Heuristiques 2

D.E ZEGOUR
École Supérieure d'Informatique
ESI

Heuristiques

Sommaire

Autre version de l'algorithme A*

Méthodes gloutonnes

Heuristiques

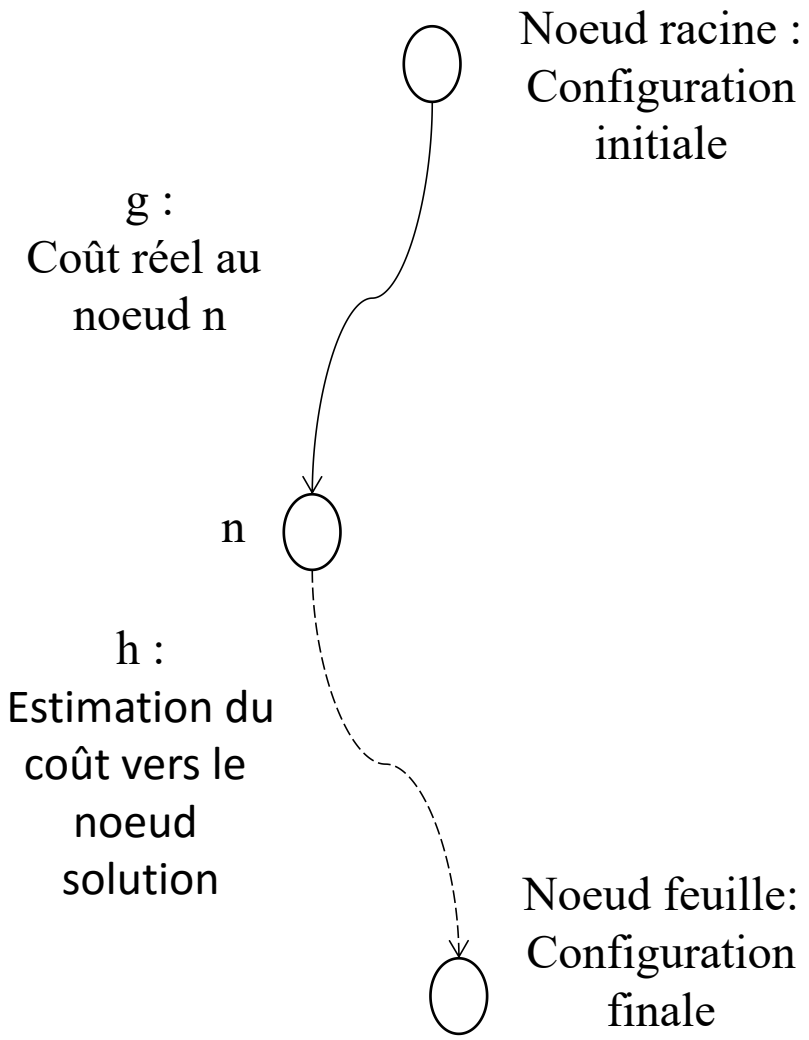
Autre version de l'algorithme A*

On maintient deux listes Ouvert(O) et Fermé(F) dont l'une est une file d'attente avec priorité(Ouvert).

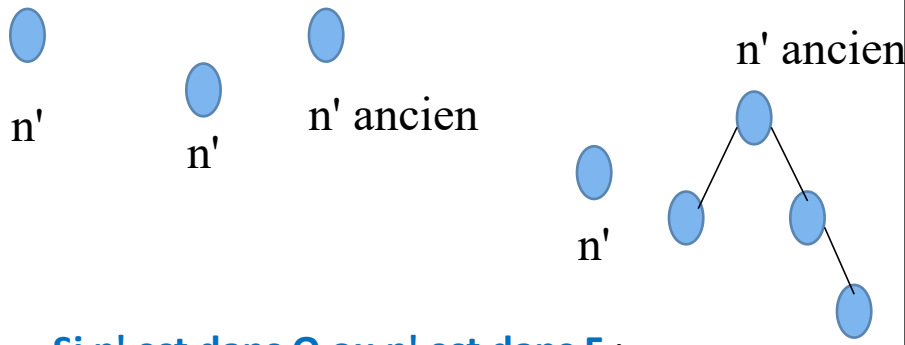
g : coût réel au nœud n

h : estimation de la distance restante

f = g + h



Heuristiques



Autre version de l'algorithme A*

1. O <-- nœud initial

2. Retirer un nœud n de Ouvert (avec a plus petite valeur de f(n))

Si c'est le nœud but, stop avec succès.
Autrement

- F <-- n

- Générer les successeurs n' de n

- Pour chaque successeur n' :

Si n' non dans O et n' non dans F :

. Calculer f(n') = g(n') + h(n')

. O <-- n'

. Attacher un pointeur arrière vers n

Fsi

Si n' est dans O ou n' est dans F :

. Changer le chaînage arrière selon le coût de la plus petite branche (g)

Fsi

Si n' est dans F et son chaînage arrière est modifié:

. l'enlever de Fermé

. Enlever tous les nœuds de la descendance de n' qui sont dans O et dans F.

. O <-- n'.

Fsi

Finpour

3. Allera 2

Heuristiques

Les numéros en rouge désignent l'ordre des villes visitées.
Les valeurs sous les nœuds désignent les cumuls + estimations des distances restantes.

Autre version de l'algorithme A* (Exemple)

F= { A , ~~D~~ , ~~F~~ , ~~E~~ , B

O= [A]

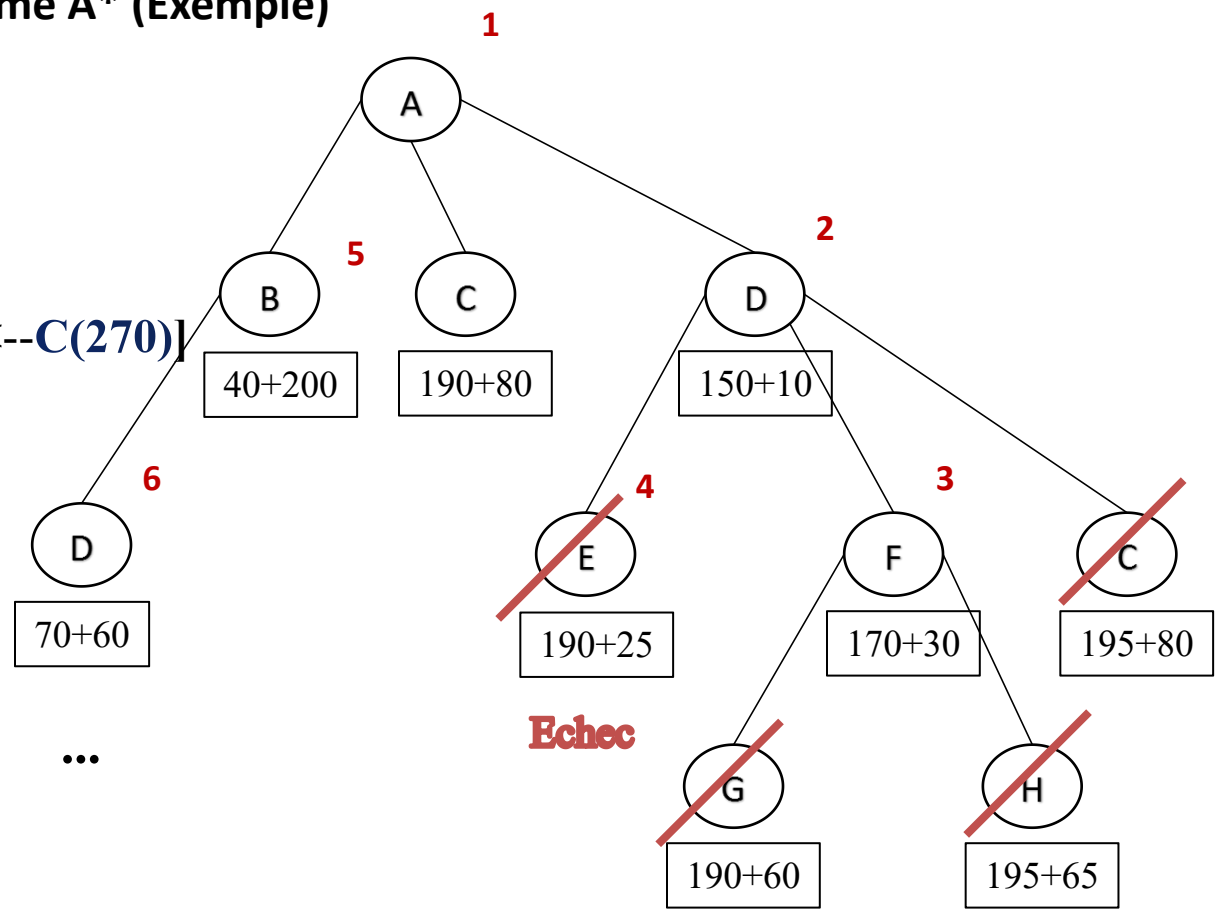
O= [A<--**D(160)**,A<--**B(240)**, A<--**C(270)**]

O= [D<--**F(200)**, D<--**E(215)**,
A<--**B(240)**, A<--**C(270)**]

O= [D<--**E(215)**, A<--**B(240)**,
F<--**G(250)**, F<--**H(260)**,
A<--**C(270)**]

O= [A<--**B(240)**, F<--**G(250)**,
F<--**H(260)**, A<--**C(270)**]

O= [B<--**D(130)**, A<--**C(270)**]



Heuristiques

Autre version de l'algorithme A* (Recapitulation)

BFS si g=1 →

g	h	
0	0	DFS (Backtracing)
#0	0	Branch and Bound
0	#0	Best First search
#0	#0	A*

← Hill Climbing si Pile au lieu de file d'attente avec priorité

Heuristiques

Autre version de l'algorithme A* (Propriétés)

Si h est parfait, A* converge immédiatement vers la solution sans recherche.

Meilleur est h, plus nous approchons de plus près de la solution.

Théorème :

Si h ne surestime jamais h (distance réelle), A* est assuré de trouver un chemin optimal vers un but s'il existe(Propriété d'admissibilté)*

h est admissible si quelque soit n : h(n) <= h(n)*

h (n) : coût réel au noeud n*

Heuristiques

Complexité de l'algorithme A*

La complexité dépend de l'heuristique

Dans le pire des cas,

Nombre de noeuds étendus : $O(b^d)$

b : facteur de branchement (nombre moyen de successeurs par étape)

d : profondeur de la solution (la plus petite branche)

Très gourmande en espace (parcours en largeur)

Heuristiques

Méthodes gloutonnes

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

Pas forcément efficace (Aucune garantie).

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

Principe :

- Construire une solution étape par étape
- A chaque étape on sélectionne l'option qui est localement optimale(selon certains critères)

Dans certains cas, peut donner la meilleure solution (Algorithmes gloutons exacts)

Dans d'autres, donne une solution satisfaisante (Heuristiques gloutonnes)

Heuristiques

Méthodes gloutonnes

Toutes les méthodes considérées

- Génère les successeurs,
- Evalue chaque successeur
- Choisit (selon une stratégie) un noeud parmi les noeuds ouverts

Méthodes gloutonnes :

- Génère à chaque étape un seul successeur
- Continue (Donc, pas de bactracking (retour arrière))

Heuristiques

Algorithme glouton :
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.
Ect...

Exemple 1 : Problème du rendu de monnaie

Exemple : on veut totaliser une somme d'argent S avec un nombre minimal de pièces.

S = 25DA70c

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

S= 1.5 DA

Pièces = {1,10DA , 50c, et 10c}

Solution naive

Enumérer toutes les combinaisons possibles, et choisir la meilleure.

Solution gloutonne

A chaque étape on choisit l'option localement optimale
(Heuristique utilisée : la plus grande pièce)

Solution : 2X10DA + 2X2DA + 1X1DA + 1X50c + 1X20c
Au total 7 pièces.

Le même algorithme donnerait :
1 X(1,10DA) et 4X(10c) [5 pièces]

Meilleure solution : 3X(50c) [3 pièces]

Heuristiques

Exemple 2 : Code de *Huffman*

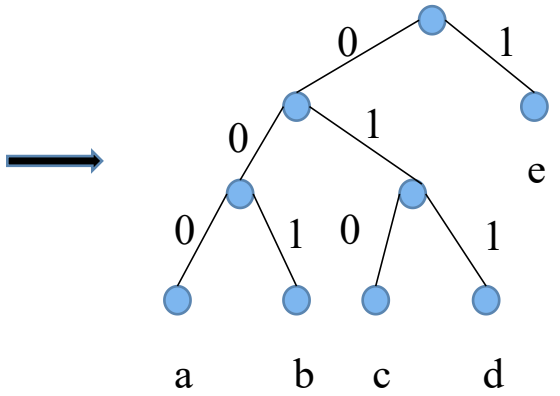
Soit A un alphabet avec les caractères a, b, c, d, e apparaissant avec des probabilités données

Coder chaque caractère en une suite de bits avec la propriété du préfixe (Aucun code n'est la préfixe d'un autre code)

La propriété du préfixe assure l'opération de décodage

A chaque code est associé un arbre binaire

a : 000
b : 001
c : 010
d : 011
e : 1



Heuristiques

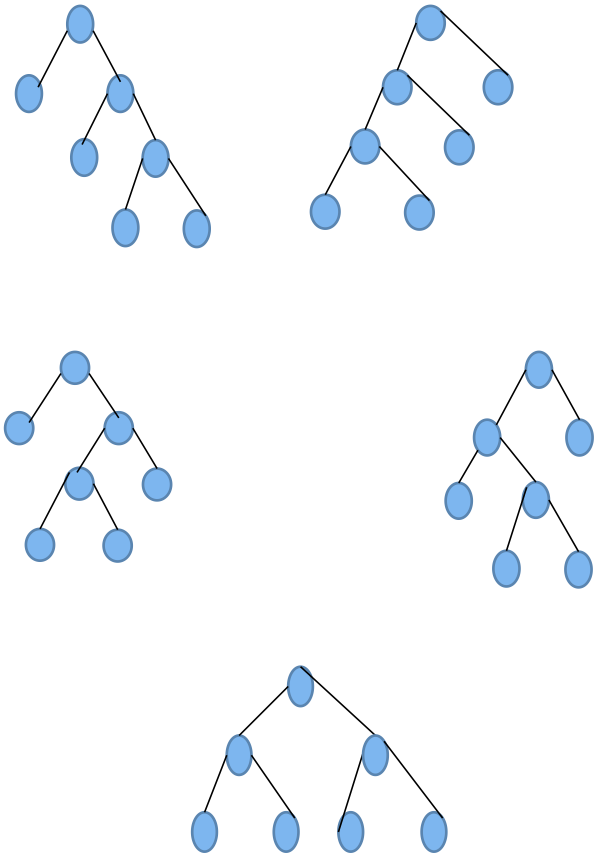
Exemple 2 : Code de *Huffman*

Solution naive:
Nombre maximal d'arbres strictement binaires ayant n+1 feuilles?

Egal au nombre de Catalan Cn
Cn= (2n)! / n!(n+1)!
C_{10}=16796 // C_{20}= 6 564 120 420

Pour n = 3
C3= (2X3)! / 3!(3+1)!
= 6! / (3!4!)
= 5

Pour n = 3



Heuristiques

Exemple 2 : Code de *Huffman*

Algorithme glouton :
1. Sélectionner deux caractères a et b ayant les plus faibles probabilités.

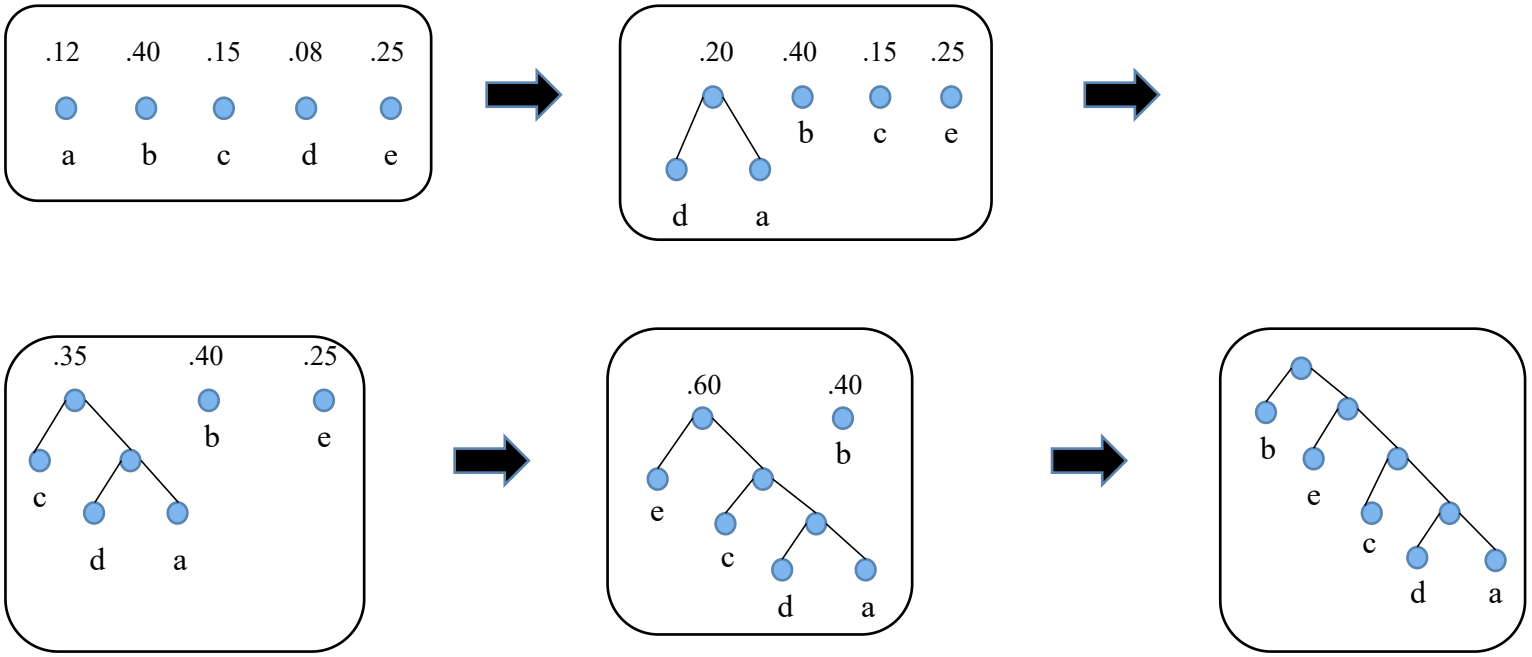
2. Remplacer les par un caractère fictif x tel que p(x) = p(a) + p(b)

3. Répéter 1. et 2. récursivement.

Théorème *: L'algorithme de Huffman produit un code optimal*

Heuristiques

Exemple 2 : Code de *Huffman (Scénario)*



Heuristiques

Exemple 3 : ***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.

Méthode naive : Générer toutes les possibiltés et choisir la meilleure (n!)

Solution gloutonne : l'algorithme de Krustal qui donne de bons résultats.

Heuristique utilisée : il choisit parmi les arêtes restantes celle de poids minimum ne provoquant pas de circuit

Heuristiques

Exemple 3: ***Le problème du voyageur de commerce(PVC)***

Algorithme :

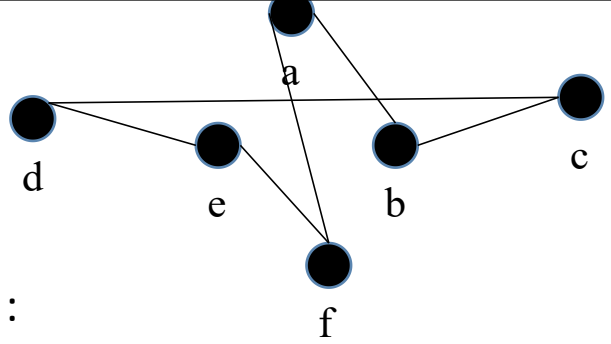
1. Trier toutes les arêtes(il existe C_n^2)

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



Exemple 3 : ***Le problème du voyageur de commerce(PVC)***

Exemple :

6 villes avec les coordonnées suivantes :

a:(0, 0) / b:(4, 3) / c:(1, 7) / d:(15, 7) / e:(15, 4) / f:(18, 0)

Il existe 6! = 720 permutations.

Par contre seulement 15 arêtes :(ab, ac, ad, ae, af, bc, be, bd, be, bf, ...).

Scénario :

Choix de l'arête (d, e) car elle a une longueur égale à 3, la plus courte.

Arêtes (b, c), (a, b) et (e, f) de poids 5: les 3 sont acceptées.

Prochaine arête la plus petite est (a, c) de poids 7,08. Comme elle forme un cycle avec(a, b) et (c, b) elle est rejetée.

L'arête (d,f) est écartée dans les mêmes conditions.

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.

- 4% que solution optimale