Intelligence Artificielle

Exploration Informée

-----Réalisé par-----

MOUHAJIR Oicila

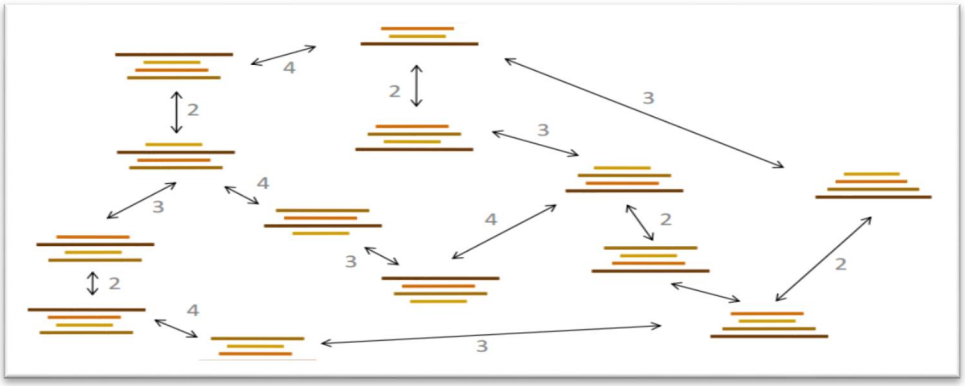
TISSOUDAL Iman

Génie des Systèmes Embarqués et Informatique Industrielle

ENSAF 2018/2019

I-Problématique :

- Le problème de crêpes consiste à mettre en ordre les crêpes avec le minimum cout possible.



- Pour résoudre ce problème l'UCS parait pourtant la solution à adopter puisqu'il est complet et optimale, mais le point faible c'est qu'il explore dans toutes les directions sans aucune information sur l'état objectif.

Comment peut-on diriger et guider la recherche vers la bonne direction afin d'approcher au GOAL ??

➔Une solution : orienter la recherche par une information heuristique !!

II-Exploration informée :

1-Heuristique :

La recherche heuristique représente une technique alternative de résolution de problèmes en intelligence artificielle.

➔Une heuristique est une fonction qui estime le coût d'un état à l'objectif .Elle utilise les caractéristiques du problème.

➔Une heuristique doit guider le choix des états à tester et les ordonner selon leurs promesses de rapprocher d'un but.

➔ Fonction heuristique h(n) : coût estimé du chemin le moins coûteux menant du nœud n à un but.

Exemple :

- **Distance de Manhattan** : $||x||1=Dm(x)=\sum|xi|=|x1|+|x2|$
- **Distance euclidienne** : $||x||2=\sqrt{(x1^2+x2^2)}$
- **Problème de crêpe** : pour résoudre le problème précédant on proposera comme heuristique le numéro de la plus large crêpe mal placée.

2-Exploration Glouton (Greedy) :

Cette technique développe le nœud le plus proche en utilisant l'heuristique h(n) qui estime le coût du chemin entre le nœud courant et le nœud le plus proche (solution).

➔Similaire à l'exploration en profondeur d'abord (BFS en considérant g(n)==h(n))

➔Une décision basée seulement sur une heuristique est fausse.

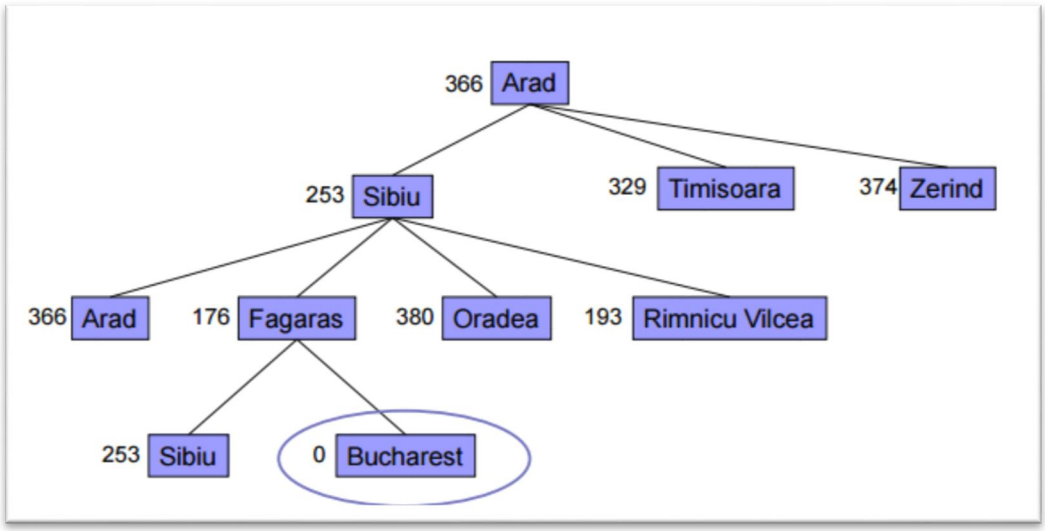
➔Cet algorithme ne produit pas toujours une solution optimale.

➔Complexité en temps et en espace : O (bm).

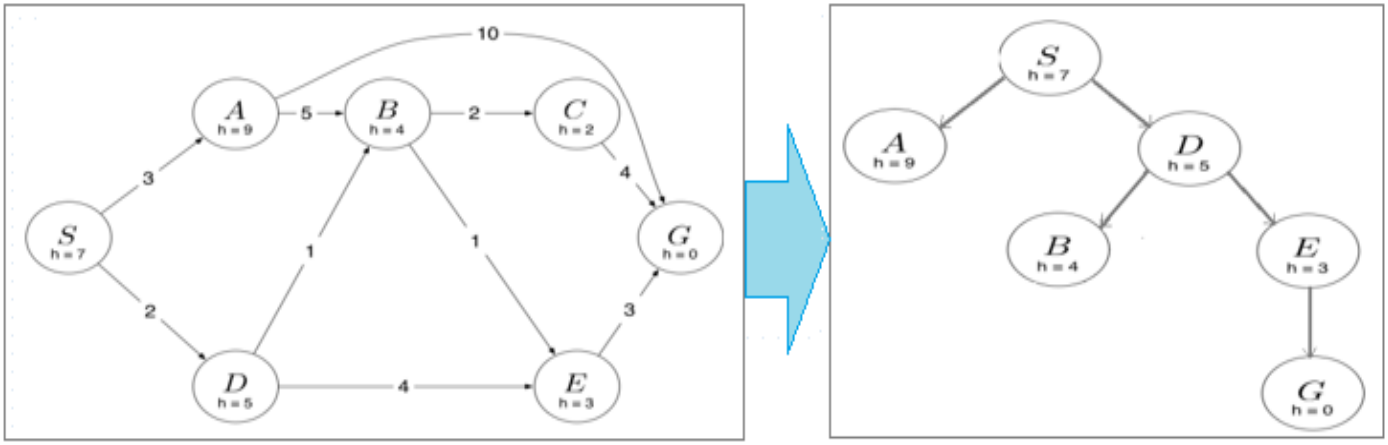
Exemple :

Le chemin calculé par l'exploration Glouton pour :

1-Le voyage en Roumanie (chemin optimal : **Arad->Sibiu->Fagaras->Bucharest**)



2-chemain de S à G (chemin optimal : **S->D->E->G)**



3-Exploration A* :

Cet algorithme combine les avantages des algorithmes de recherche en coût uniforme(**UCS**) et de recherche gloutonne (**Greedy**), en utilisant une fonction d'évaluation : **f(n) = g(n) + h(n).**

➔ A* donne une priorité selon la somme : f(n) = g(n) + h(n).

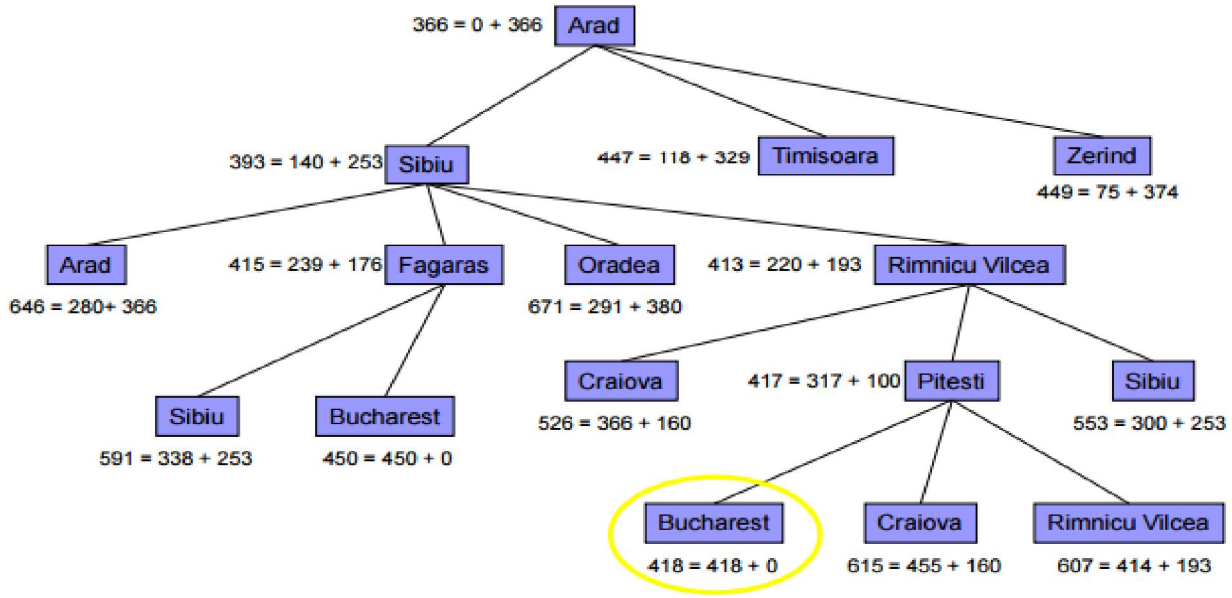
➔A* développe principalement vers l'objectif, et aussi dans les autres directions pour assurer l'optimalité.

➔On s'arrête quand on récupère l'objectif de la frontière.

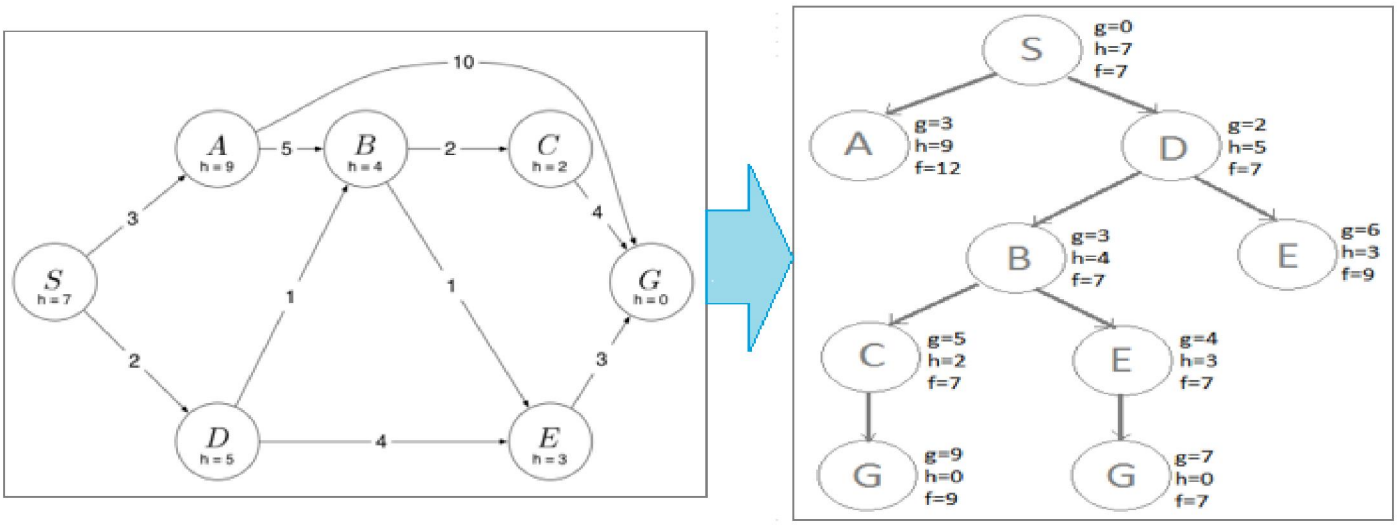
Exemple :

Le chemin calculé par l'exploration A* pour :

1-Le voyage en Roumanie (chemin optimal : **Arad->Sibiu->Rimnicu->Pitesti->Bucharest**)



2-chemain de S à G (chemin optimal : « **S->D->B->E->G** » avec « cout**=7 »)**



4-Heuristique admissible :

- Une heuristique h est admissible (optimiste) si : $0 \leq h(n) \leq h(n)^*$

Où $h(n)^*$: le coût réel à l'objectif le plus proche.

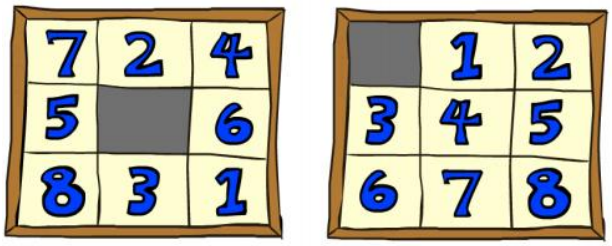
5-Création d'heuristique admissible :

-La tâche complexe dans un problème d'exploration, est de trouver une heuristique admissible.

➔Pour cela on utilise généralement la relaxation qui permet de limiter et réduire les contraintes.

Exemple : problème de puzzle :

Le but est de trouver une séquence optimale d'opérateurs qui mène les pièces de la configuration de l'état initial à l'état objectif.



Etat initial Etat objectif

➔**On peut considérer comme heuristique** : le nombre des cases mal placées.

Avec (h(Start)=8 et h(End)=0).

➔**Cardinalité** : 9 !

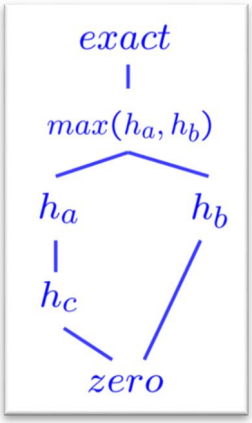
➔**Action possible** : selon la position de la case libre (gauche, droite, bas, haut).

➔**Cout :** combien de pas nécessaire pour déplacer une case (action=pas= 1).

➔**Problème relaxé** (les restrictions sur les actions seraient moindres).

III-Relations entre Heuristiques :

1-Dominance :



- Soit deux fonctions heuristiques admissibles $h_a(s)$ et $h_c(s)$. Si $\mathbf{h_a(s) \geq h_c(s)}$

On dit que h_a **domine** h_c et par la suite h_a est meilleure pour la recherche.

Remarque !

- Les heuristiques dominantes sont préférées tant qu'elles ne surestiment pas le coût minimum.

2-Heuristique incomparable :

- Généralement on ne peut pas toujours comparer deux heuristiques (pas forcément l'une domine l'autre).

- Dans ce cas on peut optimiser en définissant une heuristique composite h qui utilise la fonction la plus précise sur le nœud concerné : $\mathbf{h(n)=max\ (h_a(s), h_b(s))}$

Remarque !

- L'heuristique h est admissible et domine les deux heuristiques h_a et h_b.

3-Heuristique triviale :

- Heuristique qui renvoie toujours 0 (c'est l'objectif).

IV-Exploration Graphe :

1-Répétition des états :

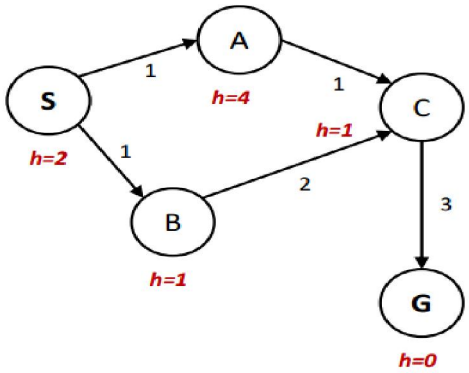
La répétition d'états fait perdre du temps, Dans le pire cas, la recherche tourne en rond dans les cycles créés. Donc pour détecter les répétitions on compare les nouveaux nœuds aux nœuds déjà développés (**Ensemble E**).

➔**Idée** : Ne pas développer un nœud deux fois.

➔**Implémentation** : Recherche en arbre + Un ensemble E pour les nœuds développés.

➔Avant de développer un nœud, on vérifie s'il existe déjà dans l'ensemble E.

Exemple : l'ensemble de nœud développé **E={S, A, B, C, G}**



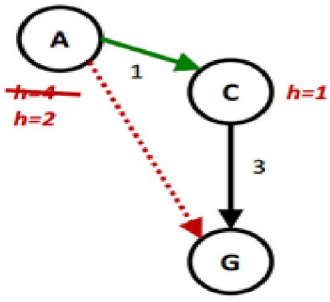
2-Consistance d'une Heuristique :

- Une heuristique est consistante si pour chaque nœud A et chaque successeur C de A, le cout estimé pour atteindre le but A est inférieur à la somme du cout pour aller de A à C et le cout estimé pour atteindre le but C.

$$h(A) \leq cost(A \text{ to } C) + h(C)$$

➔Conséquence de la consistance : la valeur de f est toujours croissante dans un chemin.

Exemple :



Pour h=4 ➔ l'heuristique est inconsistante car h(A)=4 ≥ 2

Pour h=2 ➔ l'heuristique est consistante car h(A)=2 ≤ 2

3-Optimalité :

➔**Exploration en arbre** :

A* optimale si h(n) est admissible.

➔**Exploration en graphe** :

A* optimale si h(n) est consistante.