Université du HAVRE

UFR Sciences et Techniques

Master 2 MATIS

Optimisation Combinatoire et Métaheuristiques « Problème de Bin-Packing »

Réalisé par :

- CHIBANE MOURAD

Sous la direction de :

- M. STEFAN BALEV

2012/2013

I. Introduction :

L'objectif du TP « Problème de Bin-Packing » est d'analyser le problème qui est un problème d'optimisation appelé aussi problème du conditionnement dans des boites et de proposer une modélisation appropriée pour des méta heuristiques itératives basées sur la notion de voisinage, c'est-à-dire passer de la description textuelle du problème [01], à une formulation plus au moins mathématique [02].

[01] Mettre un ensemble d'articles fixés dont on connait le poids, dans des boites de même capacité en utilisant le moins de boites possible.

[02] Soit :

- Un nombre infini de boîtes de capacité C.
- Une liste d'articles i, de poids : c_i tel que : $i \in \{1,2,\dots.,n\}$.

$$\min z(y) = \sum_{i=1}^{n} yi$$

Sous les contraintes :

$$\sum_{j=1}^{n} w_j\, x_{ij} \leq c y_j \qquad j \in N^{*+}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j \in N^{*+}$$

$$y_i, x_{ij} \in \{0,1\} \qquad i,j \in N^{*+}$$

On cherche à trouver le rangement valide pour tous ces articles qui minimise le nombre de boîtes utilisées. Pour qu'un rangement soit valide, la somme des tailles des articles affectés à une boîte doit être inférieure ou égale à C.

Ensuite il faut implémenter une solution en Java pour résoudre le problème d'une manière optimale.

II. Aspect théorique :

Il existe deux grandes catégories de méthodes de résolution de problèmes d'optimisation combinatoire : les méthodes exactes et les méthodes approchées.

Les méthodes exactes permettent d'obtenir la solution optimale à chaque fois, mais le temps de calcul peut être long si le problème est compliqué à résoudre.

Les méthodes approchées, encore appelées heuristiques, permettent d'obtenir rapidement une solution approchée, donc pas nécessairement optimale. Nous allons détailler un exemple d'algorithme de résolution de chaque catégorie.

II.1 Méthodes approchées :

Une méthode approchée a pour but de trouver une solution avec un bon compromis entre la qualité de la solution et le temps de calcul. Pour le problème du bin packing, on peut utiliser l'algorithme naïf qui consiste à remplir les sacs les uns après les autres en prenant les objets les uns après les autres.

Toujours dans la famille des méthodes approchées, un algorithme appelé ***Best Fit Deacressing – BFD –*** (meilleur remplissage par ordre décroissant) a été mis au point pour améliorer l'algorithme naïf en :

• Triant tous les objets par ordre décroissant de poids ;

• Sélectionnant les objets un à un dans l'ordre du tri et en ajoutant l'objet sélectionné dans le sac le plus lourd tel que la contrainte de poids maximal reste respectée.

Cet algorithme donne une solution optimale mais il est imparfait, sa grande qualité est sa rapidité, qui reste intacte même si le nombre d'articles augmente considérablement.

Il existe de nombreux d'autres algorithmes approchés tel que :

• ***First Fit - FF :*** on place au fur et à mesure les objets dans le premier sac possible.

• ***Worst Fit - WF :*** on place au fur et à mesure les objets dans le sac le plus léger possible.

• ***Best Fit – BF :*** on place, au fur et à mesure, les objets dans le sac le plus lourd possible.

• ***First Fit Decreasing et Worst Fit Decreasing – FFD et WFD :*** on trie les objets dans l'ordre des poids décroissant puis on applique FF / WF.

Ce type d'algorithme est aussi appelé algorithme glouton, car il mange les données au fur et à mesure et ne remet jamais en cause une décision prise

auparavant.

II.1 Méthodes exactes :

Pour trouver la solution optimale, et être certain qu'il n'y a pas mieux, il faut utiliser une méthode exacte, qui demande un temps de calcul considérable « nettement plus long que le temps de calcul des méthodes approchées ».

L'un des algorithmes le plus connu dans ce type de méthode et qui a fait ses preuves est celui nommée ***procédure par séparation et évaluation « PSE »,*** en anglais *branche and bound*.

Une PSE est un algorithme qui permet d'énumérer intelligemment toutes les solutions possibles. En énumérant que les solutions potentiellement de bonne qualité et en excluant celles qui ne vont pas améliorer la solution courante.

Un *arbre de recherche* est utilisé pour représenter une PSE, constitué :

• de nœuds ou sommets, où un nœud représente une étape de construction de la solution.

• d'arcs pour indiquer certains choix faits pour construire la solution.

Le plus grand défaut de cette méthode est la taille de l'arbre qui est exponentielle en le nombre d'articles.

III. Approche proposée pour résoudre le problème de Bin-Packing :

La lecture de différentes sources (cours…) et de quelques articles scientifiques qui traite le sujet, m'a permis d'opté pour une approche qui combine deux méthodes d'optimisation :

- La première est une méta heuristique dirigée par une fonction objective « *cherchant à minimiser au maximum l'espace vide dans la boite courante* » avant de passer aux suivantes.
- La deuxième est un algorithme gloutonne avec une liste tabou afin d'interdire certaines permutation d'objet entre deux boites.

Bien évidemment les deux méthodes sont combinées après avoir triés les articles par ordre décroissant de poids, et les boite par ordre croissant d'indice dès le début de l'exécution.

Phases d'optimisation :

L'optimisation consiste à faire tourner les deux phases d'optimisation ci-dessous plusieurs fois jusqu'à satisfaction de la fonction objective de la méta heuristique mère « globale » (minimiser le nombre de boites utilisées) :

La première phase est l'application d'une méta heuristique « sous la méta heuristique globale » qui permet un tri croissant des espaces vides dans les boites en déplaçant un article $\boldsymbol{w_i}$ d'une boite $\boldsymbol{y_j}$ vers une autre boite $\boldsymbol{y_k}$ tel que $\boldsymbol{j > k}$. C'est ce qui nous permet d'améliorer la solution actuelle en passant à

une autre solution voisine de meilleur qualité.

Juste après, vient la deuxième phase d'optimisation qui est de calculer l'ensemble des permutations possibles de sorte que les espaces vides soit croissants d'une boite $\boldsymbol{y_j}$ à une boite $\boldsymbol{y_k}$ tel que $\boldsymbol{j > k}$.

IV. Tests et exécutions :

J'ai effectué les tests, en spécifiant à chaque fois comme fichier d'entrée les différents fichiers fournis sur le site « http://www.wiwi.uni-jena.de/Entscheidung/binpp/bin3dat.htm » et en comparant à chaque fois les résultat de mon approche avec une solution de type méthode approchée ***Best Fit Deacressing « BFD ».***

Vous trouvez un fichier txt dans la racine, contenant les traces d'exécution

Référence :

http://www.mecs-press.org/ijieeb/ijieeb-v4-n2/IJIEEB-V4-N2-2.pdf
http://dept-info.labri.fr/ENSEIGNEMENT/projet2/supports/Bin-Packing/probleme-bin-packing.pdf
http://compalg.inf.elte.hu/~tony/Kutatas/BinPacking/SimchiLevi-New%20worstcase%20results%20for%20the%20bin-packing%20problem.pdf