Heuristiques

A. Introduction

B. Recherche d ’une branche

- . "Hill Climbing"
- . Beam Search
- . Best First Search

Heuristiques

Introduction

- **Une heuristique est une technique qui améliore l'efficacité d'un processus de recherche, en sacrifiant éventuellement la prétention à être complet.**
- **Pour des problèmes d'optimisation où la recherche d'une solution exacte(optimale) est difficile(coût exponentiel), on peut se contenter d'une solution satisfaisante donnée par une heuristique avec un coût plus faible.**
- **Certaines heuristiques sont polyvalentes (elles donnent d'assez bonnes résultats pour une large gamme de problèmes) alors que d'autres sont spécifiques à chaque type de problème.**

Heuristiques

Introduction

- **Dans le backtracking, dans les jeux de stratégie (jeu d'échec), on a déjà utilisé les heuristiques.**
- **Les heuristiques peuvent donner des solutions optimales, ce qui semble paradoxal (Algorithme A*).**

Heuristiques

Algorithmes de recherche utilisant les heuristiques :

1. Procédures de base simples utilisées pour trouver des branches(pas forcément les plus courtes) depuis des positions initiales jusqu'aux positions finales quand les longueurs des branches ne sont pas très importantes :

- **Hill Climbing**
- **Beam Search**
- **Best First search**

Heuristiques

Algorithmes de recherche utilisant les heuristiques :

2. Procédures plus complexes qui recherchent les plus courtes branches. Utilisé quand on donne une importance primaire au coût de la branche traversée.

- **Branch And Bound**
- **Branch and Bound avec sous estimations**
- **Branch and Bound avec programmation dynamique**
- **A***

Heuristiques

Algorithmes de recherche utilisant les heuristiques

- **Toutes ces méthodes dérivent des deux parcours suivants :**

— **Depht First Search**
— **Breadth First Search**

- **<u>Recherches guidées</u>**

- **Au niveau de chaque nœud, on a une information pour la recherche (coût, distance, ...)**

Heuristiques

Hill Climbing

- **C'est une recherche en profondeur.**
- **A chaque étape, on sélectionne le nœud avec distance (ou coût) minimale. C'est une amélioration de depht first search.**
- **Utilise une pile.**

— **1. Empiler le nœud racine**

— **2. Si la pile est vide, recherche sans succès**

— **3. Dépiler un élément e, s'il est égal à l'élément recherché, l'algorithme se termine avec succès, autrement**

— **4. *Trier les fils de e, s'ils existent, par l'estimation de la distance restante*, ensuite les empiler.**

— **5. Allera 2**

Heuristiques

Beam Search

- **Idem que breadth first search sauf que pour chaque niveau seulement les w "bons" premiers nœuds sont explorés.**

- **Beam : Rayon (ou faisceau lumineux.)**

Heuristiques

Remarque sur les deux méthodes :

- **Méthodes locales voraces (gloutonnes)**
- **Pas forcément efficace (Aucune garantie).**

- **C'est la classe des algorithmes voraces (gloutons) (du plus proche voisin)**

- **Glouton(vorace) qui mange avec avidité (qui désire avoir une solution toute de suite)**

Heuristiques

Méthodes gloutonnes : Principe

- **A chaque étape on sélectionne l'option qui est localement optimale(selon certains critères)**
- **Exemple : on veut totaliser une somme d'argent S avec un nombre minimal de pièces.**

  S = 25DA70c

  Pièce = { 10DA, 2DA, 1DA, 50c, 20c, 10c)

  Solution : 2X10DA + 2X2DA + 1X1DA + 1X50c + 1X20c

  au total 7 pièces.

Heuristiques

Méthodes gloutonnes : Algorithme

- **Prendre la pièce la plus grande < 25,70. Donc une pièce de 10DA, il reste 15,70.**
- **Prendre la pièce la plus grande < 15,70. Donc une pièce de 10DA, il reste 5,70.**
- **Prendre la pièce la plus grande < 5,70. Donc une pièce de 2DA, il reste 3,70.**
- **Ect...**

- **A chaque étape on choisit l'option localement optimale**

  Si on a des pièces de 1,10DA , 50c, et 10c et si l'on veut reconstituer 1,5 le même algorithme donnerait :

  1X(1,10DA) et 4X(10c).

  Mais la meilleure solution est 3X(50c).

Heuristiques

Méthodes gloutonnes :

- **Plusieurs algorithmes sont issus de cette technique.**
- **Algorithme de Dijkstra : il choisit le sommet le plus proche parmi tous ceux dont le plus court chemin n'est pas encore connu.**
- **Algorithme de Krustal : il choisit parmi les arêtes restantes celle de poids minimum ne provoquant pas de circuit**

Heuristiques

Méthodes gloutonnes : ***Le problème du voyageur de commerce(PVC)***

- **Étant donné n cités avec des distances entre chacune d'entre elles. Trouver un cycle Hamiltonien de poids minimale.**

- **Tous les algorithmes connus demandent un temps exponentiel. par contre, on peut donner une heuristique gloutonne dérivée de l'algorithme de Krustal qui donne généralement d'assez bons résultats.**

Heuristiques

Méthodes gloutonnes : ***Le problème du voyageur de commerce(PVC)***

- **Algorithme :**

  1. trier toutes les arêtes(il existe C_n^2)

  2. Prendre les arêtes une à une dans l'ordre en considérant les deux conditions suivantes :

  - Aucun sommet ne doit avoir un degré supérieur à 2.

  - Le seul cycle formé est le cycle final, quand le nombre d'arêtes acceptées est égal au nombre de sommets du graphe.

Heuristiques

Méthodes gloutonnes : ***Le problème du voyageur de commerce(PVC)***

- **Exemple :**

  6 villes.

  avec les coordonnées suivantes :

  c.(1, 7) **d.(15, 7)**

  e.(15, 4)

  b.(4, 3)

  a.(0, 0) **f.(18, 0)**

- **Il existe 6! = 720 permutations. Par contre seulement 15 arêtes (ab, ac, ad, ae, af, bc, be, bd, be, bf, ...).**

Heuristiques

Méthodes gloutonnes : ***Le problème du voyageur de commerce(PVC)***

- **Scénario :**
- **Choix de l'arête (d, e) car elle a une longueur égale à 3, la plus courte.**
- **On examine ensuite les arêtes (b, c), (a, b) et (e, f) de poids 5.(ordre quelconque). Elles sont toutes les 3 acceptables conformément aux critères 1. et 2.**
- **La prochaine arête la plus petite est (a, c) de poids 7,08. Comme elle forme un cycle avec(a, b) et (c, d) elle est rejetée.**
- **L'arête (d,e) est écartée dans les mêmes conditions.**

Heuristiques

Méthodes gloutonnes : ***Le problème du voyageur de commerce(PVC)***

- **Scénario :**
- **L'arête (b, e) est à son tour écartée car elle porte le degré de b et e à 3.**
- **Idem pour (b, d)**
- **L'arête suivante(c, d) est acceptée.**
- **On a maintenant un chemin a->b->c->d->e->f**
- **L'arête (a, f) est acceptée et ferme l'itinéraire.**

Heuristiques

Best First Search

- **Utilise une liste.(ou une file d'attente avec priorité)**

1. Mettre le noeud racine dans une liste.

2. Si la liste est vide, recherche sans succès.

3. Si Valeur(Prem(L)) est égal à l'élément recherché , l'algorithme se termine avec succès. Autrement

4. Supprimer le premier élément.

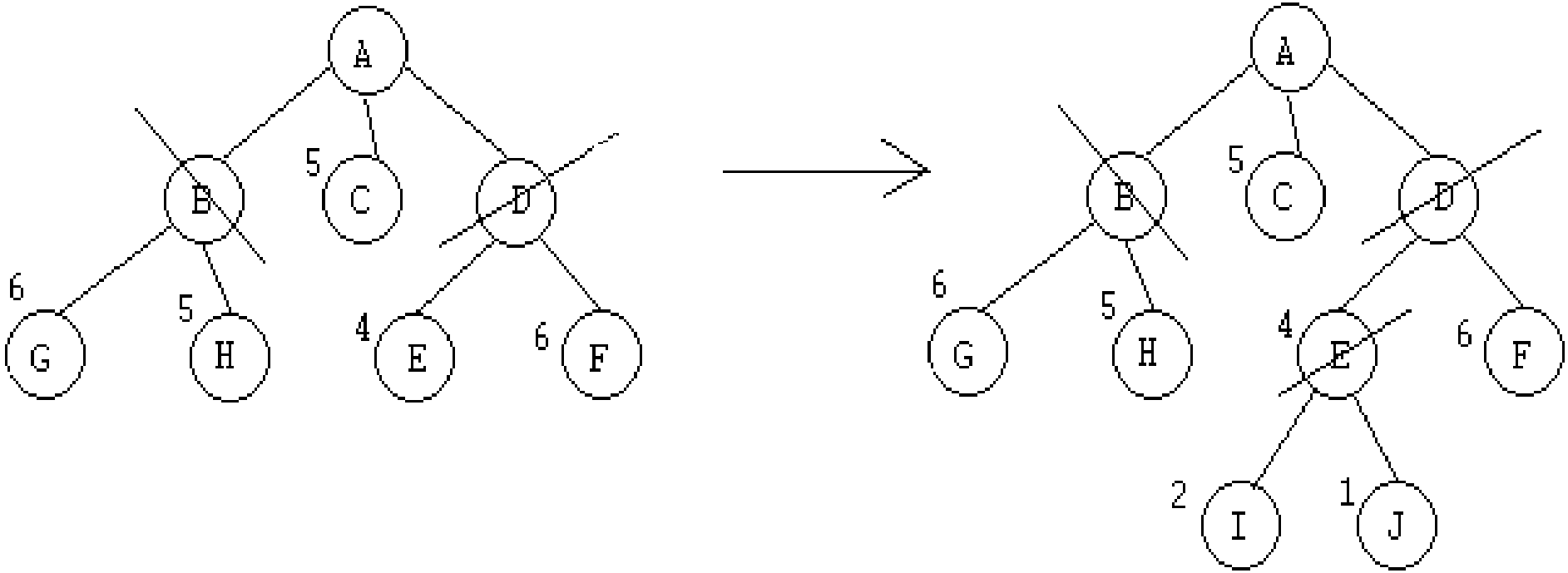
Pour chaque fils de cet élément :

calculer l'*estimation de la distance restante*

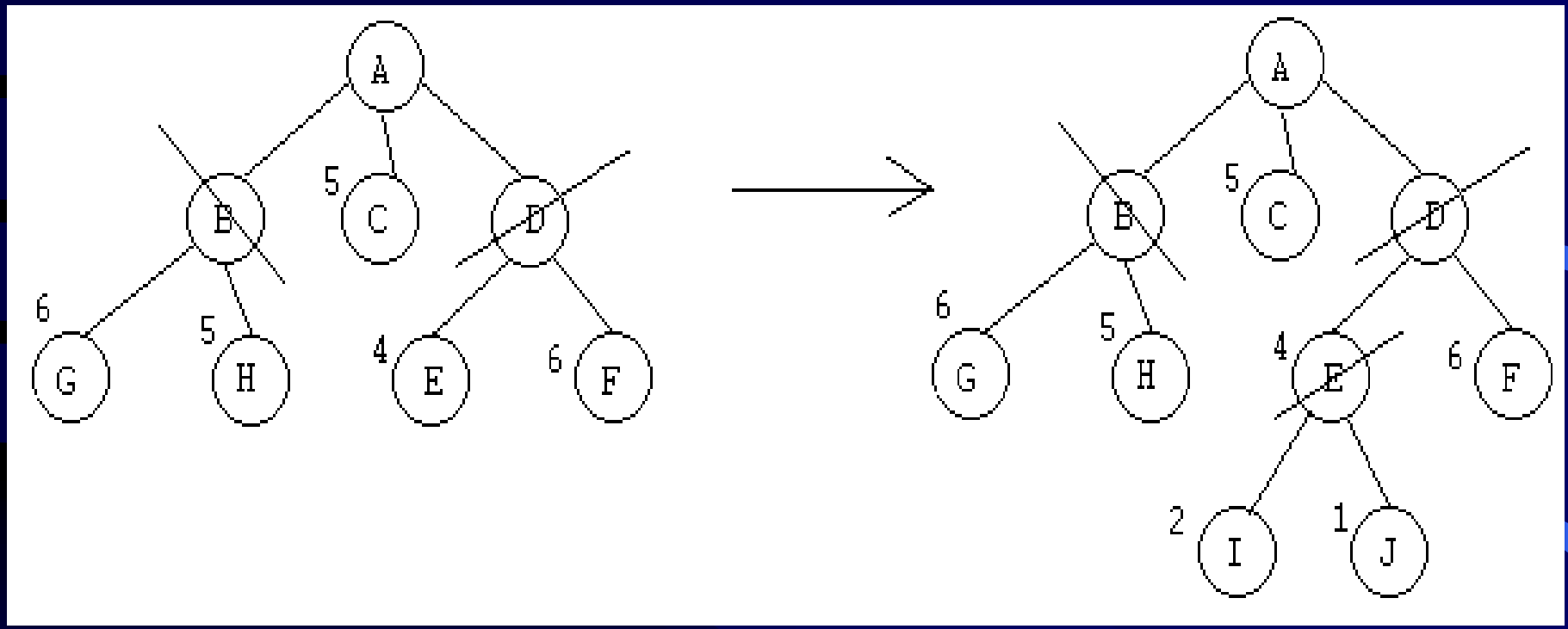
placer le dans la liste de telle sorte que la liste soit ordonnée.

5. Allera 2

Heuristiques



Heuristiques



Heuristiques

Remarques

- **Best First Search trouve toujours un bon chemin vers le but.(Avec la supposition qu'on fait une bonne estimation de la distance restante).**

- **Pour les trois méthodes on peut prendre comme exemple : Sortie d'un labyrinthe. L'estimation de la distance est donnée par la formule Rac($(x'-x)^2 + (y'-y)^2$)**