USTHB
Faculté d'Electronique et Informatique
Département d'Informatique
Master S2I, 2ème année

11 décembre 2010

Test 1
'Méta-heuristiques et Data mining'

Le problème de la tour de Hanoi consiste en 03 piquets A, B et C, et n disques de différentes tailles. Initialement les disques sont empilés sur A selon l'ordre décroissant de leur taille, le plus grand disque en bas du piquet et le plus petit en haut. Le problème est de déplacer les disques de A vers B **un à la fois** et **en évitant de placer un disque de grande taille sur un autre de plus petite taille**. Le couloir C peut être utilisé pour un stockage temporaire des disques.

1) Ecrire un algorithme itératif pour résoudre le problème de la tour de Hanoi. Illustrer votre algorithme pour n=3.

2) Parmi les méthodes heuristiques suivantes :
   a. A*
   b. ACO
   c. Algorithmes génétiques

   Quelles sont celles qui sont les plus adéquates au problème de la tour de Hanoi ? justifier votre réponse.

3) Développer un algorithme de recherche taboue simple pour ce problème :
   a. Proposer un codage de la solution
   b. Modéliser la fonction objectif
   c. Détailler les composants de votre algorithme.

Rédiger vos réponses clairement et de manière précise.

Bon courage !

Prof. Habiba Drias

Corrigé du test1
Méta-heuristiques et Data Mining

1) Version itérative :
   - Recherche en largeur d'abord ou bien
   - Recherche en profondeur d'abord

Recherche en largeur d'abord :

Structures de données : 3 tableaux d'entiers A, B et C représentant les disques et respectant la contrainte du problème.

Description de l'espace des états :
- Un état est une configuration des 3 tableaux.
- L'état initial est :
  - A : n, n-1, …, 1
  - B :
  - C :
- L'état final est :
  - A :
  - B : n, n-1, …, 1
  - C :
- L'opérateur :
  - Déplacer(i, X, Y) où X, Y ϵ {A, B, C} et X contient i au sommet et Y contient j au sommet tel que j > i.

Algorithme :

1- Insérer l'état initial s dans une liste appelée OPEN gérée FIFO
2- Si OPEN est vide alors échec sinon continuer
3- Retirer l'élément qui se trouve en tête de OPEN et l'insérer dans une liste appelée CLOSED. Soit n cet état.
4- S'il n'existe pas de successeur pour n alors aller à 2. Engendrer les successeurs de n et les insérer à la queue de OPEN. Créer un chaînage de ces nœuds vers n
5- Si parmi les successeurs, il existe un état final alors succès: la solution est obtenue en suivant le chaînage arrière de ce nœud vers la racine, sinon aller à 2

Illustration de l'algorithme pour n=3 :

(A : (3, 2, 1) ; B : (); C : ()) → (A : (3,2) ; B : (1) ; C : ()) ou bien (A : (3,2) ; B : (); C : (1))
(A : (3, 2) ; B : (1) ; C : ()) → (A : (3) ; B : (1) ; C : (2))
(A : (3, 2) ; B : (); C : (1)) → (A : (3) ; B : (2) ; C : (1))
(A : (3) ; B : (1) ; C : (2)) → (A : (3, 1) ; B : () ; C : (2)) ou bien (A : (3) ; B : () ; C : (2, 1))
(A : (3) ; B : (2) ; C : (1)) → (A : (3,1) ; B : (2) ; C : ()) ou bien (A : (3) ; B : (2, 1) ; C : ())
(A : (3, 1) ; B : () ; C : (2)) → ()
(A : (3) ; B : () ; C : (2, 1)) → (A : () ; B : (3) ; C : (2, 1))
(A : (3,1) ; B : (2) ; C : ()) → ()
(A : (3) ; B : (2, 1) ; C : ()) → ()
(A : () ; B : (3) ; C : (2, 1)) → (A : (1) ; B : (3) ; C : (2)) ou bien (A : () ; B : (3,1) ; C : (2))
(A : (1) ; B : (3) ; C : (2)) → (A : (1) ; B : (3,2) ; C : ())
(A : () ; B : (3,1) ; C : (2)) → ()
(A : (1) ; B : (3,2) ; C : ()) → (A : () ; B : (3,2,1) ; C : ())

Etat initial en rouge et état final en vert.

2) A* est plus approprié car le problème de l'admissibilité de l'état une fois engendré ne se pose pas.
ACO et AG sont moins appropriés car ce sont des processus stochastiques. Le problème de l'admissibilité se pose quand les solutions intermédiaires sont engendrées aléatoirement ou construites selon le processus stochastique propre à la méta-heuristique.

3)

a. Représentation de la solution : un vecteur de m déplacements m>n :
d1, d2, d3, …, dm

Admissibilité : corriger les déplacements qui violent la contrainte du problème par exemple en essayant d'autres déplacements possibles qui marchent.

b. Fonction objectif :
f(s) = $\max_{1 \leq = i \leq = m}$ $nb\ de\ disques\ bien\ placés\ sur\ B\ d'un\ di\ de\ s$

c. Algorithme :

début
engendrer une solution aléatoire s et la rendre admissible ;
i = 1 ;
tant que (i <= maxIter) et (f(s) < n) faire
début
soit s' un voisin de s ;
tant que (f(s')<= f(s)) et (il existe un voisin de s non visité) faire
s'= un autre voisin de s ;
si (f(s') > f(s)) **alors début** insérer s dans la liste taboue
s = s' ;
fin ;
sinon appliquer le critère d'aspiration ;
fin ;
fin ;

Obtenir un voisin : remplacer un déplacement di de s par un autre déplacement possible.

Critère d'aspiration : revenir à la solution la plus ancienne de la liste taboue, ce qui revient à gérer la liste taboue selon la stratégie FIFO puis considérer un autre voisin de meilleure qualité.

Paramètres empiriques : maxIter et la longueur de la liste taboue.

Prof. Habiba Drias