Résolution de problème par exploration:
Partie1: Formalisation d'un problème

Master 1 SID

Dr H.Belleili

Plan

- Étapes de résolution
- Formulation du problème
- Exemples: problèmes jouets
  - L'aspirateur
  - Jeu du taquin
  - Les 8 reines
- Exemples: Problèmes du monde réel
  - Problème de recherche d'un trajet
  - …
- Stratégies d'exploration
  - Définition
  - Évaluation d'une stratégie
- exercice

Étapes de résolution

1. Formulation d'un but : un ensemble d'états à atteindre.
2. Formulation du problème : les états et les actions à considérer.
3. Recherche de solution : examiner les différentes séquences d'actions menant à un état but et choisir la meilleure.
4. Exécution : accomplir la séquence d'actions sélectionnée.

Formulation du problème

- Un problème sera défini par les 5 éléments suivants :
- 1. un état initial
- 2. un ensemble d'actions
- 3. une fonction successeur, qui définit l'état résultant de l'exécution d'une action dans un état
- 4. un ensemble d'états buts
- 5. une fonction coût, associant à chaque action un nombre non-négatif (le coût de l'action)

Formulation du problème

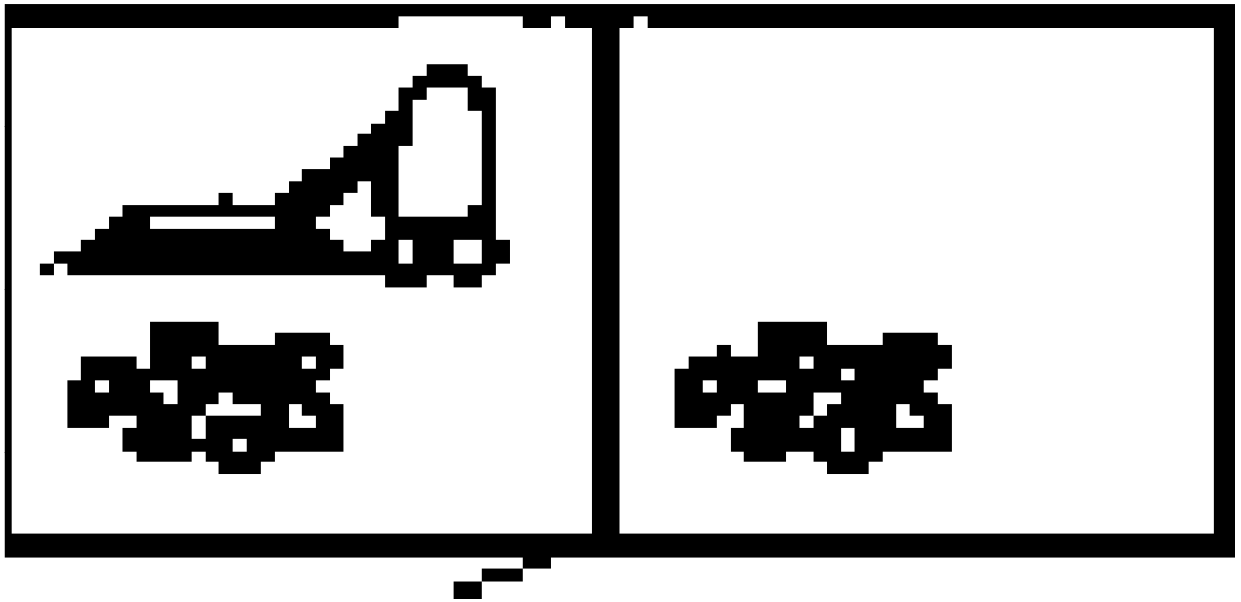
- un problème est représenté comme un graphe orienté où les noeuds sont des états accessibles depuis l'état initial et où les arcs sont des actions.
- Nous appellerons ce graphe l'espace des états.
- Une solution sera un chemin de l'état initial à un état but.
- On dit qu'une solution est optimale si la somme des coûts des actions du chemin est minimale parmi toutes les solutions du problème.

Exemple de problème d'exploration

- On distingue 2 types de problèmes:
  - Des problèmes dits jouets: servent à illustrer ou à expérimenter des méthodes de résolution de problèmes
  - Des problèmes du monde réel: ce sont des problèmes qui intéressent vraiment les gens.

Problèmes jouets (exemple 1)

- Agent aspirateur (2 emplacements)
- Formulation:
  - Ètats : l'agent peut être dans l'un des deux emplacements, les emplacements peuvent contenir ou pas de la poussière (2 x 2^2 états possibles),
  - État initial: n'importe quel état peut être pris comme état initial,
  - Fonction successeur: elle génère les états pouvant résulter de l'exécution des trois actions (gauche, droite, aspirer)
  - Test de l'état final: il vérifie que tous les carrés sont propres
  - Coût du chemin: le coût de chaque étape est 1, le coût du chemin est égal au nombre d'étapes que composent le chemin.
- Hypothèses: emplacements discrets, les saletés discrètes, un nettoyage fiable, jamais resali une fois nettoyé.



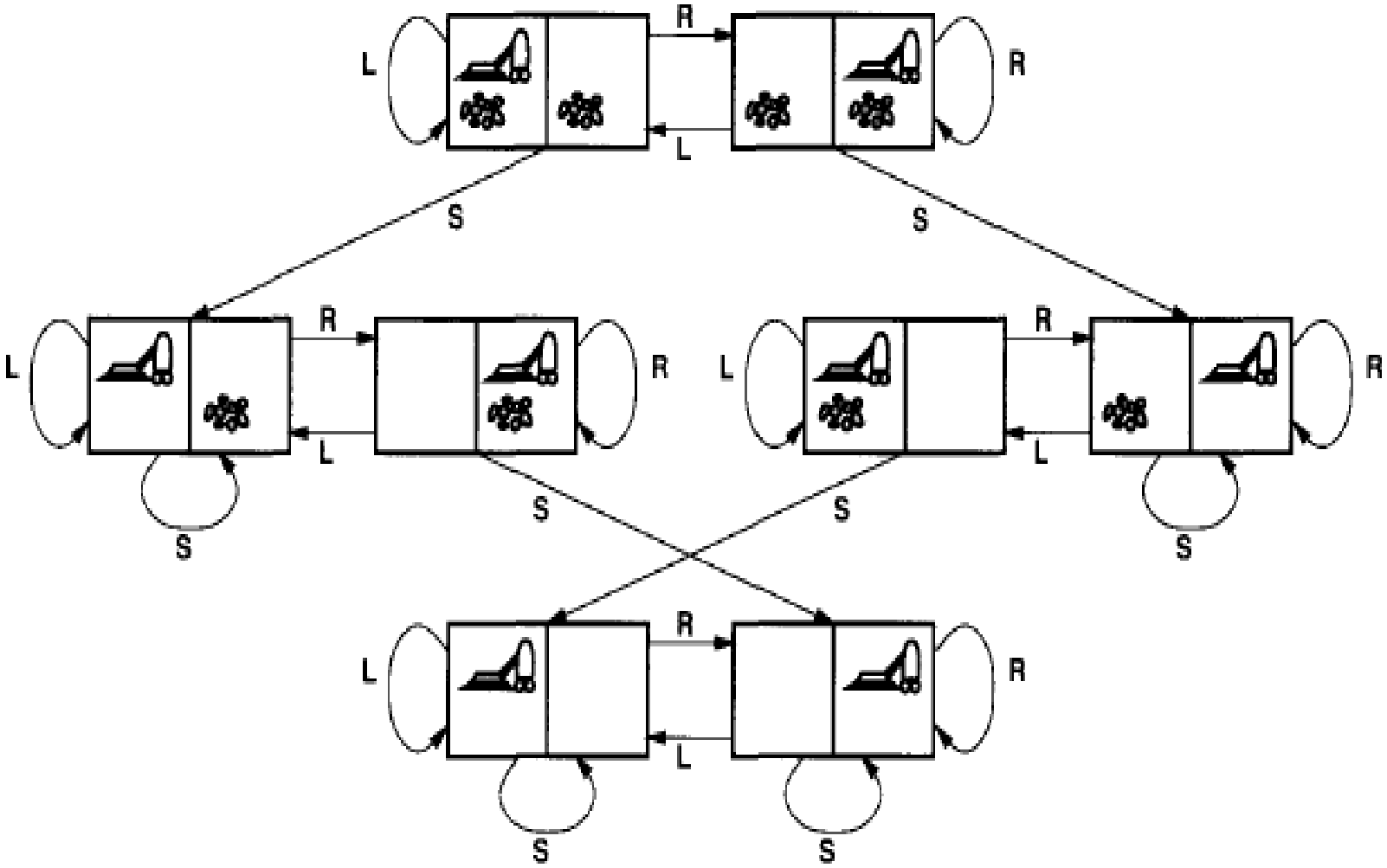
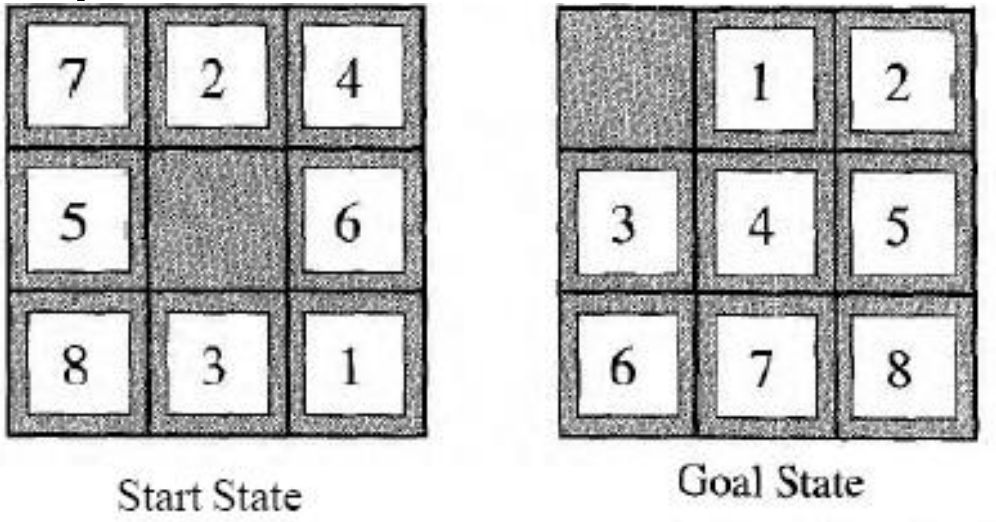


Figure 3.6 Diagram of the simplified vacuum state space. Arcs denote actions. L = move left, R = move right, S = suck.

Exemple 2: le jeu de taquin à 8 pièces

- Se compose d'un plateau 3 x 3 cases dont huit sont occupés par des numéros et une case est vide



Le taquin est souvent utilisé pour tester les algorithmes de recherche.

Taquin : Formulation

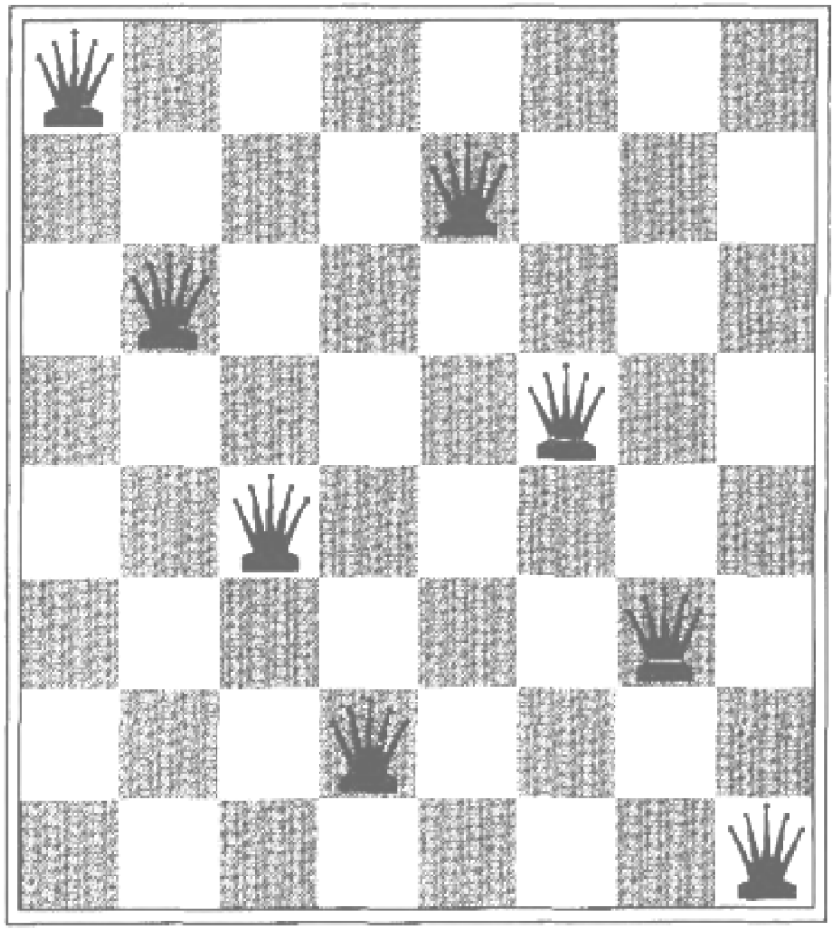
- Etats : Ce sont des configurations des huit tuiles dans les neuf cases de la grille.
- Etat initial : N'importe quel état pourrait être choisi comme l'état initial.
- Actions: Il y aura 4 actions possibles correspondant aux quatre façons de changer la position du carré vide : haut, bas, gauche, droite
- Fonction de successeur : Cette fonction spécifie les états résultants des différentes actions.
- Test de but: L'état but est unique et fixé au début du jeu
- Coût des actions: Chaque déplacement d'une tuile a un coût de 1 (pour trouver une solution avec le moins de déplacements).

Taquin 8 pièces (suite)

- Classe des problèmes NP-complets,
- Le problème à 8 pièces compte 9!/2=181 440 états possibles et est facilement résolu,
- Avec 15 pièces (16 cases) ~ 1,3 milliard d'états accessibles on arrive à le résoudre en quelques millièmes de seconde
- Avec 24 pièces ~ 10^{25} états accessibles et il est assez difficile à résoudre d'une manière optimale

Le problème des huit reines

- Objectif: placer huit reines sur un échiquier (une grille 8 × 8) tel qu' aucune reine attaque une autre reine,(il n'y a pas deux reines sur la même colonne, la même ligne, ou sur la même diagonale).



8 reines: Formulation

- Etats : Toute configuration de 0 à 8 reines sur la grille.
- Etat initial: La grille vide.
- Actions : Ajouter une reine sur n'importe quelle case vide de la grille.
- Fonction de successeur : La configuration qui résulte de l'ajout d'une reine à une case spécifiée à la configuration courante.
- Test de but: Une configuration de huit reines avec aucune reine sous attaque.
- Coûts des actions: Ce pourrait être 0, ou un coût constant pour chaque action - nous nous intéressons pas au chemin, seulement l'état but obtenu.
- Questions:
  1. quel est le nombre d'états possibles avec cette formulation?
  2. Existe-t-il une manière de réduire ce nombre d'état?

Taquin vs 8 reines

- Ces deux problèmes sont de nature assez différentes.
- Avec le taquin, nous savons depuis le début quel état nous voulons, et la difficulté est de trouver une séquence d'actions pour l'atteindre.
- le problème des huit reines, nous ne sommes pas intéressés par le chemin mais seulement par l'´etat but obtenu.
- Ces deux jeux sont des exemples de deux grandes classes de problèmes étudiés en IA : des problèmes de planification et des problèmes de satisfaction de contraintes.

Problèmes du monde réel

- Problème de recherche d'un trajet,
- Le problème du voyageur du commerce (TSP) qui une variante du problème de recherche d'un itinéraire dans lequel chaque ville doit être visitée une seule fois
- Navigation d'un robot une généralisation du problème de recherche d'itinéraire (déplacement dans un espace continu ayant un ensemble infini d'actions et d'états possibles)
- Ordonnancement automatique d'assemblage d'objets complexes (le robot FREDDY)
- Conception de protéine de synthèse (bioinformatique)
- Recherches internet

Exemple

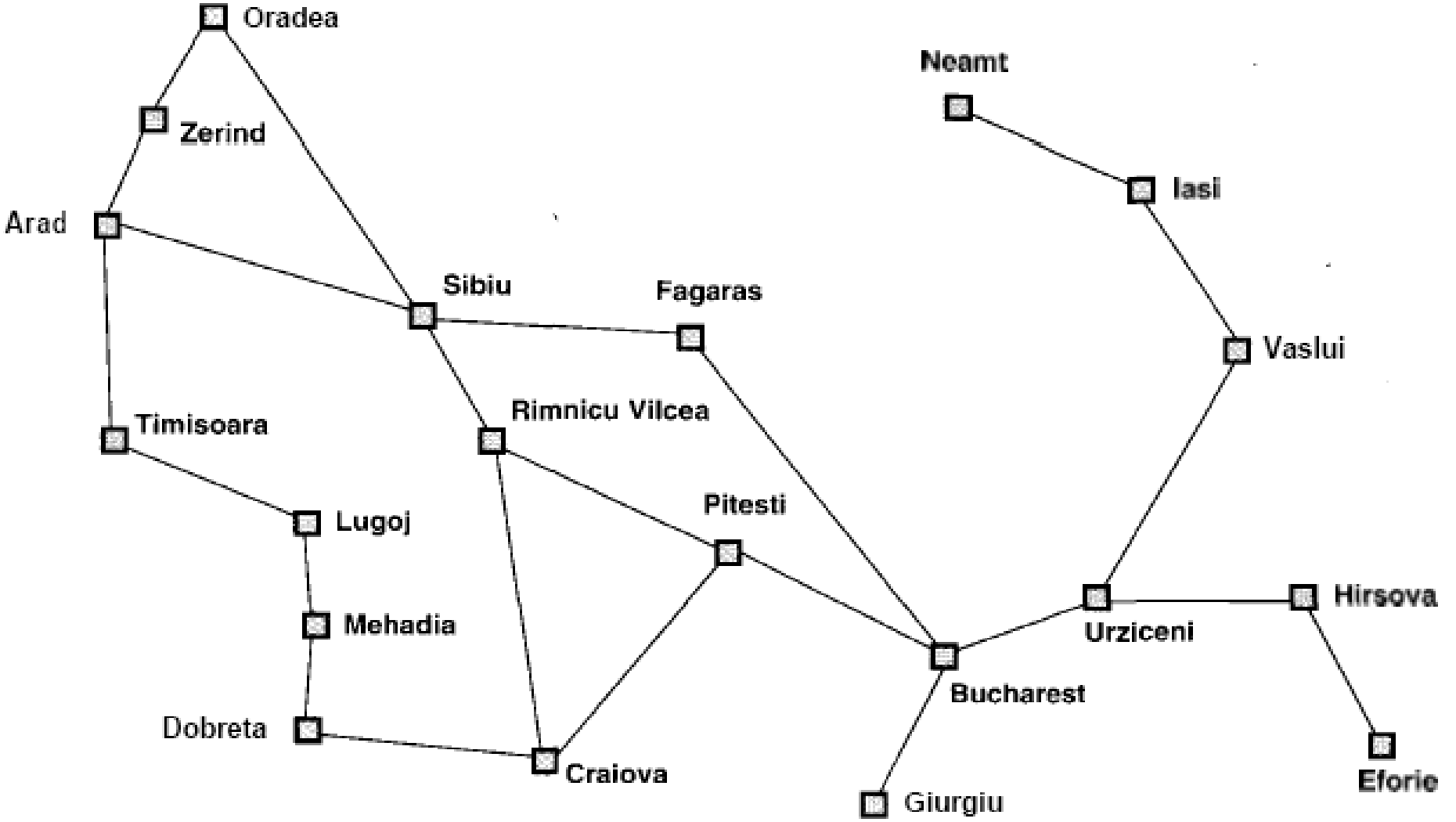


Figure 3.3 A simplified road map of Romania.

Exemple

- On est à Arad et on veut aller à Bucharest

– But: être à Bucharest

– Problème:

  – États: villes

  – Actions: aller d'une ville à une autre.

– Solution: Une séquence de villes.

  – Ex: Arad, Sibiu, Fagaras, Bucharest

  – Environnement très simple

– statique, observable, discret et déterministe

Formulation du problème

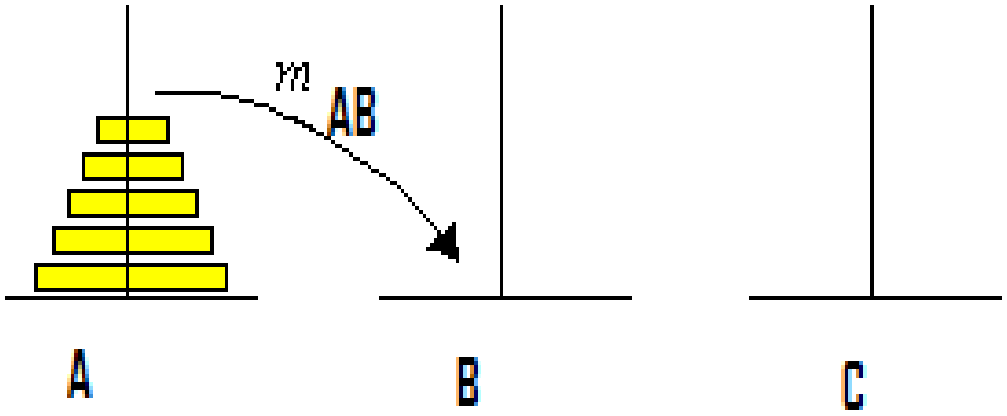
- État initial : l'état *x* dans lequel se trouve l'agent
  - ***Dans*(*Arad*)**
- Actions : Une fonction *S*(*x*) qui permet de trouver l'ensemble des états successeurs, sous forme de pair (action, successeur)

  S(*Dans* (*Arad*)) = {((*Aller*(*Sibiu*), *Dans(Sibiu*)), ((aller(Timisoara), Dans(Timisoara)), ….} etc.

- L'état initial et la fonction successeur définissent ensemble **l'espace des états du problème (états accessibles à partir de l'état initial)**
- Test de but : un test afin de déterminer si le but est atteint
  - But: {*Dans (Bucharest*)}
- Coût du chemin : noté C(x,a,y) . une fonction qui assigne un coût à un chemin. Elle reflète la mesure de performance de l'agent. La valeur est non négative.

Exercices

1. On considère le problème des Tours de Hanoi avec trois tours. On demande de déplacer les disques situés sur la première tour sur une autre, de façon qu'un disque repose toujours sur un autre disque de taille supérieure ou sur une tour libre.



Exercice (suite)

- 2- Une chèvre, un chou et un loup se trouvent sur la rive d'un fleuve ; un passeur souhaite les transporter sur l'autre rive mais, sa barque étant trop petite, il ne peut transporter qu'un seul d'entre eux à la fois. Comment doit-il procéder afin de ne jamais laisser ensemble et sans surveillance le loup et la chèvre, ainsi que la chèvre et le chou ?
- 3- On souhaite prélever 4 litres de liquide dans un tonneau. Pour cela, nous avons à notre disposition deux récipients (non gradués !), l'un de 5 litres, l'autre de 3 litres... Comment doit-on procéder ?

Partie 2: Recherche aveugle (non-informée) de solutions

Introduction

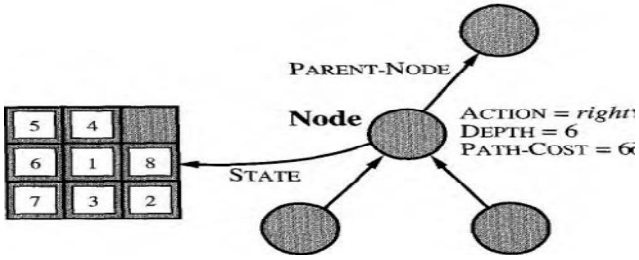
- Après avoir formulé le problème, il faut le résoudre.
- On effectue une exploration de l'espace des états
- L'exploration se fait soit
  - Dans un arbre généré explicitement par l'état initial et la fonction successeur
  - Soit dans un graphe lorsque le même état peut être atteint à partir de plusieurs chemins

Evaluation des algorithmes de recherche

- Dans la suite, nous allons voir différents d'algorithmes de recherche.
- Comment pouvons-nous les comparer ?
- Voici quatre critères que nous allons utiliser pour comparer les différents algorithmes de recherche :
  - **Complexité en temps** : Combien de temps prend l'algorithme pour trouver la solution ?
  - **Complexité en espace** : Combien de mémoire est utilisée lors de la recherche d'une solution ?
  - **Complétude:** Est-ce que l'algorithme trouve toujours une solution s'il y en a une ?
  - **Optimalité:** Est-ce que l'algorithme renvoie toujours des solutions optimales ?

Recherche de solutions dans un arbre

- Simuler l'exploration de l'espace d'états en générant des successeurs pour les états déjà explorés.
- Noeud de recherche
  - État : l'état dans l'espace d'état.
  - Noeud parent : Le noeud dans l'arbre de recherche qui a généré ce noeud.
  - Action : L'action qui a été appliquée à l'état du nœud parent pour générer l'état de ce noeud.
  - Coût du chemin : Le coût du chemin à partir de l'état initial jusqu'à ce noeud : *g(n)*
  - Profondeur : Le nombre d'étapes dans le chemin à partir de l'état initial.

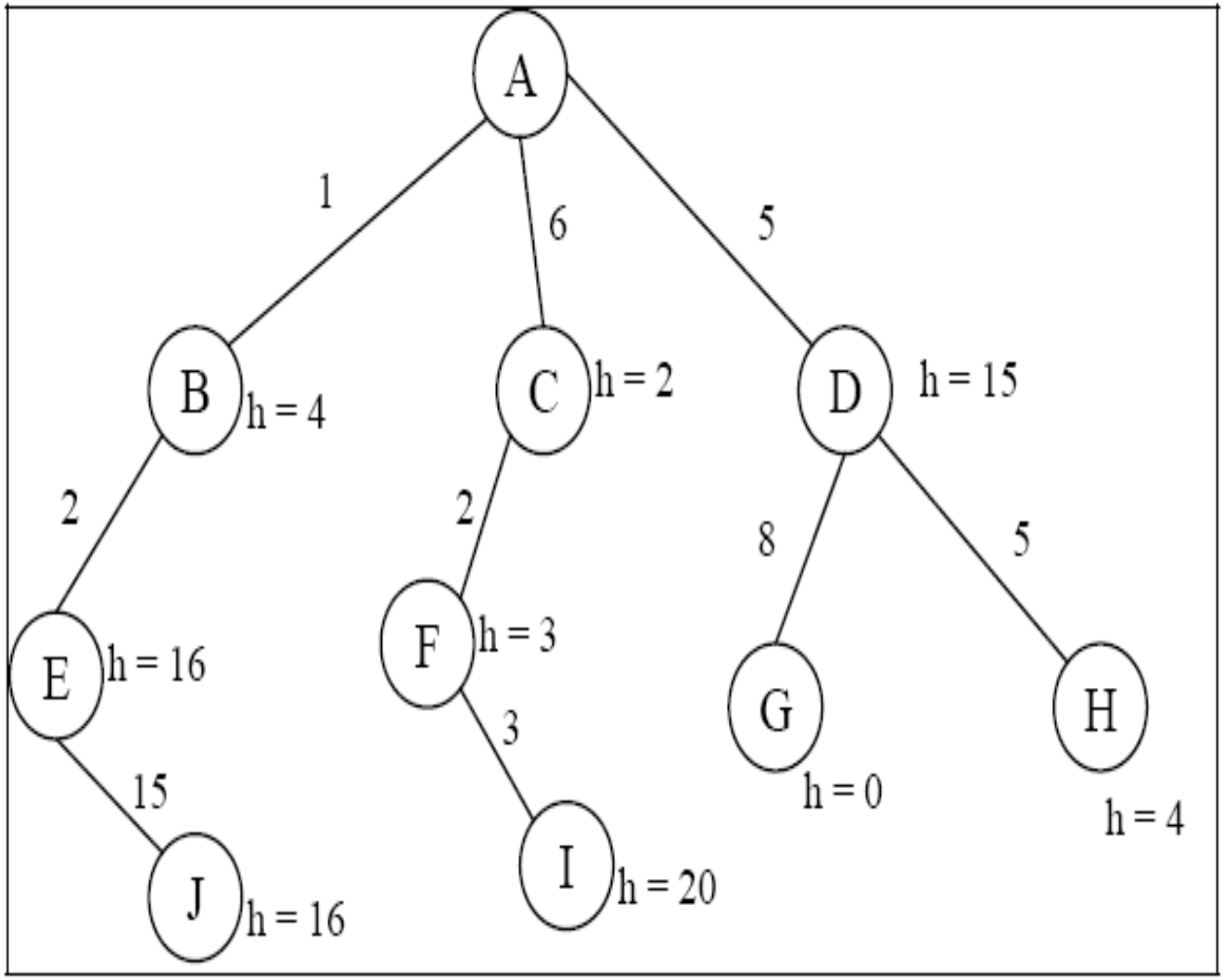


Nœud vs état

- Il est important de distinguer entre les nœuds et les états
- Un nœud est une structure de données de mémorisation qui est utilisé pour représenter l'arbre de recherche
- Un état correspond à une configuration du monde
- DONC:
  - Les nœuds sont sur des chemins particuliers, ça n'est pas le cas des états
  - Deux nœuds différents peuvent contenir le même état (si cet état est généré via deux chemins différents)

Principe Exploration

- Commencer par le nœud racine
- Développer (expand) le nœud en lui appliquant la fonction successeur
- On obtient tous les nœuds successeurs ces nœuds sont dits des nœuds générés
- Les nœuds générés mais non encore développés sont mis dans une collection appelée frontière
- Chaque élément de la frontière est un nœud feuille (ie sans les nœuds successeurs)
- La stratégie d'exploration est la fonction qui sélectionne le nœud suivant à développer dans cet ensemble de nœuds de la frontière.
- La fonction d'exploration doit examiner chacun des éléments de la frontière afin de choisir la meilleure (à développer)


A
1
6
5
B
h = 4
C
h = 2
D
h = 15
2
2
8
5
E
h = 16
F
h = 3
G
h = 0
H
h = 4
3
15
I
h = 20
J
h = 16

Structure générale d'un algorithme de recherche

- La plupart des algorithmes de recherche suivent à peu près le même schéma :
- Fonction Exploration-En-Arbre (problème, stratégie) retourne une solution, ou echec
  - **Initialiser l'arbre de recherche avec l'état initial du problème**
  - **itérer**
    - **si** il n'y a plus de nœuds candidats à développer **retourner échec**
    - **choisir un nœud feuille (dans la frontière) à développer en appliquant la stratégie**
    - **si** le nœud choisi contient un état final **alors retourner la solution correspondante**
    - **sinon**, développer le nœud et ajouter les nœuds du résultat dans la frontière

Frontière

- La Frontière est une file ayant des opérations:
  - Créer file (élément…)
  - Vide(file) retourne vrai ou faux
  - Premier(file) retourne le premier élément de la file
  - Supp-premier(file) retourne le premier(file) et le supprime de la file
  - Insérer (élément,file) insére un nœud dans la file et retourne file
  - Insérer-tout(éléments,file) insère un ensemble d'éléments dans la file et retourne la file résultante

Retour sur la mesure de performance

- La mesure de performance d'un algorithme de recherche dans un graphe est liée à la taille du graphe de l'espace d'états
- On exprime la complexité de la recherche en fonction de 3 critères:
  - Le facteur de branchement b: c'est le nombre maximal de successeurs d'un nœud donné
  - La profondeur du nœud but le moins éloigné d
  - La longueur maximale d'un chemin dans l'espace des états m
- Complexité de temps: le nombre de nœuds générés pendant la recherche.
- Complexité d'espace: le nombre maximum de nœuds conservés en mémoire.

Stratégies de recherche

- Détermine l'ordre de développement des nœuds.
- Recherches non-informées : Aucune information additionnelle. Elles ne peuvent pas dire si un noeud est meilleur qu'un autre.Elles peuvent seulement dire si l'état est un but ou non.
- Recherches informées (heuristiques) : Elles peuvent estimer si un noeud est plus prometteur qu'un autre.

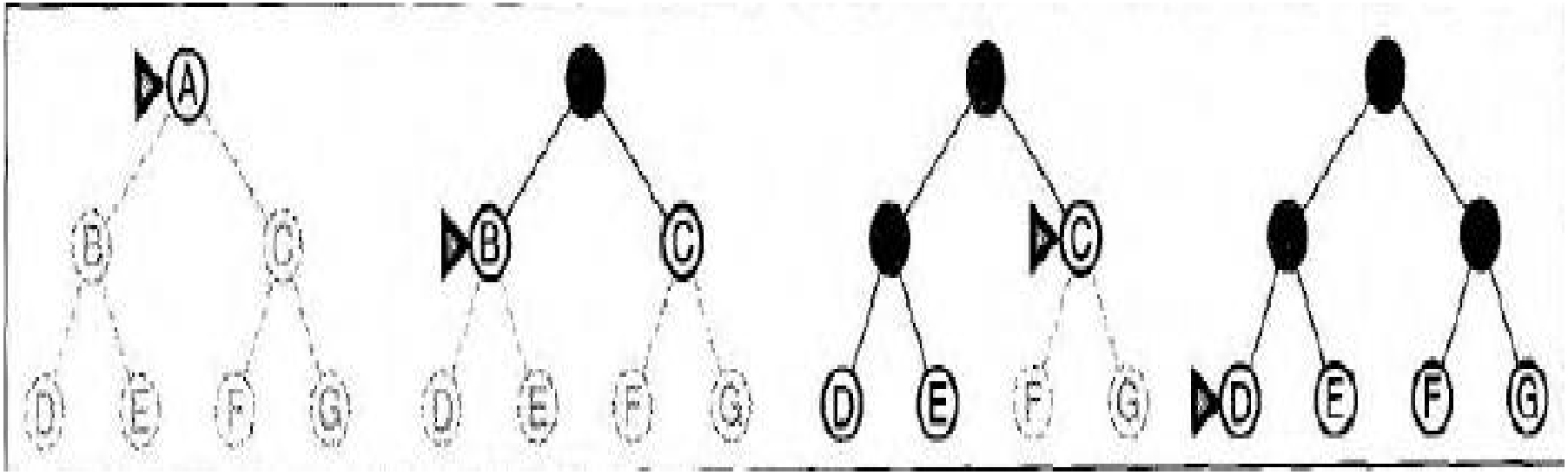
Stratégies de recherche non informées

- Largeur d'abord (Breath-first)
- Coût uniforme (Uniform-cost)
- Profondeur d'abord (Depth-first)
- Profondeur limitée (Depth-limited)
- Profondeur itératif (Iterative deepening)
- Recherche bidirectionnelle (Bidirectional search)

Parcours en largeur

- Le parcours en largeur est un algorithme de recherche très simple : nous examinons d'abord l'état initial, puis ses successeurs, puis les successeurs des successeurs, etc.
- Tous les noeuds d'une certaine profondeur sont examinés avant les noeuds de profondeur supérieure.
- Pour implementer cet algorithme, il suffit de placer les nouveaux noeuds systématiquement à la fin de la liste de noeuds à traiter.
- La frontière est gérée en FIFO qui assure que les nœuds visités en premier seront développés d'abord
- Dans une file FIFO tous les successeurs nouvellement générés sont placés à la fin, ce qui a pour effet de développer les nœuds en surface avant ceux qui sont plus en profondeur.

Parcours en largeur (exemple)



Le parcours en largeur pour un arbre binaire simple. Nous commençons par l'état initial A, puis nous examinons les noeuds B et C de profondeur 1, puis les noeuds D, E, F, et G de profondeur 2.

Largeur d'abord

- Complètude: oui, si b est fini
- Optimalité : non, le nœud but le moins profond peut ne pas être optimal
- La complexité en temps Nombre total de nœuds générés si tous les nœuds ont « b » successeurs et que le but soit à la profondeur « d » au pire cas

$$b + b^2 + b^3 + \cdots + b^d + (b^{d+1} - b) = O(b^{d+1}) .$$

- La complexité en espace est la même que la complexité en temps: chaque nœud généré doit rester en mémoire soit par parce qu'il appartient à la Frontière non encore développé) soit parce qu'il est un ancêtre d'un nœud sur la frontière.