EFREI 2011/2012
L3

Aide à la Décision
Recherche dans un espace d'états – Comparaison et conclusion

Support de cours

Propriétés des algorithmes

Analyse comparative

Critères d'évaluation des algorithmes

- Complétude
  - La méthode garantit de trouver une solution si elle existe.
- Optimalité
  - Si plusieurs solutions existent, la méthode garantit de trouver la « meilleure » d'entre-elles.
  - Mais ... :
    qu'est-ce que « la meilleure » ?

Critères d'évaluation des algorithmes

- Complexité en temps
  - Combien de nœuds (d'états) faut-il produire / analyser pour trouver la solution ?

    (ordre de grandeur)
- Complexité en espace
  - Combien de nœuds faut-il conserver en mémoire pour trouver une solution ?

    (ordre de grandeur)

- Paramètres pour complexité en temps et en espace:
  - b = facteur de branchement
    nombre (maximum) de successeurs d'état
  - d = profondeur dans l'arbre de recherche
    du nœud représentant l'état solution retenu
  - m = profondeur maximale de l'arbre de recherche

(C) Hervé Barbot, 2005-2011 7

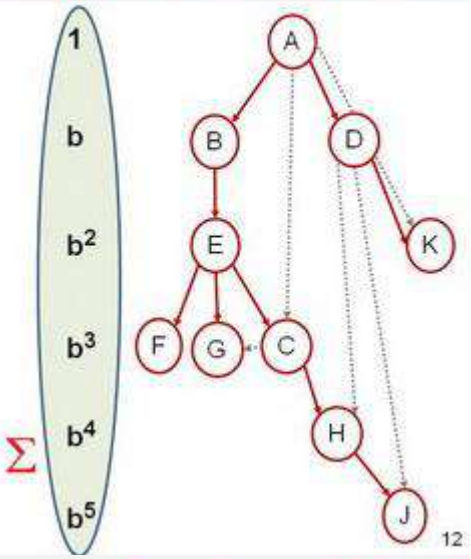
Recherche en profondeur d'abord

- Complexité en temps

  Dans le pire des cas, tous les états sont traités

  $= 1 + b + b^2 + \ldots + b^m$

  $= \Theta (b^m)$



(C) Hervé Barbot, 2005-2011 12

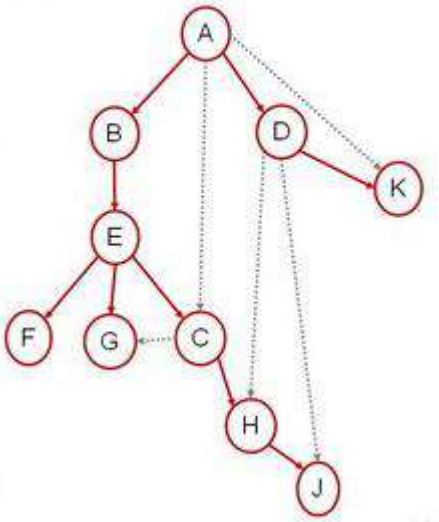
Recherche en profondeur d'abord

- Complexité en espace
  - *Théoriquement* : on peut oublier les états des sous-arbres entièrement analysés

    $= \Theta (b*m)$

    ou $\Theta (m)$

    - Mais risque de revenir sur un sous-arbre « oublié » et de le développer à nouveau



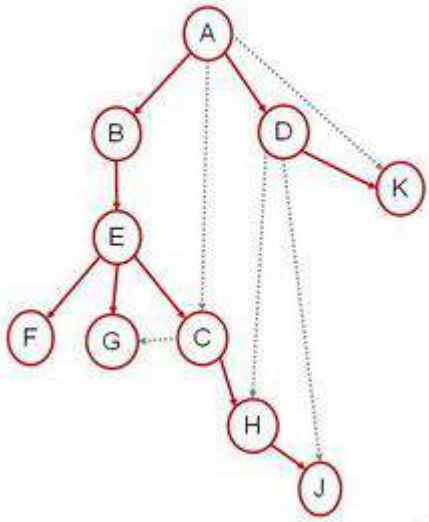
(C) Hervé Barbot, 2005-2011 14

Recherche en profondeur d'abord

- Complexité en espace
  - Si on gère les ensembles « ouverts » et « fermés » :

    $= \Theta (b^m)$

  - Plus coûteux en mémoire, mais moins risqué en temps d'exécution si les branches d'analyse se rejoignent...



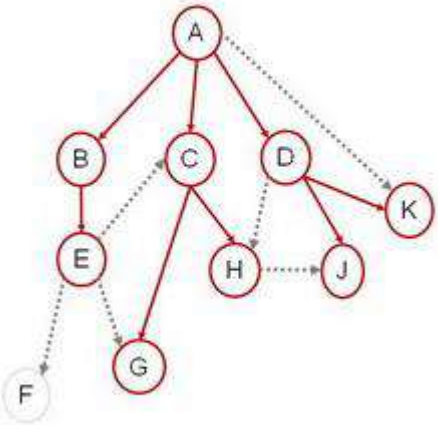
(C) Hervé Barbot, 2005-2011 15

Recherche en profondeur limitée

- Complétude :
  Oui si $L \geq d$
- Optimalité :
  NON

  car s'arrête sur le 1er état solution trouvé selon sa place dans l'arbre

  Cf. « profondeur d'abord »



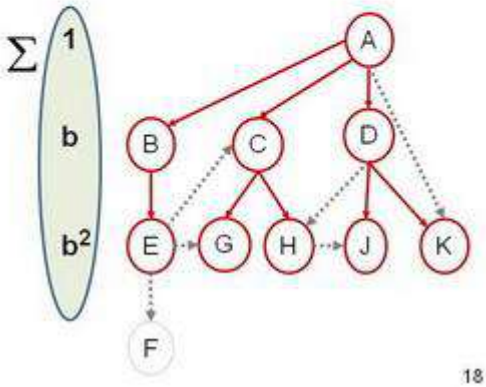
(C) Hervé Barbot, 2005-2011 17

Recherche en profondeur limitée

- Complexité en temps

  $= 1 + b + b^2 + \ldots b^L$

  $= \Theta (b^L)$



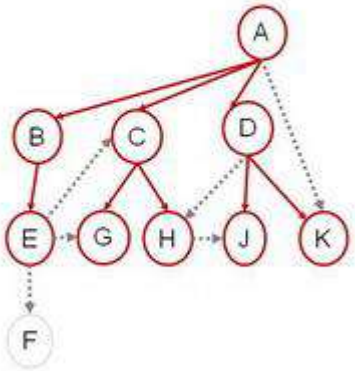
(C) Hervé Barbot, 2005-2011 18

Recherche en profondeur limitée

- Complexité en espace
  - *Théoriquement* : on peut oublier les états des sous-arbres entièrement analysés

    $= \Theta (b^*L)$

    ou $\Theta (L)$

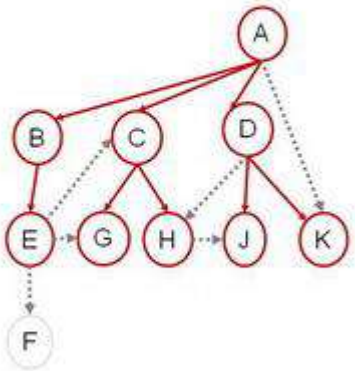


 19

Recherche en profondeur limitée

- Complexité en espace
  - *Mais* : si on gère les ensembles « ouverts » et « fermés » :

    $= \Theta (b^L)$



 20

Profondeur limitée	Approfondissement itératif
▪ Complétude Oui si L >= D	▪ Complétude Oui
▪ Optimalité Non	▪ Optimalité Non
▪ Complexité en temps $\Theta(b^L)$	▪ Complexité en temps $\Theta(b^m)$ $(\Theta(b^1)+\ldots+\Theta(b^m))$
▪ Complexité en espace $\Theta(b^L)$	▪ Complexité en espace $\Theta(b^m)$

 22

Largeur d'abord

- Complétude

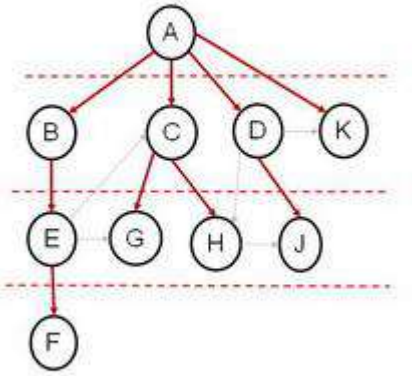
  Oui si espace fini

- Optimalité

  Non en général

  Oui si coût unitaire (ou constant) des actions

  i.e. si le « meilleur » est le nombre min d'actions



(C) Hervé Barbot, 2005-2011 24

Largeur d'abord

- Complexité en temps

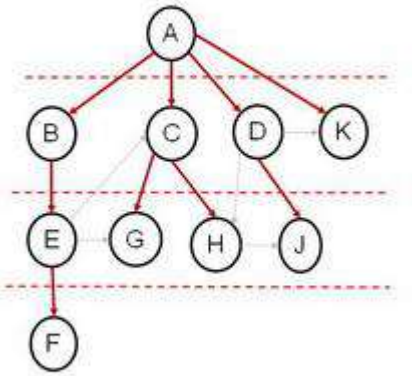
  $1 + b + b^2 + \ldots + b^d$

  $= \Theta (b^d)$

- Complexité en espace

  $1 + b + b^2 + \ldots + b^d + b(b^d-1)$

  $= \Theta (b^d)$



(C) Hervé Barbot, 2005-2011 25

Coût uniforme

- Complétude

  OUI

- Optimalité

  OUI

  Si les coûts des transitions sont positifs ou nuls !

  i.e. $\forall x \in \Gamma(e), g(x) \geq g(e)$

(C) Hervé Barbot, 2005-2011 27

Coût uniforme

- Optimalité

NON

si coûts négatifs

Car l'algorithme trouvera peut être la même « solution »,

mais pas avec le « meilleur » chemin pour y arriver !

- Cf. Pb du passage d'un nœud du graphe dans l'ensemble « CC » avec l'algorithme de Dijkstra…

(C) Hervé Barbot, 2005-2011 28

Coût uniforme

- Complexité en temps

$= \Theta (b^d)$

- Complexité en espace

$= \Theta (b^d)$

(C) Hervé Barbot, 2005-2011 29

Recherche gloutonne

- Completness

Yes

No if infinite states space

- Optimum

No in general

Yes if 'h' is perfect!

(C) Hervé Barbot, 2005-2011 32

Recherche gloutonne

- Execution Time

  $= \Theta (b^m)$

- Memory Space

  $= \Theta (b^m)$

  BUT : With a good 'h' function, these values can be greatly reduced!

(C) Hervé Barbot, 2005-2011 33

A*

- Complétude

  OUI

- Optimalité

  OUI

(C) Hervé Barbot, 2005-2011 36

A*

- Complexité en temps

  Exponentielle

- Complexité en espace

  garde tous les nœuds analysés en mémoire

En bref……

une horreur si la fonction heuristique est mal définie !

(C) Hervé Barbot, 2005-2011 37

Grille de comparaison

	Complétude	Optimalité (si applicable)	Complexité en temps	Complexité en espace
Profondeur d'abord	OUI / NON	NON	b^m	b.m ou b^m
Profondeur limitée	OUI si $L \geq d$	NON	b^L	b.L ou b^L
Approfondissement itératif	OUI (selon le pas de l'itération)	NON	b^m	b.m ou b^m
Largeur d'abord	OUI / NON	OUI / NON	b^d	b^d
Coût uniforme	OUI	OUI / NON	b^d	b^d
Recherche gourmande	OUI / NON	OUI / NON	b^m	b^m
A*	OUI	OUI	$b^?$	$b^?$

(C) Hervé Barbot, 2005-2011 38

- Introduction
- Description formelle d'un problème de recherche
- Exploration d'arbres
- Recherches aveugles
  - Principes généraux
  - Illustration
  - Recherche en profondeur d'abord
  - Recherche en profondeur limitée
  - Approfondissement itératif
  - Recherche en largeur d'abord
  - Recherche en coût uniforme
- Recherches heuristiques
  - Principes généraux
  - Recherche gloutonne / recherche gourmande
  - A*
- Propriétés des méthodes de recherche – Comparaison
- Conclusion & ouverture

(C) Hervé Barbot, 2005-2011 39

Recherche dans un espace d'états
Search in a states space

Rappels, conclusion,

et ouverture

(C) Hervé Barbot, 2005-2011 40

Algorithme de recherche ?

- Mécanisme général basé sur la modélisation de l'environnement de recherche :
  - L'environnement, le domaine d'exploration est représenté par un ensemble ou « espace » d' « états »
  - Les transformations apportées à l'environnement sont simulées par des changement d'états
    - Différentes transformations possibles à partir d'un état sont des alternatives représentées dans l'espace d'états
  - L'objectif est de trouver une suite d'actions (de transformations) permettant d'amener un environnement de son état initial à un état « acceptable »

(C) Hervé Barbot, 2005-2011 41

- Enoncé principal :
  - Etant donné une représentation de l'environnement sous forme d' « espace d'états »,
  - Etant donnée une situation initiale de départ de la recherche,
  - Quelle est la [meilleure] suite d'actions pouvant mener à une situation acceptable ?
  - La réponse à ce problème se fait par une analyse d'alternatives dans un ordre dépendant d'une « stratégie de recherche » particulière.

(C) Hervé Barbot, 2005-2011 43

- Contraintes essentielles :
  - L'ensemble des états possibles n'est a priori pas connu.
  - Il peut être très grand.
  - Dans certains cas, il peut être infini.

Donc :

Les techniques de « théorie des graphes » classiques avec représentation explicite et exhaustive sont rejetées.

(C) Hervé Barbot, 2005-2011 44

- Espace « très volumineux », voire « infini »
  - Nécessité de trouver un juste milieux entre :

- ne pas chercher indéfiniment,
- trouver une solution aussi « acceptable » que possible.

Problème de complétude !

Problème d'optimalité !

(C) Hervé Barbot, 2005-2011 45

- Paramètres à avoir en tête avant utilisation :
  - Nombre d'états dans l' « espace d'états »
    - Complexité en espace nécessaire à la recherche
  - Distribution des états solutions dans l'espace d'états
    - Une stratégie de recherche peut être meilleure qu'une autre
  - Espace mémoire nécessaire pour représenter un état
    - « x complexité en espace », cela peut faire beaucoup
  - Temps de recherche des alternatives (calcul des successeurs)
    - « x complexité en temps », cela peut aussi faire lourd…

(C) Hervé Barbot, 2005-2011 47

- Stratégie 1 : recherche aléatoire
- Stratégies 2 et 3 : largeur d'abord profondeur d'abord
- Stratégies 4 et 5 : approfondissement itératif profondeur limitée

Ce sont des stratégies

- *« informatique »*
  - *Qui ne dépendent que d'un choix de mise en œuvre de l'espace d'états*
- *« aveugles »*
  - *Qui connaissent les « états » analysés mais ne peuvent rien en déduire pour accélérer la recherche*

(C) Hervé Barbot, 2005-2011 49

- Stratégie 6
  coût uniforme

- Première approche vers la notion de « meilleure solution »
  - Si on considère que « meilleure » veut dire « à moindre coût cumulé » des différentes actions

- Limite : coûts négatifs
  - Non optimalité de la recherche

(C) Hervé Barbot, 2005-2011 50

- Stratégie 8 : recherche bi-directionnelle
  - En largeur d'abord depuis l'état initial
    +
    en largeur d'abord depuis l'état solution
  - Arrêt lorsque les deux espaces d'états se rejoignent
  - Connaissance de la fonction « action inverse » !
  - Problème technique : identification d'un état commun aux 2 espaces de recherche
    - Conservation de tous les états d'au moins un espace

(C) Hervé Barbot, 2005-2011 52

« Best first »

- Combine largeur / profondeur
- 1ère vision « basique »

```
function best_first ( problem )
returns solution or failure

set search_space = initial_state ( problem )
Loop do :
  ¬ ∃ candidate_for_expansion ( search_space )
     ⇒ return failure
  set node = choose_lower_h ( search_space )
  is_solution (node, problem )
     ⇒ return path from initial state to node
  ¬ is_solution ( node, problem )
     ⇒ set search_space = successors ( node )
```

NON !

(C) Hervé Barbot, 2005-2011 54

- Best-first :
  - Tous les états non analysés restent candidats à analyse et expansion
    - Ensemble 'ouverts'
  - A chaque itération, on choisi celui qui est le plus « prometteur »
    - i.e. celui qui a la plus faible valeur heuristique

(C) Hervé Barbot, 2005-2011 55

- Stratégie 9 : Greedy-search
  - La plus simple des stratégies « best-first »
  - Choix = valeur 'h' minimale parmi les états candidats à expansion
  - Trouve rapidement une solution si la fonction heuristique est performante
    - i.e. donne une bonne vision de la « distance restante »
  - Oublie les « coûts » effectifs des actions, donc ne trouve pas nécessairement la « meilleure » solution
    - Si « la meilleure » = le plus faible coût cumulé

(C) Hervé Barbot, 2005-2011 56

- Stratégie 10 : A*

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

  - Combine coût cumulé (calculé) et coût restant (estimé)
  - Choix = valeur 'f' minimale parmi les états candidats à expansion
    i.e. le chemin le plus prometteur de bout en bout (de l'état initial à un état solution)
  - « Meilleur » = « plus faible coût global » si la fonction heuristique est convenablement choisie…

 57

*Variantes de A**

- Problèmes souvent très complexes
- Espace de recherche très grand
- Recherche heuristique pure devient peu efficace
- Encombrement mémoire

IDA* RBFS MA* SMA*

 58