Rapport TP2 Recherche Opérationnelle

Algorithme de branch-and-Bound pour résoudre le problème du sac à dos

Kamal HAMMI
Mohamed Hamza LOTFI

20 décembre 2021

1 Implementation

1.1 Informations relatives aux fonctions utilisées dans l'imlementation

1. **SolveKnapInstance :**
   On a remplacé le modèle CLP avec un nouveau modèle qui repose sur 4 variables qui évoluent au cours de l'algorithme :
   - price_courant : Liste des coûts
   - weight_courant : Liste des poids
   - capacity_courant : La capacité maximale
   - price_constant_courant : La constante de la fonction objectif.

   En plus de certaines variables qui seront utiles pour la coordination entre les différentes focntions utilisées :
   - indices_correspond_courant : La liste des variables restantes après un choix de varibale de décision.
   - valeurs_binaires_courant : Les parties entières des éléments du vecteur solution courant.
   - list_indices_separation : La liste cumulative (après chaque itération on ajoute un élément à la liste) des indices des variables du vecteur indices_correspondant_courant dont la valeur n'est pas binaire à l'étape courante.

   Au sein de la boucle, certaines variables résultantes de la fonction **optimizer** ont été ajoutées : infaisable, cout, solutions_courant.
2. **optimizer :**
   Cette fonction calcule la solution relaxée du problème en utilisant la liste des résidus qui n'est d'autre que le rapport des coûts sur les poids. On prend à chaque étape la variable qui a le plus grand résidu tant que le poids total est inférieur à la capacité maximale, lorsqu'à une étape donnée, on est au-dessus de la capacité, on affete à la variable de cette étape le quotient de la capacité restante sur son poids. La variable "infaisable" est vraie si la capacité donnée à l'algorithme est négative, ou si la liste des coûts courants donnée à l'algorithme est vide. Le coût est calculé à partir de la fonction objective avec les valeurs des variables calculées après chaque étape.
3. **TestsSondabilite_relaxlin :**
   Cette fonction renvoie les tests de sondabilité à partir des variables "infaisable", "cout", "solutions_courant", "valeurs_binaires_courant"
4. **SeparerNoeud_relaxlin :**
   Cette fonction choisit les variables de séparation de la façon suivante et les supprime du modèle après leur utilisation dans la mise à jour de l'arbre, elle renvoie la capacité courante le coût constant courant mis à jour.

5. **ExplorerAutreNoeud_relaxlin :**
   Cette fonction gère l'exploration des autres noeuds à partir du noeud courant, tout en ajoutant à chaque retour au noeud père la dernière variable de listvars, et en la retranchant en cas de passage au noeud de droite.
6. **AfficherNotreModele** et **AfficherSolutions :**
   Deux fonctions qui affichent à chaque fois le modèle mis à jour et les solutions trouvées.

1.2 Adéquation des résultats obtenus

Dans la résolution de l'instance donnée, l'algorithme a parcouru 12 noeuds, et a trouvé deux solutions :

- La première ([1, 0, 1, 0]) avec un coût égal à 54 dans le noeud 5.
- La deuxième ([0, 1, 0, 1]) avec un coût égal à 65 dans le noeud 12.

Bien évidemment, la solution choisie est la deuxième car elle maximise le coût et donc répond à l'objectif du problème.

1.3 Comparaisons entre différentes implantations

1. Implantation avec le choix du max des résidus sur l'instance donnée :
   L'algortithme a parcouru **12 Noeuds** et a duré **0.011174 s**.
2. Implantation avec le choix du max des coûts sur l'instance donnée :
   L'algorithme a parcouru **8 Noeuds** et a duré **0.290961 s**