PROJET Recherche Opérationnelle
BIGNON Alan 685L

Résolution exacte du problème de bin-packing par des méthodes simples

Préambule :

Explication des fichiers présent dans le rendu :

- projet.jl : fichier fourni comme template
- main.jl : fichier contenant les tests de presque toutes les implémentations du projet, ce sera le seul fichier lançable du projet, tout se résout plutôt rapidement, il faudra attendre quelques minutes pour les instances B en 2D.
- question_2_1.jl : contient l'algo glouton résolvant l'Heuristique best-fit.
- question_2_2.jl : contient le solveur 1D en modélisation directe
- question 2_2_alternatif.jl : contient le solveur 1D en modélisation directe en prenant en compte le fait que des objets soit de même taille
- question_2_3_1.jl : contient l'algo qui trouve les motifs des données 1D
- question_2_3_2.jl : contient le solveur qui utilise les motifs pour résoudre le problème 1D
- question_3_1.jl : solveur 2D pour le problème à une seule bin
- question_3_2.jl : contient l'algo qui trouve les motifs 2D
- question_3_3.jl : contient le solveur du problème 2D

1. Explication des structures de données

J'ai décidé d'utiliser, pour la plupart des algorithmes présents dans le projet, des vecteur que sa soit des vecteurs de vecteur, d'int, ou d'objet1D/2D car il m'était très familier et me semblait plus facile d'utilisation, ce qui c'est avéré vrai par la suite.

2. Modèle section 2.1

L'algorithme de la section 2.1 se situe dans le fichier question2_1.jl (fonction nommée : algo_glouton, qui prend les données1D directement en paramètres).

3. Pseudo-code de l'algo qui trouve les motifs

Pseudo code de la fonction de recherche des motifs :

```
Fonction calcul_motifs (l, somme, TailleBin, motifLocal, tailleObjets,
motifs)
Variables :
Entier : l,Somme, TailleBin;
Booléen : ajouter
tableau d'Entier : motifLocal, tailleObjets;
tableau de tableau d'entier : motifs;



Début :

ajoutez <- false
Pour i de l à Taille(tailleObjets)
   Si somme + tailleObjets[i] <= tailleBin
      ajoutez <- true
      ajouter i à motifLocal
      fusionner motifs et calcul_motifs(i, Somme+tailleObjets[i],
TailleBin, MotifLocal, tailleObjets, motifs))
      supprimer dernier élement de MotifLocal
   finSi
```

```
FinPour
Si(non(ajoutez))
   ajoutez MotifLocal à motifs
FinSi
retournez motifs
Fin Fonction
```

4. Analyse synthétique des résultats

Données 1D :

L'algo glouton, de la question 2.1, pour déterminer un nombre minimum de bins est l'un des plus rapides, en moyenne 0.00015 secondes, nous verrons que le nombre de motifs obtenu est très proche de la solution exacte obtenu avec l'algo 2_3_2.

Les implémentations des algorithmes de la question 2.2, alternatif ou non, sont les plus inefficaces, après plusieurs minutes ils ne finissent toujours pas, même avec le jouet.dat.

L'implémentation de la question 2.3 est la plus efficace (pas compliqué avec les deux implémentation précédente) qui termine très rapidement pour la plupart des instances, seul deux instances prennent plus d'une seconde, la B15 qui prend 1,5 secondes et la B20 qui en prend 42.

Nous pouvons expliquer cette différence de temps de calcul par le fait que les instances B possèdent, pour un A de même chiffre, le même nombre d'objets mais leur taille est plus petite, ce qui fait que le nombre de motifs différents est beaucoup plus grand pour les instances B.

Nous pouvons conclure que, plus le nombre d'objets augmente, plus la taille de ces dits objets diminue, le temps de résolution sera plus grand, nous arrivons rapidement à des temps de résolution beaucoup trop grands.

Ce qui fait que la résolution des instances de B est donc le nombre très grand des motifs, par exemple les instances B15 et B20 ont

respectivement 9495 et 118656 motifs, les autres ne dépassant que rarement la centaine.

Ce genre de résolution avec plus d'objets, ainsi que des objets plus petits, que l'instance B20 pourrait très rapidement prendre des années.

Données 2D:

Nous avons une implémentation la 3.3 pour résoudre le problème de bin-packing bi-dimensionnel avec orientation, la 3.2 étant le calcul des motifs et la 3.1 le problème avec un seul bin.

Pour le nombre de motifs, il augmente de façon régulière en même temps que l'augmentation du nombre d'objets différents, ce qui est logique, ceci est valable pour des objets de même taille, comme pour en 1D, plus les objets sont petits, plus le nombre de motifs sera grand.

Quant à la résolution, toutes les instances de A se résolvent très rapidement mais les Instance B prennent beaucoup plus de temps (je ne les ai pas tous lancé, la B5 prend plus d'une minute).

Maintenant pour la résolution avec la 3.3, la résolution des instances de A, comme pour les motifs, se résout très rapidement contrairement aux instances B qui se laissent attendre, B5, B6 et B9 ne se résolvant même pas du tout après de longues minutes d'attente.