الجمهورية الجزائرية الديمقراطية الشعبية

RÉPUBLIQUE ALGERIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère De L'enseignement Supérieur Et De La Recherche Scientifique

UNIVERSITE SAAD DAHLEB BLIDA 1

FACULTE DES SCIENCE

Département de Mathématiques

MEMOIRE DE FIN D'ETUDE

Présenté pour l'obtention du Diplôme de **MASTER**

Domaine : Mathématique et informatique

Filière : Mathématique

Spécialité : Recherche Opérationnelle

Problème de remplissage de Boite de 2D Bin panking

Réalisé par :

HOUARI khadidja

Devant le jury composé de :

Présidente :	Boudjemaa .R	PR	USD. Blida 1
Examinateur :	Boukhari .M	MAA	USD. Blida 1
Promotrice :	Djemia .N	MAA	USD. Blida 1

Année universitaire : 2022/2023

Remerciements

D’abord avant tous, je remercie Dieu qui a illuminé mon chemin et qui m'a armé de courage pour achever mes études.

Ma reconnaissance va plus particulièrement à :

Mon promotrice M^{me} Djemia pour sa disponibilité et son aide.

Mes remerciements vont également à tous ceux qui ont participé

de près ou de loin à la réalisation de ce travail.

khadidja

Dédicace

Je remercie avant tout Dieu tout puissant de m'avoir donnée la patience et le courage et de m'avoir facilite le chemin pour achever ce fruit de mes années d'études .Je dédie ce modeste travail, qui est le fruit de ma profonde reconnaissance :

Je dédie ce travail à mon cher père « **AISSA** », qui m'a élevé, m'a enseigné et m'a soutenu dans cette vie. C'est principalement grâce à lui que j'aime les mathématiques et leur étude. Il est mon professeur et mon modèle dans la vie, et mon rêve est de devenir un professeur comme mon père.

A la lumière de ma vie « **NORA »** la femme qui a souffert sans me laisser souffrir, qui na jamais dit non a ménage aucun effort pour me rendre heureux, ma mère est la source d'espoir et de bonheur dans ma via .Je te dédie ce travail en témoignage de mon profond amour. Puisse Dieu le tout puissant te préserver et t'accorder sante longue vie et bonheur.

A mon merveilleux frère : **Mohamed.**

A mes chers amis : **Yamina**, **Chaima**, **Wafa**.

Sans oublier les personnes qui mont aide a terminer ce travail : « **ADEL »** et « **AYOUB »** , Merci beaucoup

Khadidja

ملخص

يطمح هذا العمل الى اقتراح حل مشكل تعبئة الصندوق ثنائي الابعاد حيث يتعلق بتخزين الاشياء باقل عدد ممكن من الصناديق .

يمكن تطبيق هذا المشكل على عدد كبير من القطاعات الصناعية او تكنولوجيا المعلومات .

لحل مشكل تعبئة الصناديق باقل عدد ممكن نطبق الخوارزميات التالية مثل :

Next-Fit Decreasing Height , First-Fit Decreasing Height and Best-Fit Decreasing Height

Résumé

Ce travail vise à proposer une solution au problème de remplissage des boîtes bidimensionnelles, en ce qui concerne le stockage des objets dans le moins de boîtes possible.

Ce problème peut s'appliquer à un grand nombre de secteurs industriels ou informatiques.

Pour résoudre le problème du remplissage des Boites, il existe de nombreux algorithmes tels que : Next-Fit Decreasing Height, First-Fit Decreasing Best-Fit Decreasing Height, où nous avons discuté en détail deux algorithmes First-Fit Decreasing Height et Best-Fit Decreasing Height.

Abstract

This installation hopes to solve the problem of double-dimensional scanning and registration as it relates to storing and arriving with the least possible number of luggage. This problem can be applied to a large number of industrial history, technology or information. To solve the problem of searching for lost items with the smallest possible number of algorithms:

Next-Fit Decreasing Height , First-Fit Decreasing Height and Best-Fit Decreasing Height

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

La recherche opérationnelle(RO) est la discipline des mathématiques appliquées liée à l'informatique, qui traite des questions d'utilisation optimale des ressources dans l'industrie et dans le secteur public [8].

Elle est définie comme l'ensemble des méthodes et techniques rationnelles d'analyse et de synthèse des phénomènes d'organisation utilisables pour élaborer de meilleures décisions**,** tout en proposant des modèles conceptuels permettant d'analyser et de maitriser des situations complexes pour permettre aux décideurs de comprendre et d'évaluer les enjeux afin de faire les choix les plus efficaces [18].

L'application de la Recherche Opérationnelle s'est élargie dans cette dernière décennie à divers domaines comme l'économie, la finance, le marketing et la planification d'entreprise. Plus récemment, elle a été utilisée pour la gestion des systèmes de santé et d'éducation, pour la résolution de problèmes environnementaux et dans d'autres domaines d'intérêt public.

L'optimisation combinatoire occupe une place très importante en recherche Opérationnelle. Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation et d'autre part par de nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire[1][3].

Les problèmes d'optimisation sont utilisés pour modéliser de nombreux problèmes dans différents secteurs de l'industrie (télécommunications, électronique, mécanique, chimie, transport, …).

Dans le secteur industriel, la matière première est une composante très importante lors du calcul des coûts de production. Il est donc prioritaire de minimiser les pertes et d'améliorer sans cesse son utilisation dans l'intérêt de réduire les coûts, et pour aller plus loin réduire les impacts environnementaux engendrés par l'accumulation des déchets.

Le problème de placement, découpe, conditionnement, des variantes d'un problème d'intérêt majeur qui est le problème de Bin Packing . Dans l'industrie il est l'un des problèmes d'optimisation combinatoire, où il est difficile d'assurer une solution optimale dans les cas complexes. Il s'est avéré que l'utilisation des méthodes exactes est pratiquement très difficile à réaliser puisqu'elles nécessitent un temps d'exécution insupportable, l'utilisation des heuristiques approximatives s'avère donc très intéressante [6].

Ce problème consiste principalement à placer des articles en utilisant de la manière la plus économique possible des matériaux dans des boites. Les contraintes liées aux objets varient selon la dimension du problème. Il comporte trois versions selon la dimension : la première en une dimension (1D), la deuxième en deux dimensions (2D) et la troisième en trois dimensions (3D)[1][3].

Ces problèmes appartiennent à la catégorie des problèmes NP-difficile dans la classification de la complexité des problèmes.

Objectif

Modélisation et résolution d'un problème de placement bi dimensionnelle Par deux heuristique (NFDH et BFDH) sont bien défini qui s'adapte à ce type de problème et de le résoudre avec un logiciel Python puis de comparer les résultats obtenues.

Organisation du mémoire

Afin de bien présenter notre travail nous l'avons structuré de la manière suivante :

- **Chapitre 1 : Introduction à l'optimisation combinatoire**

L'expose des techniques d'optimisation capables de résoudre un certain nombre de problèmes. Deux grandes classes des méthodes sont présentées : les méthodes exactes consistent généralement à énumérer, de manière implicite, l'ensemble des solutions de l'espace de recherche et garantissent de trouver une solution optimale, et les méthodes approchées qui traitent

généralement des problèmes de grande taille, elles n'assurent pas de trouver la solution optimale mais sont efficaces. Les méthodes les plus connues et les plus utilisées vont être détaillées dans notre travail.

- **Chapitre 2 : Le problème de bin packing**

Ce chapitre décrit concrètement le problème de bin packing, les contraintes liées au problème, ainsi que quelques-unes de ses applications.

- **Chapitre 3 : Méthodes du résolution de problème de placement**

Dans ce chapitre, nous définissons le problème de placement, nous avons présenté des méthodes de résolution des problèmes de placement avec des exemples.

- **Chapitre 4 : Application numérique et résultat**

Afin de concrétiser les méthodes présentées dans le chapitre trois nous avons traité une application numérique avec logiciel python .

Enfin, nous achevons ce mémoire par une conclusion générale qui exposera les perspectives envisagées.

Chapitre I

Introduction à l'optimisation combinatoire

1. Introduction

L'optimisation est un outil important dans la prise de décisions et dans l'analyse des systèmes physiques .En termes mathématiques, un problème d'optimisation est le problème de trouver la meilleure solution parmi l'ensemble de toutes les solutions possibles.

Les problèmes d'optimisations combinatoires se répartissent en deux catégories : ceux qui sont résolus optimalement par des algorithmes efficaces et rapides et ceux dont la résolution optimale peut prendre un temps exponentiel [4].

Les notions de complexité des problèmes sont très importantes, car si un problème est identifié comme complexe, il sera difficile d'en spécifier un modèle, on pourra même perdre espoir de trouver un algorithme pour le résoudre. Dans ce cas, on se contentera d'exigences limitées : résolution approchée du problème posé [4].

2. Formulation mathématiques

En recherche opérationnelle, une modélisation d'un problème représente une phase très importante dans le processus de sa réalisation .Elle consiste en la conversion la plus fidèle possible d'un phénomène réel (industrielle, économique, pétrochimique, physique,…), généralement très complexe en un modèle mathématique .

Les problèmes d'optimisation combinatoire peuvent être formulés comme suit:

$$\begin{cases} \min f(x) \\ g_i(x) \leq 0 \qquad i = 1, \dots, m \\ h_j(x) = 0 \qquad j = 1, \dots, p \\ x \in S \subset \mathbb{R}^n \end{cases}$$

Où f est la fonction à minimiser, appelée fonction coût ou fonction objectif, x représente le vecteur des variables d'optimisation g_i sont les contraintes d'inégalité, h_j les contraintes d'égalité et S est l'espace des variables (appelé aussi espace de recherche).

Remarque :

Remarquer que le problème de minimisation d'un problème (p) : Min f(s) ramène facilement à un problème de maximisation telle que :

Max f(s)= -Min (-f(s)) $s \in S$

3. Notions de base sur l'optimisation combinatoire

- **Minimum locale :** s'il existe un voisinage N(s) tel que :

  $\forall$ s'$\in N(s)$, f(s)$\leq$ f(s').

- **Minimum global :** si $\forall$ s'$\in$ S, f(s)$\leq$ f (s').
- **Maximum locale :** s'il existe un voisinage N(s) tel que :

  $\forall$ s'$\in N(s)$, f(s)$\geq$ f(s').

- **Maximum global :** si $\forall$ s'$\in$ S, f(s) $\geq$ f (s').
- **Voisinage :** le voisinage est une fonction notée N qui associe un sous ensemble de S à toute solution s, les voisins de s sont s'$\in N(s)$
- **Les problèmes d'optimisation avec ou sans contrainte**

Il est important de bien distinguer les problèmes où des contraintes existent sur les variables de décision. Ces contraintes peuvent être simplement des bornes et aller jusqu'à un ensemble d'équations de type égalité ou inégalité. Il est parfois possible d'éliminer une contrainte égalité par substitution dans la fonction objective [21].

- **Problème de décision**

Un problème de décision est un problème dont la solution est formulée en termes oui/non.

4. Complexité des problèmes et leur classification

Dans la littérature (voir par exemple, [22], [13]) il existe plusieurs classes de complexité, les plus connues sont les suivantes [21] :

✓ **La classe P**

- ✓ **La classe NP**
- ✓ **La Classe NP-Complet**

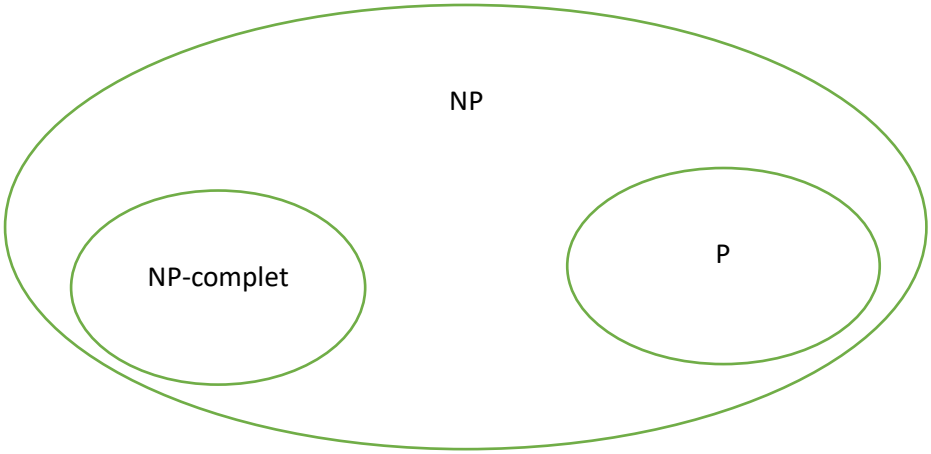


Figure 1:1.1.classification de complexité des problèmes

5. Méthodes de résolution

Les méthodes de résolution des problèmes d'optimisation combinatoire sont classées en deux catégories :

5.1. Les méthodes exactes

Les méthodes exactes ont permis de trouver des solutions optimales pour des problèmes de taille raisonnable .et le temps de calcul nécessaire pour trouver une solution risque d'augmenter exponentiellement avec la taille du problème.

Les algorithmes exacts sont utilisés pour trouver au moins une solution optimale d'un problème [16] :

-Méthode du simplexe.

- Méthode dual de simplexe.

- Méthode par séparation et évaluation progressive (branch and bound).

- **Méthode du simplexe**

La méthode de simplexe est un algorithme de recherche d'une solution optimale d'un programme linéaire donné. La mise en œuvre de la méthode du simplexe peut être divisée en trois étapes :

Première étape : Mettre le modèle sous forme standard en y introduisant des variables d'écart qui ont pour rôle de transformer les inégalités en égalités.

Deuxième étape : Etablir le premier tableau de simplexe (tableau à l'origine).

Troisième étape : Procéder une série d'itérations sur les tableaux de simplexe aboutissant à la solution optimale.

- **Méthode dual de Simplexe**

L'optimisation d'un programme linéaire à l'aide de la méthode de simplexe nous oblige parfois à introduire des variables artificielles pour obtenir une solution de départ lorsque les contraintes sont de type $Ax \leq b$ ($b \geq 0$).

- **Méthode par séparation et évaluation progressive (branch and bound)**

Consiste à énumérer ces solutions d'une manière intelligente en ce sens que en utilisant certaines propriétés du problème en question, cette technique arrive à éliminer des solutions partielles qui ne mènent pas à la solution que l'on recherche. De ce fait, on arrive souvent à obtenir la solution recherchée en des temps raisonnables. Bien entendu, dans le pire cas, on retombe toujours sur l'élimination explicite de toutes les solutions du problème.

Pour ce faire, cette méthode se dote d'une fonction qui permet de mettre une borne sur certaines solutions pour soit les exclure soit les maintenir comme des solutions potentielles. Bien entendu, La performance d'une méthode de branch and bound dépend entre autres, de la qualité de cette fonction (de sa capacité d'exclure des solutions partielles tôt.

5.2. Les méthodes approchées (heuristique et métaheuristique)

Une méthode approchée ou heuristique est une méthode d'optimisation qui a pour but de trouver une solution réalisable de la fonction objectif en un temps raisonnable, mais sans garantie d'optimalité. L'avantage

principal de ces méthodes est qu'elles peuvent s'appliquer à n'importe quelle classe de problèmes, faciles ou très difficiles.

Le mot métaheuristique est dérivé de la composition de deux mots grecs :

- heuristique.

- méta qui est un suffixe signifiant 'au -delà', 'dans un niveau supérieur'.

Les métaheuristiques se sont des méthodes inspirées de la nature, ce sont des heuristiques modernes dédiées à la résolution des problèmes. Les métaheuristiques qui se subdivisent en deux sous-classes : les méthodes de voisinage et les méthodes évolutives.

Les méthodes de voisinage

Ces méthodes partent d'une solution initiale (obtenue de façon exacte, ou par tirage aléatoire) et s'en éloignent progressivement, pour réaliser une trajectoire, un parcours progressif dans l'espace des solutions. Dans cette catégorie, se rangent :

- le recuit simulé.

- la méthode Tabou le terme de recherche locale est de plus en plus utilisée pour qualifier ces méthodes.

- **Le recuit simulé**

Le recuit simulé (simulated annealing) a été inventé par les physiciens KirKpatrick, Gellat et Vecchi en a1983 [16].Ils ont ainsi peut résoudre de maniéré quasi optimale des problèmes de voyageur de commerce a 5000 sommets, avec, il est vrai, des heures de calcule.

L'analogie historique s'inspire du recuit de métaux en métallurgie. Un métal refroidi trop vite présente de nombreux défauts microscopiques, c'est l'équivalent d'un minimum locale pour un problème combinatoire .Si on le refroidit lentement, les atomes se réarrangent, les défauts disparaissent, et le métal a alors une structure très ordonnée, équivalent du minimum global.

- **La recherche Tabou**

Les recherches taboues (Tabu ou Taboo Search) ont été inventées par Glover vers 1985 [5].

Elles sont de conception plus récente que le recuit, n'ont aucun caractère stochastique et Paraissent meilleure à temps d'exécution égale.

Elles sont caractérisées par trois points fondamentaux :

- A chaque itération en examine complètement le voisinage V(S) de la solution S', même si le cout remonte.

- On s'interdit de revenir sur une solution visitée dans un passée proche grâce à une liste Tabou (Tabou list) stockant de manière compacte la trajectoire parcourue.

On cherche donc S' dans V(S)-T.

- On conserve la meilleure solution trouvée encours de route car, contrairement au recuit, c'est rarement la dernière .On stoppe après un nombre maximal NMAX d'itération,

Ou après un nombre maximale d'itérations sans améliorer la meilleure solution trouvée, on quand V(S)-T = ∅

Ce dernier cas ne se produit que sur de très problèmes, pour lesquels le voisinage tout entrer peut se trouve enfermé dans T.

On voit qu'au cours de sa progression, une méthode taboue échappe aux minimaux locaux : même si S est un minimum local, l'heuristique va s'échappes de la régression V(S) en empruntant un col. Le schéma suivant, résume les méthodes qui permettent de déterminer la résolution d'un problème d'optimisation (Voir la **Figure 2**)

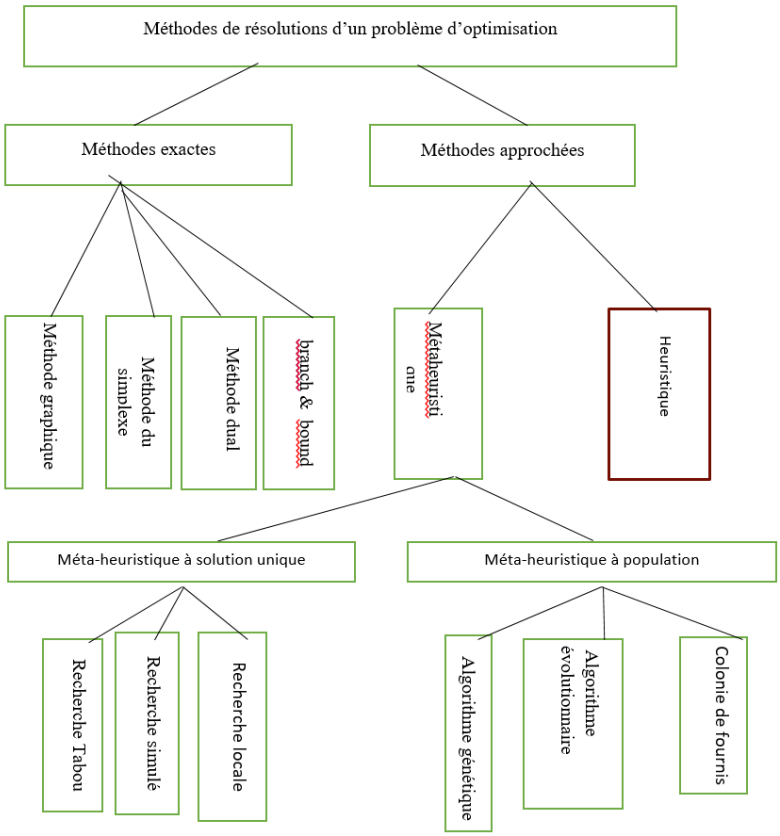


Figure 2:1.2.Quelques méthodes de résolution d'un problème d'optimisation

6. Quelques problèmes d'optimisation combinatoire

✓ Problème du sac à dos

Le problème du sac à dos (en anglais Knapsack problème), modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'un certain poids, avec tout ou une partie d'un ensemble donné d'objet ayant chacun un poids et une valeur. Les objets mis dans le sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum.

Formulation mathématique de Problème du sac à dos

$$\text{MaxZ} = \sum_{j=1}^{n} C_j X_j$$

$$\textbf{s.c} \begin{cases} \sum_{j=1}^{n} P_j X_j \leq P \\ X_j \in \{0,1\} \\ \forall j = 1, \dots, n \end{cases}$$

On connait pour chaque objet son poids Pj ainsi que l'utilité C_j que l'on peut en retirer .On connait aussi le poids maximum P du sac. L'objectif concerne la maximisation de l'utilité totale toute respectant la contrainte du poids totale limite.

Problème d'affectation

Soient n tâche (activité) à affecter à n machines de telle sorte que chaque machine soit affectée à une seule tâche et chaque tache soit affectée à une seule machine.

Le coût d'affecter la tâche j à la machine i est C_{ij}.

L'objectif est de minimiser la somme des coûts.

Si ω est l'ensemble de toutes les affectations possibles des n taches aux n machines, alors $|\omega|= n!$

Formulation mathématique de Problème d'affectation

$$\text{Min}Z = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij}X_{ij}$$

$$\sum_{j=1}^{n} X_{ij} = 1 \qquad i \in \{1,2,\dots,n\}$$

$$\sum_{i=1}^{n} X_{ij} = 1 \qquad j \in \{1,2,\dots,n\}$$

$$X_{ij} = \begin{cases} 1 & \text{si la tâche i est affectée à la machine i} \\ 0 & \text{sinon} \end{cases}$$

Problème de transport

Le problème du transport est un problème de minimisation de coût de transport. Il s'agit d'acheminer des produits de p points départ $D_1,\dots,D_p$ vers q points d'arrivées $A_1,\dots,A_q$.

Les données sont :

- La demande au point A_l est b_l (l =1,2,…,q)
- L'offre (disponibilité) au point D_K est a_k (k =1,2,…,p)
- Le coût unitaire de la route de D_k vers A_l est d_{kl}

L'objectif est de déterminer un schéma de transport optimal qui minimise le coût total du transport tout en satisfaisant la demande et respectant la disponibilité.

Formulation mathématique pour le problème de transport

Soit :

$t_{kl} \geq 0$ **:** Quantité transportée du point D_k vers le point A_t et toujours positive

ou k =1,2,…, p, l =1,2,…, q **.**

Contrainte sur la demande : $\sum_{k=1}^{p} d_{kl} \geq b_l$ (l =1,2,…,q)

Contrainte sur la disponibilité : $\sum_{l=1}^{q} t_{kl} \leq a_k$ (k =1,2,…, q)

$$\text{MinZ} = \sum_{k=1}^{p} \sum_{l=1}^{q} d_{kl} t_{kl}$$

$$\begin{cases} \sum_{l=1}^{q} t_{kl} \leq a_k & (k = 1,2,\dots,p) \\ \sum_{k=1}^{p} d_{kl} \geq b_l & (l = 1,2,\dots,q) \\ t_{kl} \geq 0 & (k = 1,2,\dots p \text{ et } l = 1,2,\dots,q) \end{cases}$$

Problème de conception de l'emploi du temps

Le problème de l'emploi du temps est un processus complexe, c'est un problème d'optimisation combinatoire très difficile à résoudre, car une solution ce type de problème est représentable par un ensemble de propriétés. Le but est d'obtenir la meilleure combinaison de cette propriété**.**

7. Conclusion

Nous avons présenté dans ce chapitre quelques méthodes de résolution des problèmes combinatoires. Ces méthodes sont généralement classées en deux catégories : les méthodes exactes et les méthodes approchées. Les méthodes exactes ont l'avantage de garantir l'obtention de la solution optimale. Cependant, elles présentent un inconvénient majeur qui celui du temps d'exécution important, car ces méthodes parcourent tout l'espace de recherche. Les méthodes approchées sont des algorithmes qui tendent à s'approcher de l'optimal en temps raisonnable. Ces algorithmes présentent l'avantage d'être rapides, mais ne garantissent pas l'obtention de la solution optimale. Pour finir nous avons donné quelques problèmes d'optimisation combinatoire.

Chapitre II

Problème de Bin Packing

1. Introduction

Le problème de bin Packing consiste d'une manière générale à trouver le rangement le plus économique possible pour un ensemble d'objets dans des boites dites « bin ». Ainsi, les problèmes de type bin packing consistent à placer des objets caractérisés par leur formes dans une ou plusieurs bins. Les variantes se distinguent selon la dimension, la connaissance à priori des objets, la forme des objets et des bins (carré, rectangulaire, circulaire), la possibilité de modifier l'orientation des objets. Elles se distinguent également par le problème de faisabilité, c'est-à-dire, savoir s'il existe un rangement réalisable des objets dans les bins, par exemple la minimisation du nombre de bin (bin packing) ou la minimisation des dimensions d'une seule bin (hauteur - strip packing, aire - rectangle packing, volume…), ou encore la maximisation de la valeur du rangement (problèmes de sac à dos – knapsack problème).

2. Présentation du problème de bin packing (BPP)

En recherche opérationnelle et en optimisation combinatoire, le bin packing est un problème algorithmique .Il s'agit de ranger des objets avec un nombre minimum des bins .Le problème classique se définit en une dimension, mais il existe de nombreuses variantes en deux ou trois dimension (Voir la **Figure 3**).

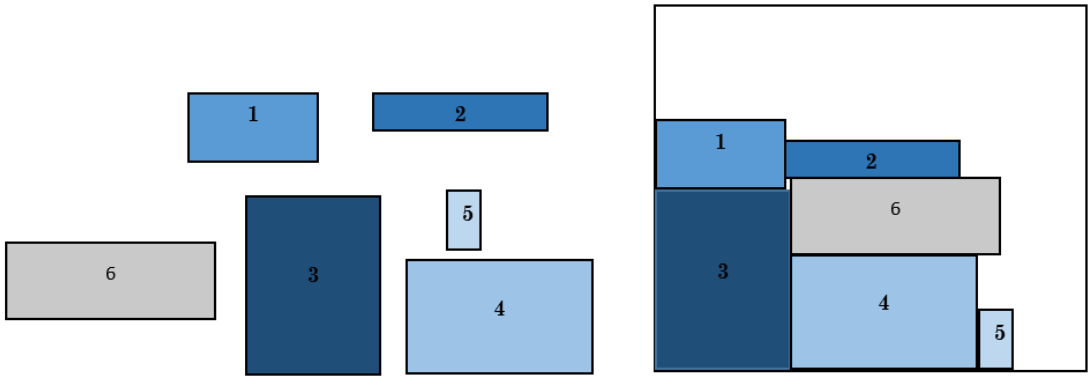


Figure 3:2.1.Exemple de Bin packing

3. Domaines d'application du problème de bin packing

- ✓ Rangement de fichiers sur un support informatique.
- ✓ Découpe de câbles.
- ✓ Remplissage de camions ou de conteneurs avec comme seule contrainte le poids ou le volume des articles.
- ✓ Placement de bin sur une palette.
- ✓ Placement dans un entrepôt.
- ✓ Rangement d'objets physiques dans des bins, un entrepôt, etc. bins, de palette.

4. Contraintes pratiques et classification

De nombreux problèmes pratiques se modélisent sous la forme d'un problème de bin packing.

Cependant, chaque problème réel présente ses propres spécificités telles que :

- ❖ **Les caractéristiques propres aux objets**
  - objets de formes homogènes ou non homogènes
  - objets de tailles uniformes ou différentes.
  - objets déformables ou non déformables,…etc.
- ❖ **Les spécificités propres au problème**
  - nombre de dimensions du problème.
  - disposer d'un seul bin (problème de minimisation ou problème de maximisation).
  - chercher à minimiser le nombre de bin à utiliser.
  - chercher à minimiser la surface ou le volume global des objets à placer,…etc.
- ❖ **Les contraintes propres au problème**
  - contraintes d'équilibre entre les objets.
  - contraintes d'orientation d'un objet.
  - contraintes de poids (par exemple, le poids d'un bin complet ne peut pas excéder une limite donnée).

- contraintes de placement, certains objets très lourds doivent être placés en bas, d'autres fragiles doivent être placés en dessus,…etc.

Dyckhoff et Finke [11] ont proposé une typologie qui permet d'organiser les problèmes de découpes et de placements en tenant compte de quatre caractéristiques principales :

- le nombre de dimensions du problème.
- le type de tâche : tous les objets et une sélection de bin, ou bien une sélection d'objets et tous les bins.
- les caractéristiques des bins : 1 seul bin, des bins de tailles identiques, ou bien des bins de tailles différentes.
- les caractéristiques des objets : objets identiques, peu d'objets de formes différentes, plusieurs objets de formes différentes ou bien des objets de formes relativement identiques.

Une typologie plus récente a été proposée par Wäscher et al [10]. Dans le but d'inclure les problématiques de découpe et de rangement, et d'établir une catégorisation complète de tous les problèmes connus dans le domaine. En ce qui concerne le problème de bin packing en deux dimensions (BPP-2D), les deux spécificités les plus rencontrées sont les suivantes :

- L'orientation : les objets peuvent être à orientation fixe (le cas orienté) ou bien ils peuvent être tournés de 90 degrés (le cas non orienté).
- La contrainte guillotine : Si elle est imposée, les Objets rangés peuvent être restitués par des coupes bout à bout parallèles aux dimensions de bins.

5. Le problème de bin packing uni-, bi- et tri-dimensionnel

Les problèmes de bin packing se distinguent, selon la dimension en :

5.1 Le problème de bin packing uni dimensionnel (BPP-1D)

Consiste à minimiser le nombre des containers uni dimensionnel (bins) nécessaire pour ranger une liste d'objets caractérisés par leurs longueur .ce problème est NP complet [15].une instance de BPP-1D, notée < I, w, W >, comprend un ensemble I= {1,2,…, n} de n objets et une fonction w qui associe à chaque objet i une valeur w_i qui Correspond à sa longueur, et W une valeur positive représentant la taille du bin.

Exemple

Placer un ensemble d'objetes de longueurs différentes dans le moins de bins possible de longueur fixée.

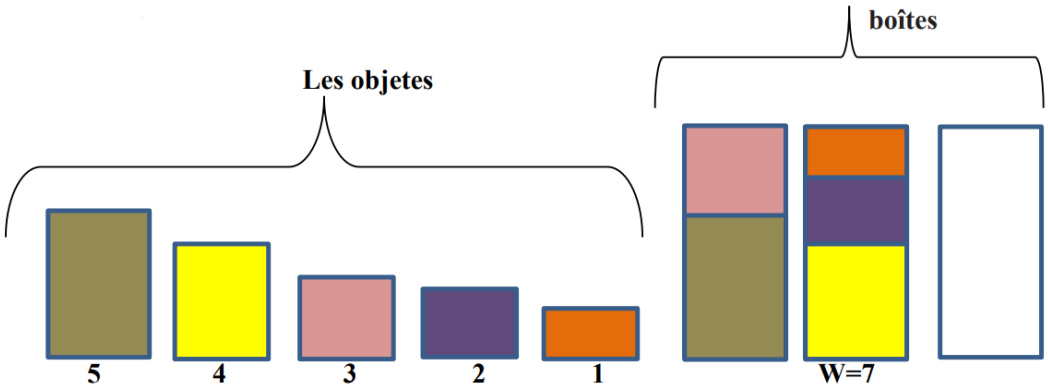


Figure 4:2.2.une instance de BPP-1D et une solution possible

Données :

N : ensemble de n objets avec w_i la longueur d'objet ($i \in \mathbb{N}$) et W longueur des bins.

Objectif : minimiser le nombre de bin utilisés.

Contraintes: ne pas dépasser la capacité d'un bin

5.2. Le problème de bin packing bi-dimensionnel (BPP-2D)

Est une généralisation naturelle de BPP-1D. Il s'agit de minimiser le nombre des bins identiques pour ranger une liste d'objet.

Les objets doivent être rangés de telle manière que les côtés des rectangles soient parallèles à ceux du bin. On note <I, w, h, W, H> une instance de BPP-2D.

I= {1,2,…, n} est la liste des objets à ranger, et une fonction w (Respectivement h) qui associe à chaque objet i une valeur w_i (respectivement h_i) qui correspond à sa longueur (respectivement sa largeur), W et H représentent la longueur et la largeur du bin **[16]**.

Exemple

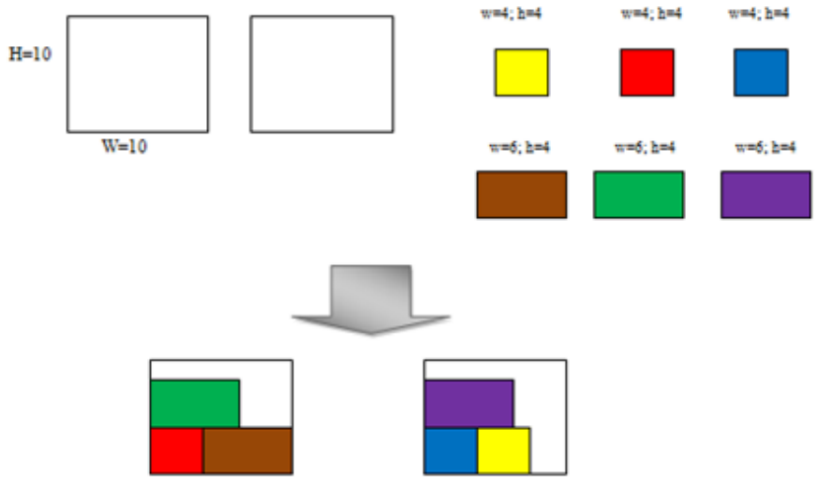


Figure 5:2.3.Une instance de BPP-2D et une solution possible

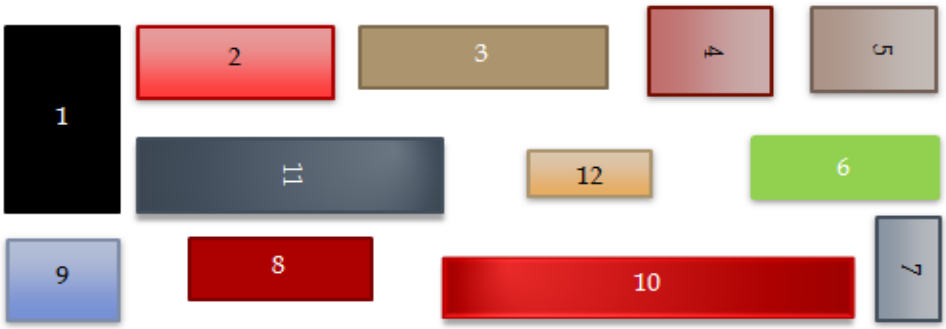
La différence entre BPP-1D et BPP-2D réside dans le problème de faisabilité car étant donné un ensemble d'objet et un bin, il s'agit de déterminer s'il existe un placement réalisable pour ces objets dans le bin. Le problème est trivial en une dimension (il suffit de sommer la longueur des objets) et NP-complet en deux dimensions.

Deux types de problèmes imbriqués sont considérés : un problème d'affectation et plusieurs problèmes de faisabilité. La question d'orientation des objets se pose pour le BPP- 2D.

Exemple

Soit l'instance BPP-2D suivante :

I= (A,B) ={ A={ a_1=(4,8), a_2=(5,4),a_3=(3,10), a_4= (3,5), a_5=(3,5), a_6=(5,3), a_7=(3,1), a_8 = (7, 2), a_9 = (2, 2), a_{10} = (11, 2), a_{11} = (2, 11), a_{12}= (3, 1) }, B = (12,10).} Telle que : A= (w, h) et B= (W, H).



Solution

1er cas : solution sans orientation

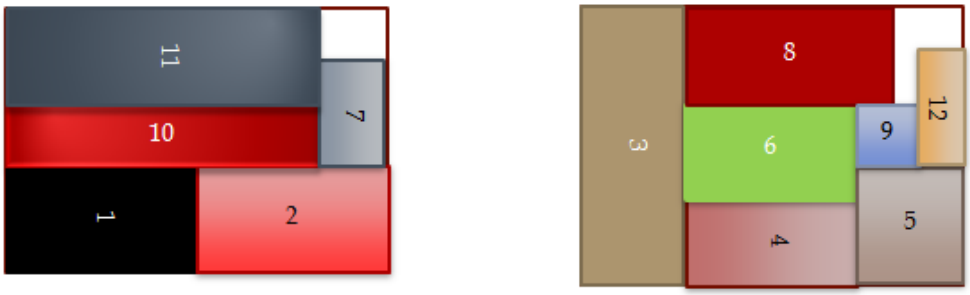


Figure 6:2.4.Bin packing le cas non orienté

2éme cas : solution avec orientation

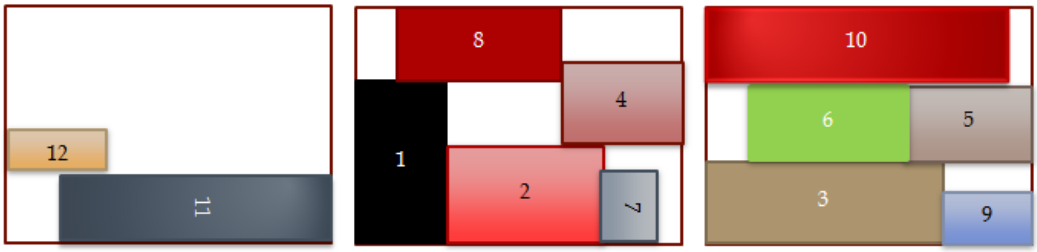


Figure 7:2.5.Bin packing le cas orienté

Un exemple de solution de cette instance dans le cas orienté est donné dans **la figure6**. **La figure 7** représente une solution pour la même

instance, mais en considérant cette fois ci le cas non orienté. Cette solution est optimale

5.3. Le problème de bin packing tri-dimensionnel (BPP-3D)

Le problème de bin packing tri- dimensionnel (BPP-3D) connu aussi sous le nom de problème de chargement de bin, est un problème combinatoire dont dérive de nombreuses applications industrielles [10]. Ce problème consiste à placer un ensemble S de n objets de dimensions (wi, hi, di), i=1,…, n, dans un nombre minimum de conteneurs identiques Rj, j=1,…, m, de dimensions (W, H, D) où W (resp.wi) est la largeur, H (resp.hi) la hauteur et D (resp.di) la profondeur des (resp des objets) tout en respectant certaines contraintes [2].

Exemple

Une instance est caractérisée par un ensemble de n objets (cylindre, petite parallélépipèdes,) et un ensemble de bin (grand parallélépipèdes).

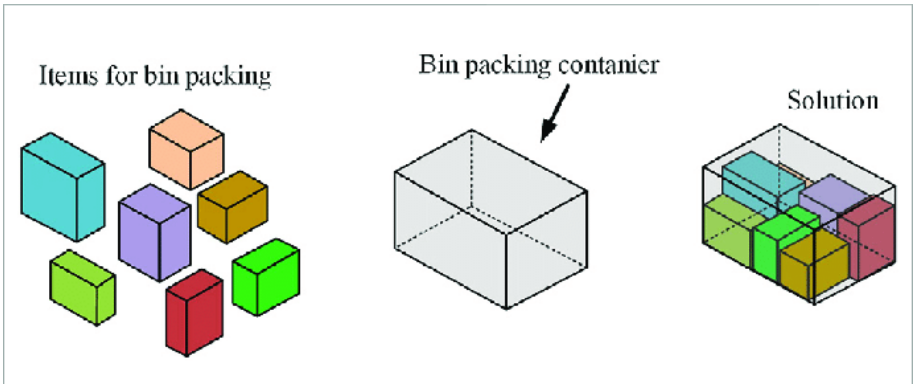


Figure 8:2.7.Illustration d'une solution réalisable pour BPP-3D

6. Formulations mathématiques

Dans cette section, nous présentons des formulations mathématiques pour le BPP-2D

6.1. Cas bi– dimensionnel

Le problème de bin packing bi-dimensionnel peut être modélisé de la manière suivante :

Les variable :

- un ensemble de m bins qu'on appelle bin.

  $B = B_1, B_2, \ldots, B_j, \ldots, B_m$
- un ensemble de n objets $Y = Y_1, Y_2, Y_i, \ldots, Y_n$.
- W : La larguer de bin B_j. (j=1,2,…m)
- H : la hauteur de bin B_j. (j=1,2,…m)
- w_i : La larguer de l'objet Y_i. (i=1,2,…n)
- h_i : La hauteur de l'objet Y_i. (i=1,2,…n)
- S_k : l'ensemble des objets Y_i qui sont dans le même niveaux k de bin B_j . (k=1,2,…t)
- h_k : la hauter de le premier objet y_i dans le niveaux k.
- $x_{ij} = \begin{cases} 1 & si\ l'objet\ Y_i\ est\ range\ dans\ le\ bin\ B_j \\ 0 & si\ l'objet\ Y_i\ ne\ peut\ etre\ range\ dans\ le\ bin\ B_j \end{cases}$
- $b_j = \begin{cases} 1 & si\ le\ bin\ B_j\ est\ utilisée \\ 0 & si\ le\ bin\ B_j\ n'est\ utilisée \end{cases}$

  On cherche à minimiser le nombre de bin à utiliser, c'est-à-dire :

$$\text{Min} \sum_{j=1}^{m} b_j$$

$$b_j = \{0,1\} \qquad j=1,..,\ m$$

Contraintes du problème:

1. Un Objet est placé uniquement dans un seule bin.

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = 1 \\ x_{ij} \epsilon \{0,1\} \\ i,j = 1 \ldots n \end{cases}$$

x_{ij}=1 impose à tous les objets d'être rangés dans un seul bin.

2. La larguer de l'ensemble des objets rangés dans le même niveaux dans une bin B_j ne doit pas dépasser la larguer de ceci.

$$\sum_{w_i \in s_k} w_i \leq W$$

3. La hauteur de l'ensemble des niveaux dans une bin B_j ne doit pas dépasser la hauteur de ceci.

$$\sum_{k=1}^{t} h_k \leq H$$

le modelé est donc le suivant :

$$\begin{cases} Min \sum_{j=1}^{m} b_j \\ \sum_{j=1}^{m} x_{ij} = 1 \\ \sum_{w_{i \in s_k}} w_i \leq W \\ \sum_{k=1}^{t} h_k \leq H \\ x_{ij} \in \{0,1\} \\ i = 1,2,\dots n\ ;\ \ j = 1,2,\dots m\ ; k = 1,2,\dots t \end{cases}$$

7. Conclusion

Un problème concret de bin packing se pose lorsque l'on cherche à remplir un ensemble fini d'objets.

Ce problème est très riche, se distinguent en : bin packing à une dimension, ou les objets sont caractérisés par une seule mesure (hauteur, le poids) bin packing à deux dimensions ou il faut ranger les objets sur un surface limitée et bin packing à trois dimensions ou les objets sont rangés dans un espace de volume fixé.

Dans ce chapitre, nous avons décrit les différents de BPP uni-bi et tri-dimensionnels, les formulations mathématiques avec les contraintes pratiques et classification. Pour finir nous avons donné quelques exemples d'application du bin packing.

Chapitre III

Méthodes de résolution du problème de placement

1. Introduction

Le problème de placement est un problème d'optimisation combinatoire d'intérêt majeur qui intervient dans des problématiques diverses. Le problème concret de placement se pose lorsque l'on cherche à remplir une ou plusieurs bins avec un ensemble fini d'objets ou lorsque l'on cherche à obtenir un ensemble fini d'objets en découpant un ou plusieurs objets de taille supérieurs et cela de la manière la plus économique possible. Dans le premier cas, il s'agit d'un problème de remplissage.

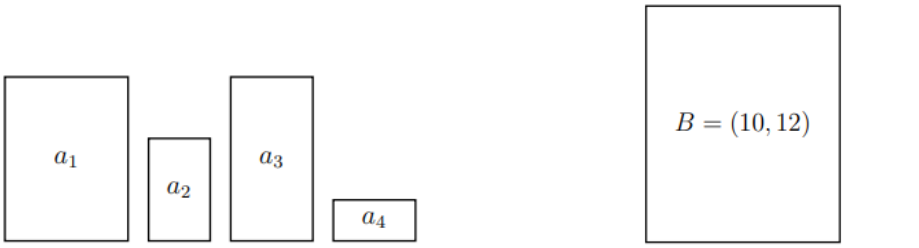
2. Définition du problème de placement

Le problème de placement est un problème d'optimisation dont l'objectif est de chercher à trouver un bon arrangement d'un ensemble d'objets dans des bins.

Figure9:3.1.Exemple du problème de placement

Exemple

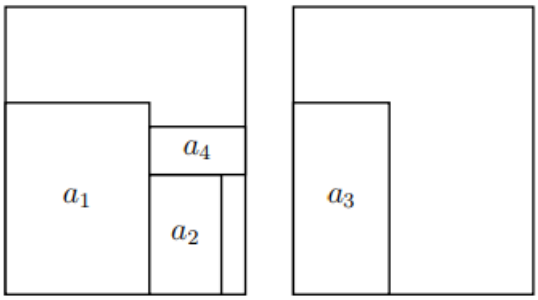
Un exemple du problème de placement on 2D telle que en à A l'ensemble des objets ,A = ({a_1= (6, 8), a_2= (3, 5), a_3= (4, 8), a_4= (4, 2), },B = (10, 12)). Ainsi pour cet exemple w_1 = 6, h_1 = 8, W = 10 et H= 12.



(a) l'ensemble des objet (b) l'ensemble des bins

Solution : On utilisant 2 bins

- premier bin : objets 1,2 et 4
- deuxième bin : objets 3



(b) Une solution pour le 2BP

3. Le problème de strip packing

3.1. Définition

Le problème de strip packing (placement sur bande SPP) en deux dimensions, est une variante du problème de bin packing qui considère un ensemble de rectangles a placer dans un strip (conteneur) de largeur donnée W (pour 'Width') et de hauteur H (pour 'Height') infinie.

Tous les rectangles doivent être stockes sans chevauchement, tels que leurs côtes soient parallèles aux bords du conteneur. La rotation des rectangles est interdite et la contrainte de découpe guillotine n'est pas imposée.

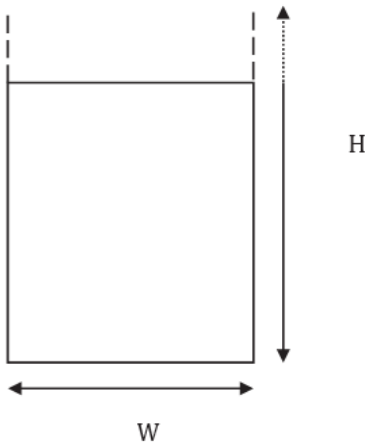


Figure:10.3.2. Principales caractéristiques du strip packing

L'objectif est de ranger tous les rectangles dans le bin en minimisant la hauteur totale a utiliser.

4. Quelques exemples d'application du problème de placement

Le problème de bin packing peut s'appliquer a un grand nombre de secteurs industriels ou informatique

Pour la version classique en deux dimension :

- Industrie manufacturière : Les usines et les ateliers de production peuvent utiliser des algorithmes de bin packing 2D pour planifier et optimiser la découpe de matériaux tels que le bois, le verre ou le métal. Cela permet d'utiliser efficacement les matériaux bruts et de minimiser les déchets.
- Coupe de tissu pour l'industrie textile : Les fabricants de vêtements et d'autres produits textiles peuvent utiliser des algorithmes de bin packing 2D pour découper des motifs de tissu de manière optimale, minimisant ainsi les pertes de matière première et maximisant le rendement de production.
- Aménagement d'espaces de bureau : Les architectes d'intérieur et les gestionnaires d'espaces de bureau peuvent utiliser des algorithmes de bin packing 2D pour optimiser l'agencement des bureaux, des salles de réunion et des espaces communs. Cela

permet de maximiser l'utilisation de l'espace et de créer un environnement de travail efficace

5. Classification des problèmes de placement

Il y a trois grandes familles de problèmes selon le nombre de dimensions des objets :

5.1. Problème de placement en un dimension

Le placement à une dimension est la version standard des problèmes de placement, il consiste à rangé un ensemble d'objets caractérisés par une seule variable soit (la hauteur, la longueur, la largeur, le poids, la taille ou autre) dans un ensemble des bins de taille fixé W dans une instance de placement une dimension 1D notée D avec l'ensemble des objets Y où Y = $\{Y_1,..., Y_n\}$

Chaque objet Y_i possède une longueur w_i inferieur à W.

La valeur optimale du nombre des bins nécessaires pour ranger tous les objets d'une instance **D** est notée OPT(**D**).

5.2 Problème de placement en deux dimensions

Plus formellement, le problème de placement en deux dimensions est défini de la façon suivante : étant donné un ensemble de ***n*** objets Y = {Y1,..., Yn} et un nombre illimité de bins de dimensions plus larges que celles des objets, les (wi, hi) les dimensions d'un objet et (W, H) les dimensions de bin [14].

Nous considérons une instance de placement 2D notée **I.**

.La valeur optimale du nombre de bins nécessaires pour ranger tous les objets d'une instance **I** est notée OPT(I).

Ces problèmes se posent en particulier chez les fabricants de verre, de tôles, et chez les fabricants de vêtements.

5.3 Problème de placement à trois dimensions :

Dans ce cas les données sont les suivantes :

Il existe N objets de volume $v_i$et M bins d'un volume V_j(éventuellement des bins toute identiques de volume **V**) Le volume total des objets est plus petit que le volume total des bins ($\sum v_i \leq \sum V_j$ **)** [18].

Figure 8:3.3.Le placement en 3D

6. Méthodes de résolution pour le problème de placement

Comme la plupart des problèmes d'optimisation combinatoire, le problème de placement étant NP-difficile, l'énumération de toutes les solutions réalisables pour trouver la meilleure solution s'avère impossible avec les méthodes exactes même pour un problème de taille moyen. Toutefois, on peut aborder la résolution par des méthodes approchées, en particulier des heuristiques et des méta-heuristiques. Dans cette partie on expose quelques méthodes exactes et approchées existent.

6.1 Méthodes exactes

Le problème de placement a été largement étudié dans la communauté de recherche opérationnelle. Il existe plusieurs méthodes exactes pour le résoudre comme la procédure par séparation et évaluation (PSE) et la méthode de génération des colonnes.

A titre d'illustration nous allons présenter ici la méthode de branch and bound pour le problème de placement [19].

6.1.1 Procédure par séparation et évaluation

La procédure par séparation et évaluation (PSE), ou en anglais branch and bound, est un algorithme qui permet d'énumérer intelligemment toutes les solutions possibles. En pratique, seul les solutions potentiellement de bonne qualité seront énumérées, les solutions qui ne

peuvent pas conduire à améliorer la solution courante ne sont pas explorées.

Pour présenter une PSE, nous utilisons un « Arbre de recherche » constitué :

- Des nœuds ou sommets, ou un nœud représente une étape de construction de la Solution.
- d'arcs pour indiquer certains choix faits pour construire la solution.

Dans notre exemple, les nœuds représentent une étape pour laquelle des objets auront été mis dans les bins et d'autres pour lesquelles aucune décision n'aura encore été prise.

Chaque arc indique l'action de mettre un objet dans le bin courante ou au contraire de ne pas le mettre dans le bin. Les nœuds sont étiquetés par une liste d'ensemble chacun correspond à un bin de plus le dernier ensemble de la liste correspond à bin que l'on est en train de remplir. La figure suivante représente l'arbre de recherche du problème donné [19].

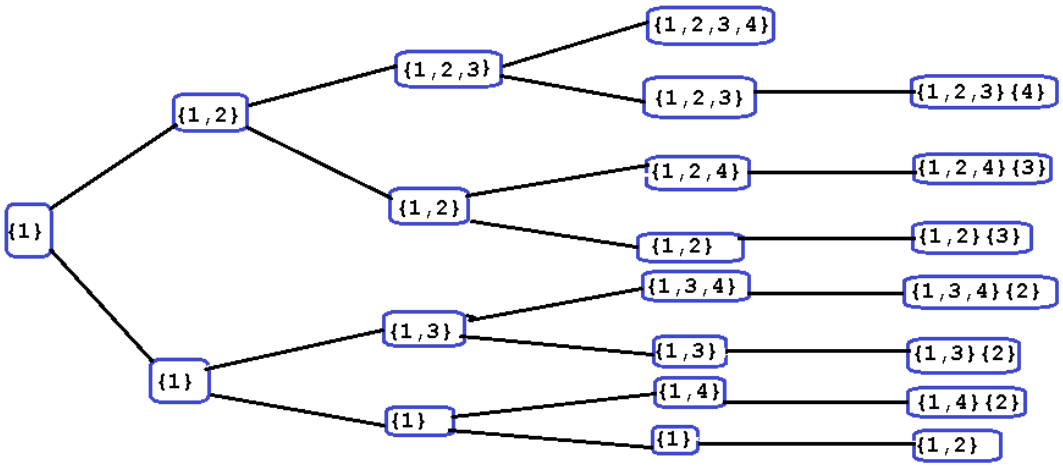


Figure 9:3.4.l'exploration des solutions dans un arbre de recherche.

Au départ, on cherche à remplir le premier bin. Tout d'abord on lui affect l'objet n°1 :

Comme il faudra bien que cet objet soit contenu dans un bin, on décide en quelque sorte de nommer 1 le bin contenant. La racine de l'arbre, dont la profondeur est par définition 0, contiendra donc l'ensemble {1} qui

représente le contenu du 1er bin. Puis pour un nœud de profondeur i on construit deux fils : celui du haut ou l'on ajoute l'objet i+1 dans la boîte et celui du bas ou la boîte reste tel quel. Lorsque l'on a considéré tous les objets pour un boîte donné, on poursuit la construction de l'arbre en passant au boîte suivant et en lui affectant initialement l'objet non sélectionné dont l'indice est le plus petit on poursuit ainsi la construction de l'arbre jusqu'à avoir rangé tous les objets. Dans l'arbre de recherche achevé, chaque feuille représente une solution potentielle mais forcement réalisable. Dans le schéma, les feuilles au bord épais représentent les propositions irréalisables car supérieures au poids maximal à ne pas dépasser.

Pour déterminer la solution, il suffit de calculer le nombre de bins pour chaque nœud feuille acceptable et de prendre la solution ayant la plus petite valeur.

- Cependant la taille de l'arbre de recherche est exponentielle en le nombre d'objets, et il n'est pas question d'implanter un tel arbre en mémoire. Aussi il existe de nombreuses techniques algorithmiques de parcours de ce type d'arbre. Ces techniques ont pour but d'augmenter la rapidité du calcul en diminuant la taille de l'arbre de recherche. Par exemple on peut remarquer que le poids du nœud interne {1,2,3} dépasse déjà le poids maximal, il n'était donc pas nécessaire de développer l'étape suivante. Les PSE permettent d'élaguer (éliminer les ensembles de solutions pour lesquels il n'existe pas d'optimum) encore plus cet arbre en utilisant des bornes inferieures et supérieures de la fonction objectif.
- Une borne inferieure est une valeur minimum de la fonction objective. Autrement dit c'est une valeur qui est nécessairement inferieur à la valeur de la meilleure [19].

6.2 Heuristiques de résolution pour le problème de Placement on 2 dimension

Le problème de bin packing en deux dimensions BPP-2D est un problème classique pour lequel plusieurs méthodes approchées

ont été proposées, ces méthodes sont en général des heuristiques ont été proposées dans le but de trouver des solutions de bonne qualité dans un temps raisonnable.

Dans cette section, nous présentons quelques méthodes heuristiques de résolution récentes proposées pour le 2BP qui utilisent des critères généralisés du 1BP. Ces heuristiques peuvent être divisées en deux principales familles :

- Les algorithmes en une phase
- Les algorithmes en deux phases

6.2.1 Les algorithmes en une phase :

Les algorithmes en une phase consistent à ranger directement les objets dans des bins. Deux règles à définir : l'ordre dans lequel les articles sont examinés, le bin et la position dans laquelle on cherche à le placer. Parmi les algorithmes qui procèdent en une phase :

L'algorithme Finite-First-Fit (FFF):

Cet algorithme consiste à trier les objets par ordre décroissant par rapport à leurs hauteurs. L'objet en cours est placé dans le niveau le plus bas du premier bin qui peut le contenir. Si aucun niveau ne peut contenir l'objet en cours un nouveau niveau est créé dans le premier bin approprié ou bien en initialisant un nouveau bin. [12]

Parmi les algorithmes qui ne rangent pas les objets par niveau on trouve **la stratégie Bottom-Left (BL)** qui consiste à placer un objet dans la position la plus en bas à gauche. La procédure est réitérée pour chaque objet dans un ordre donné. On place l'objet en cours dans la 1ère position la plus basse puis la plus à gauche possible pouvant le contenir, si on ne peut trouver cette position, dans ce cas on ajoute un nouveau bin.

6.2.2.Les algorithmes en deux phases :

Les algorithmes en deux phases débutent d'abord par ranger les objets dans un bin de hauteur infinie en cherchant une solution pour le problème de strip-packing (2SP). En une deuxième phase, un algorithme de résolution pour le problème BPP-1D consiste à utiliser la solution du 2SP obtenue pour construire le rangement dans les bins à utiliser effectivement (solution pour le 2BP.

- **Les algorithmes existants pour les problèmes de strip packing on 2D**

Les Méthodes en deux phases fonctionnent de la manière suivante: la première étape (strip packing) consiste à trier les objets suivant leurs hauteurs décroissantes (Decreasing Height) et les placer successivement dans un bin de hauteur infinie, en le remplissant couche par couche, formant ainsi des niveaux définis par la hauteur du plus grand objet par couche. Le premier objet, le plus long, est placé dans le premier niveau qui correspond à l'arête inférieure du bin. Les autres objets sont rangés suivant une stratégie NF, FF ou BF par rapport à la largeur du bin, on parle respectivement d'une stratégie NFDH, FFDH et BFDH.

- **Algorithme NFDH (Next-Fit Decreasing Height)**

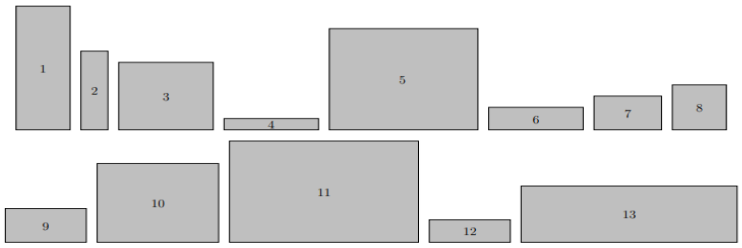
Dans l'algorithme NFDH nous suivrons les étapes suivantes :

- les rectangles sont triés selon leur hauteur de façon décroissante.
- Le premier rectangle (le plus haut) est déposé en bas à gauche ce qui détermine la hauteur de l'étage (définitivement fixé).
- Puis un autre rectangle est déposé à côté de celui-ci (qui est le plus haut rectangle parmi le reste).

Et l'on continue jusqu'à ce que le rectangle courant ne rentre plus par sa largeur. Dans ce cas, ce rectangle est déposé au-dessus à gauche de l'étagère en créant un nouvel étage. [7]Cette opération est réitérée jusqu'à ne plus avoir de rectangle.

Exemple :

Algorithme NFDH est appliquée à une série d'objets numérotée de 1à 13.



Objets	1	2	3	4	5	6	7	8	9	10	11	12	13
Hauteur	11	7	6	1	9	2	3	4	3	7	9	2	5
Largeur	4	2	7	7	11	7	5	4	6	9	14	6	16

Tableau 1:3.1.Les tailles des objets

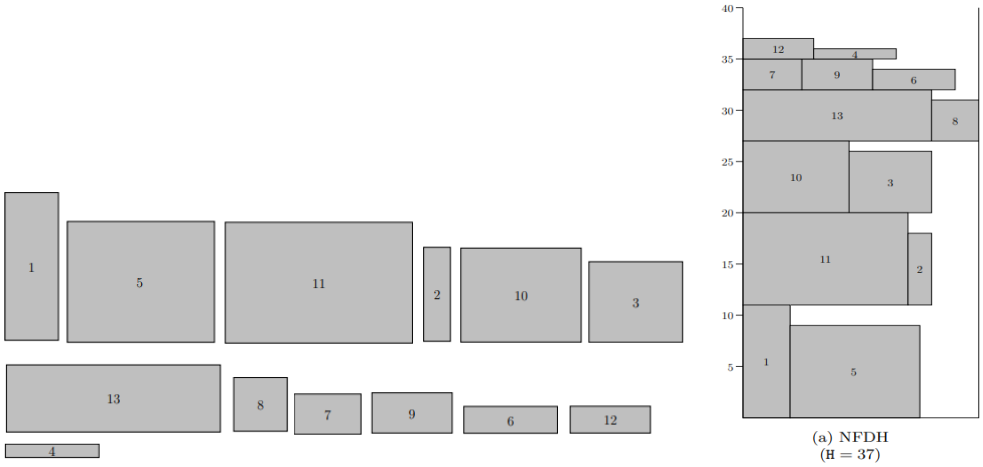


Figure13:3.5.Exemple avec solution en Algorithme de NFDH

- **Algorithme FFDH (First-Fit Decreasing Height)**

Le principe de FFDH est très similaire à celui de NFDH. Cependant avant de placer le prochain rectangle sur le niveau courant, il est nécessaire de

- vérifier la possibilité de le placer dans le niveau inférieur (le plus bas possible), ayant suffisamment d'espace pour l'accueillir[7].
- Dans FFDH, il est donc possible de revisiter un niveau inférieur, ce qui n'est pas permis dans NFDH

Exemple :

Algorithme FFDH est appliquée à une série d'objets numérotée de 1à 13.

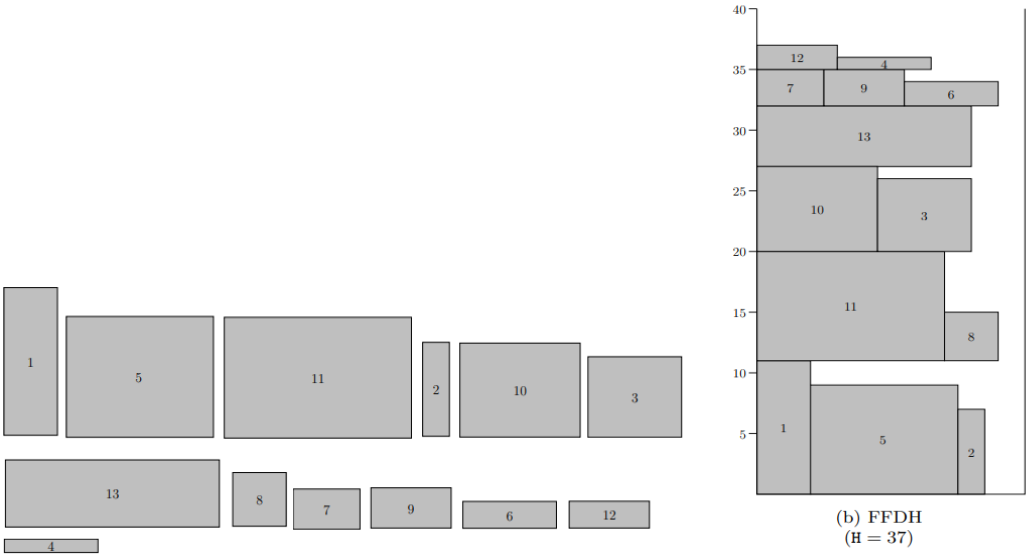


Figure14:3.6.Exemple avec solution en Algorithme de FFDH

- **Algorithme BFDH (Best-Fit Decreasing Height)**

L'algorithme BFDH agit de la même façon que FFDH, sauf que

- la recherche d'un emplacement du rectangle courant se fait de façon exhaustive sur les étagères.[20]
- Ainsi l'espace horizontal perdu par le placement d'un rectangle est calculé pour toutes les étagères.
- Le rectangle est effectivement placé dans l'étagère pour laquelle cet espace perdu est minimal.

Cet algorithme est donc logiquement plus lent que FFDH qui place les rectangles dans la première étagère trouvée.

Exemple :

Algorithme BFDH est appliquée à une serie d'objets numérotée de 1à 13.

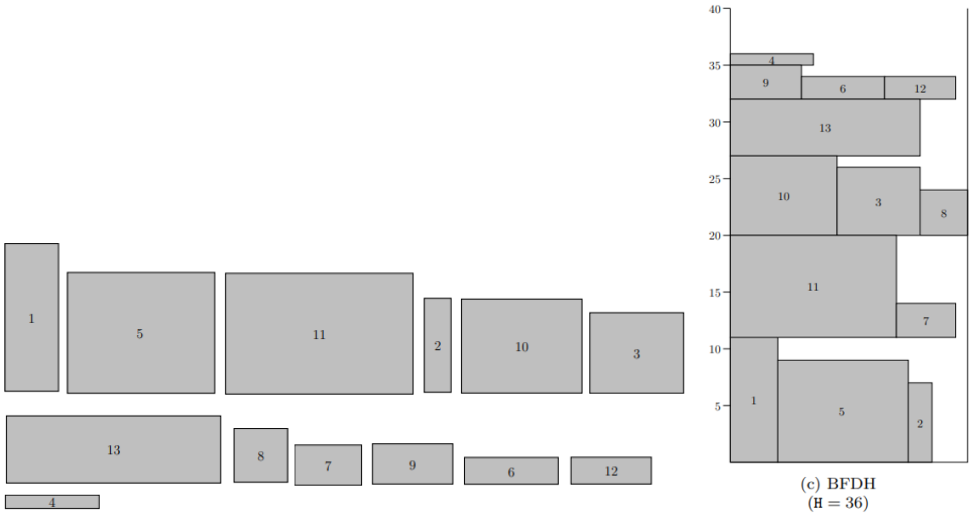


Figure15:3.7.Exemple avec solution en Algorithme de BFDH

7. Algorithme de Next Fit Dressing Height (NFDH)[24]

Inisalisation

1. lecture le nombre des objets

2.Inisialisation d'un vecteur contenant des objets de taille n elements

3. Création des objets lire w largeur h hauteur pour chaque object

4. Lire la capacité du bin (conteneur)

5.Filtrer les objects dont la taille est supérieure a la capadcitè du bin

6.Trier les objets dans l'ordre décroissant en fonction de leur hauteur

7. Initialisation d'un liste vide de bin

8. Le counts reprèsente la taille des objets ,ou i est l'indice des objets

9. Boucle pour itérer à travers tous les objects

10. Effectuer un traitement si un bin n'existe pas alors ajoutez le, si le bin existe alors ajoutez un niveau à un bin ,si un niveau n'existe pas crèez le.

12. Initialiser une variable packed pour indiquer si l'objet à été place dans le bin

14. Initialiser une variable can_packed pour determiner si l'objet peut etre place dans le bin et les niveaux

15. Si nous pouvons pas placer l'objet alors ajouter un Nouveau bin

16. l'affichage des résultats

8. Algorithm de Best-Fit Decreasing Height (BFDH) [25]

1. Boucle pour itérer à travers tous les objects
2. Effectuer un traitement si un bin n'existe pas alors ajoutez le, si le bin existe alors ajoutez un niveau à un bin ,si un niveau n'existe pas crèez le.
3. Placer toute les niveaux dans une liste de niveaux
4. Trier les niveaux par ordre croissant de la taille restante de la largeur
5.Initialiser une variable packed pour savoir si l'objet a été placé dans le bin
6.placer l' object si c'est possible
7.Initialiser une variable can_packed pour déterminer si l'objet peut être place dans le bin est les niveaux
8.Si nous ne pouvons pas placer l'objet alors ajouter un Nouveau bin
9.l'affichage des résultats

8. Conclusion

Dans ce chapitre nous avons présenté les méthodes de résolution d'un problème de placement.

En premier lieu on a parlé des méthodes de résolution exactes, comme la méthode de branch and bound pour le problème de placement, puis nous avons passé des heuristiques de résolution d'un problème bidimensionnel et leurs explications (First-fit Dressing Height ; Next-fit Dressing Height ;Best-fit Dressing Height), avec des exemples d'application.

Chapitre IV
Application numérique et résultats

1. Introduction

ce chapitre est consacré à l'implémentation des méthodes pour obtenir les résultats en utilisant le language Python soit par la programmation, soit l'utilisation de fonctions prédéfinis.

2.Python

2.1.Python

Le langage Python est un langage de programmation de haut niveau, facile à apprendre, orienté objet, totalement libre et terriblement efficace. Il est conçu pour produire du code de qualité, portable et facile à intégrer. Ainsi la conception d'un programme Python est très rapide et offre au développeur une bonne productivité. En tant que langage dynamique, il est très souple d'utilisation et constitue un complément idéal à des langages compilés. Contrairement à des langages spécifiques comme PHP qui se focalise sur un domaine précis, Python est universel. Il peut être utilisé dans un grand nombre de contextes. Un autre avantage de Python est la richesse de ses bibliothèques. C'est toutes ces nombreuses et importantes caractéristiques qui font la force de ce langage. Contrairement à certains langages comme Java, Python est facile à manipuler et peut s'apprendre sans formation. Il est aussi portable et plus général que Java ; il est utilisé dans une grande variété de domaines. Il possède aussi des bibliothèques beaucoup plus riches que celle de Java.

2.2 Flask

Flask est un framework d'application Web WSGI léger. Il est conçu pour permettre une mise en route rapide et facile, avec la possibilité d'évoluer vers des applications complexes. Il a commencé comme un simple emballage autour de Werkzeug et Jinja, et est devenu l'un des cadres d'applications Web les plus populaires en Python.

Flask propose des suggestions, mais n'impose aucune dépendance ni structure de projet spécifique. C'est au développeur de choisir les outils et bibliothèques qu'il souhaite utiliser. Il existe de nombreuses extensions fournies par la communauté qui permettent d'ajouter facilement de nouvelles fonctionnalité

Pour en savoir plus sur Flask, vous pouvez consulter le site officiel : [https://palletsprojects.com/p/flask/]

2.3.Turtle

Une tortue graphique est une manière bien connue et intuitive pour initier les enfants au monde de la programmation. Un tel module faisait partie initialement du langage de programmation Logo créé par Wally Feurzig et Seymout Papert en 1967.

Imaginez un robot sous forme de tortue partant au centre (0, 0) d'un plan cartésien x-y. Après un import turtle, exécutez la commande turtle.forward(15) et la tortue se déplace (sur l'écran) de 15 pixels en face d'elle, en dessinant une ligne. Turtle star La tortue permet de dessiner des formes complexes en utilisant un programme qui répète des actions élémentaires.

On peut donc facilement construire des formes et images à partir de commandes simples.

Turtle permet d'utiliser des primitives graphiques en utilisant un style de programmation orienté objet ou procédural. Du fait qu'il utilise la bibliothèque graphique tkinter, Turtle a besoin d'une version de python implémentant Tk.

3. Principe d'utilisation

Pour la détermination les paramètres suivants :

1. b_j : l'ensemble des bins
2. W : la largeur de bin.
3. H : la hauteur de bin .
4. w_i : La largeur des objets .
5. h_i : la hauteur des objets.

Algorithme NFDH La procédure de classements et placements

Entrée : W , H ,$w_{i,,}h_i$

Sorties : - L'ordre décroissante des objets.

- le nombre de bin

- tableau de classification des objets dans chaque bin

- la représentation graphique des résultats.

Algorithme BFDH La procédure de classements et placements

Entrée : W , H, w_i , h_i.

Sorties :- L'ordre décroissante des objets

- Le nombre de bin

- Tableau de classification des objets dans chaque bin

- la représentation graphique des résultats.

4. Implémentation et comparaison

4.1. Exemple d'application 1:

On donne un exemple de problème de bin Paking avec la larguer de bin est 500 et la hauteur de bin et 1000 qui représente les variables Y $_i$ et le nombre d'objets est 18 qui représente le nombre de contraintes alors le problème de bin Paking peut être formulé de la manière suivante :

$$\begin{cases} Min \sum_{j=1}^{m} b_j \\ \sum_{j=1}^{m} x_{ij} = 1 \\ \sum_{w_{i\in s_k}} w_i \leq W \\ \sum_{hi\in_{s_k}} h_i \leq H \\ x_{ij} \in \{0,1\} \\ i = 1,2,\dots n\ ;\ j = 1,2,\dots m\ ; k = 1,2,\dots t \end{cases}$$

La taille des objets donnés par le tableau suivant :

Objet n° :	1	2	3	4	5	6	7	8	9	10
Larguer	456	456	499	485	123	452	223	226	250	450
Hauteur	300	500	825	560	800	468	300	226	550	650

Objet n° :	11	12	13	14	15	16	17	18
Larguer	469	458	496	459	478	456	485	496
Hauteur	769	858	696	759	500	900	154	523

Tableau 2:4.1.Les tailles des objets pour l'application 1

On exécutera notre programme sur micro-ordinateur PC doté d'un système d'exploitation Windows dix 64 bits ou l'environnement PYTHON fonctionne sans problème.

Résolution du problème

La complexité de ce problème malgré toutes les avancées dans la théorie des mathématiques, on n'arrive pas à trouver une méthode exacte donnant la solution optimale.

Néanmoins plusieurs heuristiques ont été développées donnant des résultats assez Satisfaisants du point de vue temps.

Parmi ces heuristiques nous allons développer heuristique de **Nest-Fit Dressing Height** et **Best-Fit Decreassing Height.**

4.1.1. Résolution du problème en appliquant l'heuristique de Next-Fit Decreasing Height :

La figure suivante donne l'écran d'exécution de ce programme elle montre la création du fichier objet.

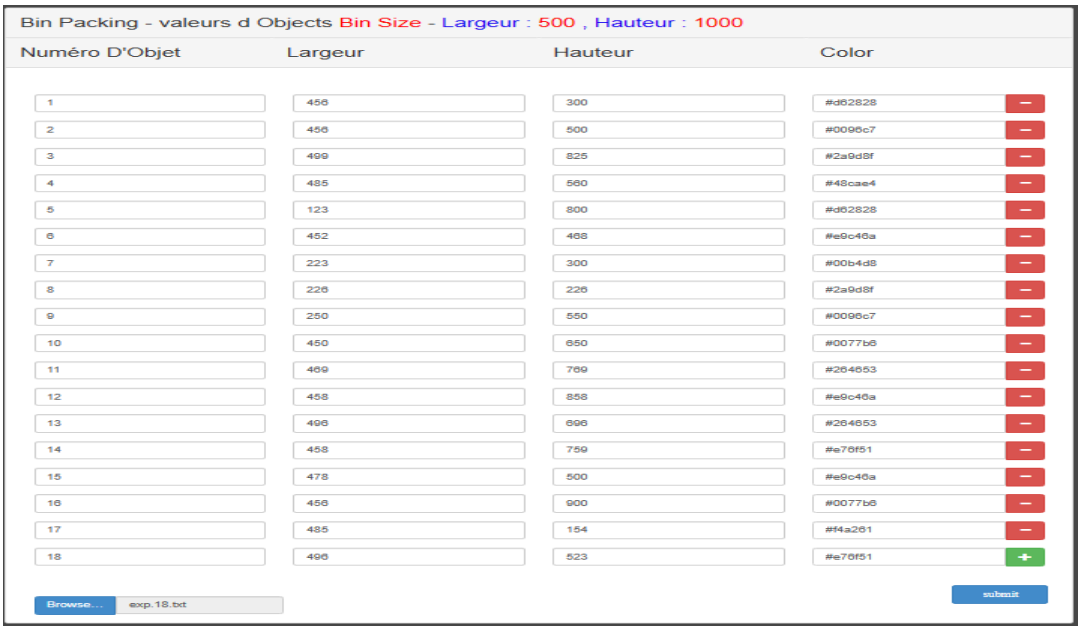


Figure 10:4.1.Création d'objets pour La méthode NFDH

La figure suivant donne Les résultats obtenus avec NFDH :

-l'ordre des objets (ordre décroissante).

-La classification des objets dans les bins .

-Le nombre de bins obtenus par la méthode NFDH.

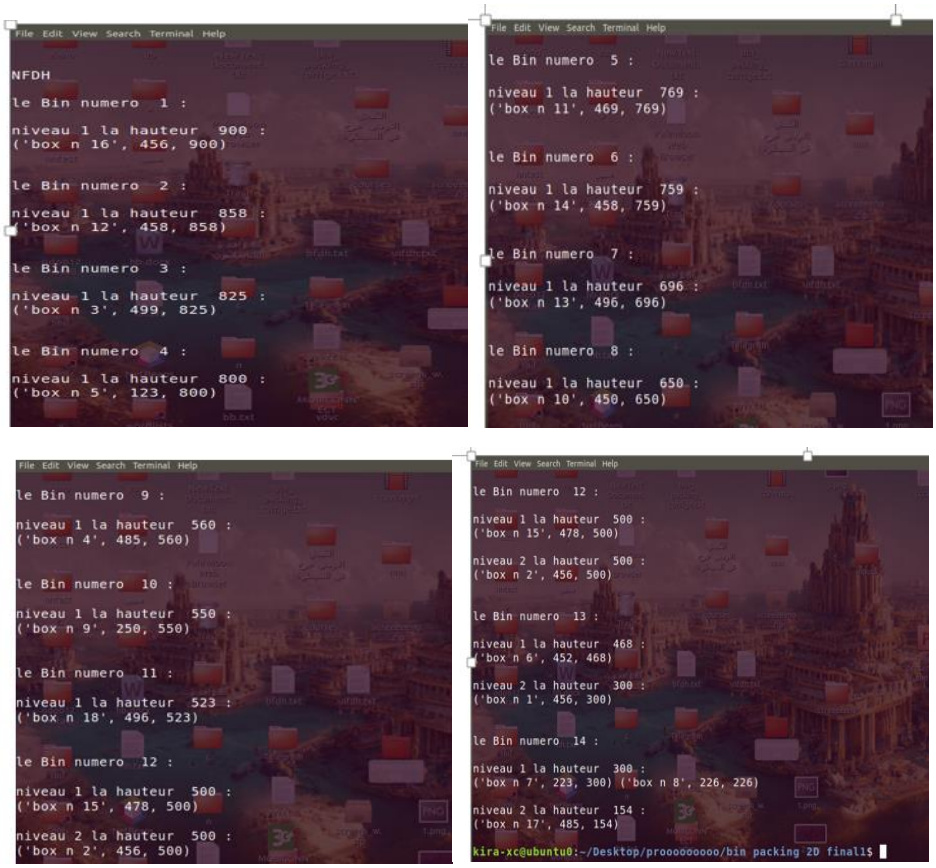


Figure 11:4.2.Les résultats obtenus avec NFDH

Résultat numérique

La figure suivant donne les résultats de problème sous forme d'un tableau et graphique.

NFDH

Bin 1		
Niveau 1 Hauteur : 900		
Numéro D'Objet	Largeur	Hauteur
16	456	900

Bin 2		
Niveau 1 Hauteur : 858		
Numéro D'Objet	Largeur	Hauteur
12	458	858

Bin 3		
Niveau 1 Hauteur : 825		
Numéro D'Objet	Largeur	Hauteur
3	499	825

Bin 4		
Niveau 1 Hauteur : 800		
Numéro D'Objet	Largeur	Hauteur
5	123	800

Bin 5		
Niveau 1 Hauteur : 769		
Numéro D'Objet	Largeur	Hauteur
11	469	769

Bin 6		
Niveau 1 Hauteur : 759		
Numéro D'Objet	Largeur	Hauteur
14	458	759

Bin 7		
Niveau 1 Hauteur : 696		
Numéro D'Objet	Largeur	Hauteur
13	496	696

Bin 8		
Niveau 1 Hauteur : 650		
Numéro D'Objet	Largeur	Hauteur
10	450	650

Bin 9		
Niveau 1 Hauteur : 560		
Numéro D'Objet	Largeur	Hauteur
4	485	560

Bin 10		
Niveau 1 Hauteur : 550		
Numéro D'Objet	Largeur	Hauteur
9	250	550

Bin 11		
Niveau 1 Hauteur : 523		
Numéro D'Objet	Largeur	Hauteur
18	496	523

Bin 12					
Niveau 1 Hauteur : 500			Niveau 2 Hauteur : 500		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
15	478	500	2	456	500

Bin 13					
Niveau 1 Hauteur : 468			Niveau 2 Hauteur : 300		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
6	452	468	1	456	300

Bin 14					
Niveau 1 Hauteur : 300			Niveau 2 Hauteur : 154		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
7	223	300	17	485	154
8	226	226			

Bin Size	
Largeur	Hauteur
500	1000

Figure 18 :4.3.Les résultats obtenus avec NFDH sous forme d'un tableau

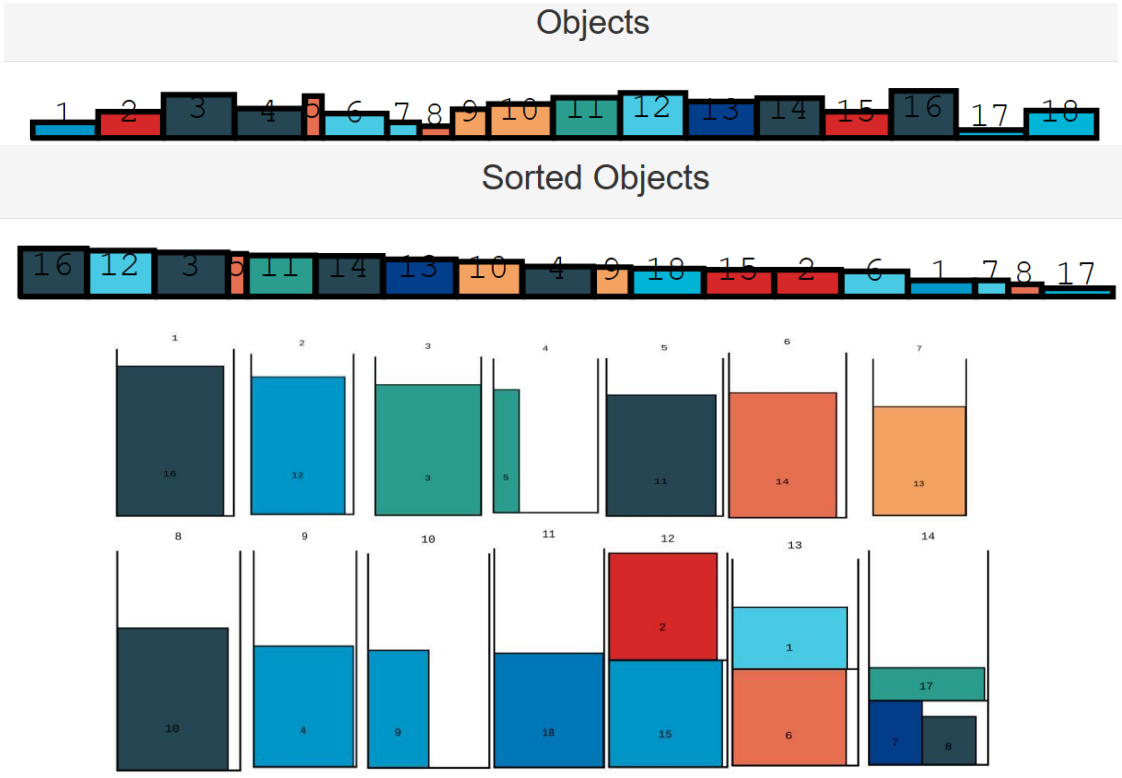


Figure *19*:4.4. L'ordre décroissante des objets et la représentation graphique du résultat avec NFDH

4.1.2. Résolution du problème en appliquant l'heuristique de Best-Fit Decreassing Height

La figure suivante donne l'écran de ce programme, elle montre la création du fichier objet.

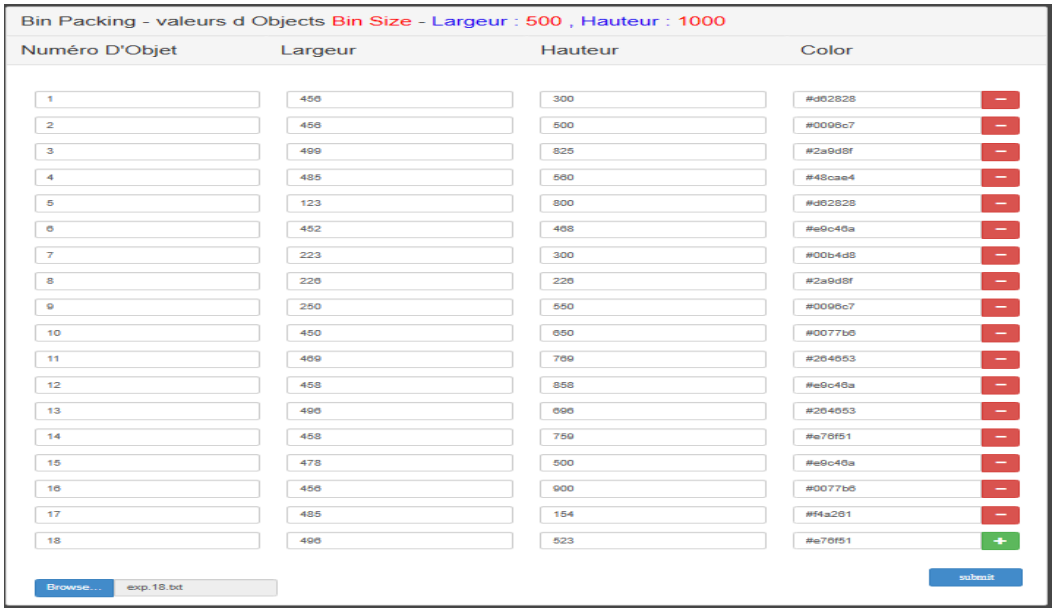


Figure 20:4.5.création d'objets pour méthode BFDH

La figure suivante donne :

-l'ordre des objets (ordre décroissante).

-La répartition des objets par les bins.

-Nombre des bins utilisée.

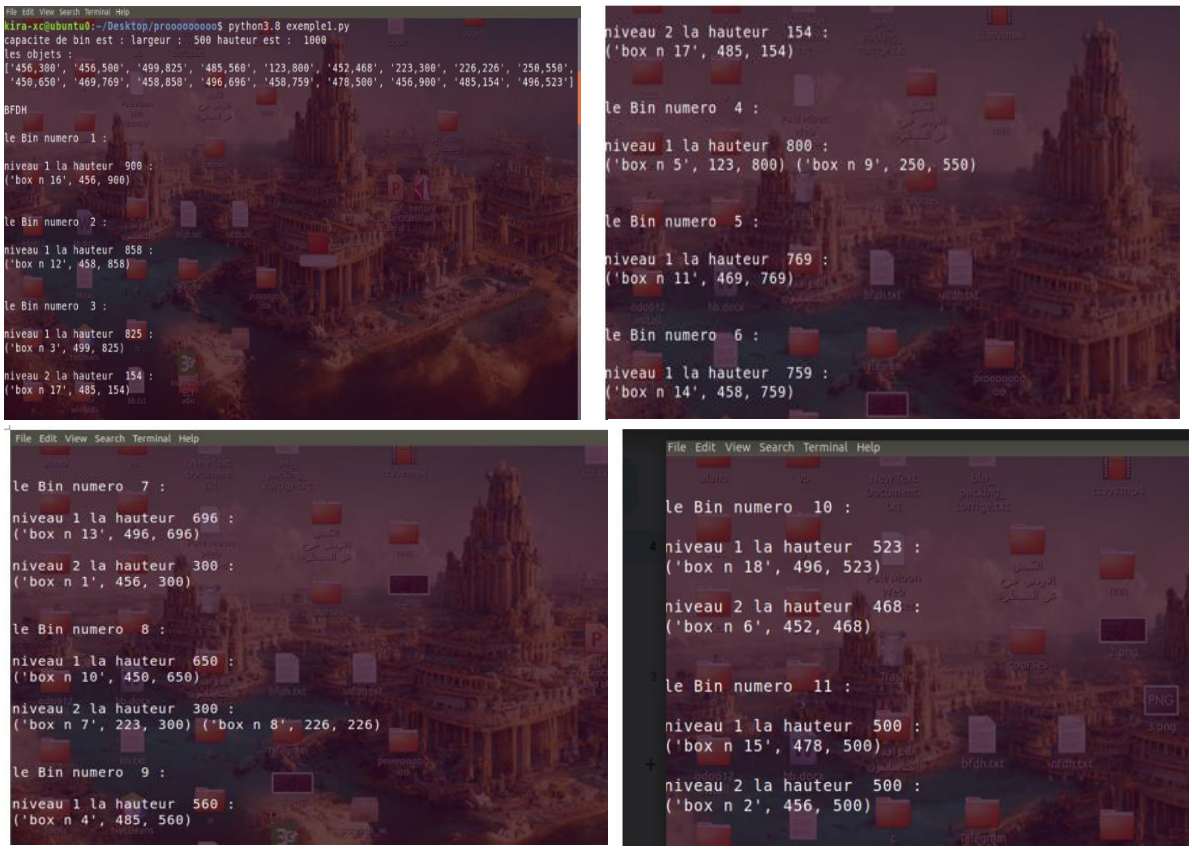


Figure 21:4.6.Les résultats obtenus avec BFDH

La figure donne :

La classification des objets dans des bins par un tableau et un graphe

BFDH

Bin 1		
Niveau 1 Hauteur : 900		
Numéro D'Objet	Largeur	Hauteur
16	456	900

Bin 2		
Niveau 1 Hauteur : 858		
Numéro D'Objet	Largeur	Hauteur
12	458	858

Bin 3					
Niveau 1 Hauteur : 825			Niveau 2 Hauteur : 154		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
3	499	825	17	485	154

Bin 4		
Niveau 1 Hauteur : 800		
Numéro D'Objet	Largeur	Hauteur
5	123	800
9	250	550

Bin 5		
Niveau 1 Hauteur : 769		
Numéro D'Objet	Largeur	Hauteur
11	469	769

Bin 6		
Niveau 1 Hauteur : 759		
Numéro D'Objet	Largeur	Hauteur
14	458	759

Bin 7					
Niveau 1 Hauteur : 696			Niveau 2 Hauteur : 300		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
13	496	696	1	456	300

Bin 8					
Niveau 1 Hauteur : 650			Niveau 2 Hauteur : 300		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
10	450	650	7	223	300
			8	226	226

Bin 9		
Niveau 1 Hauteur : 560		
Numéro D'Objet	Largeur	Hauteur
4	485	560

Bin 10					
Niveau 1 Hauteur : 523			Niveau 2 Hauteur : 468		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
18	496	523	6	452	468

Bin 11					
Niveau 1 Hauteur : 500			Niveau 2 Hauteur : 500		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
15	478	500	2	456	500

Bin Size	
Largeur	Hauteur
500	1000

Figure 22:*4.7.Les résultats obtenus avec BFDH sous forme d'un tableau*

Figure 23:4.8.la représentation graphique du résultat avec BFDH

4.2. Exemple d'application 2 :

Supposons que vous travailliez dans une entreprise de logistique qui gère l'expédition de produits à partir de différents entrepôts vers des destinations finales. L'un des défis consiste à optimiser le chargement des camions pour minimiser les coûts de transport. Vous disposez de différents types de produits, chacun ayant une taille (largeur ,hauteur) et une quantité spécifique à expédier. Vous devez trouver comment organiser ces produits dans les camions de la manière la plus efficace possible, en utilisant un espace limité.Par exemple, supposons que vous ayez les produits suivants :

De plus, vous disposez de camions avec des dimensions de 700x1400.Données :

Objet n°	1	2	3	4	5	6	7	8	9
Larguer	129	259	556	432	556	352	594	435	158
Hauteur	500	400	800	120	500	1100	1000	600	250

Objet n°	10	11	12	13	14	15	16	17	18
Larguer	350	125	225	100	450	150	200	400	100
Hauteur	500	300	600	700	500	200	800	250	350

Objet n°	19	20	21	22	23	24	25	26	27
Larguer	600	200	137	164	253	664	372	388	460
Hauteur	1000	150	254	587	358	925	524	625	387

Objet n°	28	29	30	31	32	33	34	35	36
Larguer	432	461	632	234	345	456	500	478	578
Hauteur	982	600	358	650	845	1102	325	687	900

Objet n°	37	38	39	40	41	42	43	44	45
Larguer	495	258	369	253	258	552	454	562	395
Hauteur	495	625	384	687	445	900	700	368	965

Tableau 3:4.2.Les tailles des objetsLes tailles des objets pour l'application 2

On exécutera notre programme sur micro-ordinateur PC doté d'un système d'exploitation Windows dix 64 bits ou l'environnement PYTHON fonctionne sans problème.

4.2.1. Résolution par l'heuristique NFDH :

La figure suivantes donne l'écran de ce programme, elle montre la création du fichier objet.

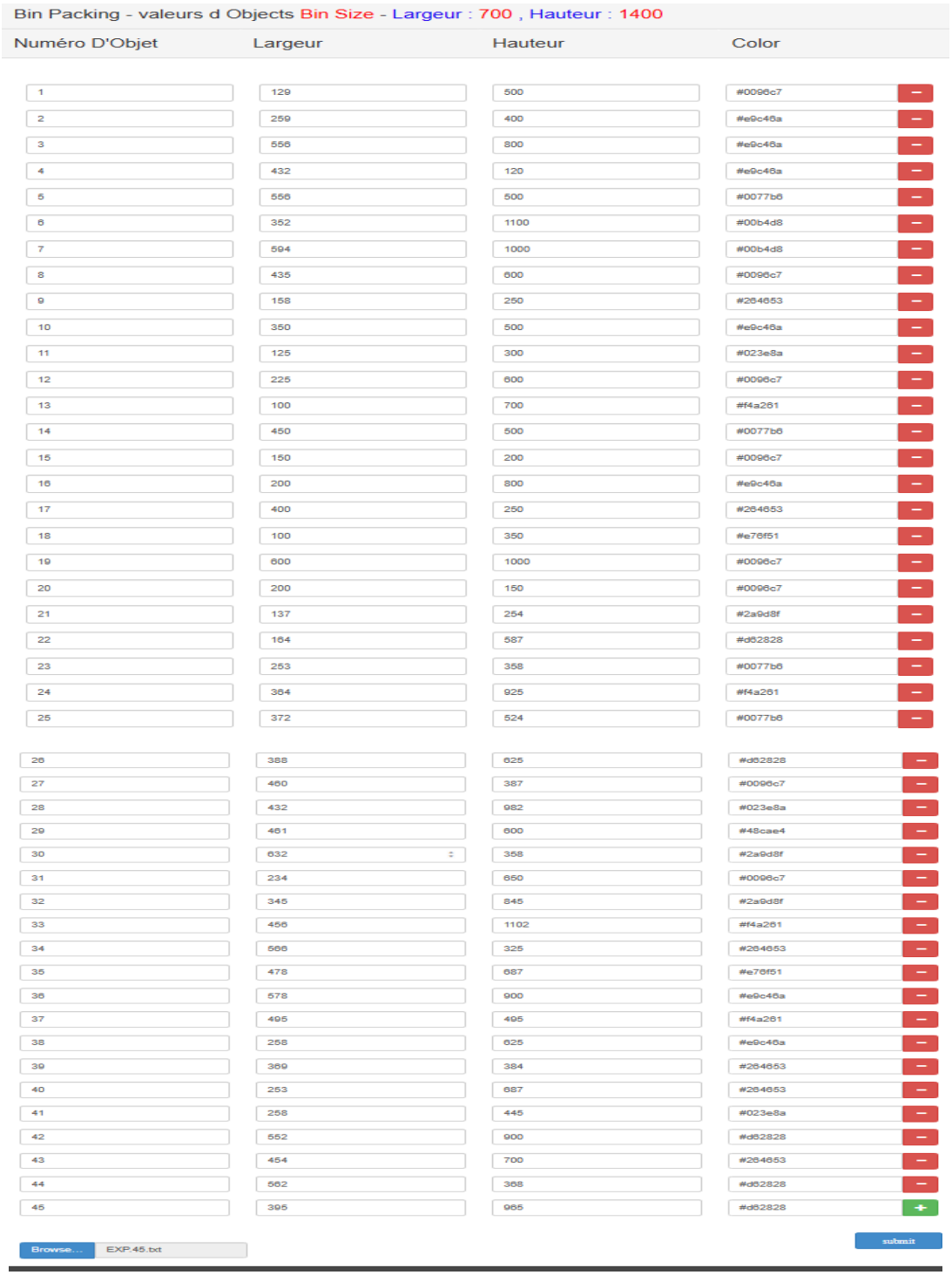
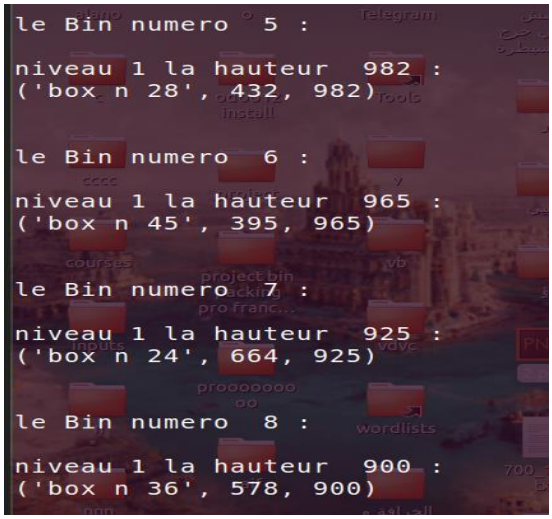
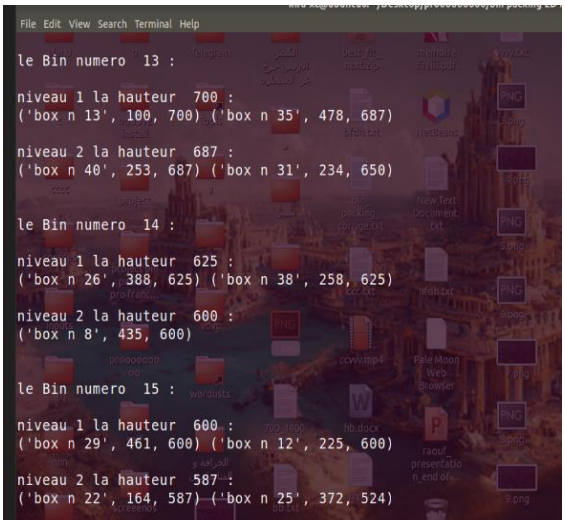
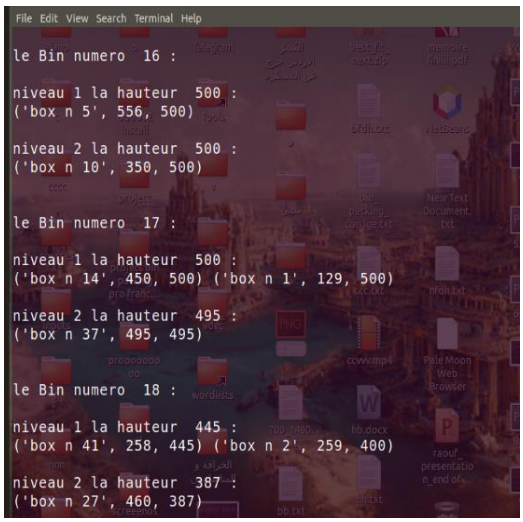


Figure 24:4.9.création d'objets pour méthode NFDH

La figure représenté classification dés objet est le nombre de bin utilisé pour méthode NFDH







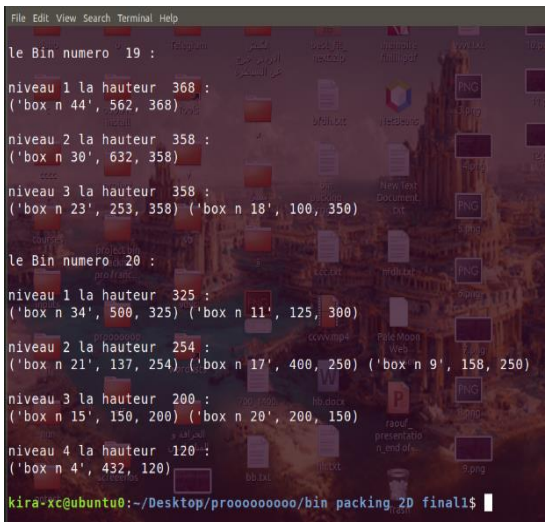


Figure 25:4.10.Les résultats obtenus avec NFDH

Le tableau suivant représenté lés objet dans les bins pour la méthode NFDH

NFDH

Bin 1		
Niveau 1 Hauteur : 1102		
Numéro D'Objet	Largeur	Hauteur
33	456	1102

Bin 2		
Niveau 1 Hauteur : 1100		
Numéro D'Objet	Largeur	Hauteur
6	352	1100

Bin 3		
Niveau 1 Hauteur : 1000		
Numéro D'Objet	Largeur	Hauteur
19	600	1000

Bin 4		
Niveau 1 Hauteur : 1000		
Numéro D'Objet	Largeur	Hauteur
7	594	1000

Bin 5		
Niveau 1 Hauteur : 982		
Numéro D'Objet	Largeur	Hauteur
28	432	982

Bin 6		
Niveau 1 Hauteur : 965		
Numéro D'Objet	Largeur	Hauteur
45	395	965

Bin 7		
Niveau 1 Hauteur : 925		
Numéro D'Objet	Largeur	Hauteur
24	664	925

Bin 8		
Niveau 1 Hauteur : 900		
Numéro D'Objet	Largeur	Hauteur
36	578	900

Bin 9		
Niveau 1 Hauteur : 900		
Numéro D'Objet	Largeur	Hauteur
42	552	900

Bin 10		
Niveau 1 Hauteur : 845		
Numéro D'Objet	Largeur	Hauteur
32	345	845

Bin 11		
Niveau 1 Hauteur : 800		
Numéro D'Objet	Largeur	Hauteur
3	556	800

Bin 12		
Niveau 1 Hauteur : 800		
Numéro D'Objet	Largeur	Hauteur
16	200	800
43	454	700

Bin 13					
Niveau 1 Hauteur : 700			Niveau 2 Hauteur : 687		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
13	100	700	40	253	687
35	478	687	31	234	650

Bin 14					
Niveau 1 Hauteur : 625			Niveau 2 Hauteur : 600		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
26	388	625	8	435	600
38	258	625			

Bin 15					
Niveau 1 Hauteur : 600			Niveau 2 Hauteur : 587		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
29	461	600	22	164	587
12	225	600	25	372	524

Bin 16					
Niveau 1 Hauteur : 500			Niveau 2 Hauteur : 500		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
5	556	500	10	350	500

Bin 17					
Niveau 1 Hauteur : 500			Niveau 2 Hauteur : 495		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
14	450	500	37	495	495
1	129	500			

Bin 18

Niveau 1 Hauteur : 445			Niveau 2 Hauteur : 387			Niveau 3 Hauteur : 384		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
41	258	445	27	460	387	39	369	384
2	259	400						

Bin 19

Niveau 1 Hauteur : 368			Niveau 2 Hauteur : 358			Niveau 3 Hauteur : 358		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
44	562	368	30	632	358	23	253	358
						18	100	350

Bin 20

Niveau 1 Hauteur : 325			Niveau 2 Hauteur : 254			Niveau 3 Hauteur : 200			Niveau 4 Hauteur : 120		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
34	500	325	21	137	254	15	150	200	4	432	120
11	125	300	17	400	250	20	200	150			
			9	158	250						

Figure 26:4.11.Les résultats obtenus avec NFDH

LaFigure suivant donne la représentation graphique d'objets dans les bins pour la méthode NFDH

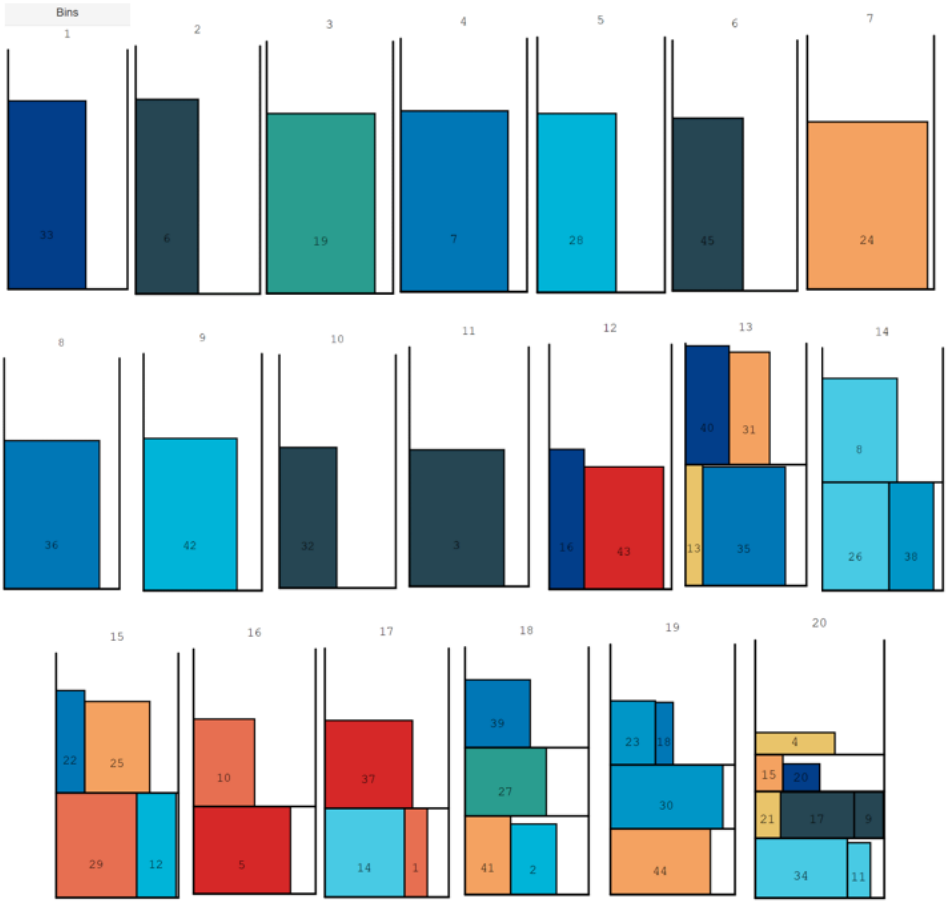


Figure 27:4.12.la représentation graphique du résultat avec NFDH

4.2.2. La Résolution de problème par la méthode BFDH

La figure suivante donne l'écran de ce programme, elle montre la création du fichier objet.

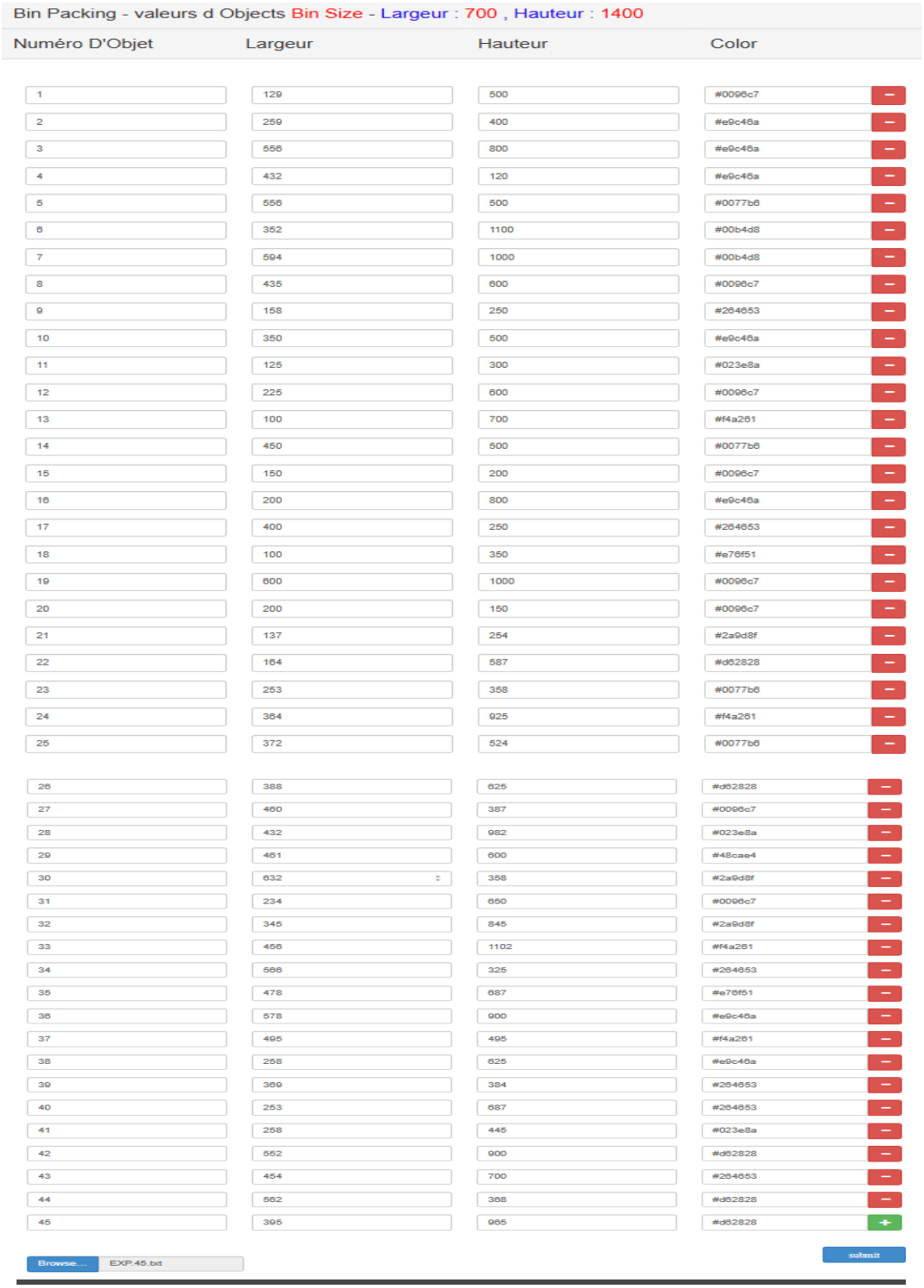


Figure 28:4.13.création d'objets pour méthode BFDH

La figure représenté classification dés objet est le nombre de bin utilisé pour méthode BFDH

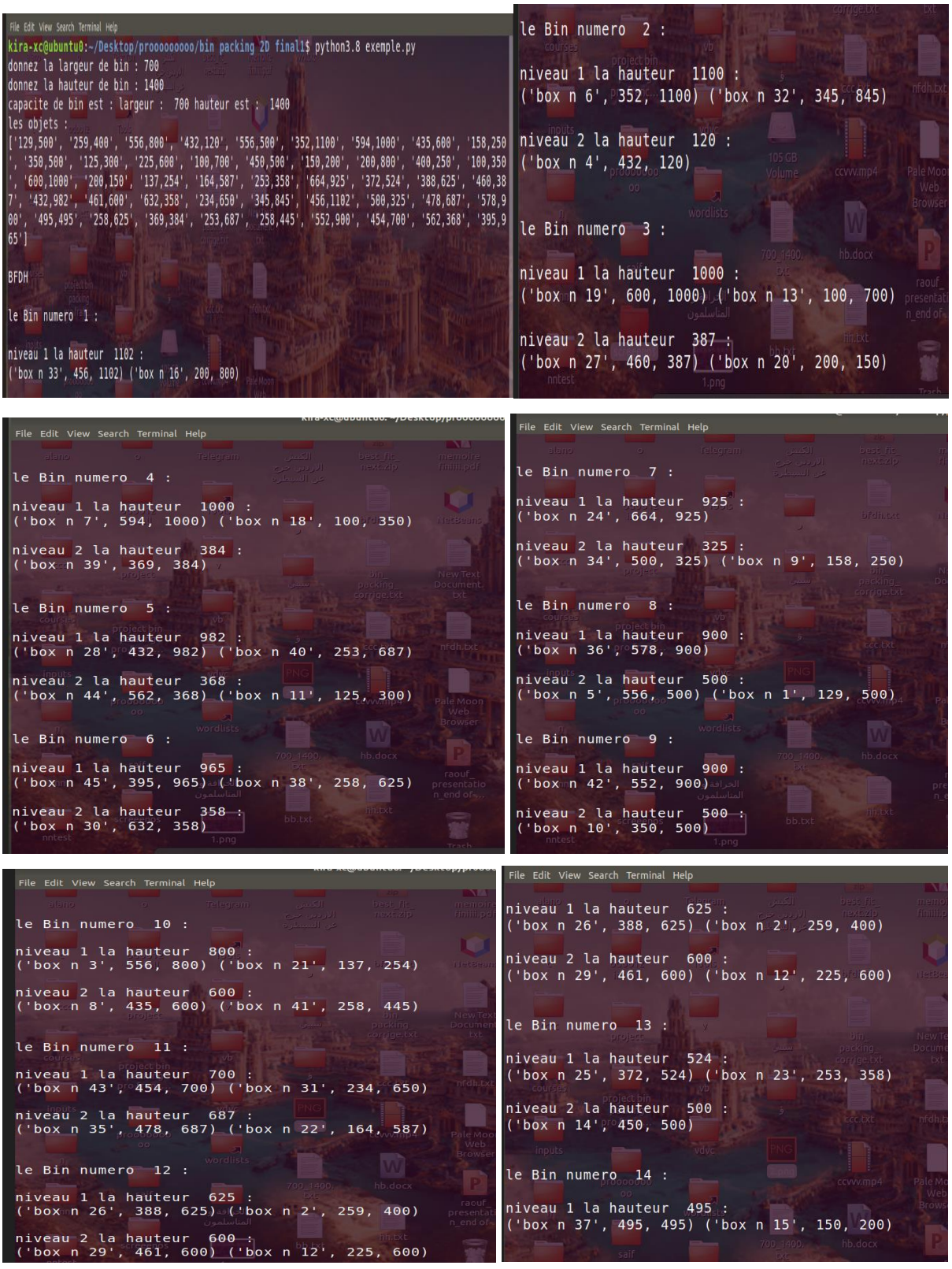


Figure 29:4.15.Les résultats obtenus avec BFDH

Le tableau suivant représenté lés objet dans les bins pour la méthode BFDH

BFDH

Bin 1					
Niveau 1 Hauteur : 1102			Niveau 2 Hauteur : 250		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
33	456	1102	17	400	250
16	200	800			

Bin 2					
Niveau 1 Hauteur : 1100			Niveau 2 Hauteur : 120		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
6	352	1100	4	432	120
32	345	845			

Bin 3					
Niveau 1 Hauteur : 1000			Niveau 2 Hauteur : 387		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
19	600	1000	27	460	387
13	100	700	20	200	150

Bin 4					
Niveau 1 Hauteur : 1000			Niveau 2 Hauteur : 384		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
7	594	1000	39	369	384
18	100	350			

Bin 5					
Niveau 1 Hauteur : 982			Niveau 2 Hauteur : 368		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
28	432	982	44	562	368
40	253	687	11	125	300

Bin 6					
Niveau 1 Hauteur : 965			Niveau 2 Hauteur : 358		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
45	395	965	30	632	358
38	258	625			

Bin 7					
Niveau 1 Hauteur : 925			Niveau 2 Hauteur : 325		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
24	664	925	34	500	325
			9	158	250

Bin 8					
Niveau 1 Hauteur : 900			Niveau 2 Hauteur : 500		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
36	578	900	5	556	500
			1	129	500

Bin 9					
Niveau 1 Hauteur : 900			Niveau 2 Hauteur : 500		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
42	552	900	10	350	500

Bin 10					
Niveau 1 Hauteur : 800			Niveau 2 Hauteur : 600		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
3	556	800	8	435	600
21	137	254	41	258	445

Bin 11					
Niveau 1 Hauteur : 700			Niveau 2 Hauteur : 687		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
43	454	700	35	478	687
31	234	650	22	164	587

Bin 12					
Niveau 1 Hauteur : 625			Niveau 2 Hauteur : 600		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
26	388	625	29	461	600
2	259	400	12	225	600

Bin 13					
Niveau 1 Hauteur : 524			Niveau 2 Hauteur : 500		
Numéro D'Objet	Largeur	Hauteur	Numéro D'Objet	Largeur	Hauteur
25	372	524	14	450	500
23	253	358			

Bin 14		
Niveau 1 Hauteur : 495		
Numéro D'Objet	Largeur	Hauteur
37	495	495
15	150	200

Bin Size	
Largeur	Hauteur
700	1400

Figure 30:4.16.Les résultats obtenus avec BFDH

La Figure suivant représentation graphique d'objets dans les bins pour la méthode BFDH

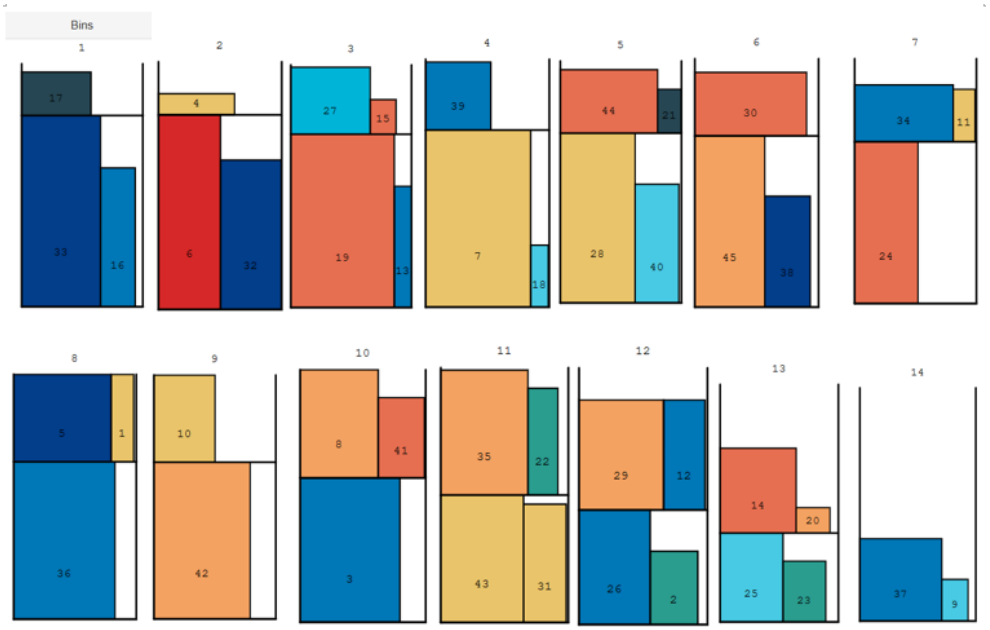


Figure 31:4.17.la représentation graphique du résultat avec BFDH

5. Simulation

Dans cette partie on essayera de voir le comportement de l'heuristique NFDH et BFDH, en variant le nombre d'objets et fixé la capacité. Les résultats de cette simulation sont résumés dans le tableau suivant :

N° d'objet	100	500	1000	2000	3000	4000	5000	10000
N° du bin avec NFDH	12	60	119	238	356	475	594	1187
N° du bin avec BFDH	10	46	92	182	273	364	455	908

Table 4:4.3.les résultats de simulation

Conclusion générale

Nous avons traité dans ce mémoire le problème de placement bidimensionnel, qui est un modèle mathématique très attrayant, et pourtant, travailler sur ce problème est étonnamment récent.

Le placement optimal en tant que sujet organisé, n'a que 35 ans environ. Les principaux pionniers et contributeurs de ce problème sont« Edward Coffman, Jr », « Michael Garey », « Ronald Graham »et « David Johnson ».

L'importance de ce problème réside par le domaine vaste de ses applications concrètes en transport - logistique et en différentes industries (papier, bois,…) ainsi que dans d'autres domaines d'intérêt publique.

Le problème de placement est un problème NP-Complet, c'est-à-dire il n'existe pas une méthode exacte qui nous permet de trouver la solution optimal dans un temps raisonnable sauf si on utilisé une méthode de parcoure de tous l'ensemble des solutions réalisable.

Ce modeste travail nous a permet de cerner et de comprendre les problèmes de l'optimisation combinatoire, les méthodes de résolution ainsi que les classes de complexités.

On a ensuite traité un problème d'optimisation combinatoire à savoir le problème de placement en présentant les diverses heuristiques de résolution puis d'implémenter l'heuristique Next Fit Decreasing Height et Best Fit Decreasing Height, l'application informatique aussi simple que soit elle nous a permet aussi de voir la rapidité de l'obtention des résultats et on trouve que l'heuristique de BFDH meilleure que l'heuristique de NFDH.

Références bibliographiques

[1] **A. Yalaoui**., Allocation de fiabilité et de redondance dans les systèmes parallèle- série et série- parallèle, thèse de doctorat, 2004.

[2] **A. Lodi, S. Martello, D. Vigo**, Heuristic and Metaheuristic Approaches for a Class of Twodimensional Bin packing problems. INFORMS Journal of Computing 11, 345-357, 1999.

[3] **D. Corne, M. Dorigo and F. Glover**, editors, New Ideas in Optimization, McGraw-Hill, 11-32

[4] **Derbala. Ali**, Optimisation combinatoire cours de Master1 departement de Mathématiques université de Blida1, 2009-2010.

[5] **D.G.Kirkptrik. Gellat, Vechi** .optimization by simulated annealing ,science, 220.pp.671-680,1983

[6] **E. Hopper**, Two-dimensional Packing utilising Evolutionary Algorithms and other Meta-Heuristic Methods, Degree of Doctor of Philosophy, University of Wales, Cardiff School of Engineering, May 2000

[7] **E. E. Bischoff, G. Wascher**, Cutting and Packing, The European Journal of Operational Research, 84, 503-505, 1995.

[8] **Frédéric.Meunier** Université Paris Est, CERMICS, Ecole des Ponts Paristech, 6-8 avenue Blaise Pascal, 77455 Marne-la-Vallée Cedex.

[9] **Guellal,Z'hor .Gaci ,Yacine**.Optimisation du transport de gaz naturel par le gazoduc GZ1 Hassi R'mel TRC Sona-trach.These de Master université de M'Hamed Bougera Boumerdes UMBB.2016

[10] **G. Wäscher, H. Haussner**, H. Schumann, An improved typology of cutting and packing problems. European Journal of Operational Research 83 {3), 1109-1130, 2007.

[11] **H. Dyckhoff et U. Finke**, Cutting and Packing in Production and Distribution; A Typologie and Bibliography, ed. Physica-verlag, A Springer-Verlag, 1992.

[12] **J. O. Berkey, P. T. Wang.** Two-dimensional finite bin-packing algorithms.

Journal of Operational Research Society, 38, 423-429, 1987..

[13] **Landauer Limit Demonstrated - IEEE Spectrum** , (consulté le 5 mai 2013) Programmation dc. Master's thesis, Université Abderrahmane Mir Bejaia, 2013/2014.

[14] **L. Kantorovich**. Mathematical methods of organizing and planning production. *Management Science*, 6 :363–422, 1960. (Cité page 18.)

[15] **Mostepha, R** : Résolution de problèmes d'optimisation combinatoire par systèmes artificiels auto-organisés. Thèse de magister, Université Mentouri de Constantine,2008

[16] **Palpant M**. : Recherche exacte et approchée en optimisation combinatoire : schémas d'intégration et applications. Thèse de Doctorat, Université d'Avignon, 2005.

[17] **P. Gilmore et R. Gomory**. A linear programming approach to the cutting stock problem - part II. Ops. Res., 11 :863–888, 1963

[18**] R. Faure, B. Lemaire et C. Picouleau**, Précis de recherche opérationnelle-Méthodes et exercices d'application, 7e éd, Dunod, 2014

[19] **S. Fekete et J. Schepers**. A combinatorial characterization of higher-dimensional orthogonal packing. Mathematics of Operations Research, 29 :353–368, 2004.

[20] **S. Martello, M. Monaci, D. Vigo**. An exact approach to the strip-packing problem. INFORMS Journal of Computing, 15 ,310-319, 2003

[21] **T. Sofiane**. Résolution de problèmes de bin packing a une dimension ou encore pound p ou number p, voir Johnson 1992

[22] **Tikalon Blog by Dev Gualtieri** , Tikalon.com (consulté le 5 mai 2013)

[23] Yann Collette , Patrick Siarry , Optimisation Multiobjectif , ÉDITIONSEYROLLE S61, Bld Saint-Germain 75240 Paris Cedex 05 ,2002.

[24] Algorithm de Nest-Fit Decreasing Height (NFDH)

[25] Algorithm de Best-Fit Decreasing Height (BFDH)

Liste des Abréviations

Lettre	abréviation	signification
B	Bl	Bottom-left
	BP	Bin acking
	BPP-1D	Bin Packing uni-dimensionnel
	BPP-2D	Bin Packing bi-dimensionnel
	BPP-3D	Bin Packing tri-dimensionnel
	BF	Best-Fit
	BFDH	Best-Fit-Decreasing –Height
F	FF	First-Fit
	FFD	First-Fit Decreasing-Height
	FFF	Finite- First-Fit
N	NF	Next-Fit
	NFDH	Next-Fit Decreasing -Height
O	OC	Optimisation Combinatoire
P	PSE	Procédure par Séparation et évaluation
R	RO	Recherche Opérationnel