



Résolution de problèmes par la recherche





Plan

■ Introduction

- Types de problèmes et approches
- Graphe de sous-problèmes

■ Résolution dans un espace d'états

- Notion d'espace d'états
- Représentation d'un problème

■ Espace d'états : stratégies

- Problématique
- Types de stratégies
- Recherche alpine et backtracking

■ Espace d'états : stratégies à graphe de recherche

- Notion de graphe de recherche
- Stratégies aveugles et recherche heuristique
- Recherche heuristique : "meilleur en premier"
- Recherche heuristique : algorithme A*



Introduction

Types de problèmes

■ Problèmes non considérés

- Existence d'une formule explicitable
- Existence d'un algorithme (polynomial)
- Non formalisables

■ Problèmes considérés

- Pas de méthode de résolution adaptée
- Bien définis
 - ➡ Une procédure pour tester une solution en un nombre fini d'étapes
 - ➡ Exemple : intégration formelle
- Une description formelle manipulable
- A fort aspect combinatoire
 - ➡ Recherche de réponses acceptables
 - ➡ Obtention dans des délais raisonnables

Approches

■ Problématique

Pas de déterminisme, mais seulement des spécifications de ce qu'est une solution !

👉 Recours :

- parcourir un espace de recherche
- construire la solution

■ Approches énumératives

Principe général :

- engendrer des solutions potentielles
- les tester

Types d'énumération :

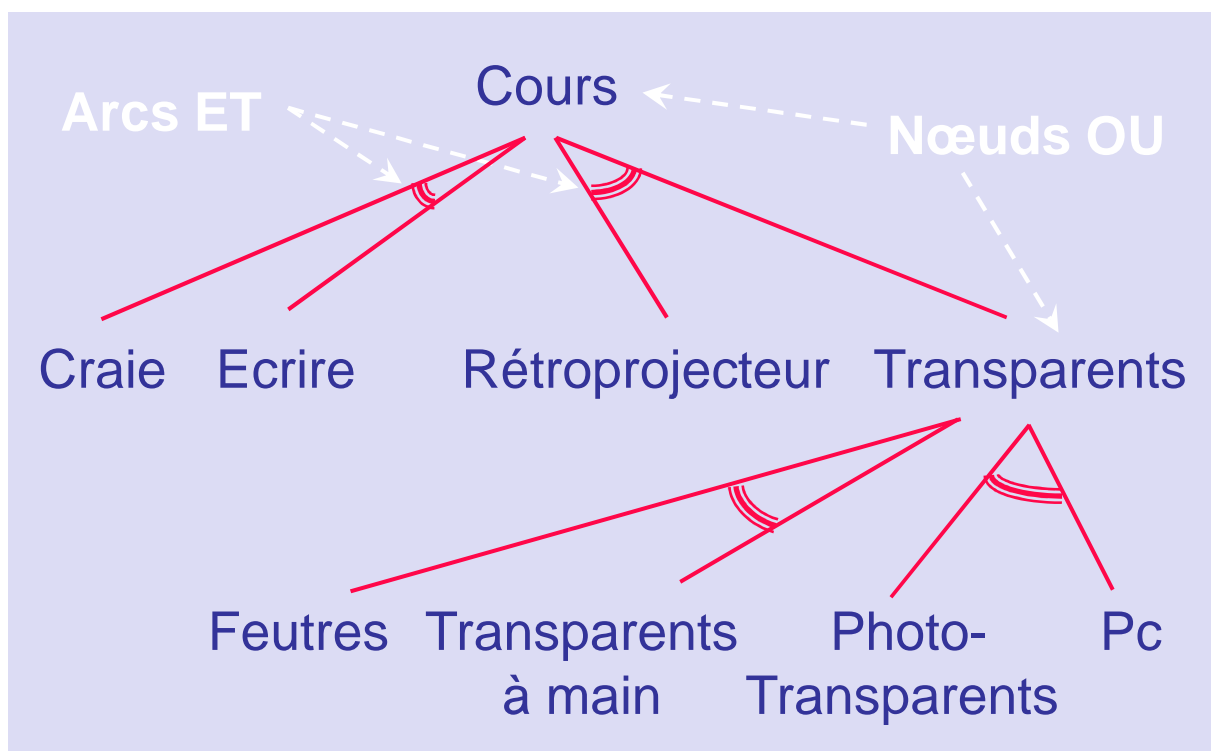
- Énumération explicite des solutions : le plus souvent non viable
- Énumération implicite : *par améliorations successives de solutions partielles*

Exemple : 8 reines

Une approche possible

Les graphes de sous-problèmes

- Réduction de problèmes : par la recherche d'opérateurs de décomposition
- Solution d'un problème : un sous-graphe. Exemple : l'intégration formelle
- Attention ! Il n'est pas toujours possible de trouver une décomposition





Résolution dans un espace d'états

Rappels sur les graphes

Vocabulaire minimal

- **Graphe** : ensemble de nœuds dont certaines paires sont connectées par des arcs
- **Graphe orienté** : graphe dont tous les arcs sont orientés \rightarrow parent, fils
- **Nœuds voisins** : n_2 est voisin de n_1 s'il existe un arc allant de n_1 à n_2 ou de n_2 à n_1
- **Arbre** : graphe où chaque nœud a au plus un parent \rightarrow racine, feuilles
profondeur
- **Chemin** : séquence de k nœuds dont chacun est fils du précédent
- **Ancêtre** : s'il existe un chemin allant de n_i à n_j , n_j est un descendant de n_i et n_i est un ancêtre de n_j

Espace d'états

Vocabulaire

- **Etat d'un problème** : ensemble d'informations décrivant une situation
- **Espace d'états** : ensemble discret de tous les états concevables, défini en extension ou en compréhension

Exemples : - base de faits d'un SE

- configuration des 8 reines

- **Etat initial** : décrit la situation de départ
- **Etats finaux** : états (ou buts) à atteindre, dont l'ensemble est défini en extension ou en compréhension

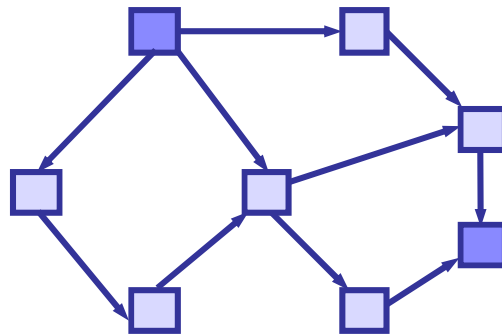
Exemples : échecs, diagnostic

- **Opérateurs** : permettent de passer d'un état à un autre

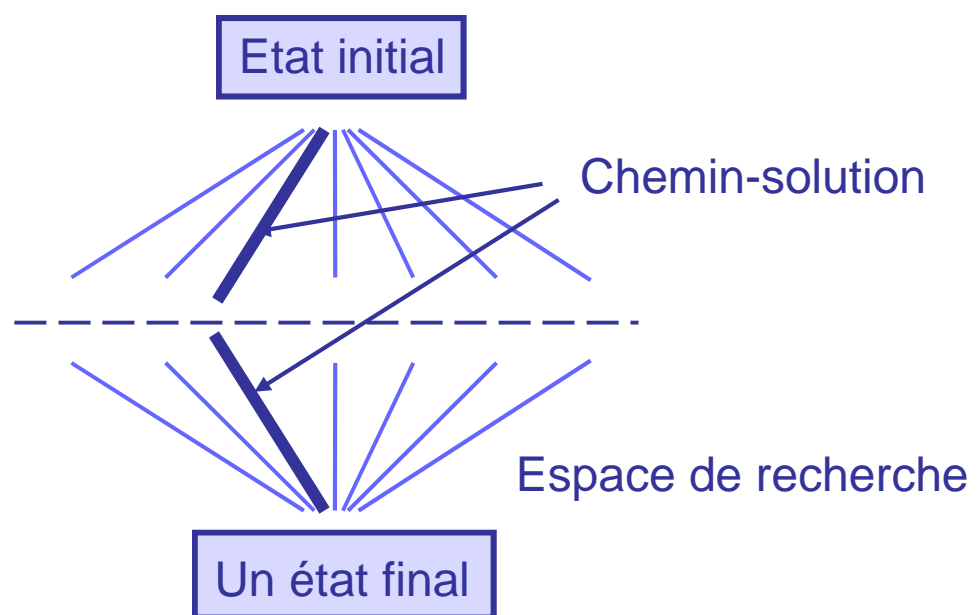
Espace d'états

Représentation

- De l'espace d'états : un graphe d'états OU dont les nœuds sont les états, et dont les arcs sont les opérateurs de transformation

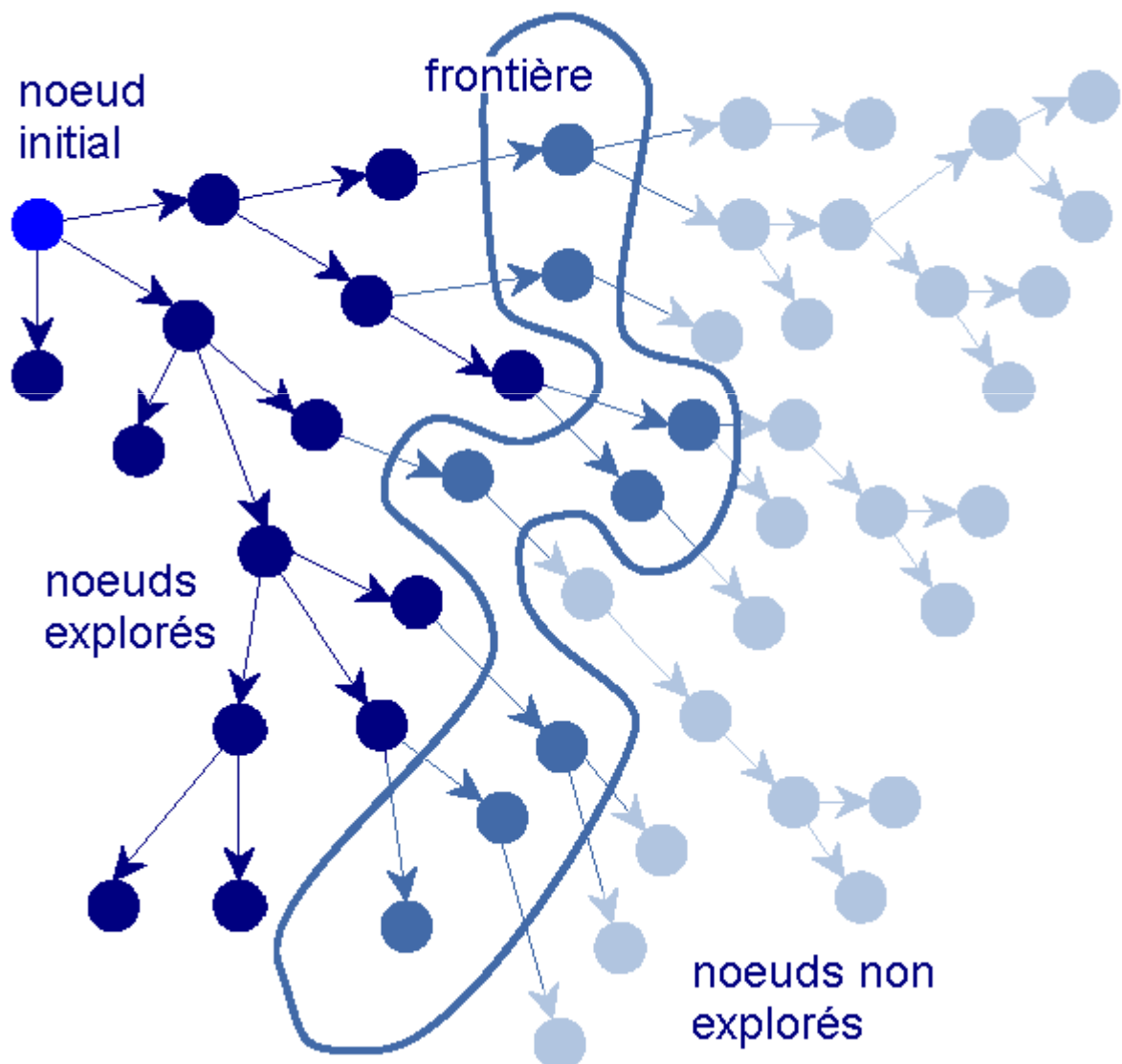


- De la solution :



Espace d'états

Recherche dans un espace d'états



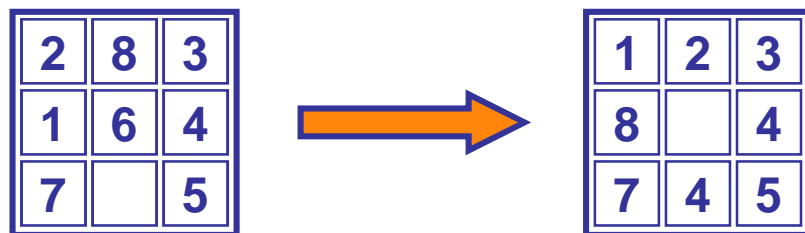
Espace d'états

Exemple : le taquin

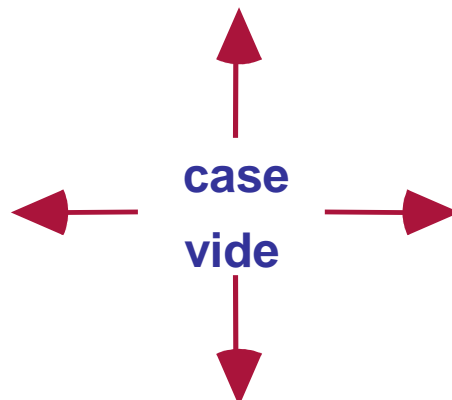
– Etat :

3	5	
2	1	8
6	4	7

– Etat initial et état final :

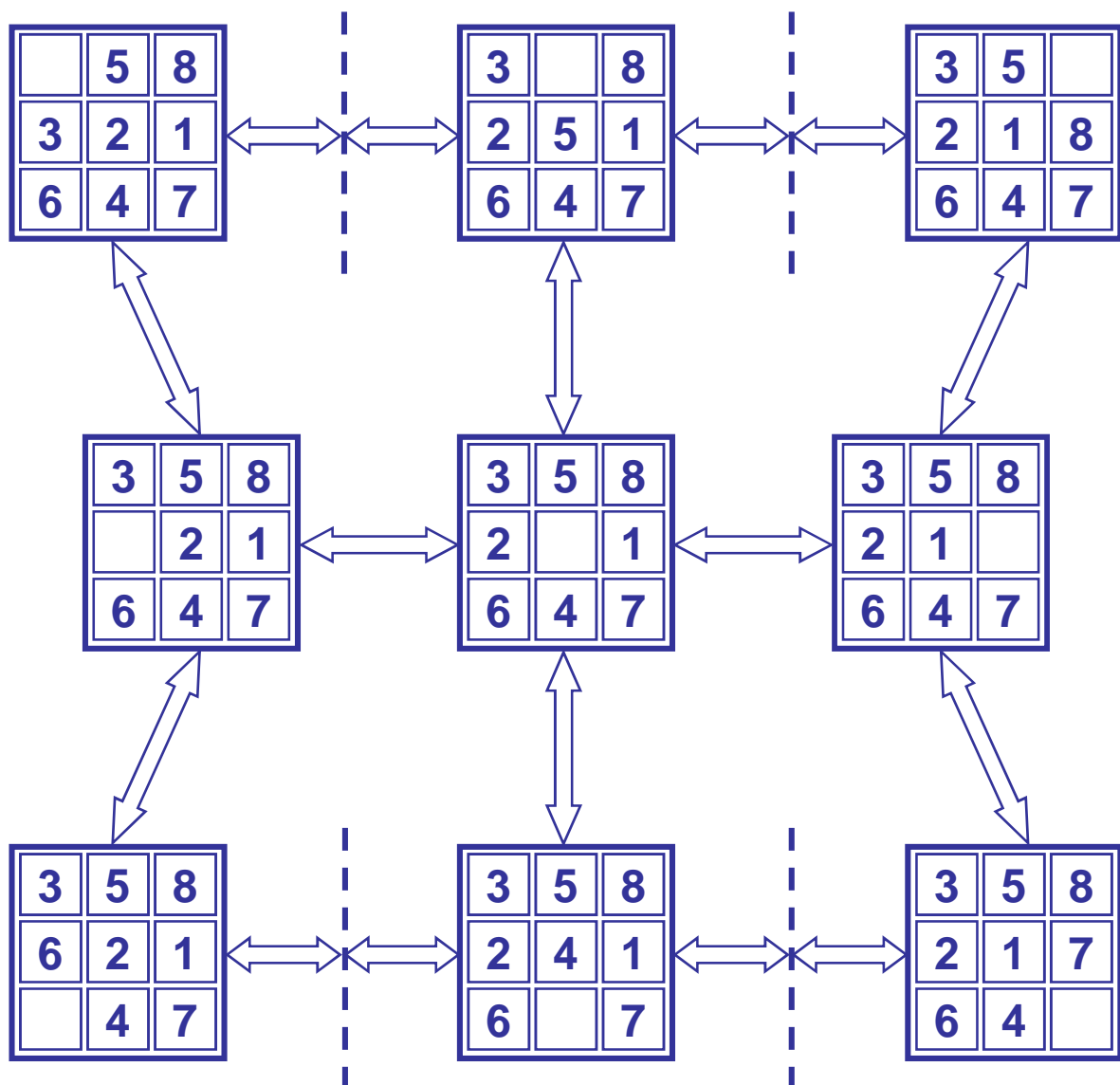


– Un opérateur de transformation :



Espace d'états

Le taquin : espace d'états



Problématique

Questions importantes

- **Sens de la recherche :**
 - *En-avant* : de l'état initial vers un but (orienté par les données)
 - *En-arrière* : d'un but vers l'état initial (orienté par les buts)
 - *Bidirectionnel*
- **Détermination des déplacements possibles :**
 - Recherche des états et des opérateurs à l'origine du prochain déplacement
 - Par appariement de formes (unification,...)
 - Importance de cette phase (filtrage)
- **Contrôle de la recherche :**
 - Quels choix effectuer ? Autrement dit, quels nœuds et quels opérateurs effectivement choisir ?
 - Importance des options stratégiques



Espace d'états : stratégies de recherche



Problématique

Stratégies de recherche

- Que demander à une stratégie ?
 - de créer un mouvement
 - de le faire de façon efficace (finalisée)
- Stratégie idéale
 - Elle minimise :
 - le coût de la solution
 - solution optimale
 - la durée de la recherche
- Types de stratégies
 - Uniformes / Informées
 - Irrévocables / Par tentatives

Types de stratégies

Stratégies irrévocables

– Caractéristiques

- ❑ Les opérateurs sont appliqués de façon irrémédiable, sans rétractation possible
- ❑ Il n'y a pas de mémorisation, donc pas de retour en arrière

– Problèmes concernés :

- ❑ Cas 1 : résolution monotone, sans remise en cause
- ❑ Cas 2 : opérateurs tous réversibles
- ❑ Cas 3 : existence d'une fonction de choix locale (bien informée)

– Exemple : recherche alpine

Types de stratégies

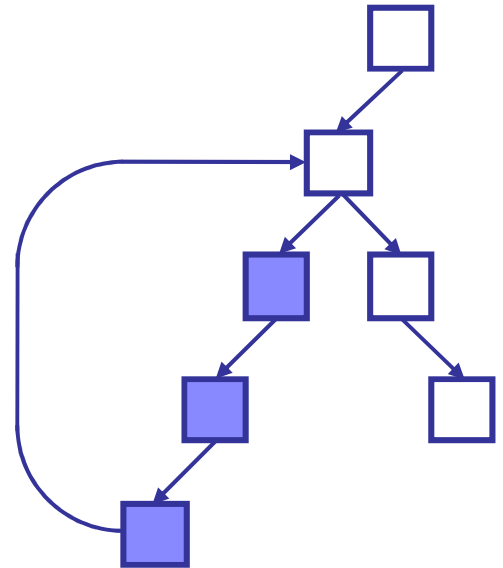
Stratégies par tentatives

– Caractéristiques

- ❑ Les opérateurs sont appliqués, avec possibilité de rétractation ultérieure
- ❑ Il y a mémorisation, partielle ou totale, d'états de la recherche

– Problèmes concernés :

- ❑ Cas 1 : description du problème incomplète
- ❑ Cas 2 : certains opérateurs ont des effets irréversibles
- ❑ Cas 3 : évolution temporelle



Types de stratégies

Stratégies par tentatives

Questions liées au retour-en-arrière :

- *L'instant*
 - en cas d'impasse
 - de recherche de toutes les solutions
 - de dépassement d'un seuil
 - de détection d'une contradiction....
- *Le point de retour*
 - application d'un principe déterministe
 - utilisation d'un critère de choix

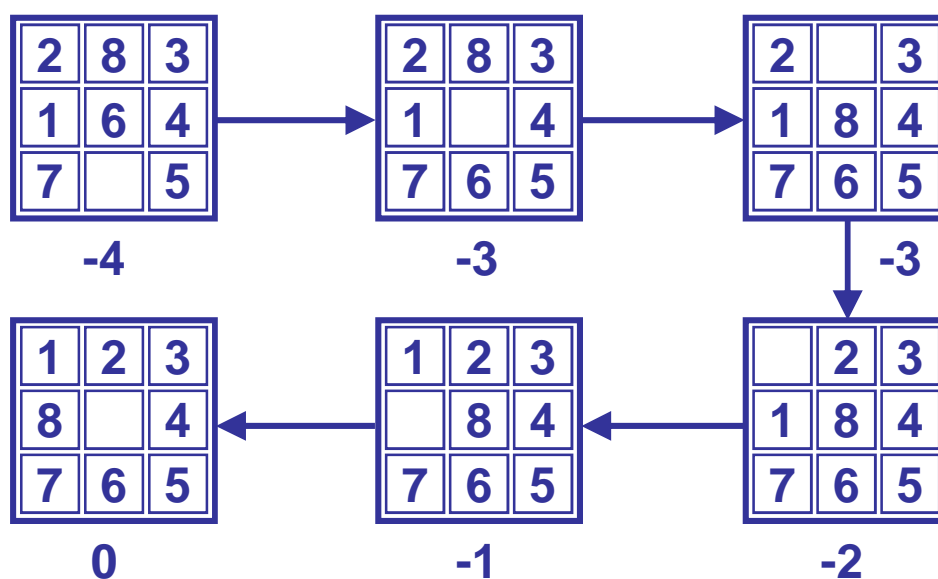
Exemples :

- Backtracking
- À graphe de recherche

Recherche alpine

Principe de base

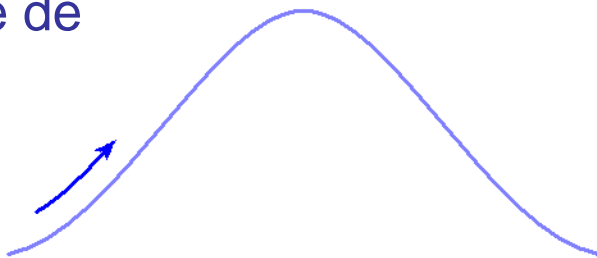
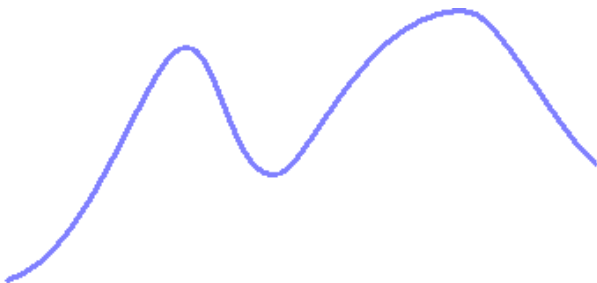
- Stratégies irrévocables, à choix local, Une fonction de potentiel f , mesurant la valeur intrinsèque de chaque état, est à maximiser
- **Algorithme** :
 1. n est le noeud initial
 2. Si la valeur de n est supérieure à celle de ses enfants, retourner n
 3. Sinon, l'enfant de plus haute valeur devient n , et retour en 1



Recherche alpine

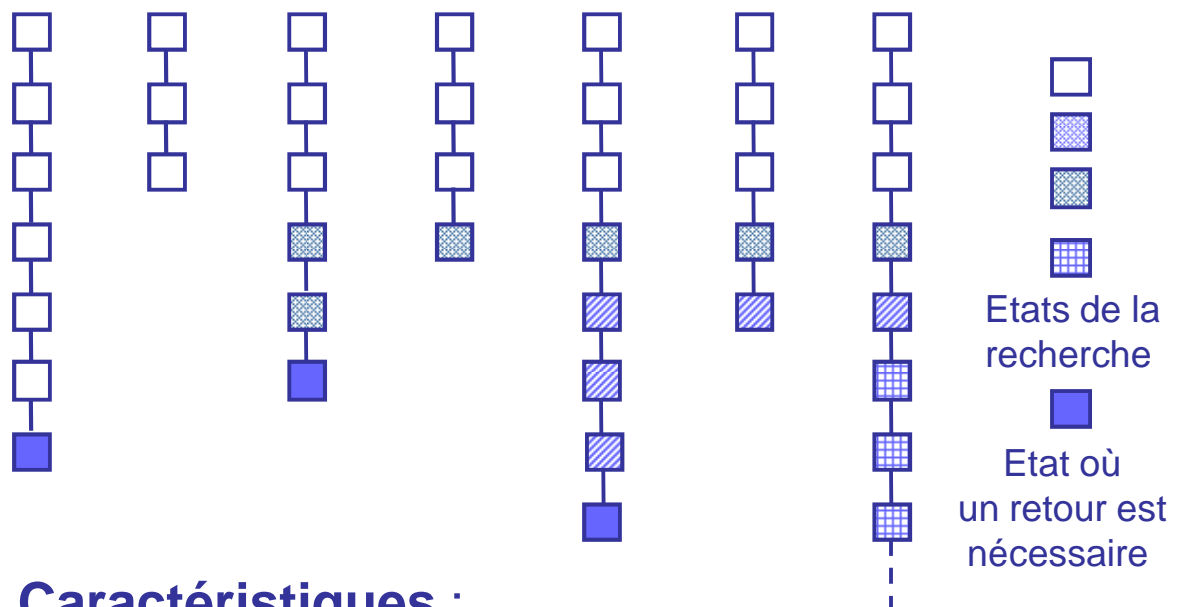
Problèmes

« Revient à grimper l'Everest dans le brouillard et en souffrant d'une amnésie »

- Pour cette topologie de l'espace, obtention d'une solution optimale :
- Risque de solution non optimale si présence de maxima locaux
- **Solution** : faire des grands pas, des sauts
 - Monte-Carlo : par génération de points aléatoires
 - Recuit simulé : avec ajustement d'un paramètre de température

Backtracking

Le backtracking est une stratégie par tentatives qui en cas de retour en-arrière renvoie sur un état situé sur le chemin courant



Caractéristiques :

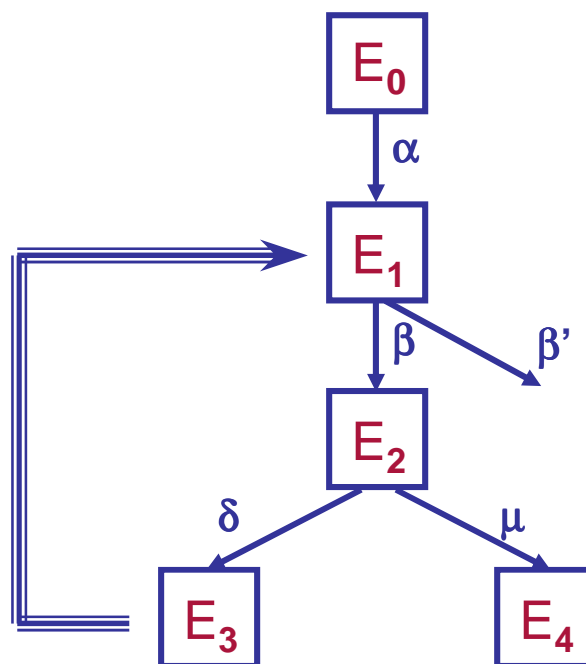
- ❑ États mémorisés : ceux du chemin courant
- ❑ Pas de garantie de trouver une solution
- ❑ Effort de recherche important
- ❑ Utilisation d'un seuil, de critères d'arrêt de développement d'une branche, etc....

Backtracking

Problèmes liés au retour-en-arrière

TRASHING

→ pour le backtracking chronologique



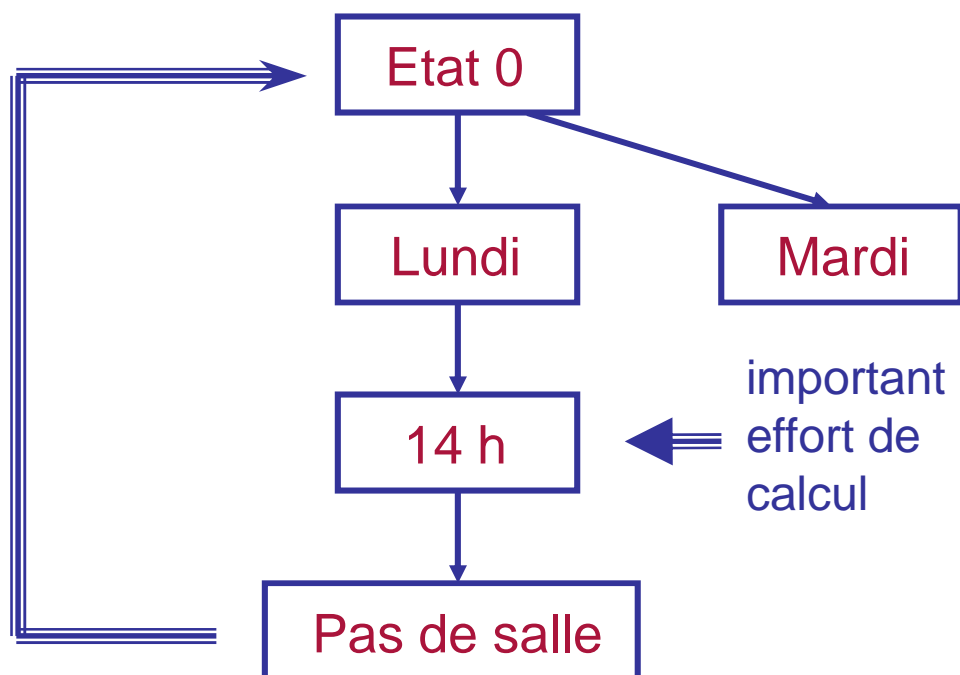
Si l'état E_3 contient une contradiction, due à l'application successive de α et β , il est alors inutile d'appliquer μ

Backtracking

Problèmes liés au retour-en-arrière

REVISION DES CONNAISSANCES

- ❑ Faut-il tout éliminer toutes les inférences ou peut-on en conserver certaines ?
- ❑ Cas de la date d'une réunion :



→ Systèmes de maintien du raisonnement



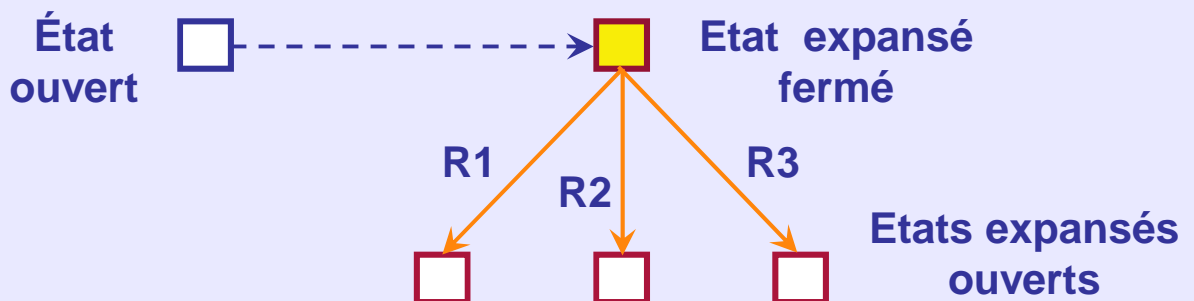
Espace d'états : stratégies à graphe de recherche

Graphe de recherche

Introduction

Une stratégie à graphe de recherche est une stratégie par tentatives qui mémorise les états rencontrés dans un graphe d'états, construit au cours de la résolution selon la stratégie choisie

Principe d'expansion :



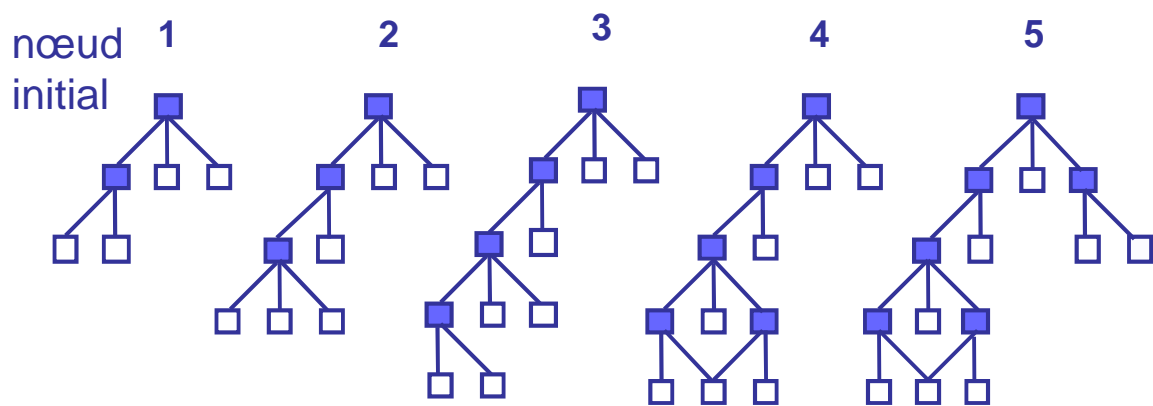
Principe de choix :



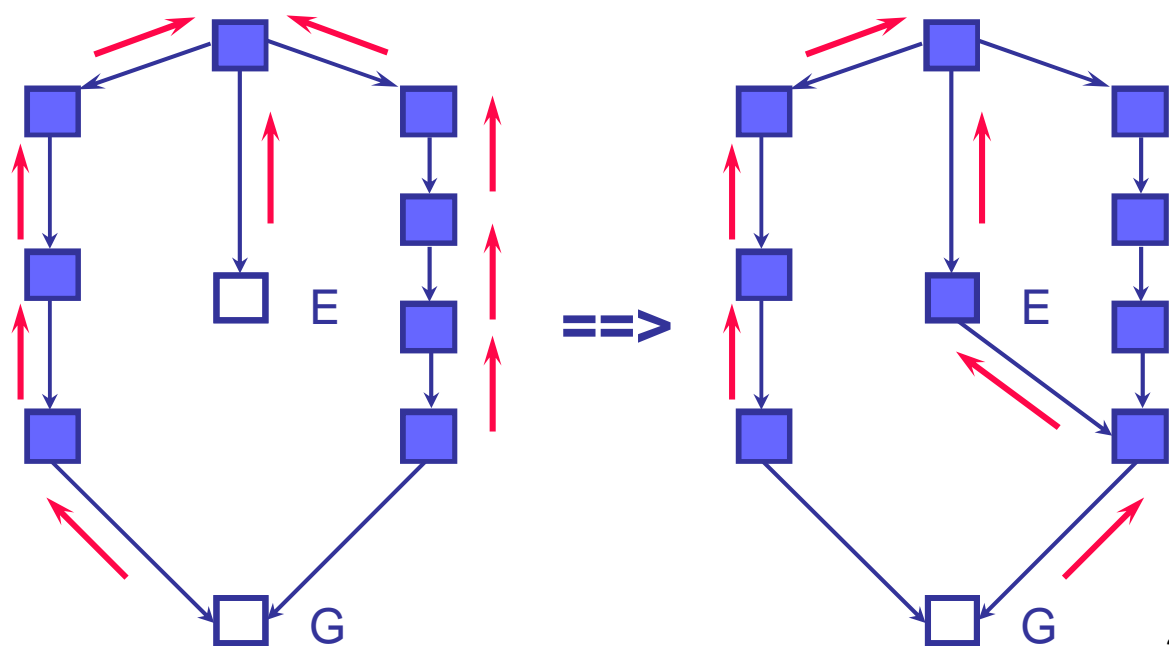
Graphe de recherche

Développement d'arborescences

- D'un graphe de recherche :



- Simultanément d'un arbre de recherche précisant les meilleurs chemins :



Graphe de recherche

Création des arborescences : algorithme

Soit E un ensemble de nœuds ouverts, G le graphe courant de recherche, et s l'état initial.

1. Sélection d'un nœud n de E
2. Si c'est un but, tracer la solution en suivant les pointeurs de n vers s
3. Expanser le nœud n en traçant tous les successeurs qui ne sont pas des ancêtres de n
4. Établir un pointeur entre les successeurs qui n'étaient pas dans G et n
5. Pour chaque successeur de n qui était déjà dans G , voir s'il faut diriger son pointeur vers n
6. Pour chaque successeur déjà expansé, voir s'il faut "rediriger" ses descendants

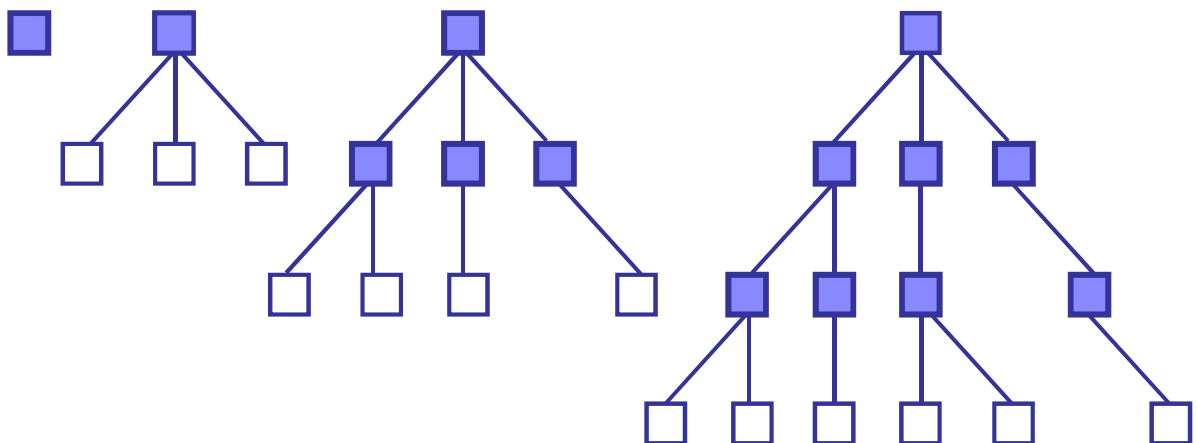
Stratégies aveugles

Stratégies qui se fondent sur le passé, et ne tiennent pas compte du domaine ou du problème

Stratégies en largeur d'abord (fifo)

Expansion en premier des nœuds les plus anciens, ceux-ci étant ordonnés arbitrairement

- Garantissent une solution et même la solution la plus courte
- Complexité exponentielle (temps, mémoire)

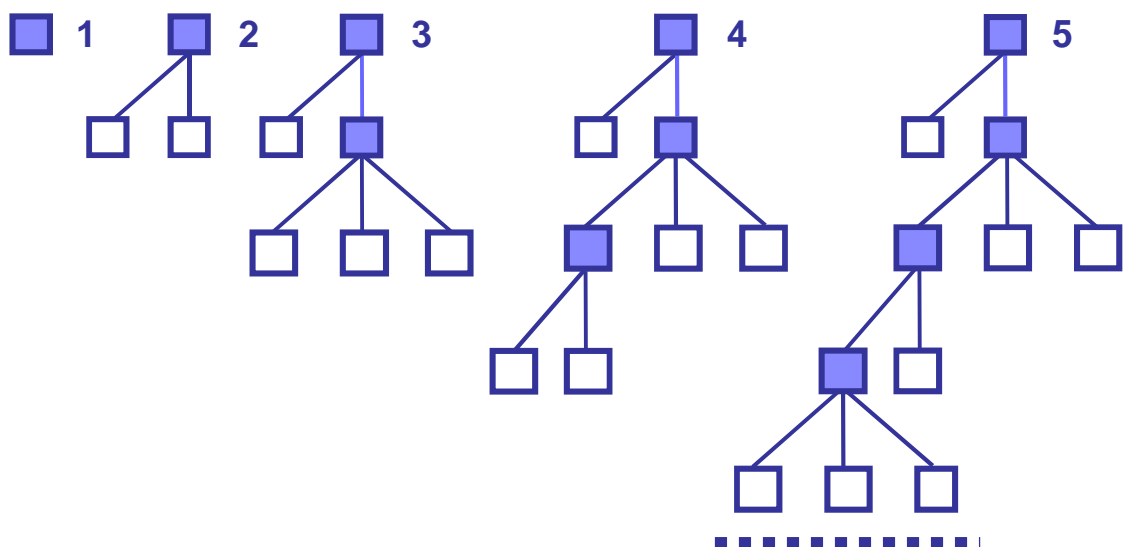


Stratégies aveugles

Stratégies en profondeur d'abord (lifo)

Expansion en premier des nœuds les plus récents, ceux-ci étant ordonnés arbitrairement

- Principe : la *focalisation de l'attention*
- Pas de solution garantie, notamment en présence de chemins de longueur infinie (fixation d'un seuil) ou de cycles
- Peuvent être inefficaces (buts ignorés)



Recherche heuristique

Recherche basée sur l'utilisation de connaissances ou de fonctions *heuristiques*

Notion d'heuristique

Une heuristique est une règle de l'art ou une fonction qui, utilisée à bon escient, permet de réduire la complexité d'un problème. En contrepartie, elle peut s'avérer parfois inefficace.

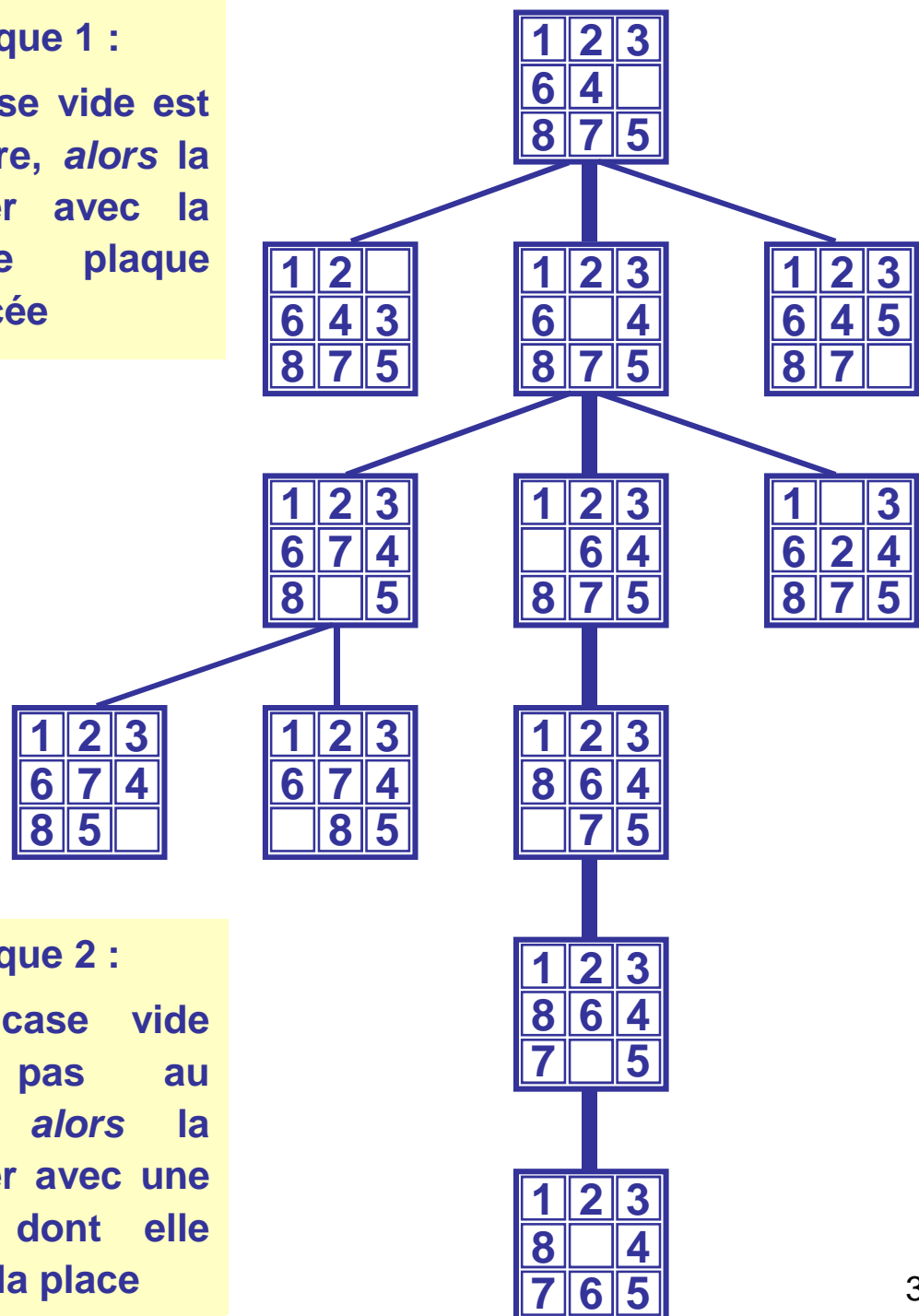
- ❑ Une bonne heuristique « se trompe rarement » et résulte souvent d'une simplification du problème
- ❑ **Exemples** : voyageur de commerce, experts et systèmes experts, calculateurs, sens commun, jeux, etc.....
- ❑ Mise en œuvre : règles, meilleur en premier

Recherche heuristique

Règles heuristiques

Heuristique 1 :

Si la case vide est au centre, *alors* la permuter avec la première plaque mal placée



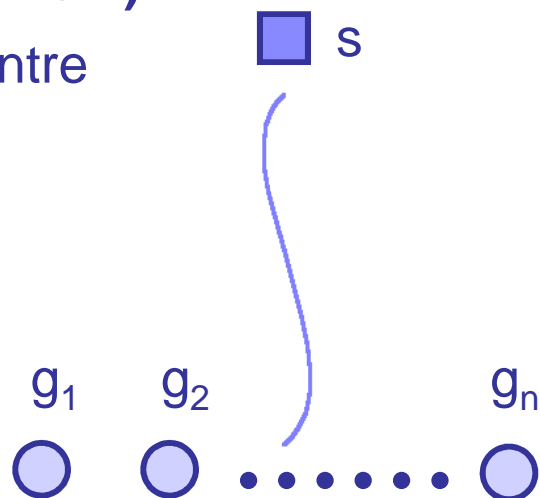
Heuristique 2 :

Si la case vide n'est pas au centre, *alors* la permuter avec une plaque dont elle occupe la place

Meilleur en premier

Principes

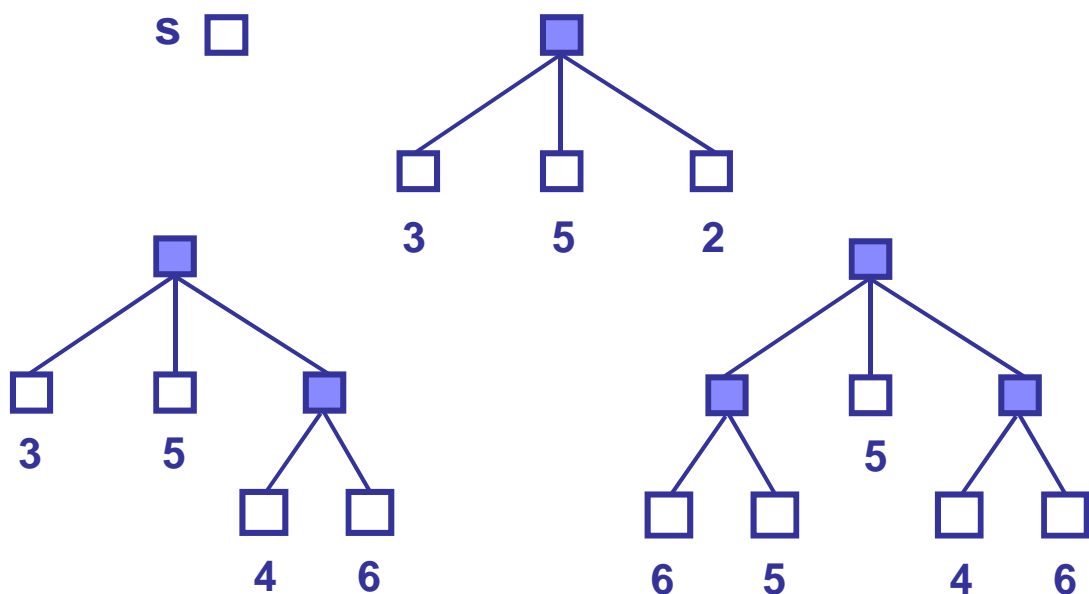
- Forme de recherche heuristique importante :
 - par ses applications
 - par son support théorique
- Elle repose sur une « prévision de l'avenir », par la prise en compte du(es) but(s)
- Les algorithmes explorent le meilleur chemin, quitte à l'abandonner s'il devient moins intéressant que d'autres chemins
- **Objectifs (concomitants ?) :**
 - Trouver un chemin entre s et un but
 - Trouver le meilleur chemin
 - Minimiser l'effort de recherche



Algorithme A

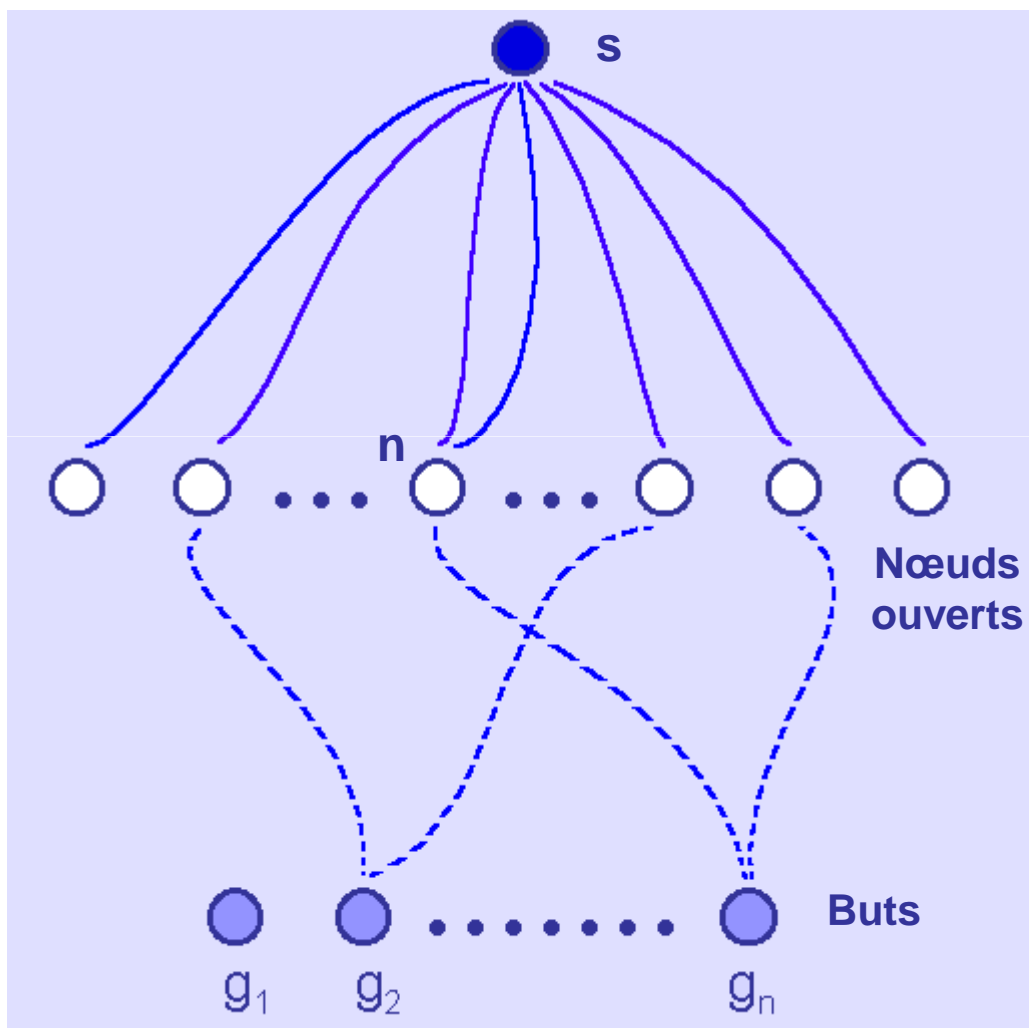
Principes de l'algorithme A

- De type « meilleur en premier »
- On utilise une fonction d'évaluation f à caractère heuristique
- La frontière (ensemble des états ouverts) est alors ordonnée par valeurs croissantes
- On expande alors l'état de plus faible valeur
- **Exemple :**



Algorithme A

Construction de f



- Coût d'un chemin : Σ coûts des arcs (≥ 0)
- Meilleur chemin : celui de coût minimal

Algorithme A

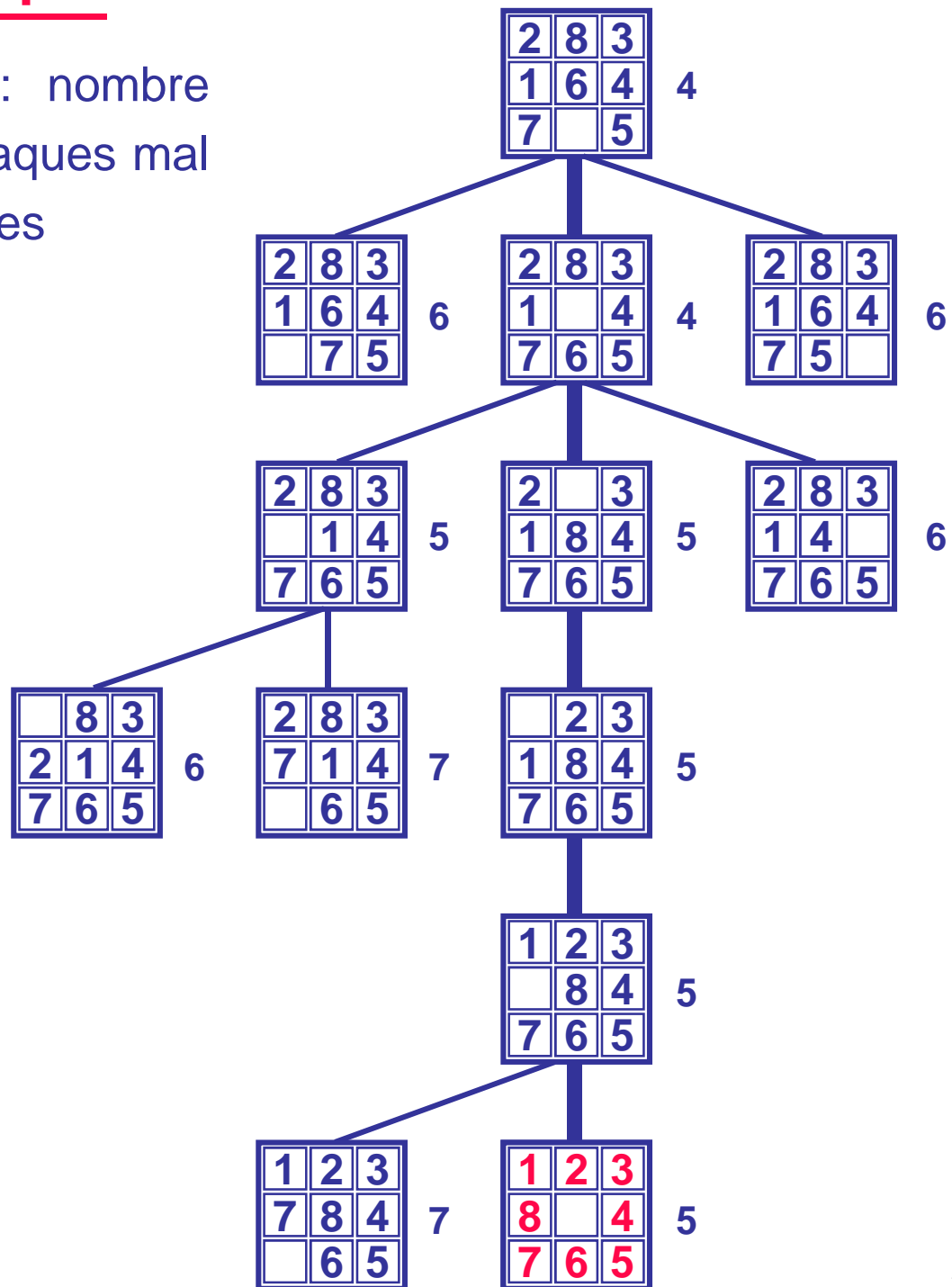
Construction de f

- $f^*(n)$ coût réel du chemin de coût minimum passant par n : $f^*(n) = d^*(n) + h^*(n)$
- $f(n)$: estimation de ce coût réel
$$f(n) = d(n) + h(n)$$
 - $d(n)$ estimation du coût $d^*(n)$ du chemin de coût minimum entre s et n
 - $h(n)$ estimation du coût $h^*(n)$ du chemin de coût minimum entre n et l'un des buts
- $d(n)$: somme minimale des coûts des arcs déjà développés entre s et n
 - $d(n) \geq d^*(n)$
 - calcul de $d(n)$ par un algorithme
- $h(n)$: choisi après analyse du problème. Ce choix est déterminant !

Algorithme A

Exemple

$h(n)$: nombre
de plaques mal
placées



Algorithme A

Choix de l'heuristique h

h est *minorante* ssi, pour tout nœud n , on a :
 $h(n) \leq h^*(n)$. A est alors appelé A*

Un algorithme est *admissible* s'il trouve toujours la solution optimale (si elle existe)

Propriété : si le facteur de branchement est fini, et si les coûts élémentaires sont strictement positifs, l'algorithme A* est admissible

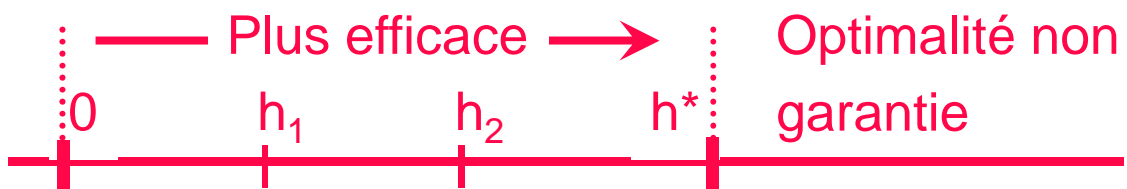
h^* étant inconnue, comment choisir h ?

- $h = 0$: développement proche du largeur d'abord, donc heuristique très inefficace
- h trop élevé : solution optimale peut-être non découverte
- Comment prouver que h est minorante ?

Algorithme A

■ Comparaison de deux heuristiques

A^* muni de h_2 développe moins de nœuds que A^* muni de h_1 si et seulement si, pour tout nœud n , on a : $h_1(n) \leq h_2(n)$



■ Condition de restriction monotone

Condition : pour tous nœuds n_1 et n_2 tels que n_2 est fils de n_1 , on a : $h(n_1) - h(n_2) \leq c(n_1, n_2)$

Propriétés :

- Si h vérifie cette condition et si pour tout état n terminal $h(n)=0$, alors A est admissible
- Quand un nœud est *choisi*, son chemin optimal est alors déjà trouvé

Algorithme A

Complexité

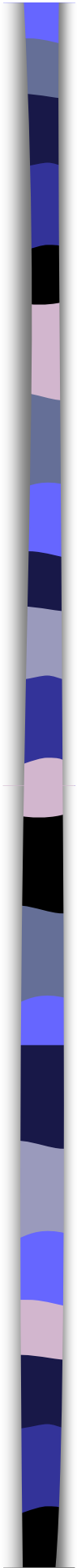
Soit B le nombre de branches par nœud, L la longueur de chemin, N le nombre de nœuds à explorer.

On a : $N = O(B^L)$

Problématique :

- Plus l'algorithme est informé, plus le nombre de nœuds à explorer est faible
- En revanche, cette information requiert un temps de calcul plus grand

Donc : recherche d'un compromis entre le temps consacré à l'exploration et le temps consacré à l'évaluation



Problématique (option)

Le sens de la recherche

Trois formes :

- *Déplacement en-avant* : de l'état initial vers un but (orienté par les données)
- *Déplacement en-arrière* : des buts vers l'état initial (orienté par les buts)
- *Déplacement bidirectionnel* : simultanément à partir de l'état initial et d'un but. Problème : les frontières se rencontrent-elles ?

Exemples :

Raisonnement mathématique (progressif et régressif)... Systèmes à base de règles, GPS...

Choix d'un sens :

- Complexité de la recherche en b^k (b facteur de branchement) pour les déplacements mono-directionnels
- selon la topologie de l'espace de recherche