Centre Universitaire de Mila

Institut des sciences et de la technologie

1ère Master STIC

Année : 2018-2019

Module :

Contrôle d’optimisation combinatoire et

métaheuristiques

**Exercice 1: 5.5 points**

1. La différence entre une heuristique et une métaheuristique : Généralement une heuristique est conçue pour un problème particulier tandis que les métaheuristiques peuvent être utilisées pour différents types de problème.’
2. En utilisant comme exemple le problème de sac à dos, expliquer les caractéristiques des problèmes de la classe NP.

Un algorithme non déterministe polynomial pour choisir la meilleure solution

Un algorithme polynomial pour vérifier une solution

{x1=1, x2=0,…}

{x1=1, x2=0,…}

..

.

.

{x1=0, x2=0,…}

S

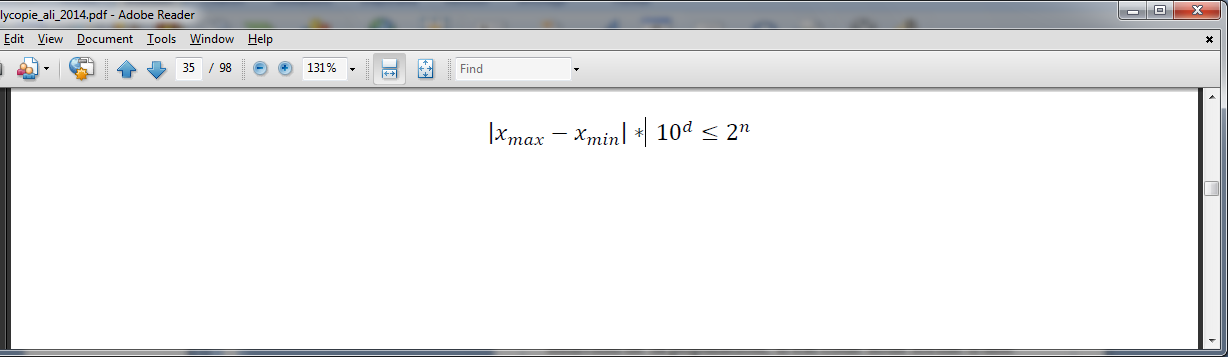
Ensemble

De solution

1. Soit P1 et P2 deux problèmes d’optimisation :

* Quand on peut dire que P1 est NP-complet ?
  + On peut dire que P1 est NP complet si P1 possède la propriété que tout problème dans NP peut être réduit en P1 en temps polynomial.
* Si P1 est NP-complet et P1 se réduit à P2. Qu’est-ce qu’on peut dire sur P2.
  + On peut dire que P2 est NP-difficile.

1. Nous cherchons à trouver le maximum d’une fonction *f* (x) sur l’intervalle [0, 128] en utilisant un algorithme génétique. Calculer la taille du chromosome (représentation binaire) dans le cas où *x* est un entier et dans le cas où *x* est un réel avec deux nombre après la virgule.



1. 128-0 \* 100 ≤2n  🡺 128≤2n 🡺 27≤2n 🡺 n=7
2. 128-0 \* 102 ≤2n🡺27 \*102<2n

On a 26<102<27 🡺 n=14

**Exercice 2: 3.5 points**

On cherche à trouver le plus court chemin entre le nœud n0 et n6.

1. Résoudre ce problème par l’algorithme A\*.

|  |  |  |
| --- | --- | --- |
|  | **Open** | **Closed** |
| **1** | (n0, 9, void) | Vide |
| **2** | (n1, 5, n0), (n3, 6, n0), (n2, 7, n0) | (n0, 9, void) |
| **3** | (n3, 6, n0), (n2, 7, n0), (n5, 12, n1) | (n0, 9, void), (n1, 5, n0), |
| **4** | (n2, 7, n0), (n4, 9, n2) ,(n5, 12, n1) | (n0, 9, void), (n1, 5, n0), (n3, 6, n0) |
| **5** | (n3, 5, n2), (n4, 6, n2) ,(n5, 12, n1) | (n0, 9, void), (n1, 5, n0), (n2, 7, n0) |
| **6** | (n4, 6, n2) ,(n5, 12, n1) | (n0, 9, void), (n1, 5, n0), (n2, 7, n0), (n3, 5, n2), |
| **7** | (n6, 7, n4) ,(n5, 12, n1) | (n0, 9, void), (n1, 5, n0), (n2, 7, n0), (n3, 5, n2), (n4, 6, n2) |
| **8** | (n5, 12, n1) | (n0, 9, void), (n1, 5, n0), (n2, 7, n0), (n3, 5, n2), (n4, 6, n2) , (n6, 7, n4) |

**Solution:** n0, n2, n4, n6

1. Est-ce que la solution trouvée est optimale ? justifier.

On ne peut pas dire si elle est optimale ou non parce que l’heuristique est non admissible.

**Exercice 3: 5 points**

1. La formulation mathématique de ce problème.

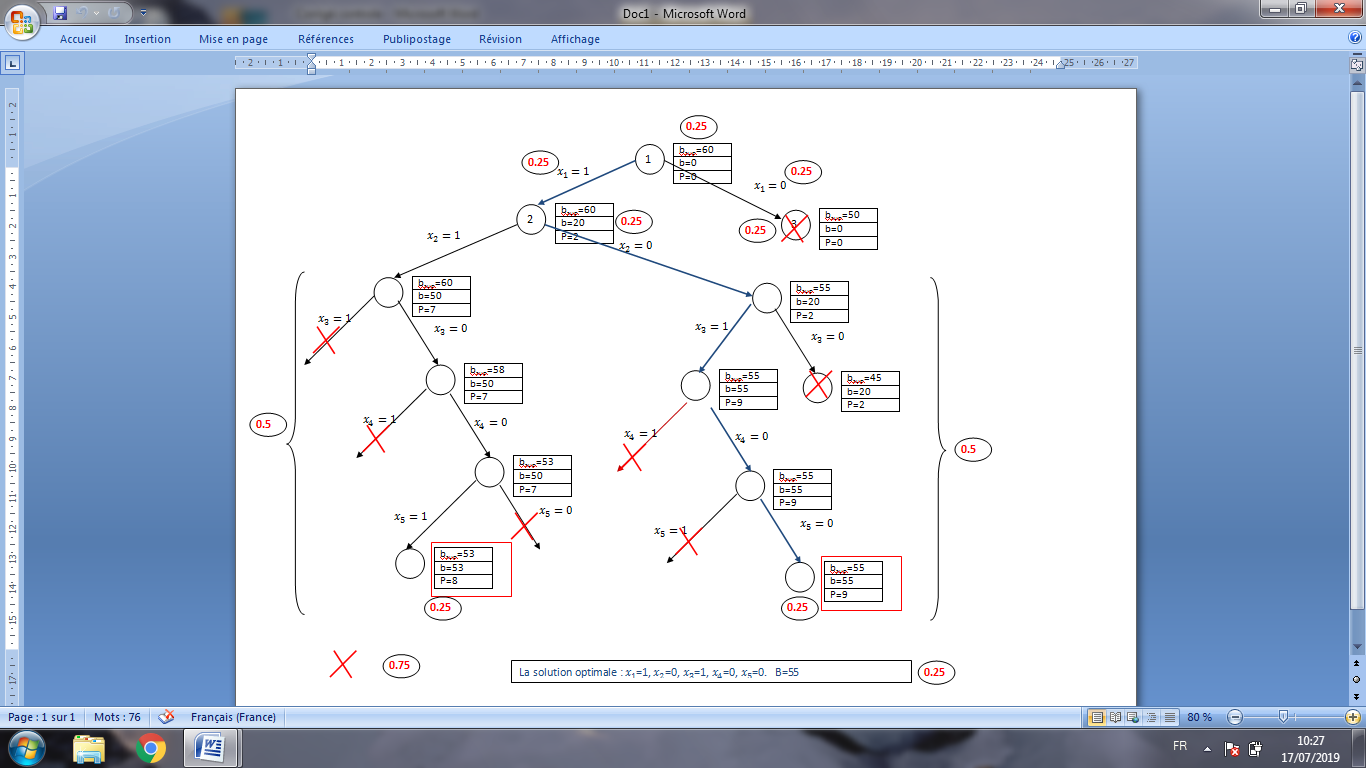
Variable de décision : *x1, x2, x3, x4, x5.*  Pour chaque objet nous avons un variable de décision *xi*

Fonction objectif :

Contraintes : a)

b)

1. En choisissant un parcours en profondeur, appliquer l’algorithme branch and bound pour la résolution de ce problème. Préciser la méthode de calcule de la borne supérieure pour les trois premiers nœuds dans l’arbre.

****

**Exercice 4: 6**

1. Si on utilise la méthode de recuits simulés avec T=10. Calculer pour chacun des voisins de la solution s la probabilité d’être choisi comme solution courante.

S= (T1🡨P1, T2🡨P3, T3🡨P4, T4🡨P2) f(s)= 9+8+9+7 =33

|  |  |
| --- | --- |
| N(s) | probabilité |
| (T1🡨P3, T2🡨P1, T3🡨P4, T4🡨P2) | f(s0)= 23 <f(s) 🡺 P=1 |
| (T1🡨P1, T2🡨P4, T3🡨P3, T4🡨P2) | f(s0)= 23 <f(s) 🡺 P=1 |
| (T1🡨P1, T2🡨P3, T3🡨P2, T4🡨P4) | f(s0)=24 <f(s) 🡺 P=1 |
| (T1🡨P2, T2🡨P3, T3🡨P4, T4🡨P1) | f(s0)=31 <f(s) 🡺 P=1 |

1. La méthode de recherche taboue :

Initialisation :

S🡨 (T1🡨P1, T2🡨P3, T3🡨P4, T4🡨P2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T | (T1🡨P1, T2🡨P3, T3🡨P4, T4🡨P2) |  |  |  |

Best🡨(T1🡨P1, T2🡨P3, T3🡨P4, T4🡨P2)

Itération 1 :

N(s) ={ (T1🡨P2, T2🡨P1, T3🡨P4, T4🡨P2), (T1🡨P1, T2🡨P4, T3🡨P3, T4🡨P2), (T1🡨P1, T2🡨P3, T3🡨P2, T4🡨P4), (T1🡨P2, T2🡨P3, T3🡨P4, T4🡨P1)}

S0= (T1🡨P1, T2🡨P4, T3🡨P3, T4🡨P2)

f(s0)=23

best=(T1🡨P1, T2🡨P4, T3🡨P3, T4🡨P2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T | (T1🡨P1, T2🡨P3, T3🡨P4, T4🡨P2) | (T1🡨P1, T2🡨P4, T3🡨P3, T4🡨P2) |  |  |

Itération 2 :

N(s) ={ (T1🡨P4, T2🡨P1, T3🡨P3, T4🡨P2), (T1🡨P1, T2🡨P3, T3🡨P4, T4🡨P2), (T1🡨P1, T2🡨P4, T3🡨P2, T4🡨P3), (T1🡨P2, T2🡨P4, T3🡨P3, T4🡨P1)}

S0= (T1🡨P4, T2🡨P1, T3🡨P3, T4🡨P2)

f(s0)=17

best=(T1🡨P4, T2🡨P1, T3🡨P3, T4🡨P2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T | (T1🡨P1, T2🡨P3, T3🡨P4, T4🡨P2) | (T1🡨P1, T2🡨P4, T3🡨P3, T4🡨P2) | (T1🡨P4, T2🡨P1, T3🡨P3, T4🡨P2) |  |

1. Après la définition des types de données nécessaires, écrire en algorithme la fonction f(s, M) permettant d’évaluer une solution s où M est la matrice des coûts.

Structure solution

T : tableau d’entier

f : entier

fin

fonction f (s : solution, M : matrice d’entier)

f,i : entier

Début

f🡨0 ;

Pour i allant de 1 à 4 faire

f🡨f + M[i][s.T[i]] ;

Fin pour

Retourne f ;

fin