

IA41

**Concepts fondamentaux en Intelligence Artificielle
et langages dédiés**

CM #10

**Les stratégies de recherche
dans les graphes :
A* & Co.**

Fabrice LAURI

Plan

- **Introduction**
- **Espace d'états et système de production**
- **Définition, types et propriétés des graphes et arbres**
- **Stratégies de recherche**
 - **Stratégies de recherche non informées**
 - **Stratégies de recherche informées (avec heuristique)**
 - **Exemple de résolution d'un problème avec A***
 - **Propriétés des fonctions heuristiques**
- **Mesure de performances des stratégies de recherche**
- **Analyse de problèmes**

Plan

- **Introduction**
- **Espace d'états et système de production**
- **Définition, types et propriétés des graphes et arbres**
- **Stratégies de recherche**
 - **Stratégies de recherche non informées**
 - **Stratégies de recherche informées (avec heuristique)**
 - **Exemple de résolution d'un problème avec A***
 - **Propriétés des fonctions heuristiques**
- **Mesure de performances des stratégies de recherche**
- **Analyse de problèmes**

Introduction

En IA, certains problèmes peuvent être formalisés de manière très précise :

- les puzzles (tours de Hanoi, taquin, missionnaires & cannibales)
- la démonstration de théorèmes, par ex. en (LP) ou (PP)
- problème du type « représentant de commerce »
- jeux à deux adversaires (othello, dames, échecs, etc.) : déterminer une séquence de mouvements gagnants ou le coup gagnant
- planification de chemins : trouver le chemin d'un point à un autre dans un environnement discret en évitant les obstacles
- ...

Introduction

La résolution d'un problème **bien formalisé** revient généralement à :

- 1) définir l'**espace d'états** du problème
- 2) déterminer l'**état initial** et les **états finaux**
- 3) définir le **système de production** pour se déplacer dans l'espace
- 4) définir/choisir la **stratégie de résolution** pour obtenir une solution

Plan

- **Introduction**
- **Espace d'états et système de production**
- **Définition, types et propriétés des graphes et arbres**
- **Stratégies de recherche**
 - **Stratégies de recherche non informées**
 - **Stratégies de recherche informées (avec heuristique)**
 - **Exemple de résolution d'un problème avec A***
 - **Propriétés des fonctions heuristiques**
- **Mesure de performances des stratégies de recherche**
- **Analyse de problèmes**

Espace d'états

Définition d'un état

Ensemble des variables représentant les paramètres du problème.

A un état s sera associé un ensemble de règles permettant de passer de s vers d'autres états.

Définition d'un espace d'états

Graphe orienté dans lequel :

- un noeud = un état du problème
- un arc = une transition entre deux états

Etat initial et états finaux

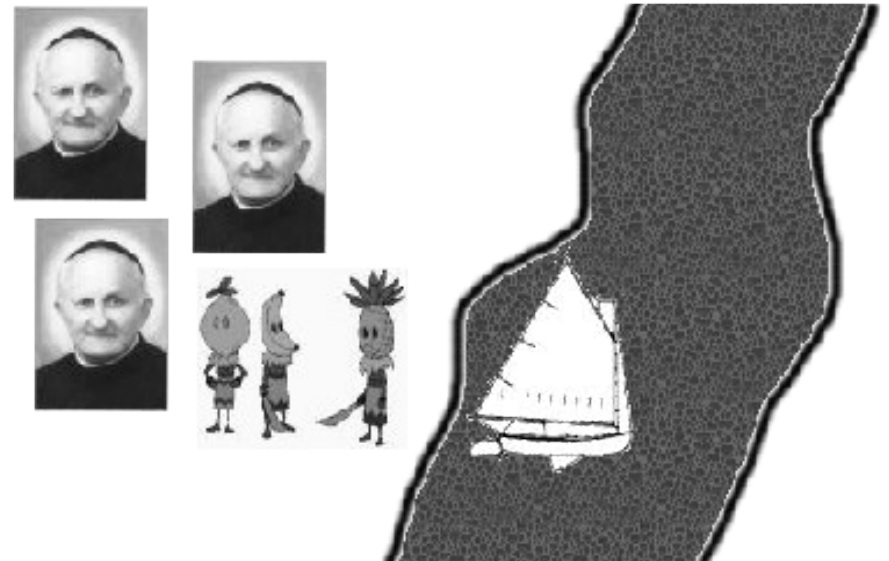
La résolution d'un problème ne concerne qu'un seul état initial et éventuellement plusieurs états finaux.

Des états initiaux différents impliquent des instances différentes d'un même problème.

Exemple #1 : les missionnaires et les cannibales

Questions :

- Comment peut être codé un état ?
- Etat initial ?
- Ensemble des états finaux ?



Exemple #2 : démonstration de théorèmes

Questions :

- Comment peut être codé un état ?
- Etat initial ?
- Ensemble des états finaux ?

Exemple #3 : le taquin

Questions :

- Comment peut être codé un état ?
- Etat initial ?
- Ensemble des états finaux ?

2	8	3
1	6	4
7		5

Exemple #4 : le jeu d'échecs

Questions :

- Comment peut être codé un état ?
- Etat initial ?
- Ensemble des états finaux ?



Systeme de production (1/2)

Définition d'un système de production

Ensemble de *règles* qui permettent, à partir d'un état donné, de générer l'ensemble des états que l'on peut légalement atteindre à partir de lui.

Raisonnement déductif, en chaînage avant

Partir de l'état initial à l'état final, en appliquant les règles de production.

Raisonnement inductif, en chaînage arrière

Passer d'un état final à l'état initial en respectant les règles de production.

Choix du raisonnement ? En fonction du facteur de branchement.

Chaînage mixte : par exemple, démonstration de théorèmes.

Systeme de production (2/2)

Définition d'un système monotonique

Systeme monotonique si l'application d'une règle à l'instant t laisse toutes les autres règles applicables à l'instant $t' > t$.

Systeme partiellement commutatif

Soient X un état, et une séquence $S = (s_1, s_2, \dots, s_n)$ de règles de production telles que :

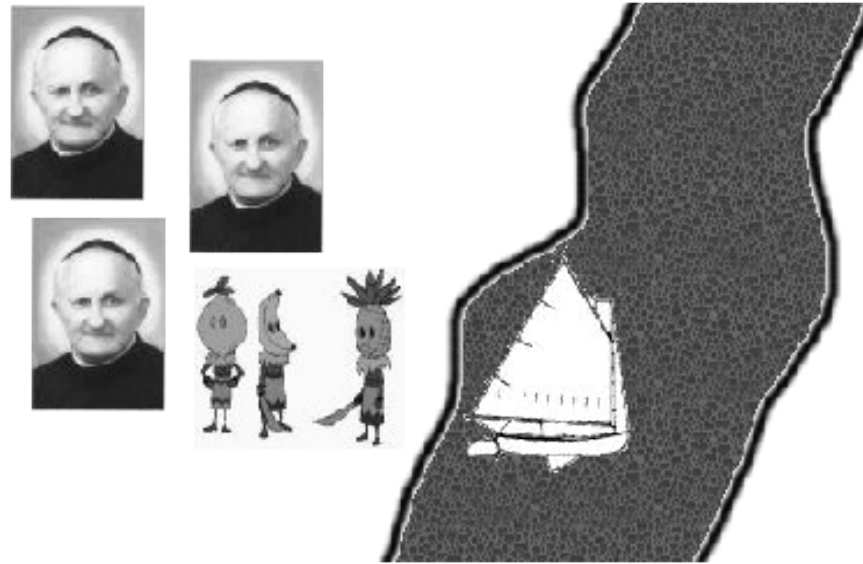
$$X \xrightarrow{s_1} X_1 \xrightarrow{s_2} X_2 \xrightarrow{s_3} \dots \xrightarrow{s_{n-1}} X_{n-1} \xrightarrow{s_n} Y$$

Systeme partiellement commutatif s'il existe une permutation S^* de S telle que Y puisse être atteint à partir de X avec S^* .

Systeme commutatif

Systeme monotonique + partiellement commutatif

Exemple de système de production : les missionnaires et les cannibales



Un exemple d'état : (3,3,G)

Exemple de système de production : le taquin

2	8	3
1	6	4
7		5

Un exemple d'état initial : ((2,3),(2,8,3,1,6,4,7,0,5))

Exemple de système de production : le jeu d'échecs



Un exemple d'état...

Représentations de l'espace d'états construits par un système de production

Arbres

Etats reliés entre eux **sans** test d'occurrence.

Avantage : rapidité de génération

Inconvénients : duplication d'états, bouclage

Graphes OU

Etats reliés entre eux **avec** test d'occurrence.

Duplication en creux des avantages et inconvénients des arbres.

Graphes ET-OU

Contient des arcs OU et des arcs ET (plusieurs arcs reliés entre eux).

Pour résoudre certains noeuds, il faut résoudre tous les noeuds reliés par un arc ET.

Plan

- **Introduction**
- **Espace d'états et système de production**
- **Définition, types et propriétés des graphes et arbres**
- **Stratégies de recherche**
 - **Stratégies de recherche non informées**
 - **Stratégies de recherche informées (avec heuristique)**
 - **Exemple de résolution d'un problème avec A***
 - **Propriétés des fonctions heuristiques**
- **Mesure de performances des stratégies de recherche**
- **Analyse de problèmes**

Définition d'un graphe

Un graphe fini $G = (V, E)$ est défini par :

- un ensemble V fini $V = \{ v_1, v_2, \dots, v_n \}$ dont les éléments sont appelés **noeuds** (ou sommets),
- un ensemble E fini $E = \{ e_1, e_2, \dots, e_m \}$ dont les éléments sont appelés **arêtes** (graphes non orientés) ou **arcs** (graphes orientés)

Une arête e de l'ensemble E est définie par une paire **non ordonnée** de sommets, appelés les extrémités de e .

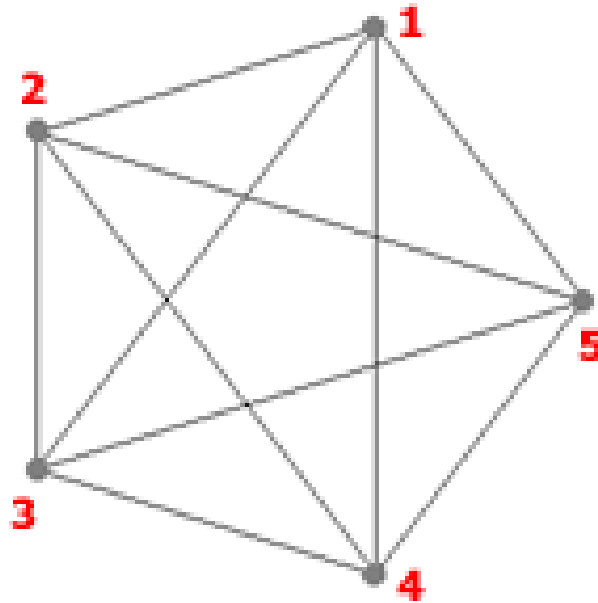
Si l'arête e relie les sommets a et b , on dira que ces sommets sont **adjacents**, ou **incidents avec e** , ou encore que l'arête e est incidente avec les sommets a et b .

Un arc e de l'ensemble E est défini par une paire **ordonnée** de sommets. Lorsque $e = (u, v)$, on dira que l'arc e va de u à v .

On dit aussi que u est l'extrémité initiale et v l'extrémité finale de e .

Un type de graphe particulier

Graphe complet : graphe dans lequel chaque sommet est relié directement à tous les autres sommets.



Graphe complet
d'ordre 5

Quelques définitions (1/5)

En théorie des graphes :

- **Ordre d'un graphe** = le nombre de noeuds du graphe.
- **Degré d'un noeud** = le nombre d'arêtes incidentes avec ce noeud. On note $d(v)$ le degré du noeud v .
- **Degré d'un graphe** : degré maximum de tous ses sommets.

Dans les stratégies de recherche :

- **Facteur de branchement d'un noeud** = degré d'un noeud, c-à-d nombre de décisions possibles (fils) à partir de ce noeud.

Quelques définitions (2/5)

Définition d'une *chaîne / chemin* (g. non orienté/g. orienté)

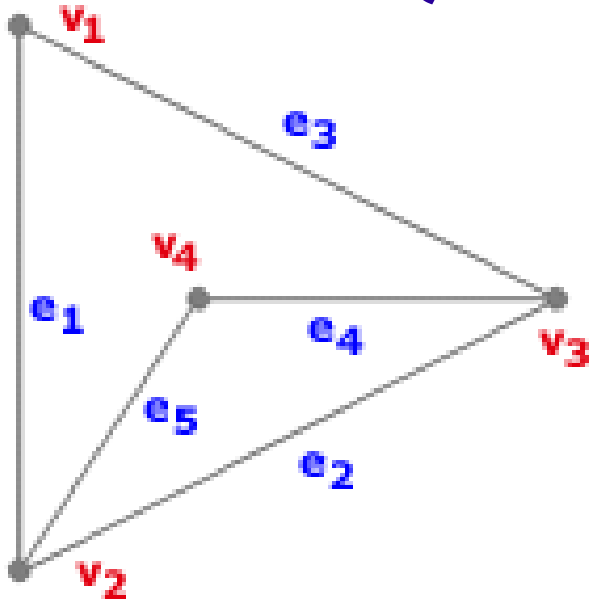
Suite de la forme $(v_0, e_1, v_1, e_2, \dots, v_{k-1}, e_k, v_k)$, ayant pour éléments alternativement des sommets (v_i) et des arêtes/arcs (e_i) , commençant et se terminant par un sommet, et telle que les extrémités de e_i sont v_{i-1} et v_i , $i = 1, \dots, k$.

Si $v_0 = a$ et $v_k = b$, on dira que la chaîne/chemin relie a et b .

En outre, on dira que la chaîne/chemin est de longueur k , k étant le nombre d'arêtes/arcs de la chaîne/chemin.

Une chaîne/chemin doit comporter au moins une arête/arc.

Quelques définitions (3/5)



Ce graphe contient par ex. les chaînes
(v_1, e_3, v_3, e_4, v_4) et
($v_4, e_4, v_3, e_2, v_2, e_1, v_1$).

On ne change pas une chaîne en inversant l'ordre des éléments dans la suite correspondante :

(v_1, e_3, v_3, e_4, v_4) et (v_4, e_4, v_3, e_3, v_1) sont identiques.

On appelle distance entre deux sommets la longueur de la plus petite chaîne les reliant.

On appelle diamètre d'un graphe la plus longue des distances entre deux sommets.

Quelques définitions (4/5)

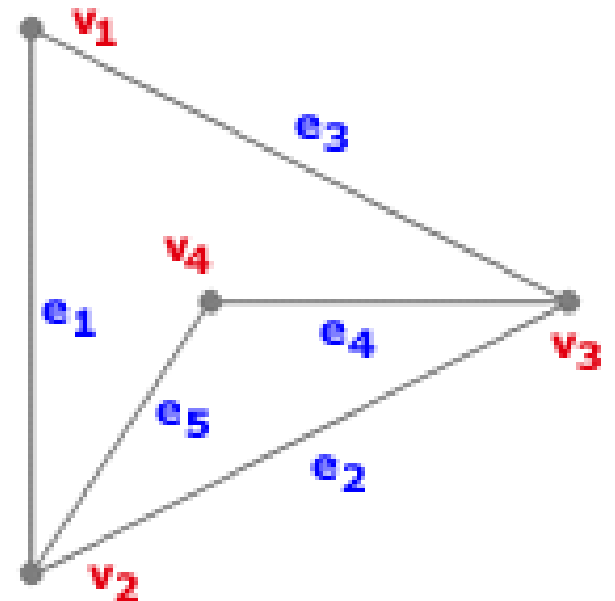
Définition d'une *chaîne élémentaire*

Une chaîne est élémentaire si **chaque sommet** y apparaît au plus une fois.

Définition d'une *chaîne simple*

Une chaîne est simple si **chaque arête** apparaît au plus une fois.

Dans ce graphe,
(v_1, e_1, v_2, e_2, v_3) est une chaîne simple et élémentaire.



Quelques définitions (5/5)

Définition d'une *chaîne fermée* / *circuit* (g. n. o. / g. o.)

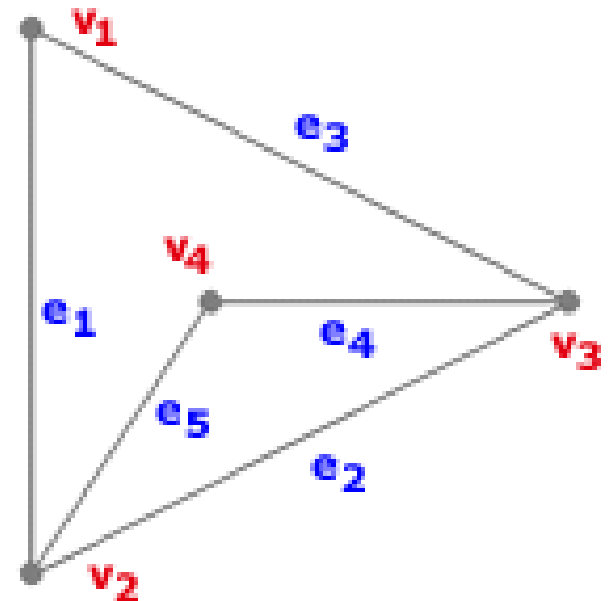
Une chaîne telle que $v_0 = v_k$ est appelée chaîne fermée.

Définition d'un *cycle*

Une chaîne fermée simple est appelée cycle si seul le sommet de départ apparaît deux fois dans la chaîne.

Dans ce graphe,
($v_2, e_2, v_3, e_4, v_4, e_5, v_2$) est un cycle,

($v_2, e_2, v_3, e_4, v_4, e_4, v_3, e_2, v_2$)
est une chaîne fermée.



Définition d'un arbre

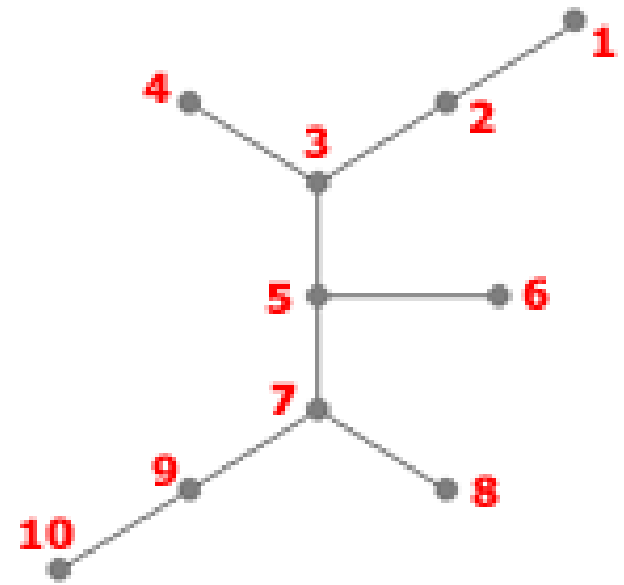
Un arbre est un **graphe connexe sans cycle**.

Théorème :

Soit un graphe $G = (V, E)$ à n sommets.

Les affirmations suivantes sont alors équivalentes :

1. G est un arbre,
2. G est sans cycles et connexe,
3. G est sans cycles et comporte $n-1$ arêtes,
4. G est connexe et comporte $n-1$ arêtes,
5. Chaque paire $\{u, v\}$ de sommets distincts est reliée par une seule chaîne simple (et le graphe est sans boucles).



Un arbre

Plan

- **Introduction**
- **Espace d'états et système de production**
- **Définition, types et propriétés des graphes et arbres**
- **Stratégies de recherche**
 - **Stratégies de recherche non informées**
 - **Stratégies de recherche informées (avec heuristique)**
 - **Exemple de résolution d'un problème avec A***
 - **Propriétés des fonctions heuristiques**
- **Mesure de performances des stratégies de recherche**
- **Analyse de problèmes**

Stratégies de résolution

Techniques de résolution non informées

- Approche du British Museum
- Profondeur d'abord avec retour arrière,
- Largeur d'abord

Techniques de résolution informées

- Meilleur d'abord
- A^* (graphe OU)
- AO^* (graphe ET-OU)

Technique de recherche dans un arbre de jeu

- Algorithme *MinMax*,
- Procédure *Alpha-Beta*

Espace de recherche

Chaque stratégie/algorithmes de recherche est caractérisé par l'espace de recherche qu'il explore.

Définition d'un espace de recherche

Un espace de recherche est l'ensemble des chemins possibles d'un graphe d'états.

Un chemin = une solution

Principes des algorithmes

Ces algorithmes procèdent par expansions successives de l'espace de recherche.

Etape d'expansion

- Choisir l'extrémité n d'un chemin de l'espace de recherche
- Etendre ce chemin en générant des (ou tous les) successeurs du noeud n .

Plan

- **Introduction**
- **Espace d'états et système de production**
- **Définition, types et propriétés des graphes et arbres**
- **Stratégies de recherche**
 - **Stratégies de recherche non informées**
 - **Stratégies de recherche informées (avec heuristique)**
 - **Exemple de résolution d'un problème avec A***
 - **Propriétés des fonctions heuristiques**
- **Mesure de performances des stratégies de recherche**
- **Analyse de problèmes**

Approche *British Museum*

Tire son nom du postulat théorique qu'un singe pourrait reproduire en intégralité tous les ouvrages du *British Museum*, si on lui en laissait le temps...

Dans le jargon informatique, dire qu'un algorithme est de type *British Museum* revient à dire qu'il est mauvais !

Idée

Examiner toutes les possibilités les unes après les autres et choisir la meilleure.

Principe

A chaque étape, on génère un seul noeud en choisissant au hasard une règle de production. A partir de ce noeud, on itère le processus jusqu'à obtention éventuelle d'une solution.

Si à un instant donné, on aboutit à une impasse, on recommence l'opération **depuis la racine...**

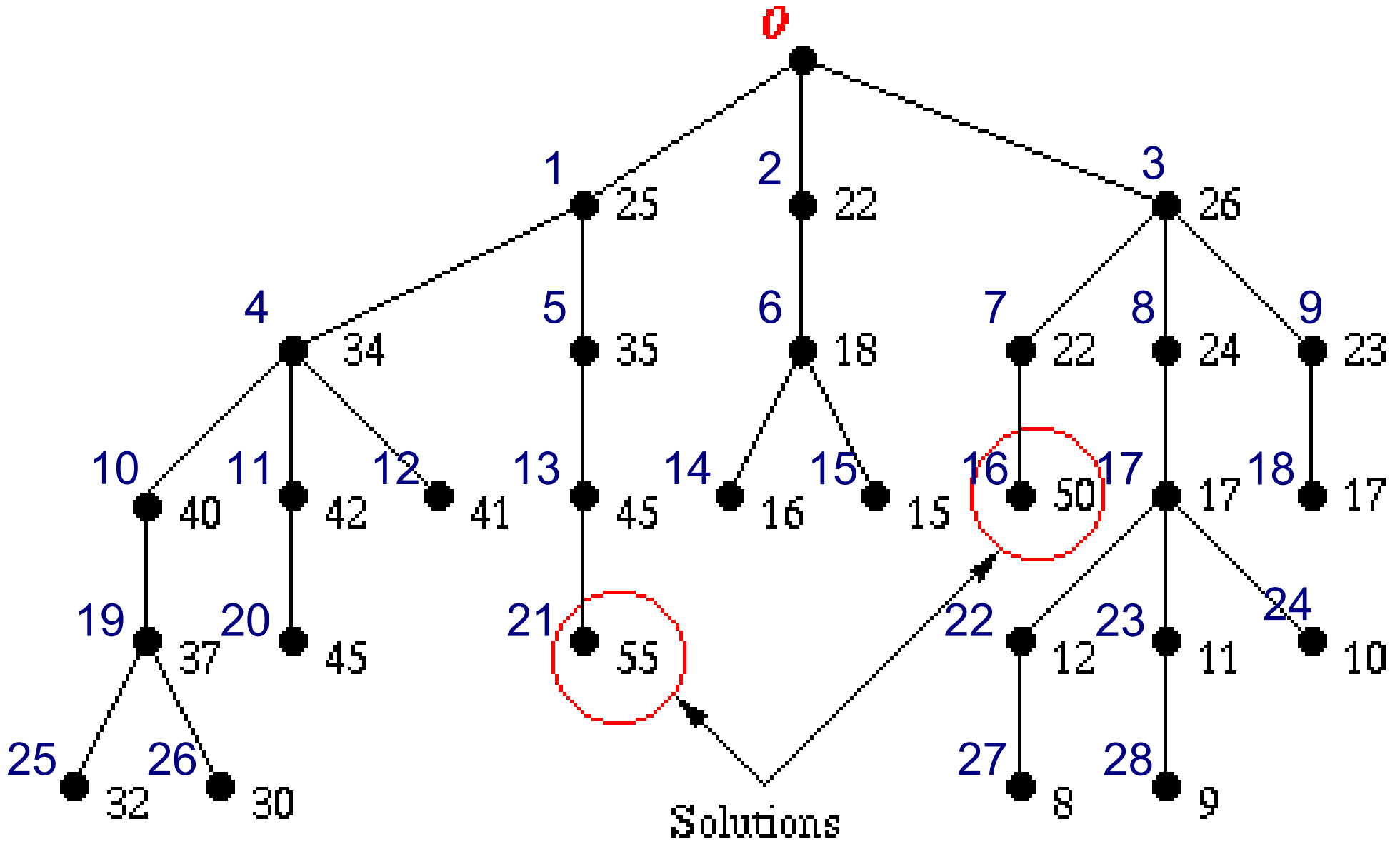
Largeur d'abord (*Breadth-First Search*)

BFS étend et examine systématiquement tous les noeuds d'un graphe à la recherche d'une solution. La recherche est exhaustive. BFS n'utilise pas d'heuristique.

Algorithme :

- 1) Créer une **liste FIFO** formée d'un seul élément : le noeud racine.
- 2) Faire jusqu'à ce que la liste soit vide ou que le but soit atteint
Si la tête de la liste n'est pas le but Alors
 Enlever le noeud en tête de liste
 Ajouter en **queue** de liste tous les fils du noeud supprimé, s'il en existe.
 Pour chaque noeud ajouté, spécifiez son noeud père.
 FinSi
 FinFaireJusqu'à
- 3) Si le but a été trouvé Alors **Succès** Sinon **Echec**.

Largeur d'abord (*Breadth-First Search*)



Ordre dans lesquels les noeuds sont étendus ?

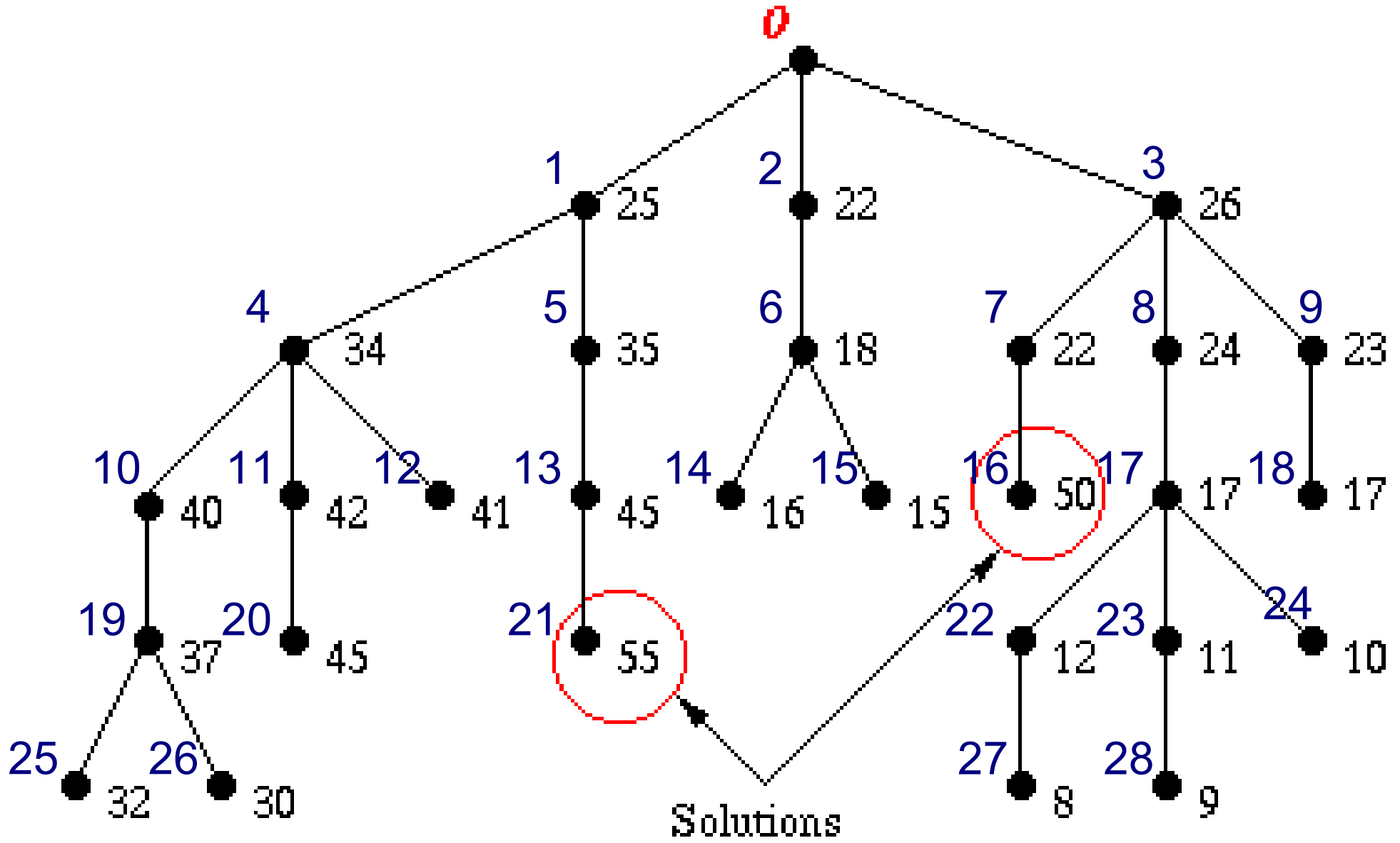
Profondeur d'abord (*Depth-First Search*)

DFS est un algorithme qui étend le noeud d'un graphe et ses successeurs le plus longtemps possible jusqu'à atteindre le noeud but, ou jusqu'à atteindre un noeud n'ayant plus de fils. Dans ce cas la recherche retourne au noeud le plus récent qui n'a pas été encore exploré : *backtracking*.

Algorithme :

- 1) Créer une **liste LIFO** formée d'un seul élément : le noeud racine.
- 2) Faire jusqu'à ce que la liste soit vide ou que le but soit atteint
Si la tête de la liste n'est pas le but Alors
 Enlever le noeud en tête de liste
 Ajouter en **tête** de liste tous les fils du noeud supprimé, s'il en existe.
 Pour chaque noeud ajouté, spécifiez son noeud père.
FinSi
FinFaireJusqu'à
- 3) Si le but a été trouvé Alors **Succès** Sinon **Echec**.

Profondeur d'abord (*Depth-First Search*)



Ordre dans lesquels les noeuds sont étendus ?

Plan

- **Introduction**
- **Espace d'états et système de production**
- **Définition, types et propriétés des graphes et arbres**
- **Stratégies de recherche**
 - **Stratégies de recherche non informées**
 - **Stratégies de recherche informées (avec heuristique)**
 - **Exemple de résolution d'un problème avec A***
 - **Propriétés des fonctions heuristiques**
- **Mesure de performances des stratégies de recherche**
- **Analyse de problèmes**

Propriétés d'une recherche informée

Pourquoi ?

- Eviter le parcours complet de l'espace d'états
- Eviter le parcours complet de l'espace d'actions
- Eviter l'énumération complète des solutions

Comment ?

- Améliorer la recherche en :
 - Explorant d'abord les solutions partielles prometteuses
 - Mettant à l'écart les solutions partielles mauvaises

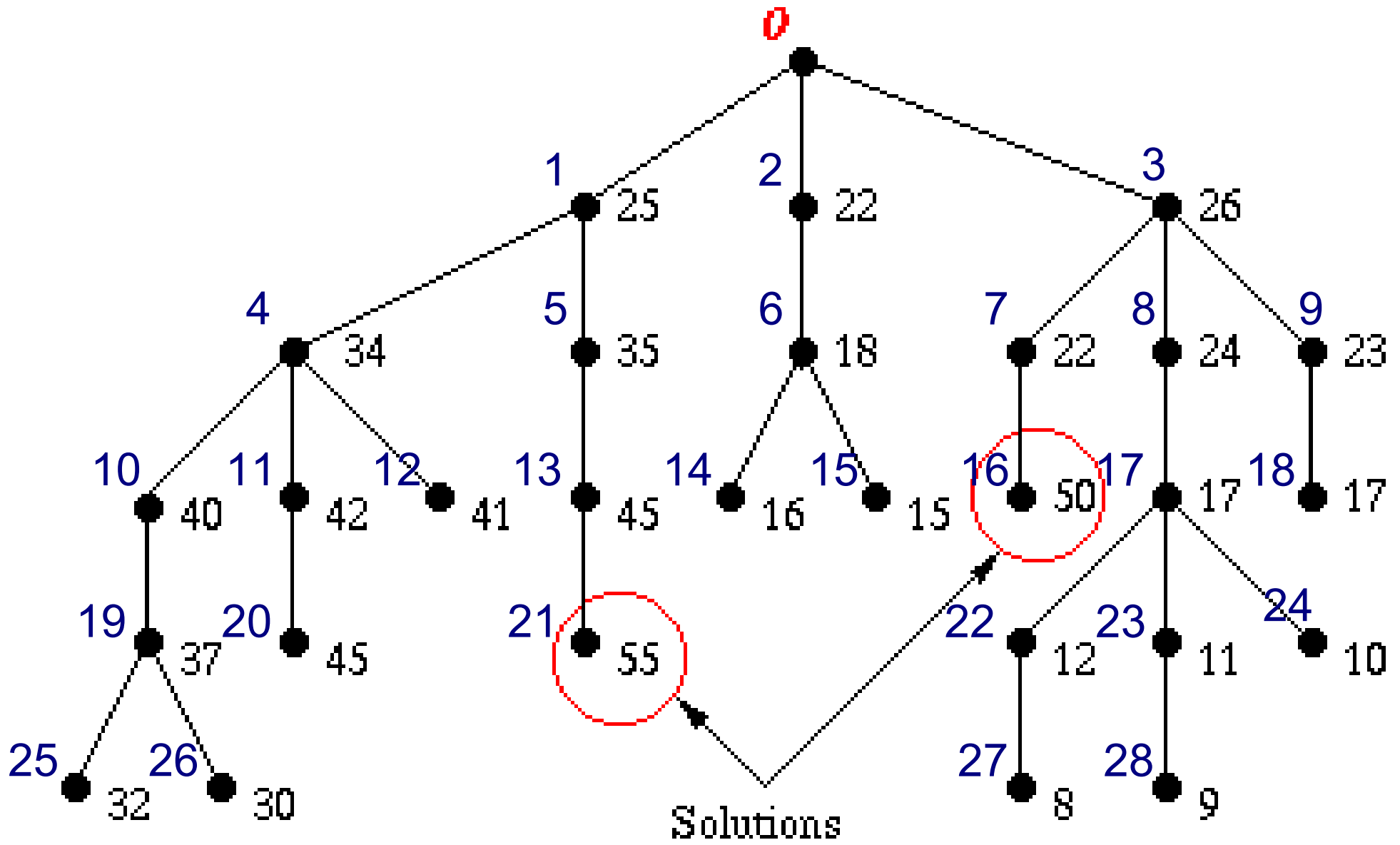
Meilleur d'abord (*Best-First Search*)

A chaque étape, on fait progresser d'un niveau le meilleur chemin (selon la distance estimée), en créant autant de nouveaux chemins qu'il y a de fils à ce noeud.

Algorithme :

- 1) Créer une **liste FIFO** formée d'un seul élément : le noeud racine.
- 2) Faire jusqu'à ce que la liste soit vide ou que le but soit atteint
Si la tête de la liste n'est pas le but Alors
 - Enlever le noeud en tête de liste
 - Ajouter en **queue** de liste tous les fils du noeud supprimé, s'il en existe.
 - Trier toute la liste en estimant la distance restant à parcourirFinSi
FinFaireJusqu'à
- 3) Si le but a été trouvé Alors **Succès** Sinon **Echec**.

Meilleur d'abord (*Best-First Search*)



Ordre dans lesquels les noeuds sont étendus ?

Notion d'heuristique

Du grec *euristikê* : art de découvrir.

Idée

Diriger la recherche dans un espace d'états, de manière à réduire le nombre de noeuds générés et donc le temps de résolution.

Motivation

Une solution satisfaisante suffit.

Fondement

Introduit des mécanismes qui dérivent de connaissances spécifiques au problème.

Une des notions clefs en I.A.

Algorithme A* (1/2)

(Hart & Nilsson & Raphael, 1968)

Objectif

Rechercher le **plus court chemin** dans un graphe pour aller au noeud but E en partant d'un noeud initial S.

Principe

Utilise une **heuristique** $h(x)$, qui donne une **estimation** de ce qu'il reste à parcourir à partir du noeud x pour arriver au noeud but E.

A chaque noeud x est associée une valeur $f(x) = g(x) + h(x)$, où $g(x)$ représente le coût du chemin pour arriver jusqu'à x .

$c(x,y)$ représente le coût de passage du noeud x vers le noeud y .

Les noeuds sont visités selon la valeur $f(x)$, en privilégiant les noeuds x pour lesquels $f(x)$ est la plus petite.

Algorithme A* (2/2)

$Open \leftarrow \{ s \}; Closed \leftarrow \emptyset; g(s) \leftarrow 0; f(s) \leftarrow h(s)$

Tant que $Open \neq \emptyset$ Faire

Extraire de $Open$ l'**élément x** tel que **$f(x)$ est minimale**

Insérer x dans $Closed$

Si $x \in T$ Alors Fini : Solution trouvée

Sinon

Pour tout y successeur de x Faire

Si $y \notin Closed \cup Open$ Ou $g(y) > g(x) + c(x,y)$ Alors

$g(y) \leftarrow g(x) + c(x,y)$

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

$père(y) \leftarrow x$

Insérer-selon- f y dans $Open$

FinSi

FinPour

FinSi

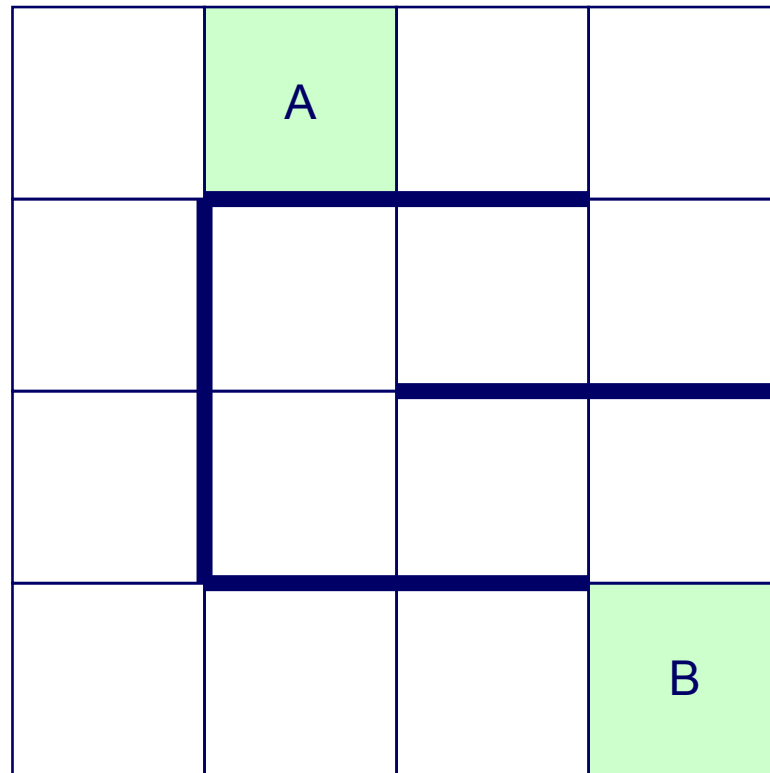
FinTantQue

Plan

- Introduction
- Espace d'états et système de production
- Définition, types et propriétés des graphes et arbres
- Stratégies de recherche
 - Stratégies de recherche non informées
 - Stratégies de recherche informées (avec heuristique)
 - **Exemple de résolution d'un problème avec A***
 - Propriétés des fonctions heuristiques
- Mesure de performances des stratégies de recherche
- Analyse de problèmes

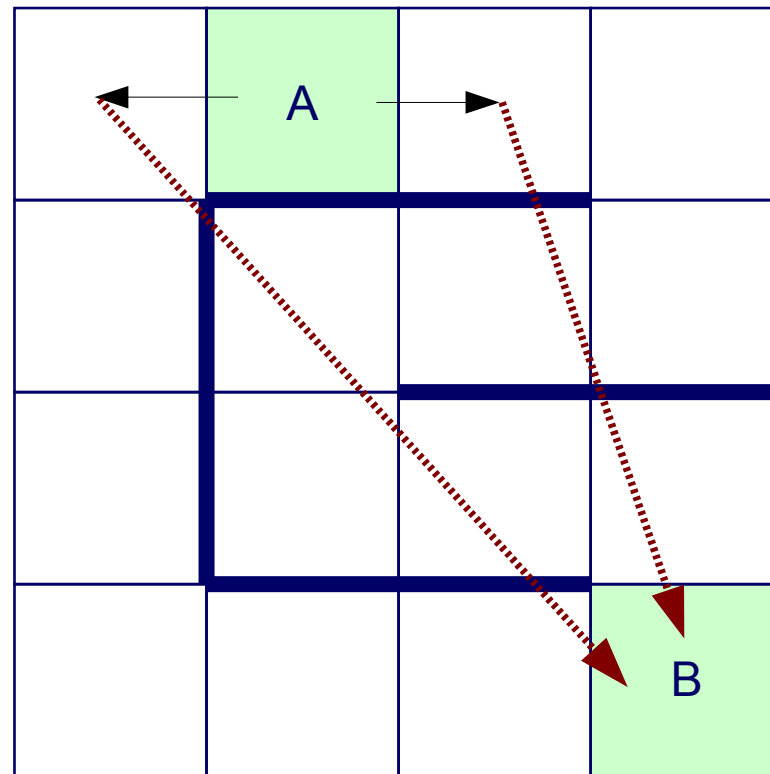
Exemple : recherche du plus court chemin dans un labyrinthe (1/14)

On recherche le plus court chemin entre A et B



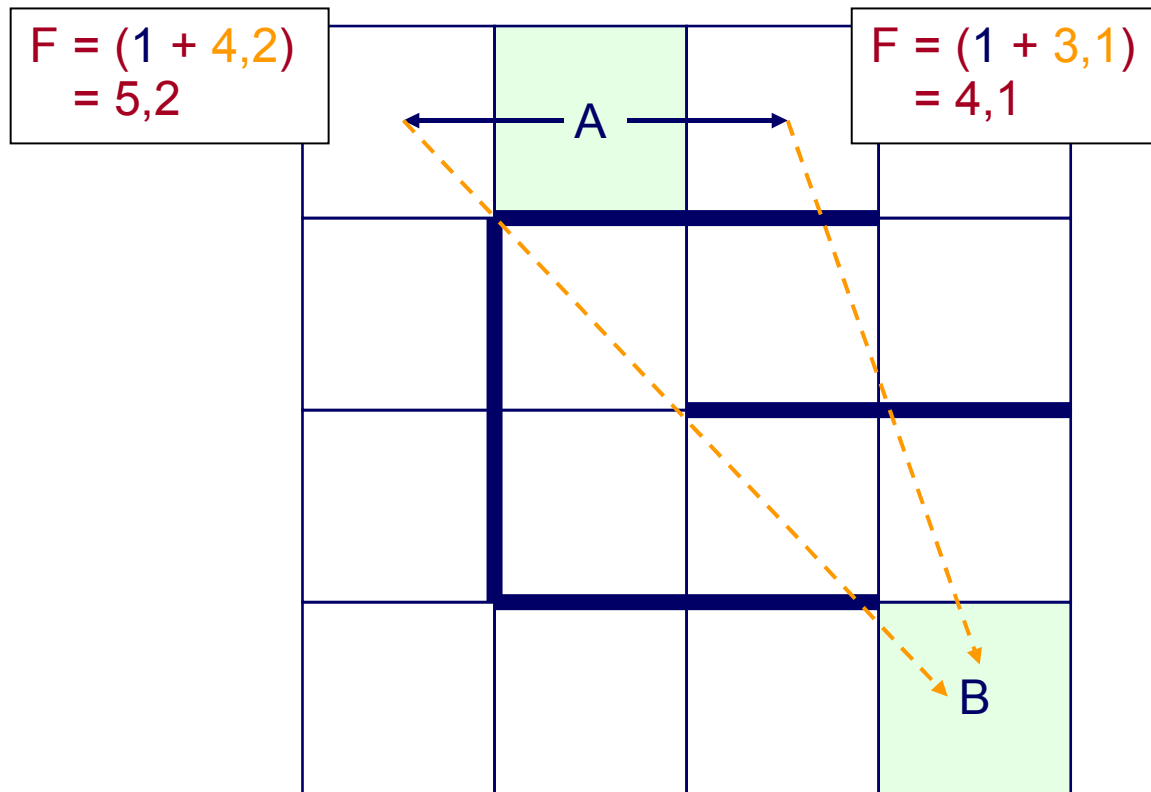
Exemple : recherche du plus court chemin dans un labyrinthe (3/14)

- coût $x \rightarrow y$: $c(x,y) = 1$
- evaluation : $f(x) = g(x) + h(x)$

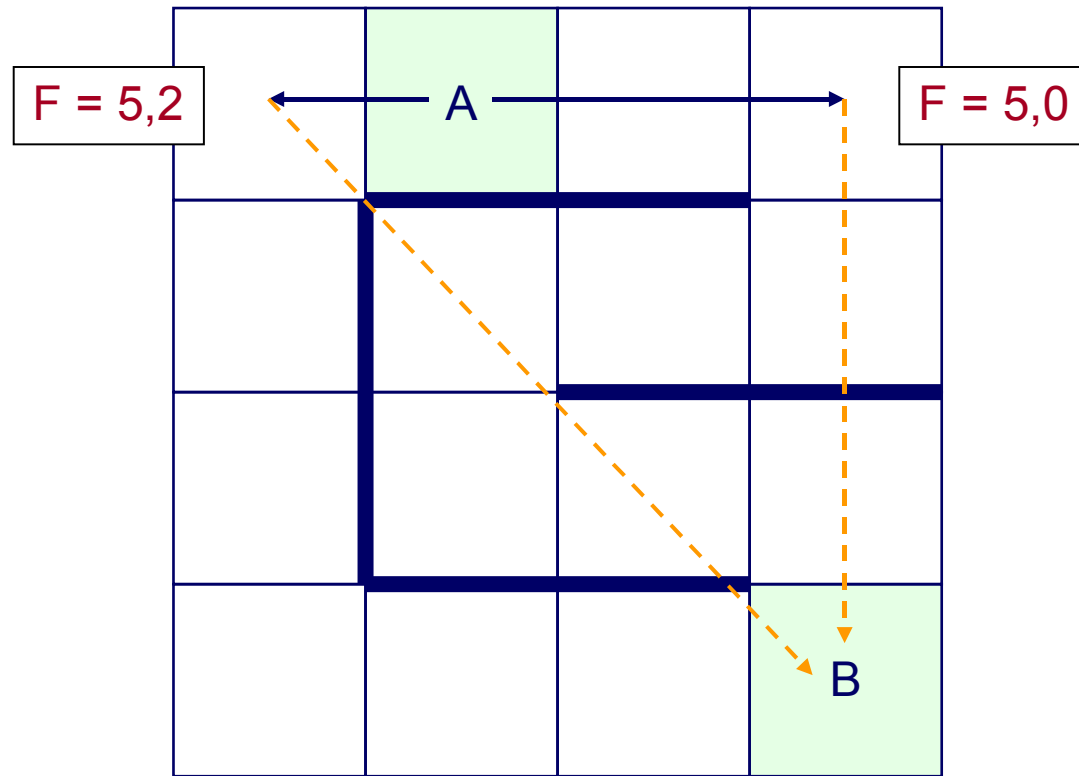


$g(x)$: coût du chemin parcouru pour arriver à x
 $h(x)$: distance euclidienne pour aller de x à B

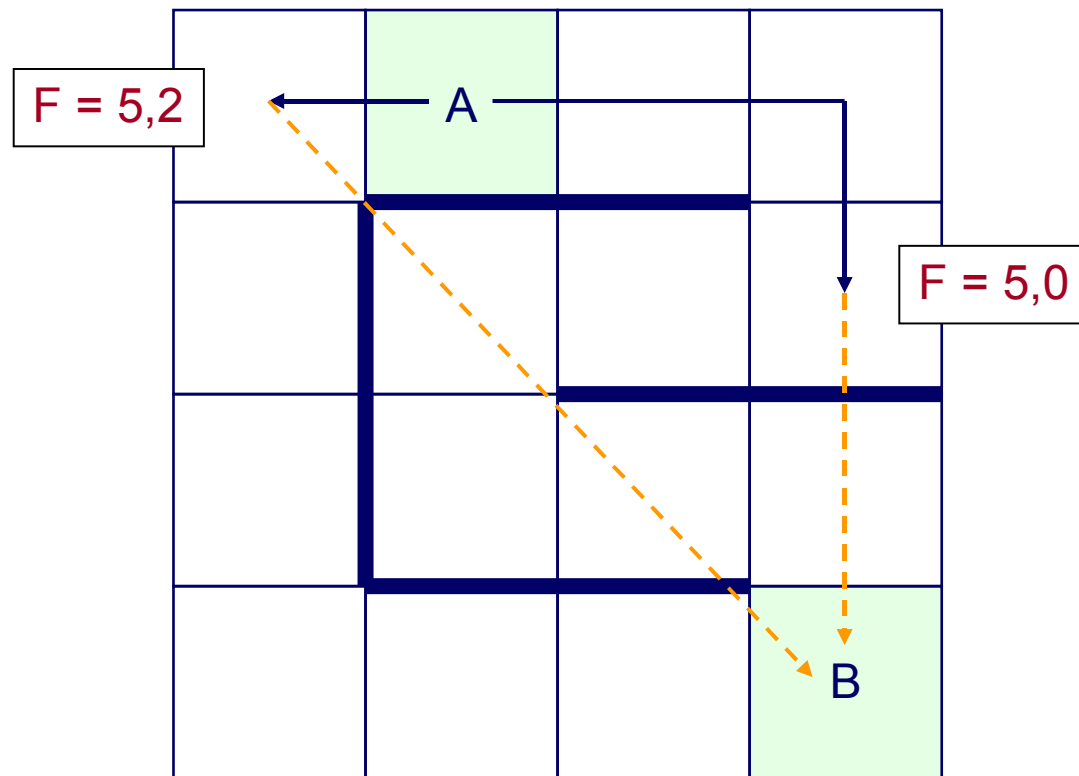
Exemple : recherche du plus court chemin dans un labyrinthe (4/14)



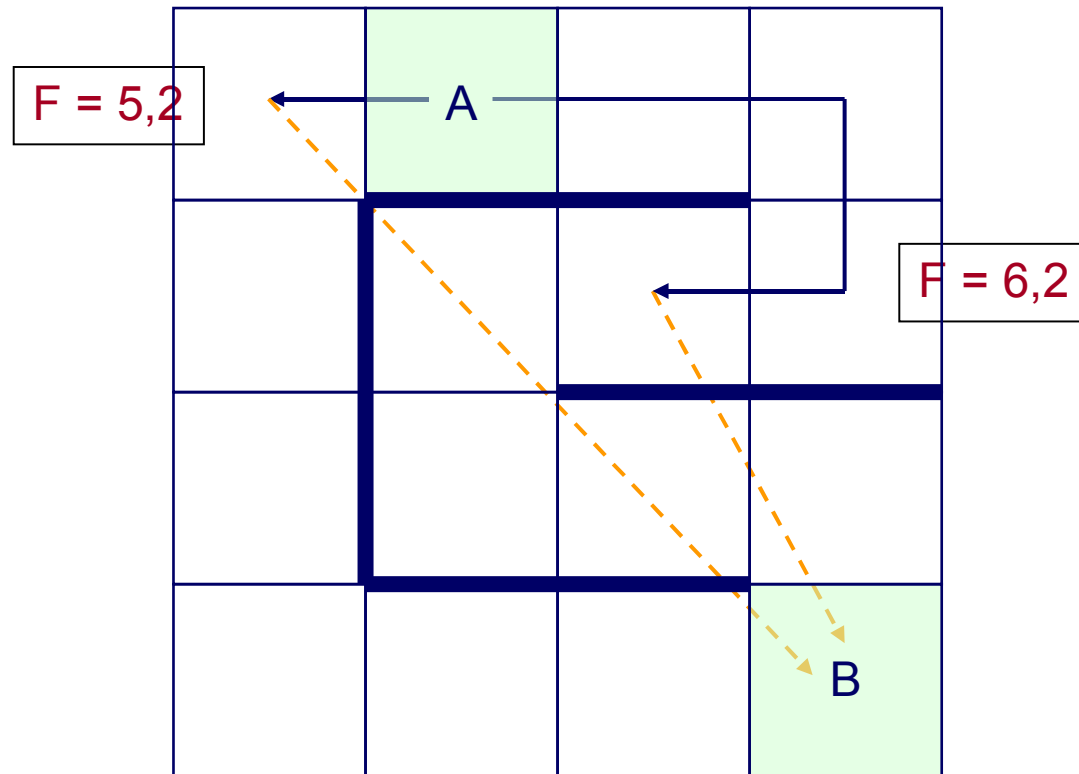
Exemple : recherche du plus court chemin dans un labyrinthe (5/14)



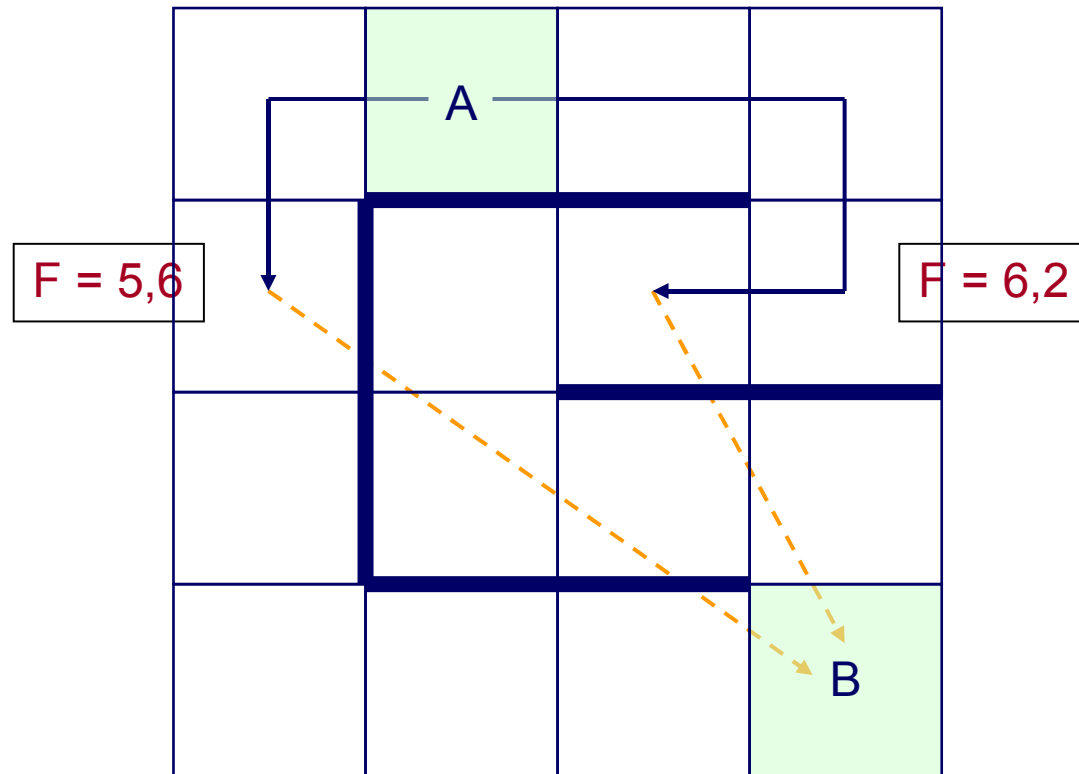
Exemple : recherche du plus court chemin dans un labyrinthe (6/14)



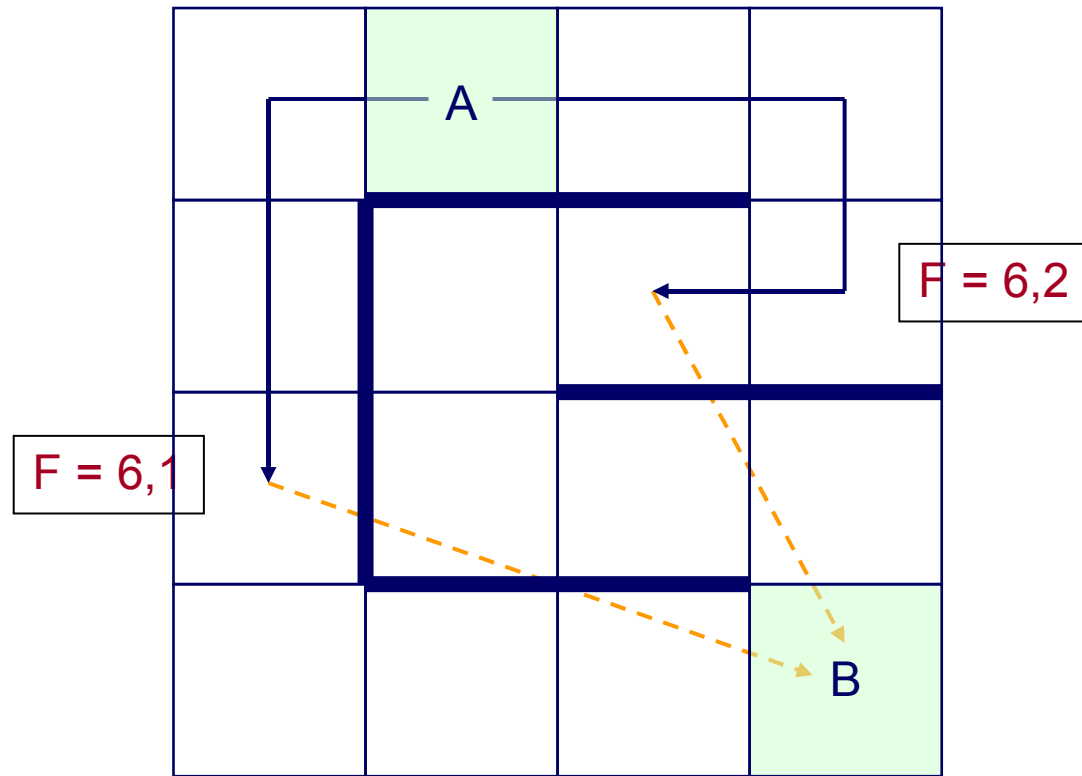
Exemple : recherche du plus court chemin dans un labyrinthe (7/14)



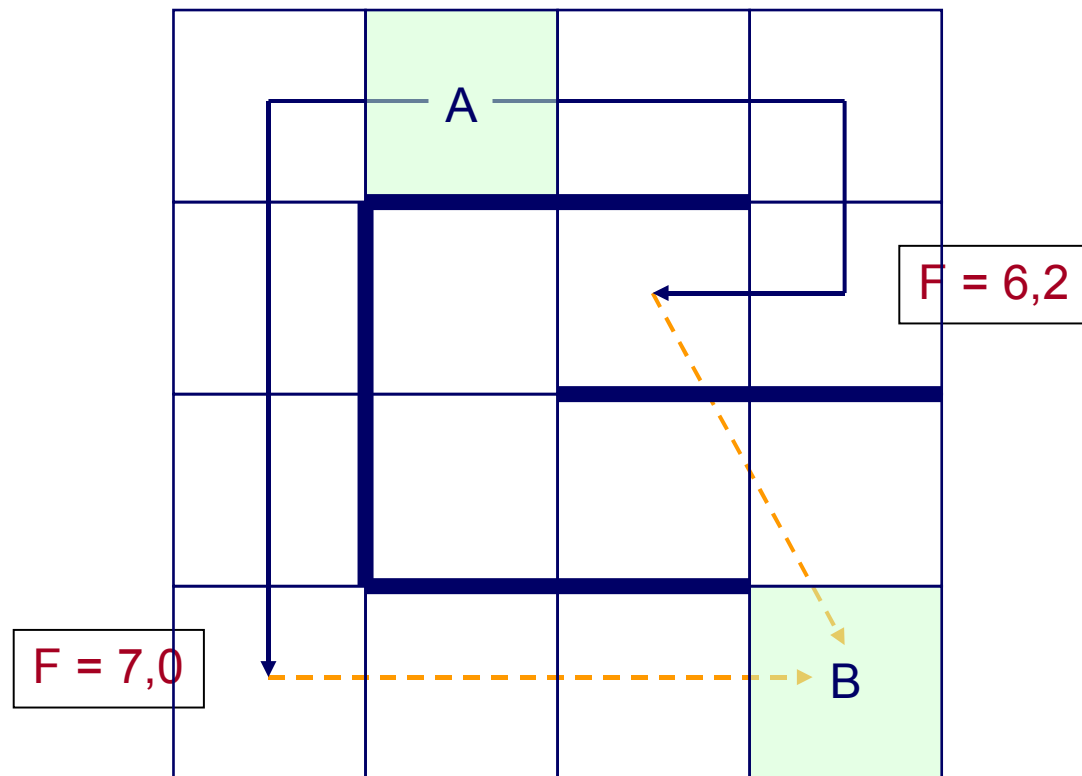
Exemple : recherche du plus court chemin dans un labyrinthe (8/14)



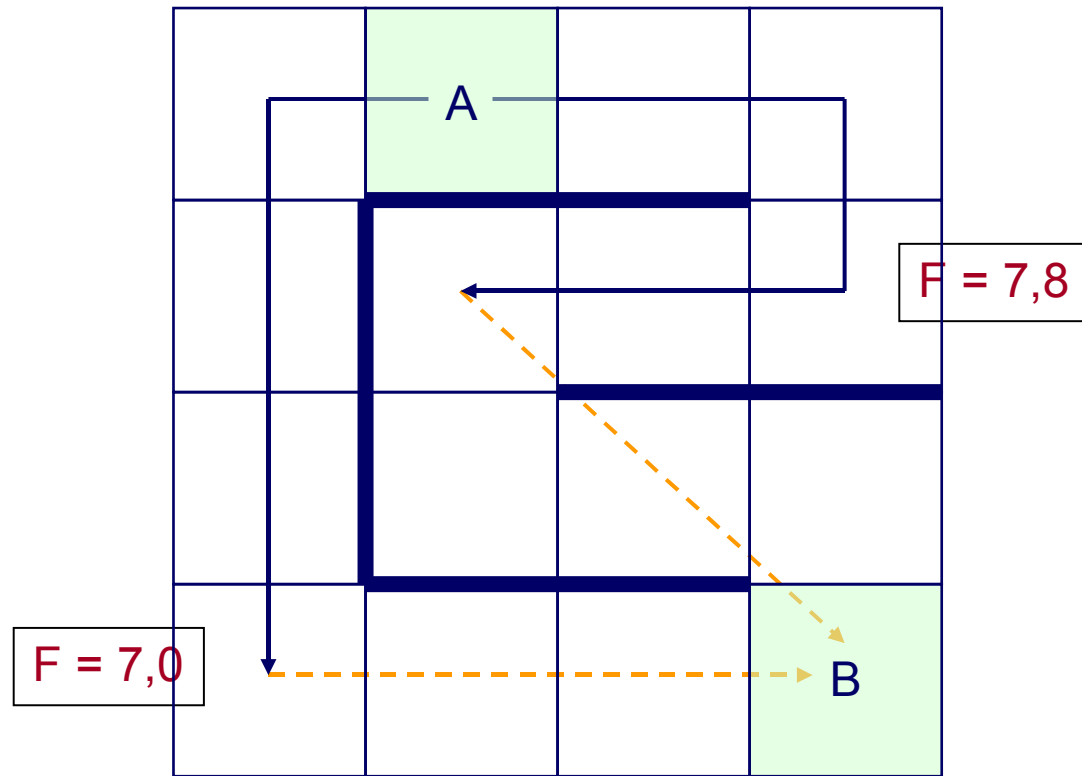
Exemple : recherche du plus court chemin dans un labyrinthe (9/14)



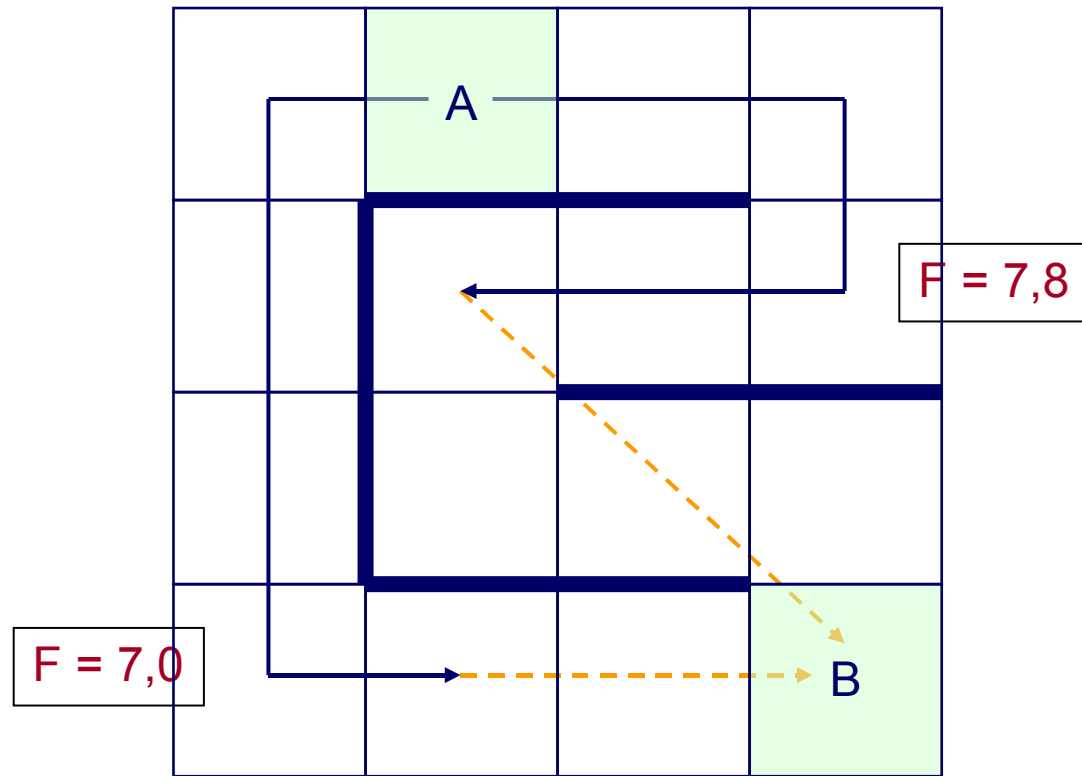
Exemple : recherche du plus court chemin dans un labyrinthe (10/14)



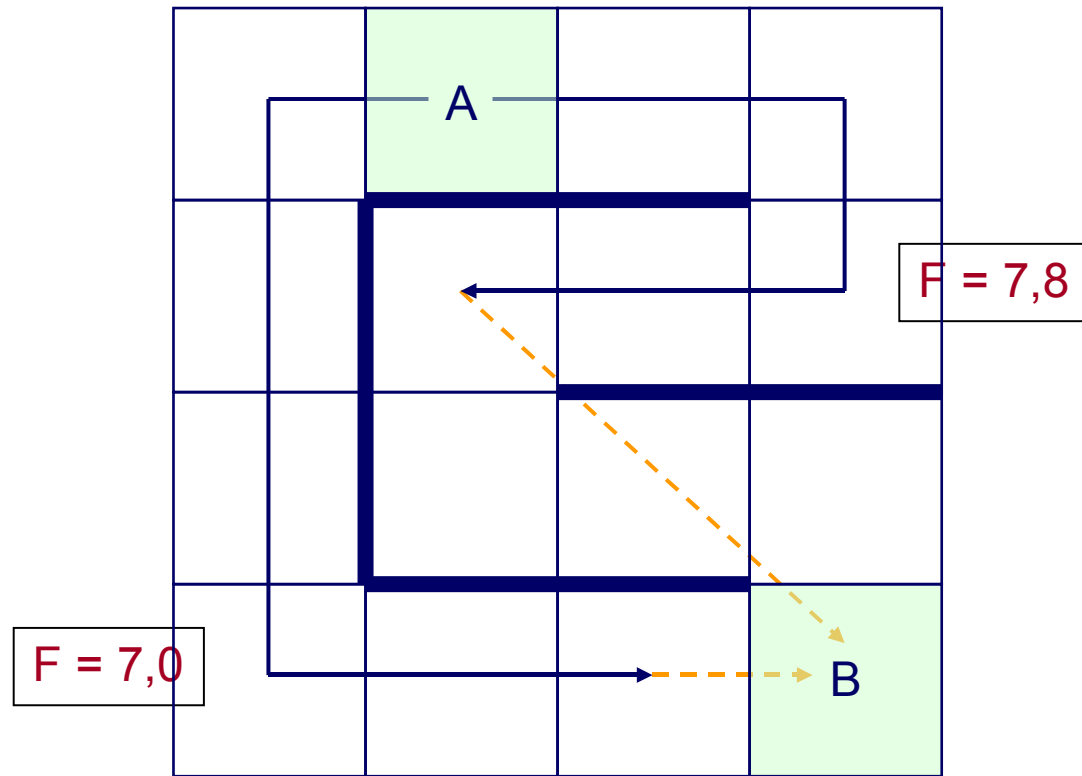
Exemple : recherche du plus court chemin dans un labyrinthe (11/14)



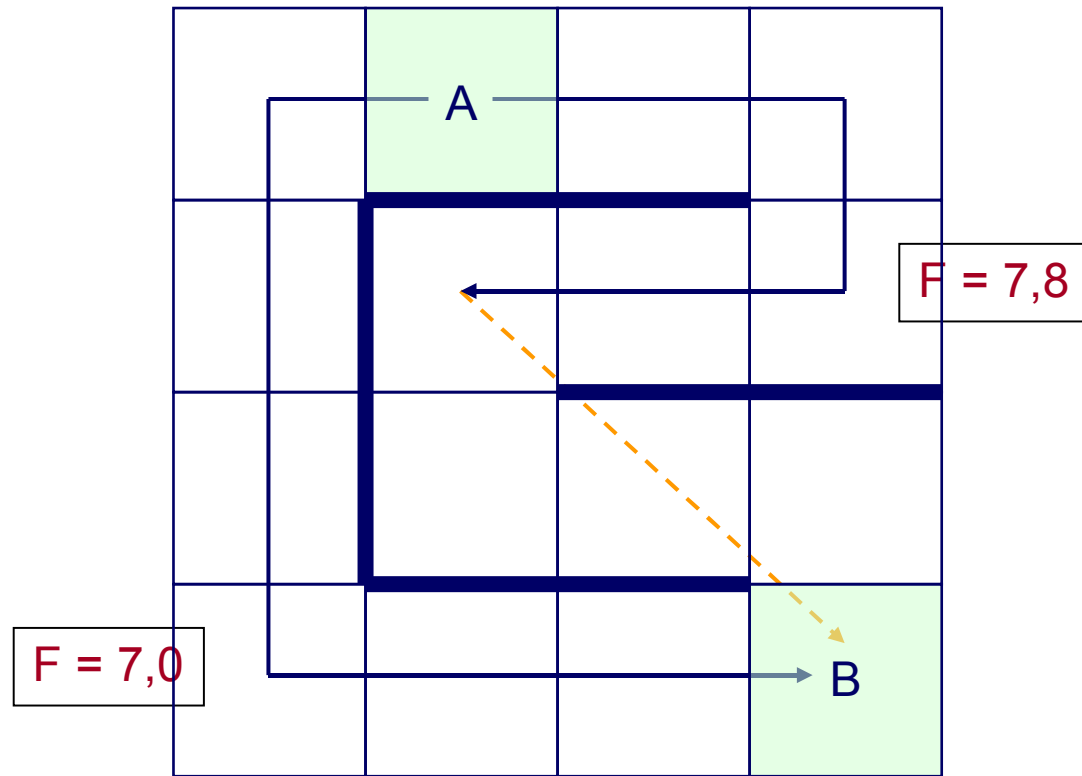
Exemple : recherche du plus court chemin dans un labyrinthe (12/14)



Exemple : recherche du plus court chemin dans un labyrinthe (13/14)



Exemple : recherche du plus court chemin dans un labyrinthe (14/14)



Plan

- **Introduction**
- **Espace d'états et système de production**
- **Définition, types et propriétés des graphes et arbres**
- **Stratégies de recherche**
 - **Stratégies de recherche non informées**
 - **Stratégies de recherche informées (avec heuristique)**
 - **Exemple de résolution d'un problème avec A***
 - **Propriétés des fonctions heuristiques**
- **Mesure de performances des stratégies de recherche**
- **Analyse de problèmes**

Propriétés d'une heuristique

Définition formelle d'une heuristique $h(x)$

Tente d'estimer sur tous les chemins (x, x_1, \dots, x_n, y) la somme $c(x, x_1) + c(x_1, x_2) + \dots + c(x_n, y)$, avec y noeud but.
Soit $h^*(x)$ la valeur de ce minimum.

Définition : Heuristique coïncidente

Pour tout $x \in T$, $h(x) = 0$ avec T ensemble des états terminaux

Définition : Heuristique parfaite

Pour tout x et y , $h(x) = h(y)$ équivaut à $h^*(x) = h^*(y)$

Définition : Heuristique monotone

Pour tout x et pour tout y descendant de x , $h(x) \leq c(x,y) + h(y)$

Définition : Heuristique admissible

Pour tout x , $h(x) \leq h^*(x)$

Plan

- **Introduction**
- **Espace d'états et système de production**
- **Définition, types et propriétés des graphes et arbres**
- **Stratégies de recherche**
 - **Stratégies de recherche non informées**
 - **Stratégies de recherche informées (avec heuristique)**
 - **Exemple de résolution d'un problème avec A***
 - **Propriétés des fonctions heuristiques**
- **Mesure de performances des stratégies de recherche**
- **Analyse de problèmes**

Mesure de performances des stratégies de recherche

Une stratégie de recherche s'évalue selon 4 critères :

- **Complétude** : s'il existe une solution, la stratégie la trouve.
- **Optimalité** : est-ce que la stratégie retourne la solution optimale, c'est-à-dire telle que la longueur du chemin entre l'état initial et l'état final est minimale.
- **Complexité en terme de temps de calcul** : combien de temps cela prend pour trouver une solution.
- **Complexité en terme d'espace mémoire** : nombre maximum de noeuds stockés

Mesure de performances des stratégies de recherche

Dans le cadre de l'étude des stratégies de recherche, la complexité en terme de temps de calcul s'évalue selon la taille de l'espace d'états, et plus particulièrement selon :

- b : facteur de branchement
- d : profondeur du noeud le plus près du noeud initial et contenant l'état final
- m : la taille du plus grand chemin dans l'espace d'états

Mesure de performances des stratégies de recherche

	Complet ?	Optimal ?	Complexité calcul	Complexité espace
Largeur d'abord	Oui	Oui	$O(b^{d+1})$	$O(b^{d+1})$
Profondeur d'abord	Non	Non	$O(b^m)$	$O(bm)$

Propriétés de A*

- **Complet** : s'il existe une solution, A* la trouve.
- **Optimal** : si l'heuristique est **admissible**, A* trouvera le chemin optimal.
Si l'heuristique est **parfaite**, A* converge immédiatement vers la solution.
Si l'heuristique est **monotone**, tout noeud développé fait partie du chemin optimal.
- **Optimalement efficace** : aucun autre algorithme optimal ne peut trouver la solution en développant moins d'états.

Mais :

- **Complexité**

Le nombre d'états développés est **exponentiel** en la longueur de la solution, dans le pire des cas (mauvaise heuristique).

Plan

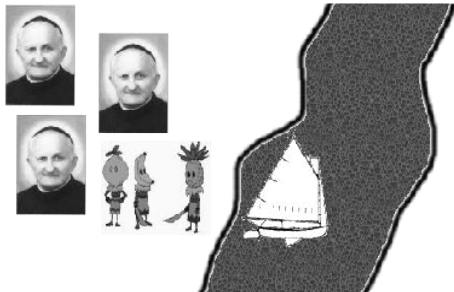
- **Introduction**
- **Espace d'états et système de production**
- **Définition, types et propriétés des graphes et arbres**
- **Stratégies de recherche**
 - **Stratégies de recherche non informées**
 - **Stratégies de recherche informées (avec heuristique)**
 - **Exemple de résolution d'un problème avec A***
 - **Propriétés des fonctions heuristiques**
- **Mesure de performances des stratégies de recherche**
- **Analyse de problèmes**

Analyse de problèmes (1/3)

Le choix de la stratégie de résolution à adopter dépend de la nature du problème.

Complexité du problème

- Largeur et profondeur d'abord : pour des problèmes assez simples.
- A* : pour des problèmes de complexité plus grande, mais pas trop quand même...



?

Démonstration
de théorèmes

?

2	8	3
1	6	4
7		5

?



?

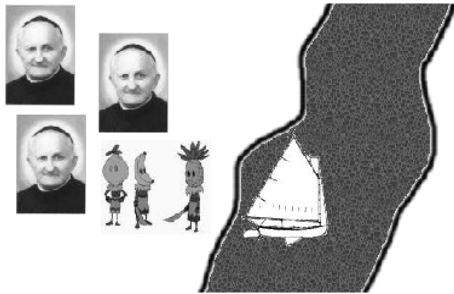
Analyse de problèmes (1/3)

Problèmes décomposables

Problème décomposable s'il peut être réduit en problèmes plus petits pouvant être résolus séparément.

Réduit considérablement la complexité du problème.

→ Approche *Divide and Conquer*.



?

Démonstration
de théorèmes

?

2	8	3
1	6	4
7		5

?



?

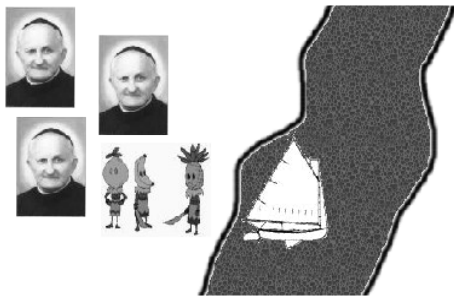
Analyse de problèmes (2/3)

Problèmes ignorables, récupérables ou irrécupérables

Problème *ignorable* si l'application d'une règle produisant un état inutile n'influence en rien la suite de la résolution.

Problème *totalelement récupérable* si la production d'un état inutile peut être annulée.

Problème *partiellement récupérable* si la stratégie de résolution utilisée implémente une technique permettant de revenir avant le point incorrect.



?

Démonstration
de théorèmes

?

2	8	3
1	6	4
7		5

?



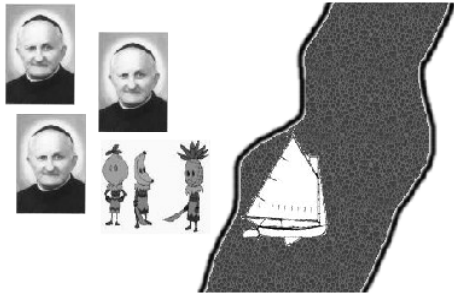
?

Analyse de problèmes (3/3)

Problèmes d'optimisation ou de satisfiabilité

Problème d'*optimisation* : trouver la meilleure solution.

Problème de *satisfaction* : trouver une solution (pas forcément la meilleure) respectant des contraintes.



?

Démonstration
de théorèmes

?

2	8	3
1	6	4
7		5

?



?