Modélisation du Problème de Bin Packing

MATTAR Omar – GRANGE Pierre

Introduction

Ce projet consiste à résoudre le problème de bin packing en deux dimensions en utilisant plusieurs algorithmes d'optimisation. Le problème de bin packing en 2D consiste à placer un ensemble d'items de tailles différentes dans des bins de taille fixe de manière à minimiser le nombre de bins utilisés.

Variables

1. **Items (Objets) :**

   - $I = \{1, 2, ..., n\}$: Ensemble des objets.
   - w_i : Largeur de l'objet i.
   - h_i : Hauteur de l'objet i.

2. **Bins (Bacs) :**

   - $B = \{1, 2, ..., m\}$: Ensemble des bacs.
   - W : Largeur maximale d'un bac.
   - H : Hauteur maximale d'un bac.

3. **Variables de décision :**

   - $x_{ij} = \begin{cases} 1 & \text{si l'objet } i \text{ est placé dans le bac } j \\ 0 & \text{sinon} \end{cases}$
   - $y_j = \begin{cases} 1 & \text{si le bac } j \text{ est utilisé} \\ 0 & \text{sinon} \end{cases}$

Fonction de Fitness

La fonction de fitness vise à minimiser le nombre total de bacs utilisés.

$$\text{Fitness} = \sum_{j=1}^{m} y_j$$

Contraintes

1. **Contrainte de capacité des bacs :**
   - Pour chaque bac, la somme des largeurs des objets placés dans ce bac ne doit pas dépasser la largeur maximale du bac.
   - Pour chaque bac, la somme des hauteurs des objets placés dans ce bac ne doit pas dépasser la hauteur maximale du bac.

$$\sum_{i=1}^{n} w_i x_{ij} \leq W \cdot y_j \quad \forall j \in B$$
$$\sum_{i=1}^{n} h_i x_{ij} \leq H \cdot y_j \quad \forall j \in B$$

2. **Contrainte d'affectation :**
   - Chaque objet doit être placé dans exactement un bac.

$$\sum_{j=1}^{m} x_{ij} = 1 \quad \forall i \in I$$

3. **Domaines des variables :**
   - Les variables x_{ij} et y_j sont binaires.

$$x_{ij} \in \{0,1\} \quad \forall i \in I, j \in B$$
$$y_j \in \{0,1\} \quad \forall j \in B$$

.

Objectifs

Les objectifs principaux du projet étaient :

1. Implémenter et comparer différents algorithmes de bin packing.
2. Mesurer la performance et l'efficacité de chaque algorithme.

3. Visualiser les solutions obtenues par chaque algorithme.

Algorithmes Implémentés

Quatre algorithmes ont été implémentés pour résoudre le problème de bin packing :

1. **First Fit**
2. **Next Fit**
3. **Tabu Search**
4. **Simulated Annealing**

First Fit

L'algorithme First Fit place chaque item dans le premier bin disponible où il peut entrer. S'il n'y a pas de bin où l'item peut entrer, un nouveau bin est créé.

Next Fit

L'algorithme Next Fit place chaque item dans le dernier bin utilisé. S'il ne peut pas entrer dans ce bin, un nouveau bin est créé.

Tabu Search

Tabu Search est un algorithme d'optimisation heuristique qui explore l'espace des solutions en utilisant une mémoire (la liste Tabu) pour éviter de revisiter les solutions précédemment explorées.

Simulated Annealing

Simulated Annealing est un autre algorithme d'optimisation heuristique qui explore l'espace des solutions en acceptant des solutions moins optimales avec une certaine probabilité, qui diminue au fil du temps.

Composants du Projet

Classes Principales

Reader

Cette classe est responsable de lire les données d'items et de bins à partir des fichiers de données. Elle utilise un patron de conception singleton pour s'assurer qu'une seule instance de la classe Reader est utilisée.

Bin

Cette classe représente un bin et contient des méthodes pour ajouter et supprimer des items, vérifier si un item peut entrer dans le bin, et réorganiser les items à l'intérieur du bin.

Item

Cette classe représente un item avec ses dimensions (largeur et longueur) et sa position dans le bin.

OneBinOneItem

Cette classe crée une solution initiale où chaque item est placé dans un bin séparé.

Fitness

Cette classe calcule la fitness d'une solution en fonction de l'espace inutilisé total dans les bins.

Neighborhood

Cette classe génère le voisinage d'une solution en déplaçant des items entre les bins.

Algorithmes de Bin Packing

- **FirstFit** et **NextFit** : Implémentent les algorithmes de First Fit et Next Fit respectivement.
- **TabuSearch** et **SimulatedAnnealing**: Implémentent les algorithmes de Tabu Search et Simulated Annealing respectivement.

Afficheur

Cette classe utilise Java Swing pour visualiser les solutions obtenues par les différents algorithmes. Elle permet également de zoomer et de dézoomer sur l'affichage des bins.

Résultats

Les algorithmes ont été testés sur un ensemble de données spécifiques (dataset numéro 13). Voici les résultats obtenus :

First Fit

- Nombre de bins : 5
- Temps d'exécution : 2ms

Next Fit

- Nombre de bins : 5
- Temps d'exécution : 0ms

Tabu Search

Cette algorithme a éte executé avec un liste de Tabu de taille 10 et de nombre d'itération de 1000

- Nombre de bins : 4
- Temps d'exécution : 680ms

Simulated Annealing

Cette algorithme a éte executé avec une température de 50, taux de refroidissement de 0.9, de nombre de fois de changement de température de 10 et un nombre d'itération pour chaque température de 100

- Nombre de bins : 4
- Temps d'exécution : 740ms

Conclusion

Les algorithmes heuristiques avancés (Tabu Search et Simulated Annealing) ont montré de meilleures performances en termes de nombre de bins utilisés par rapport aux algorithmes plus simples (First Fit et Next Fit), bien qu'ils nécessitent plus de temps de calcul. La visualisation des solutions a permis de mieux comprendre comment chaque algorithme place les items dans les bins. L'algorithme Simulated Annealing a montré des meilleures solutions que Tabu Search puisqu'il est probabiliste mais nécessite plus de temps pour l'éxecution.

Donc, on déduit que le meilleur algorithme pour la problème bin Packing est le Simulated Annealing.

Résultats pour tous les DataSet

DataSet 1

- First Fit: 5
- Next Fit: 6
- Tabu Search: 5
- Simulated Annealing : 5

DataSet 2

- First Fit: 8
- Next Fit: 11
- Tabu Search: 8
- Simulated Annealing : 7

DataSet 3

- First Fit: 11
- Next Fit: 15
- Tabu Search: 11
- Simulated Annealing : 11

DataSet 4

- First Fit: 21
- Next Fit: 25
- Tabu Search: 20
- Simulated Annealing : 18

DataSet 5

- First Fit: 4
- Next Fit: 5
- Tabu Search: 4
- Simulated Annealing :4

DataSet 6

- First Fit: 8
- Next Fit: 11
- Tabu Search: 9
- Simulated Annealing : 8

DataSet 7

- First Fit: 15
- Next Fit: 20
- Tabu Search: 14
- Simulated Annealing :13

DataSet 8

- First Fit: 20
- Next Fit: 24
- Tabu Search: 18
- Simulated Annealing :19

DataSet 9

- First Fit: 4
- Next Fit: 4
- Tabu Search: 4
- Simulated Annealing : 4

DataSet 10

- First Fit: 10
- Next Fit: 13
- Tabu Search: 9
- Simulated Annealing : 9

DataSet 11

- First Fit: 11
- Next Fit: 16
- Tabu Search: 11
- Simulated Annealing :11

DataSet 12

- First Fit: 20
- Next Fit: 28
- Tabu Search: 22
- Simulated Annealing : 20

Utilisation des Algorithmes de Bin Packing

Pour résoudre le problème de Bin Packing avec des algorithmes spécifiques, vous pouvez ouvrir le fichier BinPacking.java et suivre les instructions ci-dessous

Étape 1 : Chargement du Dataset

Pour charger un dataset spécifique, modifiez le paramètre de la fonction `getInstance(n)` où `n` est le numéro du dataset.

Étape 2 : Paramètres de l'Algorithme de Tabu Search

Pour utiliser l'algorithme de Tabu Search, vous devez spécifier deux paramètres dans la fonction TabuSearch(**tailleListeTabu**,**nombreIterations**) :

1. **tailleListeTabu** : La taille de la liste taboue.
2. **nombreIterations** : Le nombre d'itérations.

Étape 3 : Paramètres de l'Algorithme de Recuit Simulé (Simulated Annealing)

Pour utiliser l'algorithme de Recuit Simulé, vous devez spécifier quatre paramètres dans la fonction SimulatedAnnealing(**temperatureInitiale**, **tauxRefroidissement**, **nombreDeChangementsDeTemperature**, **nombreDIterationsParTemperature**) :

1. **temperatureInitiale** : La température initiale.
2. **tauxRefroidissement** : Le taux de refroidissement.
3. **nombreDeChangementsDeTemperature** : Le nombre de fois que la température change.
4. **nombreDIterationsParTemperature** : Le nombre d'itérations pour chaque température.

Étape 4 : Affichage d'une Solution dans l'Interface Graphique

Pour afficher une solution dans l'interface graphique, vous devez décommenter la ligne correspondante de l'affichage à la fin de fichier et choisir l'un des algorithmes comme paramètre dans `new Afficheur(...)`.

Un de ces variables peuvent être choisi comme paramètres : **binsFirstFit**, **binsNextFit**, **binsTabuSearch** et **binsSimulatedAnnealing**