, each up to a preset limit.
- **Progressive deepening search**

Cours IA (A. Cornuéjols)

18/61

## 2. Profondeur itérative (iterative deepening)



Richard Korf popularise et étudie cette technique en 1985.

Cours IA (A. Cornuéjols)

19/61

## 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 ?**

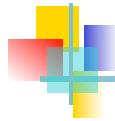
$$\begin{aligned} & b^k + 2b^{k-1} + 3b^{k-2} + \dots + kb \\ &= b^k(1 + 2b^{-1} + 3b^{-2} + \dots + kb^{1-k}) \\ &\leq b^k \left( \sum_{i=1}^{\infty} ib^{(i-1)} \right) \\ &= b^k \left( \frac{b}{b-1} \right)^2 \end{aligned}$$

Erreur :  $b / (b-1)^2$
- **Optimalité ?**
  - Oui (si le coût des arcs = 1)

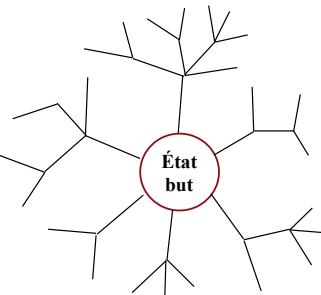
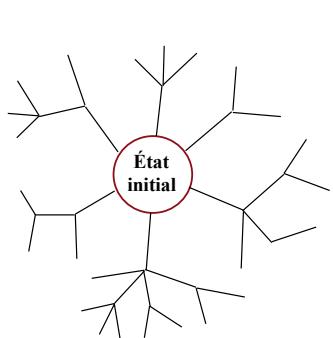
→ **Méthode préférée** quand grand espace de recherche et profondeur de la solution inconnue

Cours IA (A. Cornuéjols)

20/61



## 2. Recherche bidirectionnelle



Cours IA (A. Cornuéjols)

21/61



## 2. Coût uniforme

Cours IA (A. Cornuéjols)

22/61



## 2. Les méthodes en meilleur d'abord

```

OUVERT ← {état initial} ;   FERME ← Ø      ;   Succès ← Faux
Tant que OUVERT ≠ Ø et Succès = Faux
    Etape 1 : Choix d'un noeud n dans OUVERT
    Si n est un état terminal alors Succès ← Vrai
        et retourner le chemin solution trouvé
    Sinon
        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)← n
            Sinon mise à jour éventuelle du père de s
        Echec si OUVERT = Ø
    
```

Cours IA (A. Cornuéjols)

23/61



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

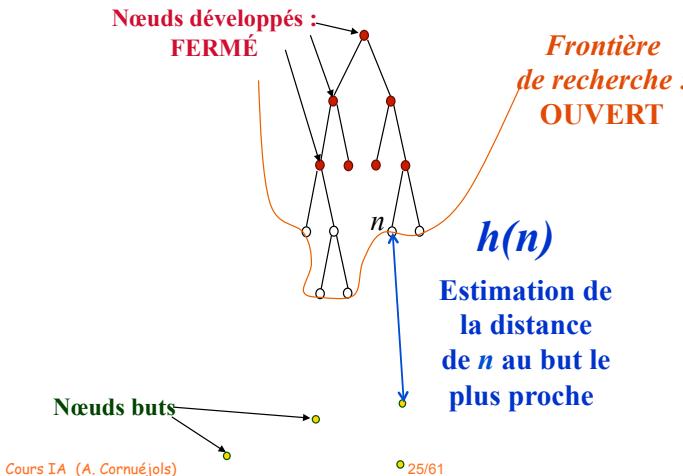
- Le choix du noeud à développer dépend d'une fonction d'évaluation  $f(n)$  estimant le mérite de *n*
- Les noeuds *n* sur la frontière de recherche sont ordonnés dans une liste OUVERT par ordre croissant

Cours IA (A. Cornuéjols)

24/61

## 2. Les méthodes informées

Disposent d'une information sur la proximité au but



## 2. La méthode A ( $A^*$ )

Inventée en 1968  
[Hart, Nilsson & Raphael]

Fonction d'évaluation :

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

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

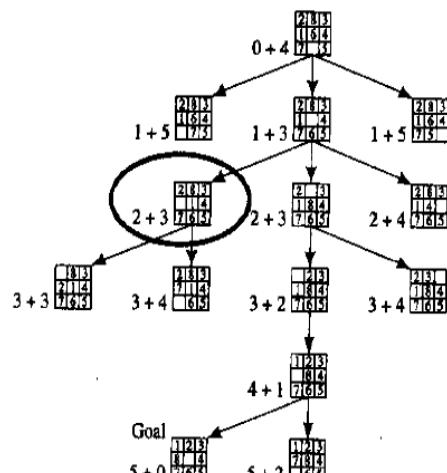
Coût du chemin le plus court connu actuellement pour aller de la racine au nœud  $n$  : fonction dynamique

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

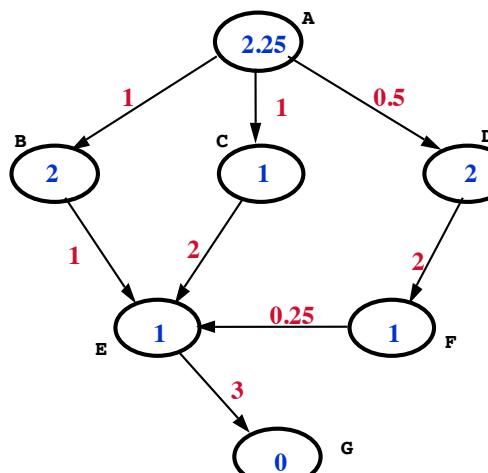
Cours IA (A. Cornuéjols)

26/61

## 2. La méthode A ( $A^*$ )



## 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)	<pre> graph TD     A((A)) --&gt; B((B))     A --&gt; C((C))     A --&gt; D((D))   </pre>
FERMÉ : C(2), A(2.25) OUVERT : D(2.5), B(3), E(4)	<pre> graph TD     A((2.25)) --&gt; B((2))     A --&gt; C((1))     A --&gt; D((2.5))     B --&gt; E((1))   </pre>

Cours IA (A. Cornuéjols)

29/61

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

FERMÉ : C(2), A(2.25), D(2.5) OUVERT : B(3), F(3.5), E(4)	<pre> graph TD     A((2.25)) --&gt; B((2))     A --&gt; C((1))     A --&gt; D((2.5))     C --&gt; E((1))     C --&gt; F((1))   </pre>
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	<pre> graph TD     A((2.25)) --&gt; B((2))     A --&gt; C((1))     A --&gt; D((2.5))     B --&gt; E((1))     B --&gt; F((1))     C --&gt; G((0))   </pre>

Cours IA (A. Cornuéjols)

30/61

## 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)	<pre> graph TD     A((2.25)) --&gt; B((2))     A --&gt; C((1))     A --&gt; D((2.5))     B --&gt; E((1))     B --&gt; F((1))     C --&gt; G((0))   </pre>
FERMÉ : C(2), A(2.25), D(2.5), B(3), E(3) par B, F(3.5) OUVERT : E(3.75) par F, 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	<pre> graph TD     A((2.25)) --&gt; B((2))     A --&gt; C((1))     A --&gt; D((2.5))     B --&gt; E((1))     B --&gt; F((1))     C --&gt; G((0))   </pre>

Cours IA (A. Cornuéjols)

31/61

## 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 :

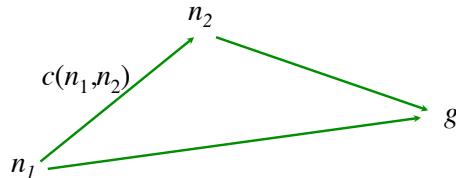
Cours IA (A. Cornuéjols)

32/61

## 2. La consistance

Propriété (*inégalité triangulaire*) :

$$\forall n_1, n_2, \quad h(n_1) \leq c(n_1, n_2) + h(n_2)$$



## 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 noeuds développés par  $A_2^*$  l'ont également été par  $A_1^*$  (et peut-être d'autres).

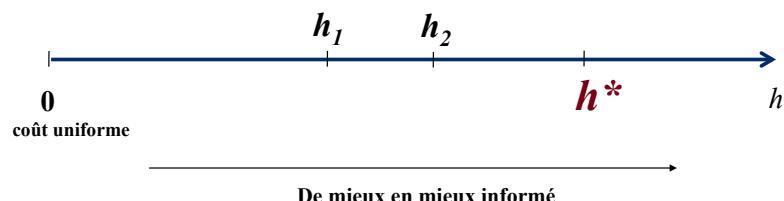
→  $A_2^*$  est plus efficace que  $A_1^*$

## 2. Algorithme plus ou moins informé

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

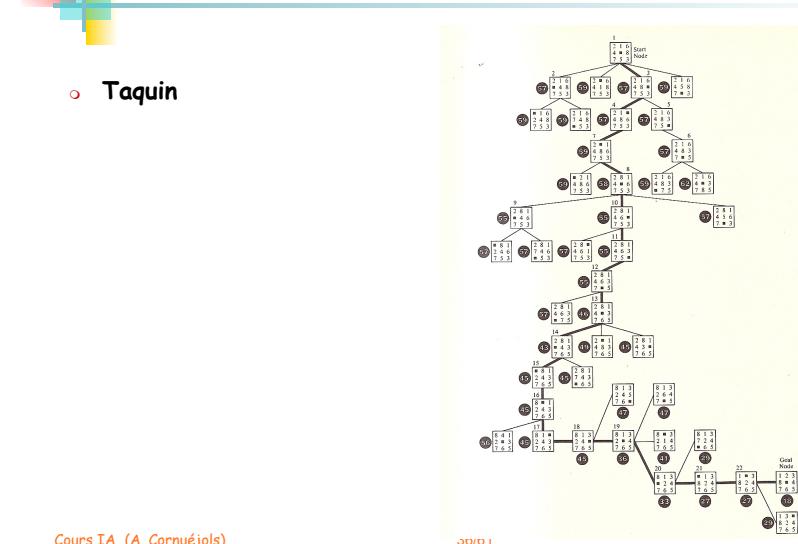
*Admissible*

*Non admissible*



## 2. Algorithme plus ou moins informé : illustration

○ 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]

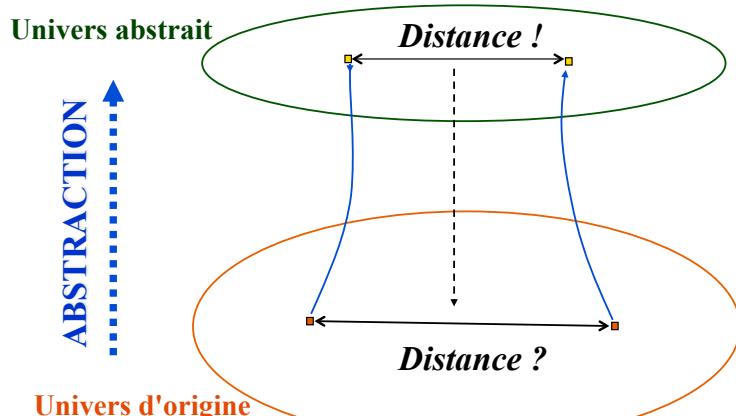


## 2. La découverte de fonction heuristique

- **Approches possibles**

- Par abstraction
- Par essais et erreurs
- Par apprentissage

## Principe :



## 2. La découverte de fonction heuristique par abstraction

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

#### Soient les *prédictats de description* :

- o  $\text{sur}(X, Y)$  : la tuile X est sur la case Y
- o  $\text{vide}(Y)$  : il n'y a pas de tuile sur la case Y
- o  $\text{adj}(Y, Z)$  : les cases Y et Z sont adjacentes

#### Soit l'*opérateur de déplacement* $\text{bouger}(X, Y, Z)$ défini par :

- o Préconditions :  $\text{sur}(X, Y) \ \& \ \text{vide}(Z) \ \& \ \text{adj}(Y, Z)$
- o Ajouts :  $\text{sur}(X, Z) \ \& \ \text{vide}(Y)$
- o 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, c'est le nombre d'opérations nécessaires

## 2. La découverte de fonction heuristique par abstraction

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

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

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



## 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

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

- Par abstraction

- ABSOLVER [Priedetis, 1993]

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

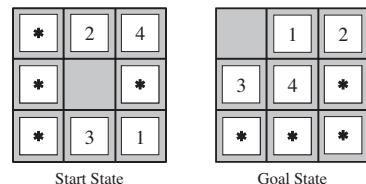
- Par apprentissage

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



## Recherche avec utilisation de « patterns »

### Un sous-problème



### 1ère idée :

- Principe

- Mémoriser la solution de tous les sous-problèmes, par exemple les sous-problèmes avec 4 tuiles → génère une « Pattern database »
- Calculer une heuristique admissible pour chacun de ces sous-problèmes
- Prendre l'heuristique qui est le max de toutes ces heuristiques

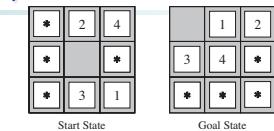
- E.g. le taquin à 15 tuiles est résolu en moyenne 1000 fois plus rapidement qu'avec l'heuristique de Manhattan

- Mais la taille d'une « pattern database » croît très vite avec la taille du problème



## Recherche avec utilisation de « patterns »

### Un sous-problème



### 2ème idée :

- Pourrait-on additionner les heuristiques au lieu d'en prendre le max ?

- Non. Car les sous-problèmes sont généralement non indépendants

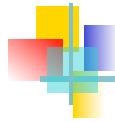
- Il se pourrait que les mouvements pour résoudre le sous-problème avec les tuiles 1-2-3-4 soient communs avec la solution pour le sous-problème 5-6-7-8.

- Principe. Calculer des « disjoint pattern databases »

- Pas simple. Voir [Ariel Felner, Korf & Hanan « Additive Pattern database heuristics », JAIR, 22, 279-318, (2004)]

- E.g. le taquin à 15 tuiles est résolu en moyenne 10 000 fois plus rapidement qu'avec l'heuristique de Manhattan

- Le taquin à 24 tuiles est résolu 1 000 000 de fois plus rapidement.



## LifeLong A\*

$S_{start}$	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	2	3	4	5	6	7	8	9
3									
4	4	5	4	7					
5	5				12	12	12	13	
6	6	6	7	8		11	11	12	13
					10	11	12	13	
12	11	10	9	9		11		13	
12	11	10	10	10		12	12		$S_{goal}$

Changed Eight-Connected Gridworld

$S_{start}$	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	2	3	4	5	6	7	8	9
3									
4	4	5	6	7					
5	5				12	11	10	10	
6	6	6	7	8		11	11	11	11
					10	11	12	12	
7	7	8	8	9		11	11	12	13
9	9	9	9	9		11		13	
9	9	9	9	9		12			$S_{goal}$

Cours IA (A. Cornuéjols)

49/61



## LifeLong A\*

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

1	2	3	4	5	6
8	9	15	14	13	13
5	6	13	12	12	12
6	6	7	12	11	11
5	5	6	11	10	10
4	4	4	9	8	8
A	7	7	7	16	16
B	3	2	1	15	15
C	3	2	1	14	14
D	5	5	6	10	10
E	6	6	7	9	11
F	7	7	7	14	14

Changed Eight-Connected Gridworld

1	2	3	4	5	6
10	10	15	14	13	13
9	8	7	13	12	12
10	9	8	12	11	11
10	9	8	11	10	10
10	9	8	10	9	9
A	9	8	9	10	10
B	3	2	1	15	15
C	3	2	1	14	14
D	5	5	6	10	10
E	6	6	7	9	11
F	7	7	7	14	14

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

[Fedon & Cornuéjols, 2008]

Cours IA (A. Cornuéjols)

50/61



## Extensions

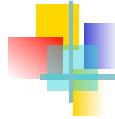


## Recherche de plus court chemin multi-agents

Problème : risque d'interférences « destructrices »

### 1. Algorithmes avec replanification

- Recherche indépendante des chemins de chaque agent
- Replanification si obstacle rencontré
  - Pb : si inter-blocages
  - Heuristique : ajout de « bruit »
    - Position des agents
    - Heuristique de distance utilisée



## Recherche de plus court chemin multi-agents

Problème : risque d' **interférences** « destructrices »

### 2. Recherche coopérative

- Recherche indépendante des chemins de chaque agent
  - Dans un **ordre prédéterminé**
  - Avec **table de réservation** remplie incrémentalement
- Ne permet pas de résoudre **certains blocages**
- Coûteux en temps
- Heuristique :
  - *Changer l'ordre*
  - *Introduire de la replanification*



## Recherche en temps réel

### o Algorithmes

- RTA\*
- LRTA\* (Learning RTA\*)
  - Modification de l'heuristique par apprentissage



## Recherche avec cible mouvante

### o Algorithmes

- Moving target search
  - Une heuristique pour chaque position possible de la cible
    - Table d'heuristiques



## Recherche hiérarchique de chemin



## Les graphes ET/OU

### 2. Les graphes ET/OU

- Motivation
- Illustrations
- Différences avec les algorithmes A

- On ne maintient plus des chemins mais **des sous-graphes**
- On a donc une **nouvelle fonction d'évaluation**
  - tenant compte du **mérite des noeuds frontières** (sous-problèmes)
  - mais aussi des **coûts de combinaison** des sous-problèmes

### 2. Algorithme AO\*: illustration

- Structure chimique

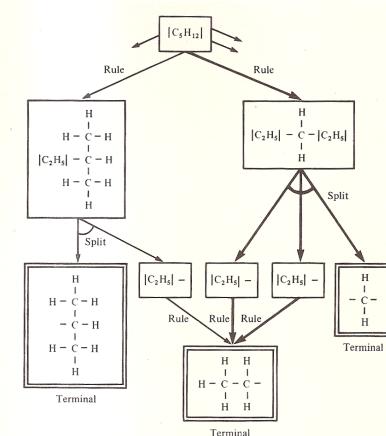
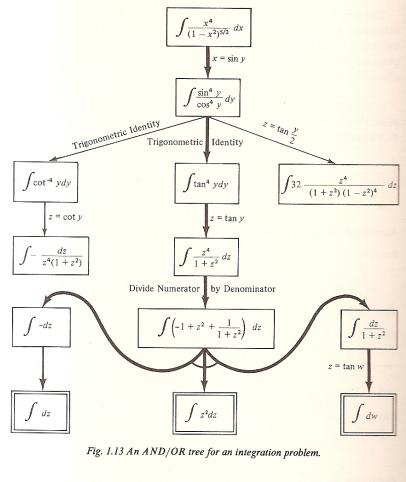


Fig. I.11 An AND/OR tree for a chemical structure problem.

## 2. Algorithme AO<sup>\*</sup> : illustration

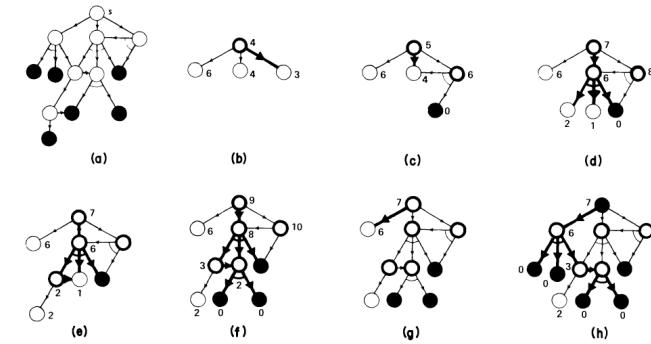
### ○ Calcul d'intégrale



Cours IA (A. Cornuéjols)

61/61

## 2. Algorithme AO<sup>\*</sup> : illustration



**Figure 2.9**  
Successive steps in the execution of general-best-first (GBF) search on the implicit AND/OR graph of part (a). Solid circles represent solved nodes, heavy hollow circles nodes in CLOSED, and thin circles nodes in OPEN. The heavy lines stand, at each stage, for the current most promising solution base.

Cours IA (A. Cornuéjols)

62/61

## Sources documentaires

### ○ Ouvrages / articles

- R. Callan (2003) : *Artificial Intelligence*. Palgrave MacMillan.
- T. Cazenave (2011) : *Intelligence Artificielle. Une approche ludique*. Ellipses.
- I. Millington (2006) : *Artificial Intelligence for Games*. Morgan Kaufmann.
- Nilsson N. (1998) : *Artificial Intelligence : A new synthesis*. Morgan Kaufmann.
- J. Pearl (1984) : *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison Wesley.
- Russel S. & Norvig P. (2009) : *Artificial Intelligence : A modern approach*. Prentice Hall, 2009 (3rd Ed.).

### ○ Sites web

- <http://www.gameai.com/clagames.html>

Cours IA (A. Cornuéjols)

63/61