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

Option : Recherche Opérationnelle

Problème de Bin Packing unidimensionnel

Réalisé par :

Mr : OUAD FATEH

Mr : GROUCI ADEL

Devant le jury composé de :

Présidente :	S. ARRACHE	MAA	USD. Blida 1
Examinatrice :	N. MESSAOUDI	MCB	USD. Blida 1
Promotrice :	N. DJEMIA	MAA	USD. Blida 1

Année universitaire : 2020/2021

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.

A l’ensemble des enseignants qui m’avoir enseigné et honorer le jury.

Mes remerciements vont également à tous ceux qui ont participé
de près ou de loin à la réalisation de ce travail.

Fateh &Adel

Dédicace

Je dédie ce modeste travail, qui est le fruit de ma
profonde reconnaissance :
A mon père bien-aimé pour son amour, ses efforts et
Son Soutien.
En l'honneur de ma chère et affectueuse maman.
A mon merveilleux frère : Saleh.
A ma chère sœur : Siham
Aux deux gentils fils de mon frère : Fares et Ishaq.
A mes chers amis : Ayoub, Abdou, Meftah , Adam.
Adel et Rachid pour leurs encouragements constants.

Fateh

Dédicace

Je dédie ce mémoire de fin d'étude :

A mes très chers parents,

A mon père, qui m'a aidé à construire l'homme et la personne que je suis aujourd'hui, sans lui toute ces années n'auraient jamais pu aboutir, grâce à son soutien moral et l'éducation qu'il m'a donné, il m'a appris de ne jamais abandonner et de me battre coute que coute.

A ma mère, l'incarnation de la douceur et de la bienveillance, celle qui m'a appris à aimer le travail, aimer les autres et aimer la vie, dans les moments difficiles tu as été la lumière qui m'a guidé et que j'ai suivie sans me poser de questions, je te dédis toute ces années de dur labeur et ce mémoire que j'ai façonné à ton image.

A mon frère et à ma sœur, pour leurs encouragements permanents, et leur soutien moral.

Mes amis, qui m'ont accompagné à chaque pat que je marchais, leurs réconfort inconditionnel et leur bonnes idées.

A tous mes camarades ainsi qu'aux professeurs qui m'ont enseigné tout au long de mon parcours Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fuit de votre soutien infaillible,

Merci d'être toujours là pour moi.

Adel

Résumé

Ce travail vise à proposer une solution au problème de remplissage des boîtes unidimensionnelles, 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 : Best-Fit, Best-Fit Decreassing, Next-Fit, Worst Fit et First-Fit où nous avons discuté en détail deux algorithmes Best-Fit et Best-Fit

Decreassing,

ملخص

يطمح هذا العمل الى اقتراح حل مشكل تعبئة الصندوق احادي 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 توجد العديد من الخوارزميات مثل :

Best-Fit, Best-Fit Decreassing and Next-Fit, Worst- Fit

حيث تطرقنا بالتفصيل الى خوارزمية :

Best-Fit, Best-Fit Decreassing .

Abstract

The aim of the work is to propose a solution to Bin Packing one dimensional box problem, which consist of using the minimum number of bins to store objects.

This problem exists in many sectors such as industrial and computational ones.

To solve the problem of filling the Boxes, there are many algorithms such as: Best-Fit, Best-Fit Decreasing, Next-Fit, Worst Fit and First-Fit. We have applied two algorithms Best-Fit and Best- Fit Decreasing,

Table des matières

Remerciements 2
Résumé 5
Liste des figures 10
Liste des tableaux 11
Introduction générale 12
Chapitre I 15
1. Introduction 16
2. Formulation mathématiques 16
3. Notions de base sur l'optimisation combinatoire 18
4. Complexité des problèmes et leur classification 18
5. Méthodes de résolution 20
5.1. Les méthodes exactes 20
5.2. Les méthodes approchées (heuristique et métaheuristique) 22
6. Quelques problèmes d'optimisation combinatoire 26
7. Conclusion 28
Chapitre II 29
1. Introduction 30
2. Présentation du problème de bin packing (BPP) 30
3. Domaines d'application du problème de bin packing 31
4. Contraintes pratiques et classification 31
5. Le problème de bin packing uni-, bi- et tri-dimensionnel 33
5.1 Le problème de bin packing uni dimensionnel (BPP-1D) 33
5.2. Le problème de bin packing bi-dimensionnel (BPP-2D) 34
5.3. Le problème de bin packing tri-dimensionnel (BPP-3D) 36
6. Formulations mathématiques 37
6.1. Cas uni – dimensionnel 38
7. Conclusion 39
Chapitre III 40
1. Introduction 41
2. Définition du problème de placement 41
3. Quelques exemples d'application du problème de placement 42

4. Classification des problèmes de placement 42
4.1. Problème de placement en un dimension 43
4.2 Problème de placement en deux dimensions 43
4.3 Problème de placement à trois dimensions : 43
5. Méthodes de résolution pour le problème de placement 44
5.1 Méthodes exactes 44
5.1.1 Procédure par séparation et évaluation 44
5.2 Heuristiques de résolution pour le problème de Placement en une dimension 46
6. Algorithme de Best Fit (BF) 50
7. Algorithme de Best-Fit Decreasing (BFD) 52
8. Conclusion 55
Chapitre IV 56
1. Introduction 57
2. Matlab 8.6 57
4. Implémentation et comparaison 58
4.1. Exemple d'application 1: 58
4.1.1. Résolution du problème en appliquant l'heuristique de Best-Fit 59
4.1.2. Résolution du problème en appliquant l'heuristique de Best-Fit Decreassing 62
4.2. Exemple d'application 2 : 65
4.2.1. Résolution par l'heuristique BF : 66
4.2.2. La Résolution de problème par la méthode BFD 70
5. Simulation 74
Conclusion générale 75
Références bibliographiques 76
Annexe 78
Liste des Abréviations 85

Liste des figures

Figure 1:1.1.classification de complexité des problèmes.....20
Figure 2:1.2.Quelques méthodes de résolution d'un problème d'optimisation 25
Figure 3:2.1.Exemple de Bin packing 30
Figure 4:2.2.une instance de BPP-1D et une solution possible 33
Figure 5:2.3.Une instance de BPP-2D et une solution possible.....34
Figure 6:2.4.Bin packing le cas non orienté 35
Figure 7:2.5.Bin packing le cas orienté 36
Figure 8:2.6.Illustration graphique du BPP-3D 37
Figure 9:2.7.Illustration d'une solution réalisable pour BPP-3D.....37
Figure 10:3.1.Exemple du problème de placement.....41
Figure 11:3.2.Le placement en 3D 43
Figure 12:3.3.l'exploration des solutions dans un arbre de recherche.....45
Figure 13:3.4.Exemple avec solution en stratégie NF 47
Figure 14:3.5.Exemple avec solution en stratégie FF.....48
Figure 15:3.6.Exemple avec solution en stratégie BF.....49
Figure 16:4.1.Création d'objets pour La méthode BF 59
Figure 17:4.2.Les résultats obtenus avec BF.....60
Figure 18:4.3.la représentation graphique du résultat avec BF 61
Figure 19:4.4.création d'objets pour méthode BFD.....62
Figure 20:4.5.Les résultats obtenus avec BFD 63
Figure 21:4.6.la représentation graphique du résultat avec BFD 64
Figure 22:4.7.création d'objets pour méthode BF 66
Figure 23:4.8.Les résultats obtenus avec BF.....67
Figure 24:4.9.la représentation graphique du résultat avec BF 69
Figure 25:4.10.création d'objets pour méthode BFD.....70
Figure 26:4.11.Les résultats obtenus avec BFD 71
Figure 27:4.12.la représentation graphique du résultat avec BFD 73

Liste des tableaux

Table 1:3.1.les objets et ses poids 41
Table 2:4.1.Les tailles des objets pour l'application 1..... 58
Table 3:4.2.Les tailles des objetsLes tailles des objets pour l'application 2 65
Table 4:4.3.Les résultats obtenus avec BF..... 68
Table 5:4.4.Les résultats obtenus avec BFD 72
Table 6:4.5.les résultats de simulation..... 74

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 [1].

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 [2].

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[3][4].

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 [5].

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)[34].

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 Uni dimensionnelle Par deux heuristique (BF et BFD) bien défini qui s'adapte à ce type de problème et de le résoudre avec un logiciel Matlab 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 de 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 matlab.
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 [6].

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 [6].

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 .Il comprend trois étapes [7] :

1ere Etape : Définition des variables de décision

Les variables de décision représentent comme leurs noms l'indiquent, les décisions à prendre afin de satisfaire le ou les objectifs. Dans certains cas, les variables de décision sont sous la forme d'entiers ou de binaires, le problème d'optimisation est dit dans ce cas discret.

Les variables peuvent prendre n'importe quelle valeur. Le problème d'optimisation est dit continue sont à résoudre. Un problème d'optimisation mêlant variables continues et variables discrètes est dit mixte [8].

2éme Etape : Détermination de la fonction objectif

Appelée aussi fonction coût ou fonction économique, permettant d'évaluer l'état du système. Autrement dit, c'est le but à atteindre.

Les problèmes mono-objectifs sont définis par une unique fonction objectif les problèmes multi objectifs existent quand un compromis est à rechercher entre plusieurs objectifs.

3éme partie : Identification des contraintes imposées au variable de décision

Ecrire les contraintes sous formes d'égalités ou d'inégalités qui traduisent la relation entre les objets du système.

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)) $\qquad\qquad$ $s \in S$

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 [8].

- **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, [9], [10]) il existe plusieurs classes de complexité, les plus connues sont les suivantes [8] :

- ✓ **La classe P**

C'est la classe des problèmes de décision relativement faciles c'est-à-dire ceux pour lesquels on connait des algorithmes polynomiaux ou bien efficaces. (Le sigle P signifie « polynomial time »).

- ✓ **La classe NP**

C'est la classe qui contient tous les problèmes de décision qui peuvent être exprimés sous la forme d'une question dont la réponse est soit oui, soit non. Le problème appartient à la classe NP si on arrive à l'aide d'un algorithme de polynomiale complexe que n'importe quelle instance (donnée) oui du problème bien corrigée. (NP est l'abréviation pour "temps polynomial non déterministe"). (La vérification est polynomiale).

- ✓ **La Classe NP-Complet**

Un problème de décision (*) est NP-Complet s'il satisfait les deux conditions suivantes :

(*) Appartient à NP, et tout problème NP se réduit à (*) en temps polynomial.

La classe NP-complet contient un ensemble de problèmes les plus difficiles de NP, de tels problèmes apparaissent régulièrement. Citons entre autres, le problème du voyageur de commerce, celui du stable maximal, le calcul du nombre chromatique...etc. tous ces problèmes correspondent à une difficulté pratique pour trouver un algorithme efficace pour de très grands graphes (les seuls algorithmes déterministes connus étant de complexité non polynomiale, souvent exponentielle).

Cette difficulté n'empêche pas de mettre au point des heuristiques, c'est-à-dire qui n'assurent pas à coup sûr l'obtention d'une solution du problème, mais y parviennent assez souvent ou assez près. Après avoir utilisé une méthode heuristique, on peut conclure la recherche avec une méthode exhaustive, c'est-à-dire d'examen a priori des toutes les configurations possibles.

La grande question ouverte dans ce domaine, et qui est classée par l'institut de mathématique. Clay parmi les 7 problèmes du prix du millénaire, concerne l'inclusion des classes P et NP, la question est traditionnellement formulée ainsi :<Est-ce-que N = N P ?

Cette question n'ait pas encore trouvé de réponse car de très nombreux problèmes fondamentaux s'avèrent être < NP -difficiles >, et aucun algorithme polynomial n'existe pour les résoudre, sauf si P=NP. Puisqu'il est tout de même important de pouvoir les résoudre, les scientifiques introduisent et utilisent diverses méthodes pour attaquer ces problèmes. (Voir la figure 1.1).

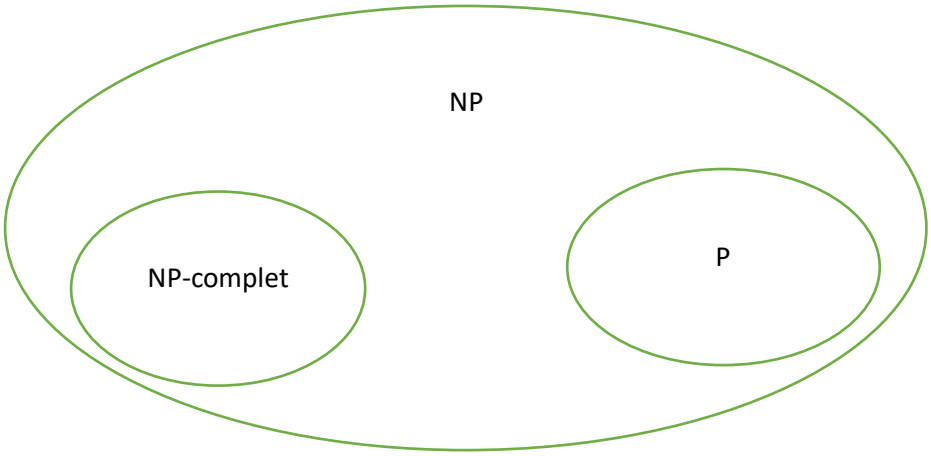


Figure 1:1.1.classification de complexité des problèmes

5. Méthodes de résolution

Les méthodes de résolution des problèmes d'optimisation combinatoire sont classées en deux catégories : les méthodes exactes et les méthodes approchées.

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 [11] :

-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.

Algorithme du simplexe

Pas 0 : initialisation

Mettre le problème sous forme standard (contraintes d'égalités et variables positives).

Trouver une première solution de base réalisable. Si aucune solution de base réalisable n'existe, alors STOP. Le problème est irrésoluble.

Pas 1 : Choix de la variable entrante

- Choisir comme variable entrante une variable hors base qui améliore la valeur courante de l'objectif.
- Si aucune variable hors base n'est candidate, alors STOP. La solution de base courante est une 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 $Ax \leq b$ ($b \geq 0$).

Algorithme dual de simplexe

- Mettre le problème sous forme standard (contraintes d''égalités et variables positives)
- Trouver une solution de base dual-réalisable (tous les couts relatifs cj sont positifs (négatifs) dans le cas d'un problème de minimisation (maximisation))

Pas 1 : Choix de la variable sortante

- Choisir comme variable sortante une variable en base dont la valeur est strictement négative.

Soit x_{ir} cette variable, b_r la valeur de cette variable et *r* le numéro de la contrainte qui donne cette valeur.

- Si aucune variable de base n'est strictement négative, alors STOP. La solution de base dual réalisable courante est primal réalisable et optimale.
- Si aucune variable de base n'est strictement négative, alors STOP. La solution de base dual réalisable courante est primal réalisable et optimale.

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 [11].Ils ont ainsi pu 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 [12].

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**)

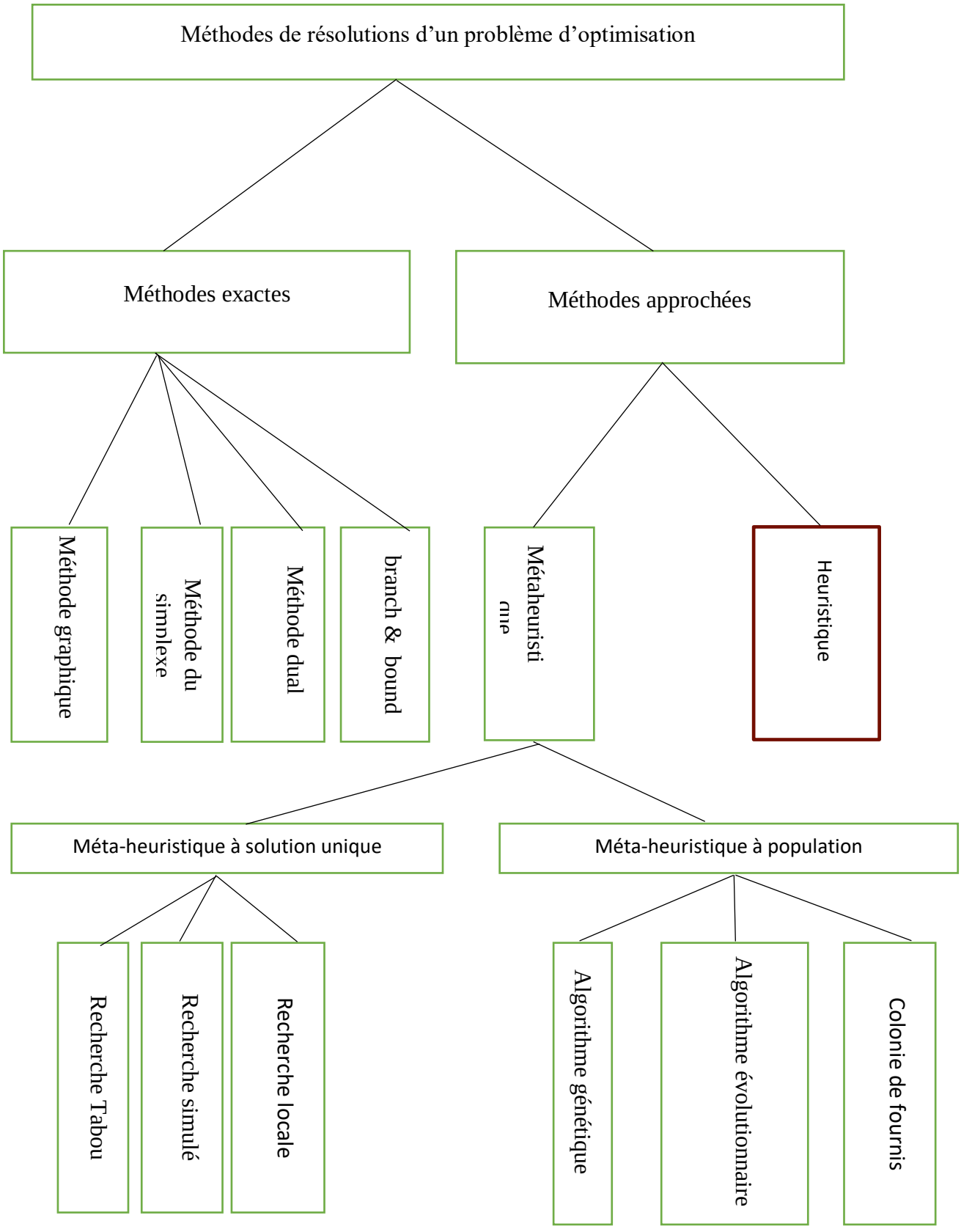


Figure 2:1.2.Quelques méthodes de résolution d'un problème d'optimisation

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

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

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

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

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

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

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

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

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 $D_1,\ldots,D_p$ vers q points d'arrivées $A_1,\ldots,A_q$.

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)

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

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

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

Le problème de Bin Packin

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 « bins ». Ainsi, les problèmes de type bin packing consistent à placer des objets caractérisés par leur formes dans une ou plusieurs boîtes. Les variantes se distinguent selon la dimension, la connaissance à priori des objets, la forme des objets et des boîtes (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 boîtes, par exemple la minimisation du nombre de boîtes (bin packing) ou la minimisation des dimensions d'une seule boîte (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.

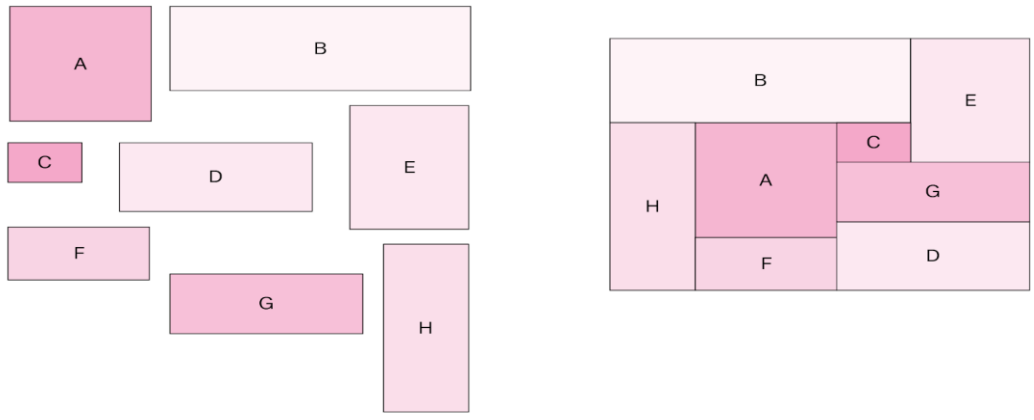


Figure 3:2.1.Exemple de Bin packing

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.
- ✓ Dans la fabrication des tubes de tailles différentes. Au lieu d'acheter plusieurs machines pour produire des tubes de tailles différentes, une seule machine produit des tubes de tailles fixées et une seconde machine découpe les tubes en petits tubes de tailles différentes. Dans ce cas, le problème bin packing est utilisé pour découper plus efficacement afin de minimiser le coût de fabrication. Différents logiciels industriels pour la découpe des tubes, découpe 2D/3D pour le découpe automatique sont largement commercialisés.
- ✓ Placement de boîtes sur une palette.
- ✓ Placement dans un entrepôt.
- ✓ Rangement d'objets physiques dans des boîtes, un entrepôt, etc. boîtes, 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 décision 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 [13] 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 [14]. 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 de boîtes de longueurs différentes dans le moins de bins possible de longueur fixée.

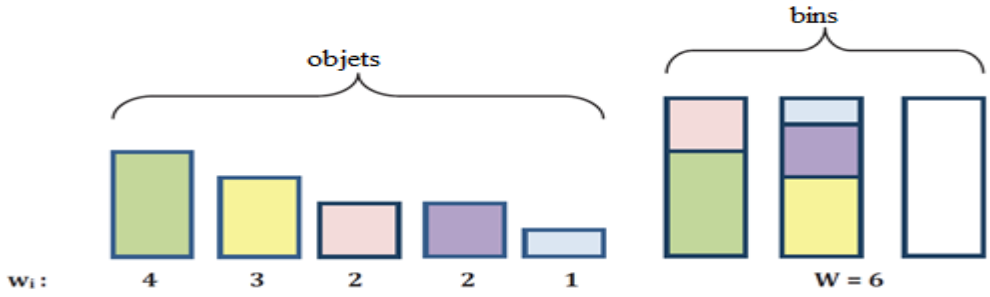


Figure 4:2.2.une instance de BPP-1D et une solution possible

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 **[11]**.

Exemple

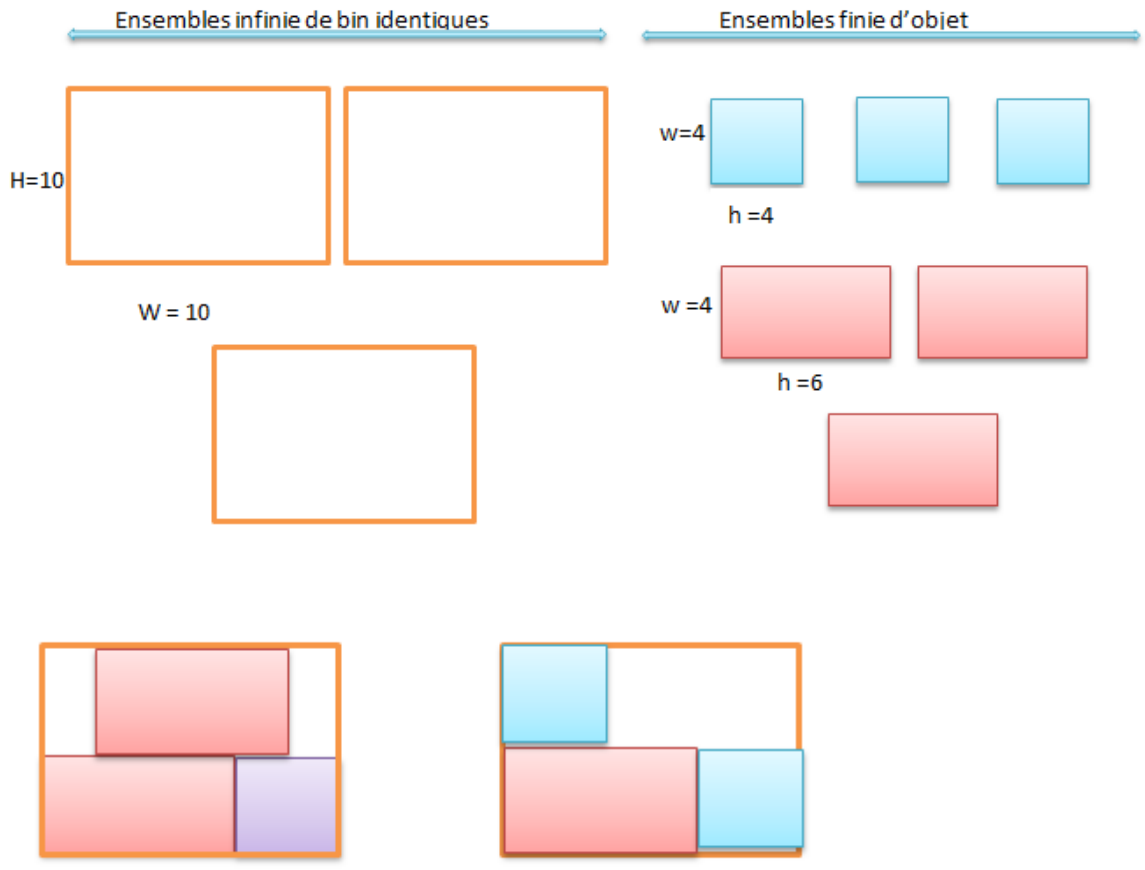


Figure 5:2.3.Une instance de BPP-2D et une solution possible

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.

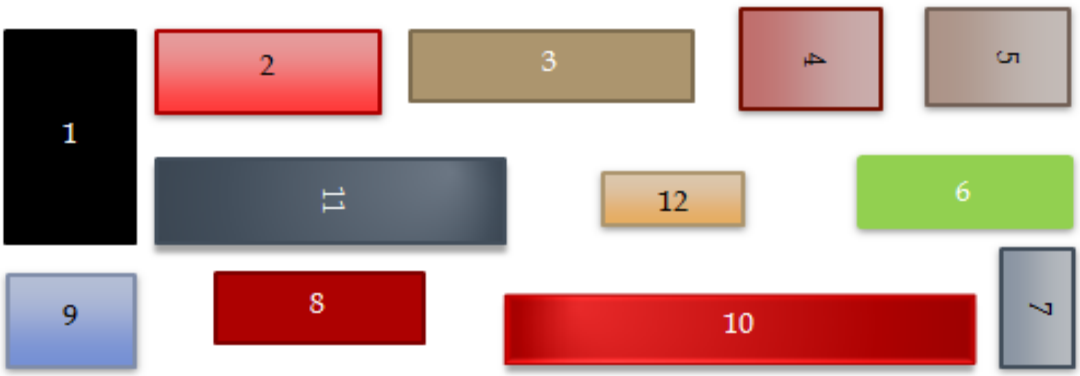
Le problème est trivial en une dimension (il suffit de sommer la longueur des objets) et NP-complet en deux dimensions.

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

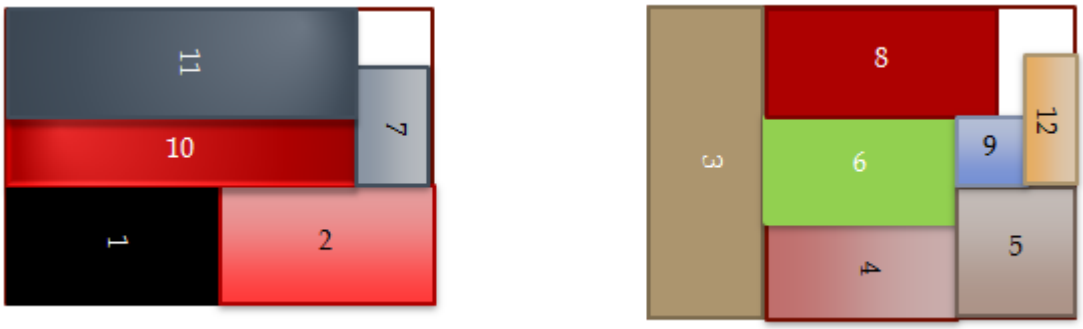


Figure 6:2.4.Bin packing le cas non orienté

2éme cas : solution avec orientation

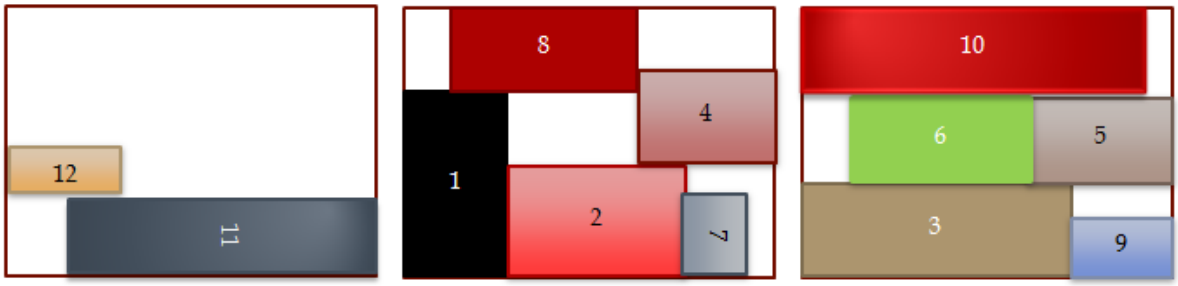


Figure 7:2.5.Bin packing le cas orienté

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 conteneurs, est un problème combinatoire dont dérive de nombreuses applications industrielles [14]. 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 [16].

Exemple

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

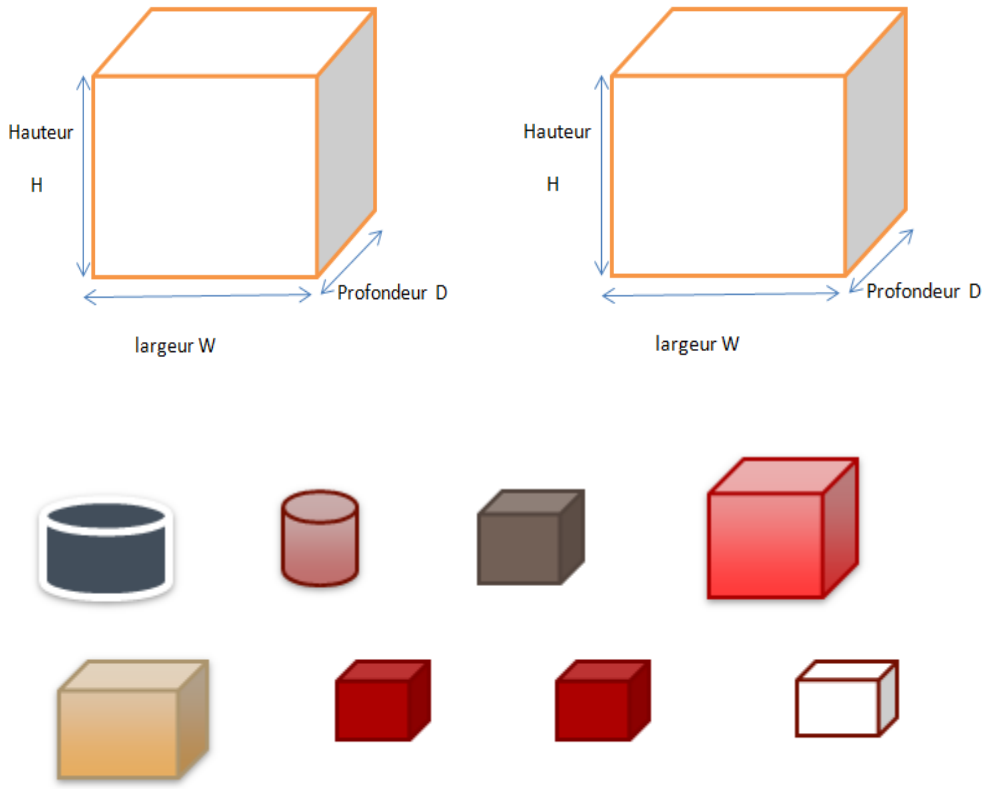


Figure 8:2.6.Illustration graphique du BPP-3D

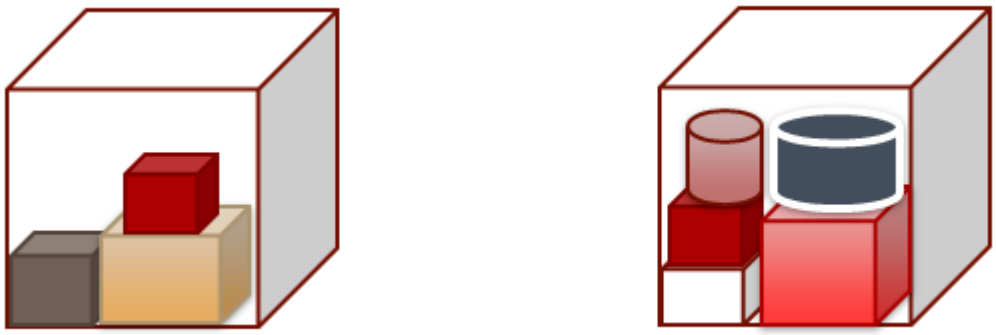


Figure 9:2.7.Illustration d'une solution réalisable pour BPP-3D

6. Formulations mathématiques

Dans cette section, nous présentons des formulations mathématiques pour le BPP-1D

6.1. Cas uni – dimensionnel

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

- un ensemble de n objets qu'on appelle bin.

$$B = B_1, B_2, \ldots, B_j, \ldots, B_n$$

- un ensemble de n objets $Y = Y_1, Y_2, Y_i, \ldots, Y_n$.
- W_j : La capacité de bin B_j. (j=1,2,…n)
- w_i : La taille de l'objet Y_i. (i=1,2,…n)
- $x\text{ij} = \begin{cases} 1 & si\ l'objet\ Y_i\ est\ range\ dans\ le\ bin\ B_j \\ 0 & si\ l'objet\ Y_i\ ne\ peut\ etre\ range\ dans\ le\ bin\ B_j \end{cases}$
- $b_j = \begin{cases} 1 & si\ le\ bin\ B_j\ est\ utilisée \\ 0 & si\ le\ bin\ B_j\ n'est\ utilisée \end{cases}$

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

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

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

On suppose que les bins ont la même taille, c'est-à-dire :

$$\forall_j, k \in N, W_j = W_k$$

• Contraintes du problème:

1. Un Objet est placé uniquement dans une seule Boîte.

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

$x$ij =1 impose à tous les objets d'être rangés dans un seul bin.

2. La taille de l'ensemble des objets rangés dans une boite B_j ne doit pas dépasser la longueur de ceci.

$$\sum_{i=1}^{n} w_i * x_{ij} \leq W$$

Remarque :

Toute solution pour laquelle les deux équations sont vérifiées est dite réalisable.

Le modèle est donc le suivant :

$$\begin{cases} \text{Min} \sum_{j=1}^{n} b_j \\ \sum_{j=1}^{n} x_{ij} = 1 \\ \sum_{i=1}^{n} w_i * x_{ij} \leq W \\ x_{ij} \epsilon \{0,1\} \\ i = 1, \dots, n \, , j = 1, \dots, n \end{cases}$$

La modélisation décrite plus haut a été proposée par Leonid Kantorovitch en 1960 [17].

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 PBB 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.

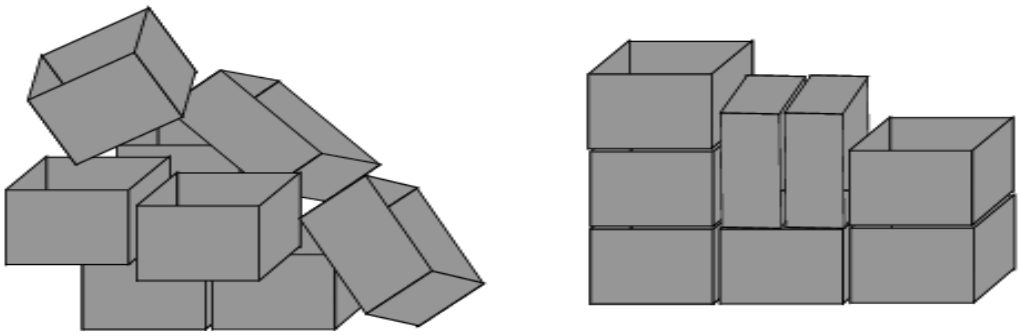


Figure 10:3.1.Exemple du problème de placement

Exemple

Supposons que nous avons des bins (récipients) de taille 10 et Supposons que les objets sont numérotée de 1 à 10.

Quel nombre de bins nécessaires pour ranger les objets de poids suivant :

objet	1	2	3	4	5	6	7	8	9	10
poids	3	6	2	1	5	7	2	4	1	9

Table 1:3.1.les objets et ses poids

Solution : La solution est utiliser 4 bins

- premier bin : objets 2 et 8
- deuxième bin : objets 4,10
- troisième bin : objets 1,6
- quatrième bin : objets 3, 5, 7,9

3. Quelques exemples d'application du problème de placement

- Sauvegarde des données sur un fichier informatique

On souhaite sauvegarder le contenu disque dur sur des CD-Rom de 700 Mo. Chaque dossier a une taille donnée, et on ne souhaite pas découper les dossiers. Comme il ne nous reste plus beaucoup de CD, on cherche le minimum de disques qu'on peut utiliser, notez bien que dans cette version du problème, on ne cherche pas à regrouper spécialement des fichiers par genre dans ce cas on a un problème de placement en une dimension à résoudre.

- Découpe de bois

Un menuisier a reçu une commande d'un ensemble d'étagères de tailles différentes. Dans le but de répondre à cette commande au moindre coût, le menuisier voudrait utiliser le minimum de planches possibles pour découper les rectangles ayant la même largeur nécessaires pour fabriquer les étagères. Il s'agit d'un problème de placement à une dimension.

- Organisation d'une fête

On a une fête à organiser, et on doit inviter un certain nombre de familles, chaque famille est composée d'un ou plusieurs personnes, comme il ne nous reste pas beaucoup d'argents, on cherche le minimum de tables à utiliser pour ce dîner. Sachant qu'on ne peut pas séparer une personne de sa famille.

- Réalisation des taches

On doit réaliser un ensemble de taches à faire, chaque tache doit être exécutée dans un temps quelconque. Des machines ont utilisées pour réaliser ces taches dans un délai fixé. On a besoin de minimiser les machines utilisées sans passer le délai.

4. Classification des problèmes de placement

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

4.1. 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**).

4.2 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 [18].

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

.La valeur optimale du nombre de boites 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.

4.3 Problème de placement à trois dimensions :

Dans ce cas les données sont les suivantes :

Il existe N objets de volume $v_i$et M boites 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 11:3.2.Le placement en 3D

5. 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.

5.1 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].

5.1.1 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].

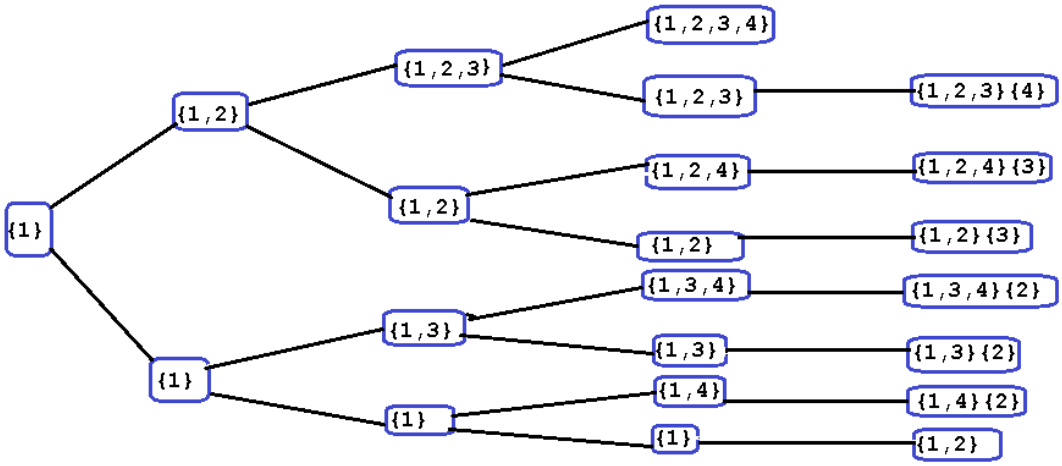


Figure 12:3.3.l'exploration des solutions dans un arbre de recherche.

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].

5.2 Heuristiques de résolution pour le problème de Placement en une dimension

Comme dans tous les problèmes combinatoires, des méthodes heuristiques ont été proposées pour le problème de placement dans le but de trouver des solutions de bonne qualité dans un temps raisonnable sans garantie d'optimalité.

Heureusement pour le problème en une dimension on trouve des algorithmes avec garantie de performance qui sont connus sous le nom de **First-fit (FF)** ou **Best-fit (BF), Next-fit** et **Worst-fit (WF),** on parle de first-fit decreasing (FFD) ou best-fit decreasing (BFD), next-fit decreasing (NFD) et worst-fit decreasing(WFD) quand il s'agit de trier les objets à ranger dans l'ordre décroissant.

5.2.1 Stratégie Next-Fit(NF)

Dans cette stratégie on ne considère qu'une boite (bin ouverte à la fois). Les objets sont traités selon un ordre donné. Les objets sont rangés successivement dans la boite ouverte tant qu'il y a de la place pour l'objet en cours, sinon la boite en cours est fermée et une nouvelle est ouverte.

- Une heuristique qui adopte une stratégie NF à l'avantage d'avoir une complexité temporelle linéaire en fonction du nombre d'objets a placé.

Par contre, le fait de ne considérer qu'une seule boite à la fois cause beaucoup de perte d'espaces exploitables.

On parle de Next-Fit-Decreasing (NFD) quand il s'agit de trier les objets à ranger dans l'ordre décroissant de hauteurs avant de les ranger suivant une stratégie NF.

Exemple

La strategie NF est appliquée à une serie d'objets numérotée de 1à 8 .

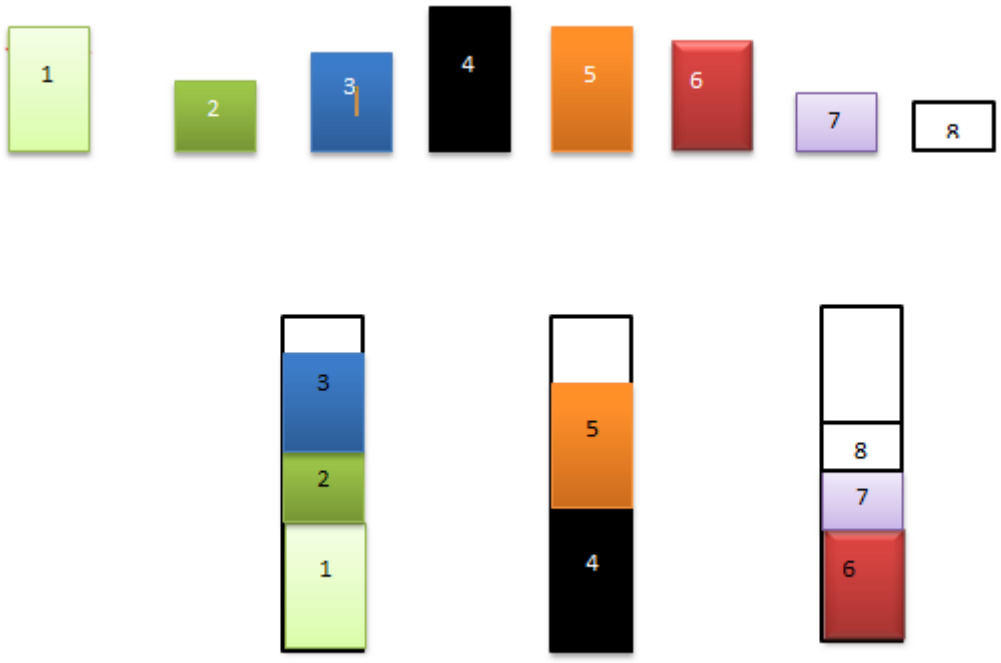


Figure 13:3.4.Exemple avec solution en stratégie NF

Next-Fit Decreasing(NFD) consiste à trier les objets par ordre décroissant de hauteurs et appliquer la stratégie NF pour les ranger

5.2.2 Stratégie First Fit (FF)

Initialement une seule boîte est considérée, et les objets sont traités selon un ordre donné.

Quand il n'y a plus de la place dans la première boîte pour ranger l'objet en cours, une deuxième est alors ouvert mais sans fermer la première.

- Dans une étape intermédiaire où on dispose de k boites ouvertes numérotées de 1 à k selon l'ordre de leur première utilisation, un objet ai en cours est rangé dans la boite du plus faible numéro qui peut le contenir.
- Dans le cas où aucune boite ne peut contenir une nouvelle boite

(k + 1) est alors utilisé sans fermer les autres.

L'ordre selon lequel on traite les objets est crucial pour la qualité de la solution. Un choix heuristique consiste à trier les objets par ordre décroissant de hauteurs, on parle dans ce cas d'heuristiques First- Fit Decreasing

Exemple

La stratégie de FF est appliquée à une série d'objets numérotée de 1 à 8

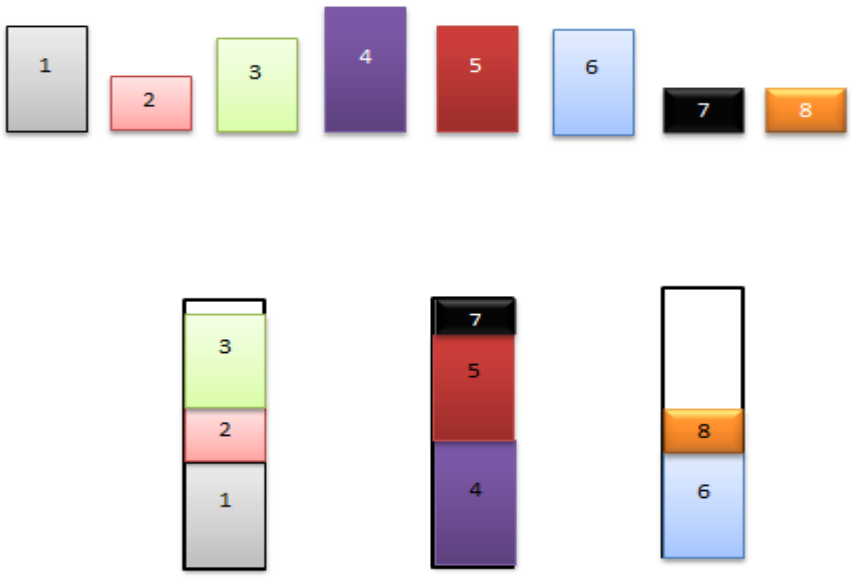


Figure 14:3.5.Exemple avec solution en stratégie FF

First-Fit Decreasing (FFD) est un choix heuristique qui consiste à trier les objets par ordre décroissant de hauteurs.

5.2.3 Stratégie Best Fit (BF)

Comme dans la stratégie FF, les algorithmes BF laissent les boites toujours ouvertes.

Cependant, le choix de la boite dans laquelle l'objet ai en cours va être placé dépend des valeurs des gaps (espace libre restant dans la boite) (hauteurs non utilisées) présentes dans les boites. Ainsi, sera placé dans la boite qui présente le moindre gap parmi les boites qui peuvent le contenir.

- Les heuristiques BF et FF peuvent être implantées en O (nlog (n)) en utilisant une structure de données appropriée Johnson (1973).
- On parle de Best-Fit-Decreasing (BFD) quand il s'agit de trier les objets à ranger dans l'ordre décroissant de hauteurs avant de les ranger suivant une stratégie BF.

Exemple

La stratégie BF est appliquée à une série d'objets numérotée de 1 à 6

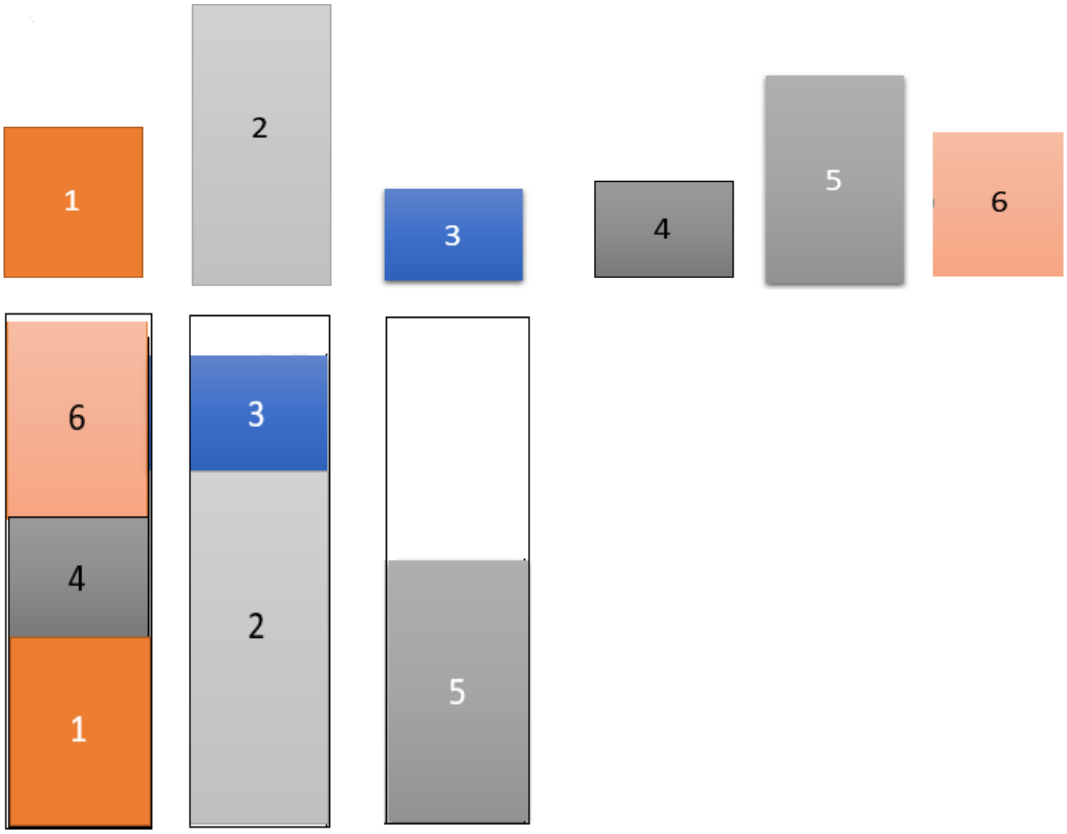


Figure 15:3.6.Exemple avec solution en stratégie BF

Le Best-Fit Decreasing (BFD) consiste à trier les objets à ranger dans l'ordre décroissant de hauteurs avant de les placer selon la stratégie BF

5.2.4 Stratégie de Worst-Fit

Cette approche est similaire à Best Fit mais on place un élément dans une des boites ouvertes pour que le reste de la capacité disponible de la boite soit maximum.

- on parle de worst fit decreasing (WFD) quand il s'agit de trier les objets à ranger dans l'ordre décroissant.

Remarques

Ces algorithmes ne sont pas optimaux, mais ils permettent d'obtenir de très bons résultats en pratique.

- ✓ pour calcule le nombre de boites optimales (OPT) nécessaires pour placer les objets en sommant la taille de tous les objets divisés par la capacité de bin.
- ✓ Les algorithmes Best Fit Decreasing et First Fit Decreasing n'utilisent jamais plus ($\lceil OPT \rceil + 1$) bins.
- ✓ Une version plus efficace de FFD utilise au plus (71/60 **OPT** + 1) bins.
- ✓ Dans ce travail nous vous a choisi la premier heuristique Best Fit BF et Best Fit Decreasing BFD sont meilleurs dans l'implémentation.

A partir les stratégies de BF et BFD on donne les algorithmes suivant :

6. Algorithme de Best Fit (BF)

Début

1. lecture la capacité de bin

Lire (capacité) ;

2. lecture le nombre des objets

Lire (n) ;

3. Création des objets

Pour i de 1 à n faire

Lire (objet(i));

Fait ;

4. Initialisation la liste des bins= []

bins = []

5. initialisation l'indice des objets i et l'indice des bins k, la somme des tailles T

T ←0;

k←1;

i ←1;

6. Début de traitement de l'algorithme BF

7. Parcourt tous les objets

Tanque i<=n faire

8. Variable booléen pour vérifier dernier élément

test_dernier_élement ← faux ;

9. Boucle pour passer les objets qui on a les tailles supérieures à la capacité de bin

tanque taille (objet(i)) > capacité faire

i ← i+1;

fait ;

10. La somme des objets T pour remplir un bin

T=T+ taille(objet(i));

11. Vérification de la somme T est supérieure a la capacité

Si T>capacité alors

si T – taille (objet(i))<>0 alors

12. Vérification de la somme T avant le dernier élément si est inférieure à la capacité si oui remplir le bin

si T – taille (objet(i))<= capacité

bins[k] ← T;

k ← k+1;

T ← 0;

si i > 1 alors

i←i-1;

finsi;

finsi;

13. si T= capacité de remplir le bin

Sinon si T = capacité alors

bins[k]=T;

k ←k+1;

T ← 0;

test_dernier_element ← vrai;

finsi;

finsi;

14. Vérification que le dernier elt est acceptable

si i=n et test_dernier_element = faux et taille (objet(n)<= capacité alors

bins[k]=T;

k←k+1

finsi;

i ← i+1;

Fait ;

si k <> 1 alors

k ← k - 1;

Sinon k ← 0;

finsi;

15. Affichage

Ecrire (" nombre de bins est : ", k);

Pour i de 1 a k faire

écrire (" le bin N° ",i," = ",bins(i));

FIN

7. Algorithme de Best-Fit Decreasing (BFD)

Début

1. Lecture la capacité de bin

Lire (capacité);

2. Lecture le nombre des objets

lire(n);

3. Création des objets

objet[n];

Pour i de 1 à n faire

lire (objet(i));

Fait ;

4. Initialisé la liste des bins=[]

bins = []

5. Initialisation l'indice des objets i et l'indice des bins, la somme des tailles T

T ← 0;

k ←1;

i ←1;

6. Début de traitement de l'algorithme BFD

7. Triage les elts en ordre décroissante avec les indices (en utilisant tri à bulles par exemple)

objet=tri décroissante(objet);

8. Parcourt tous les objets

Tanque i<=n faire

9. Variable booléen pour vérifier dernier élément

test_dernier_element ← faux ;

10. Boucle pour passer les objets qui ona les tailles supérieures a la capacité de bin

tanque taille (objet(i)) > capacité faire

i ← i+1;

Fait ;

11. La somme des objets T pour remplir un bin

T=T+ taille(objet(i));

12. Vérification de la somme T est supérieure à la capacité

Si T>capacité alors

si T - taille (objet(i))<>0 alors

13. Vérification de la somme T avant le dernier élément si est inferieure à la capacité si oui remplir le bin

si T – taille (objet(i))<= capacité

```
            bins[k] ← T;
            k←k+1;
            T ← 0;
            si i > 1 alors
               i ←i-1;
            finsi;
         finsi;
```

14. si T= capacité remplir le bin

```
   Sinon si T = capacité alors
        bins[k]=T;
        k ←k+1;
        T ←0;
        test_dernier_element ← vrai ;
        finsi;
   finsi;
```

15. Vérifier que le dernier elt est acceptable

```
si i=n et test_dernier_element = faux et taille (objet(n)<= capacité alors
 bins[k]=T;
 k← k+1
 finsi;
 i ←i+1;
Fait ;
si k <> 1 alors
   k ←k - 1;
Sinon
k ←0;
finsi;
```

16. Affichage

```
écrire ( " nombre de bins est : ", k);
Pour i de 1 a k faire
écrire ( " le bin N° ",i," = ",bins(i));
FIN.
```

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 unidimensionnel et leurs explications (First-fit ; Next-fit ;Best-fit ;Worst fit), avec des exemples d'application.

Chapitre IV

Application numérique et résultats

1. Introduction

ce chapitre est consacré à l'implémentation des résultats obtenus en utilisant le langage Matlab 8.6 soit par la programmation, soit l'utilisation de fonctions prédéfinis.

2. Matlab 8.6

Matlab 8.6 est un langage de programmation et un environnement de développement développé et commercialisé par la société américaine : Math Works

Matlab est utilisé dans le domaine de recherche et de l'industrie pour permet le calcul numérique mais aussi dans les phases de développement de projets. C'est un outil très pratique et vaste d'utilisation. Il est doté de nombreuses boites à outils (toolbox) ; incluant les possibilités données par d'autre langage de programmation comme C^{++}, Fortran ou java

3. Principe d'utilisation

Pour la détermination les paramètres suivants :

1. b_j : l'ensemble des bins
2. W : la capacité de bin.
3. w_i : La taille des objets

Algorithme BF La procédure de classements et placements

Entrée : W , w_i

Sorties :- le nombre de bin

- tableau de classification des objets dans chaque bin
- la somme des tailles d'objet dans le bin
- la représentation graphique des résultats.

Algorithme BFD La procédure de classements et placements

Entrée : W , w_i

Sorties :- L'ordre décroissante des objets

- Le nombre de bin
- Tableau de classification des objets dans chaque bin
- la somme des tailles d'objet dans le 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 les capacités de bins est 500 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 :

$$\begin{cases} \mathbf{Min} \sum_{j=1}^{n} b_j \\ \sum_{j=1}^{n} x_{ij} = 1 \\ \sum_{i=1}^{n} w_i * x_{ij} \leq W \\ x_{ij} \epsilon \{0, 1\} \\ i = 1, \dots, n \, , j = 1, \dots, n \end{cases}$$

La taille des objets donnés par le tableau suivant :

Objet n° :	1	2	3	4	5	6	7	8	9
Tailles	456	425	499	485	123	452	223	226	250

Objet n° :	10	11	12	13	14	15	16	17	18
Tailles	450	469	458	496	459	478	456	485	496

Table 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 MATLAB 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 **Best-Fit** et **Best-Fit Decreassing.**

4.1.1. Résolution du problème en appliquant l'heuristique de Best-Fit

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

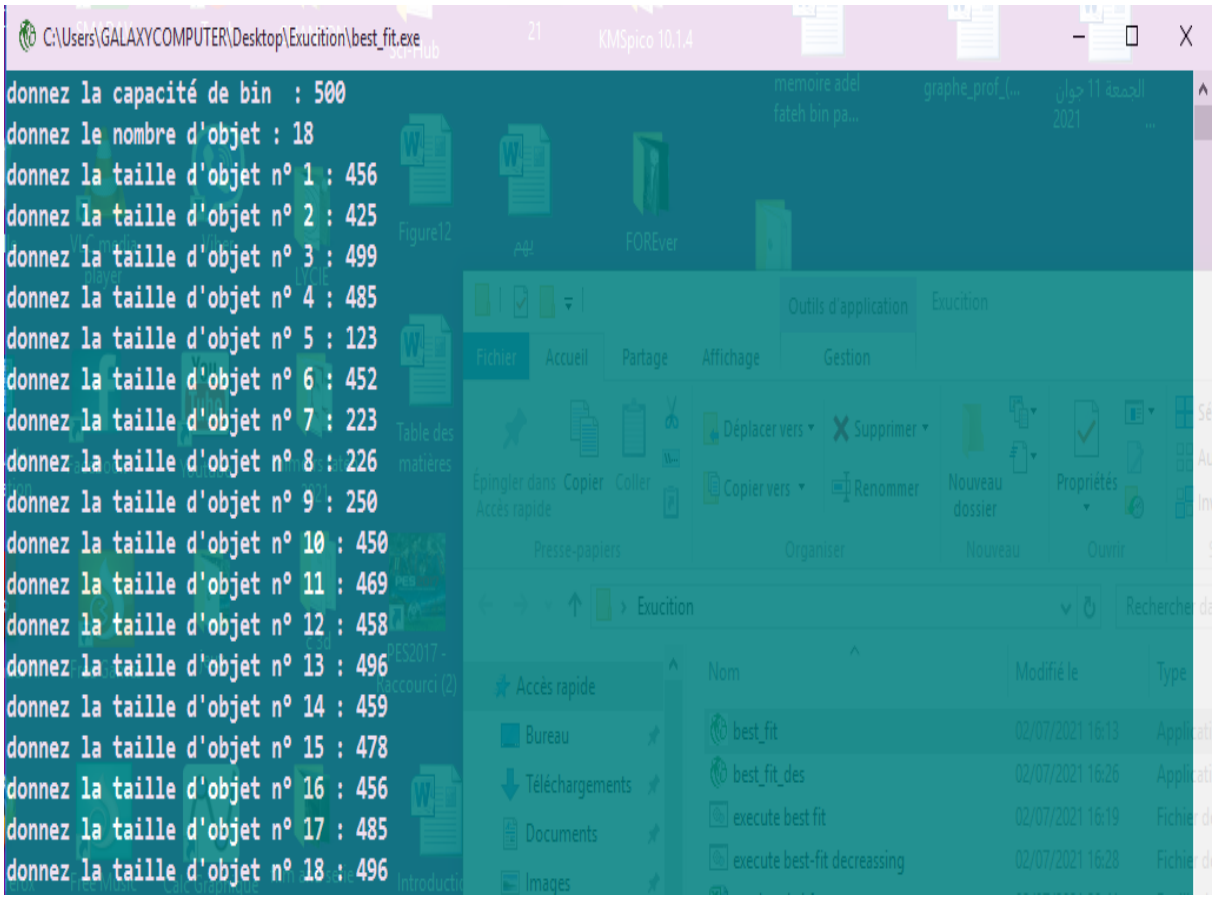


Figure 16:4.1.Création d'objets pour La méthode BF

La figure suivant donne Les résultats obtenus avec BF :

-La classification des objets dans les bins .

-Le nombre de bins obtenus par la méthode BF.

-la somme des tailles des objets utilisée dans chaque bins.

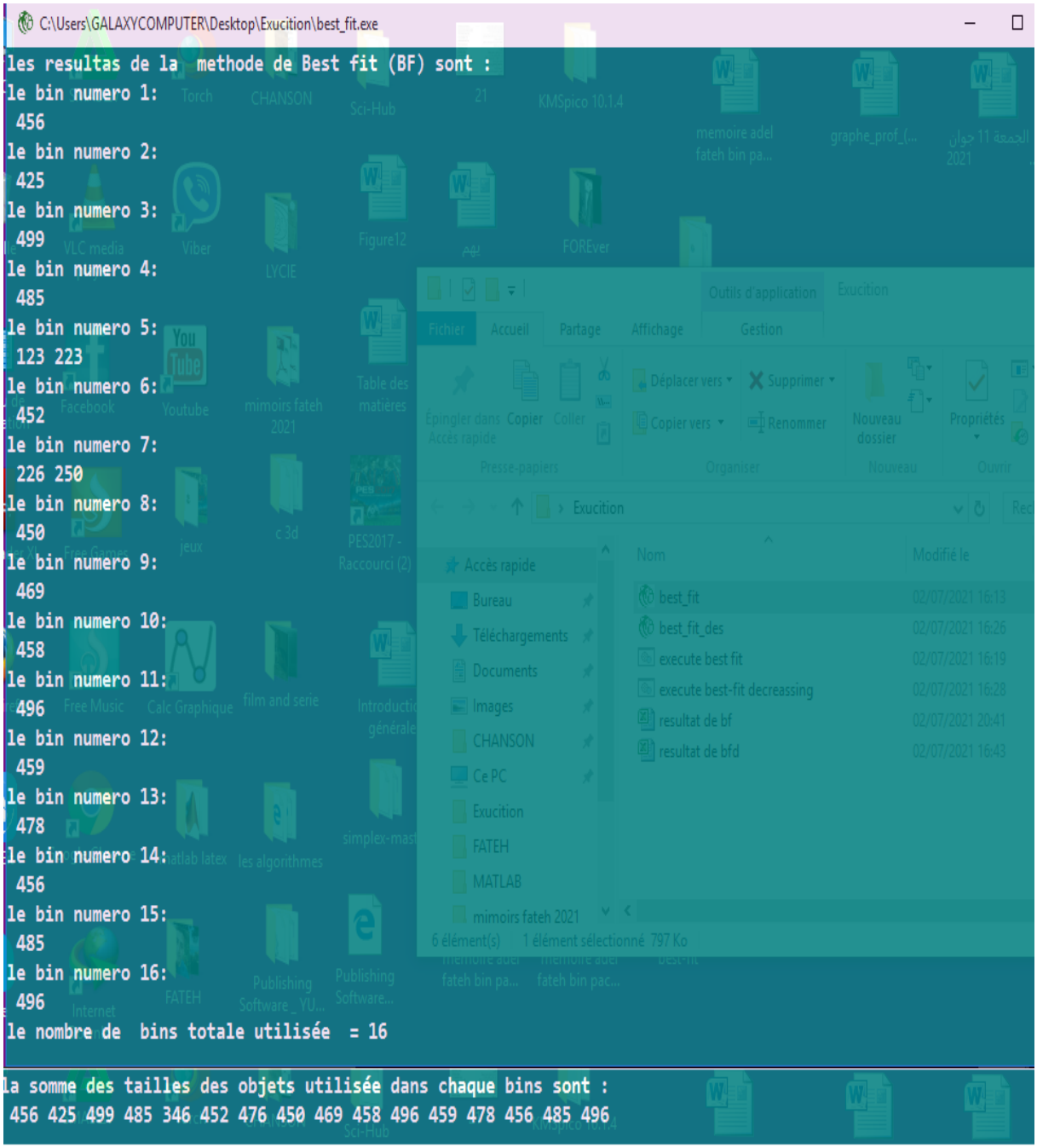


Figure 17:4.2.Les résultats obtenus avec BF

Résultat numérique

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

Bins	Bin 1	Bin 2	Bin 3	Bin 4	Bin 5	Bin 6	Bin 7	Bin 8	Bin 9	Bin 10	Bin 11	Bin 12	Bin 13	Bin 14	Bin 15	Bin 16
objets	456	425	499	485	123	452	226	450	469	458	496	459	478	456	485	496
					223		250									

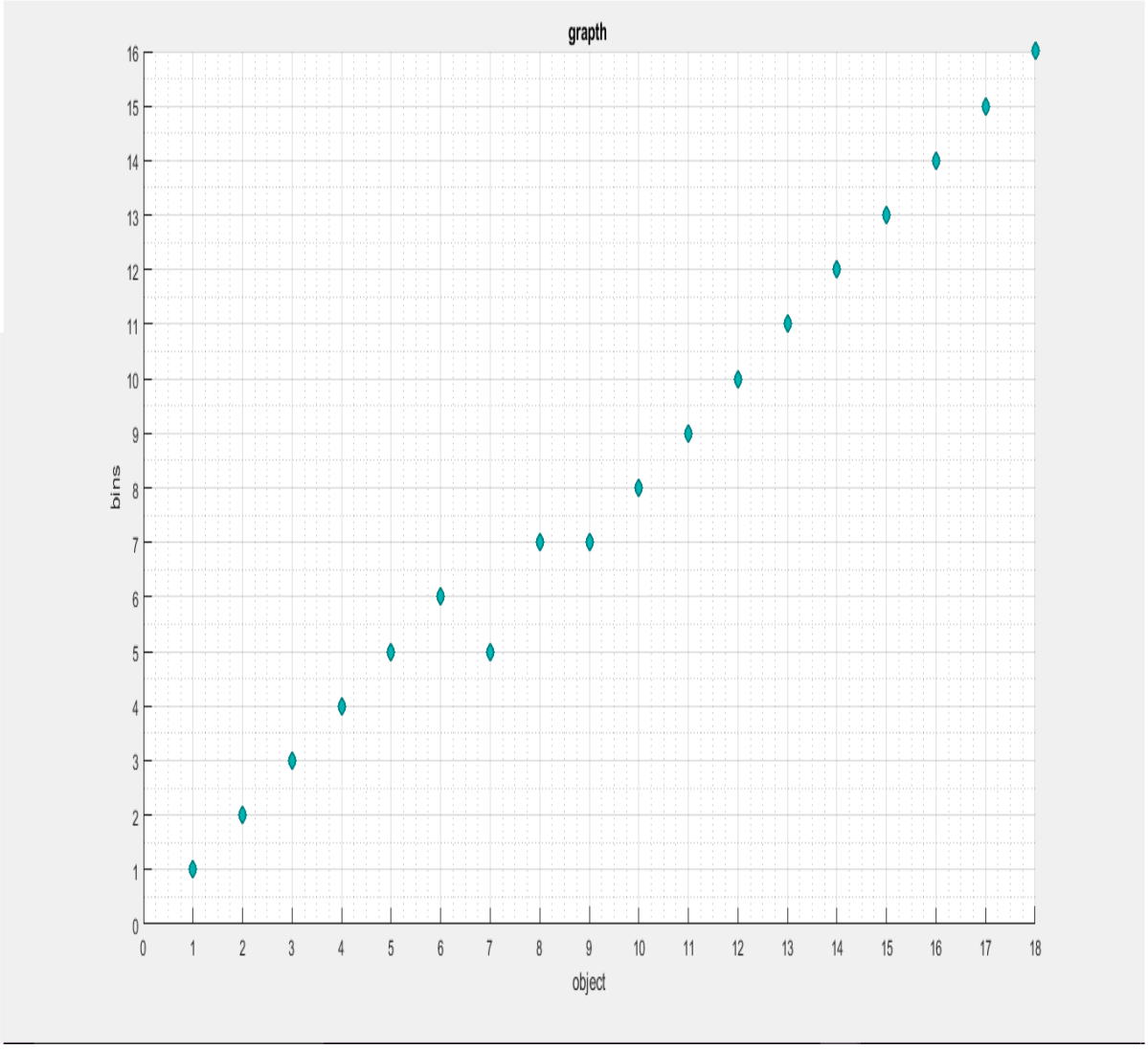


Figure 18:4.3.la représentation graphique du résultat avec BF

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

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

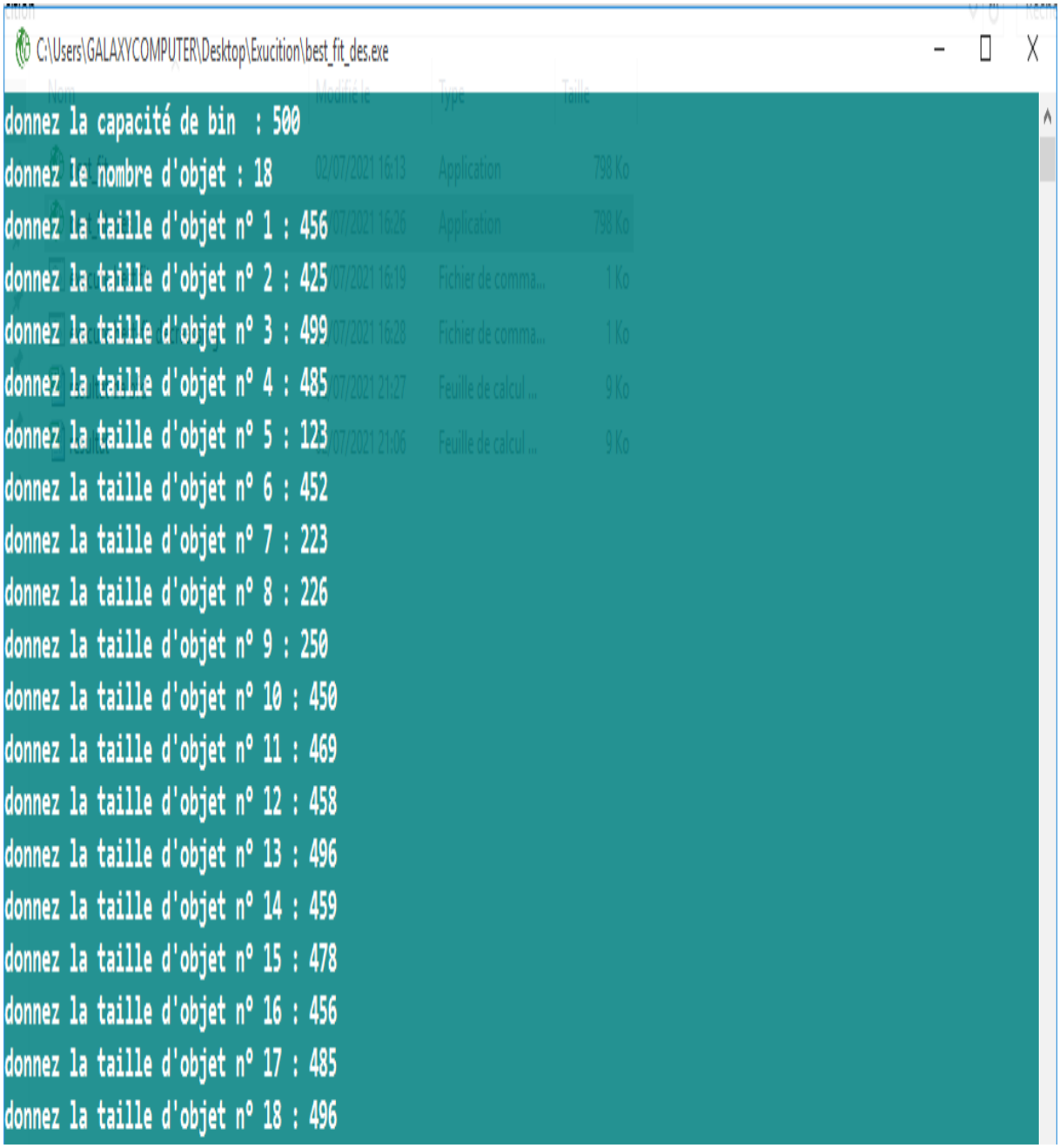


Figure 19:4.4.création d'objets pour méthode BFD

La figure suivante donne :

-l'ordre des objets (ordre décroissante).

-La répartition des objets par les bins.

-Nombre des bins utilisée.

-La taille totale de chaque bins.

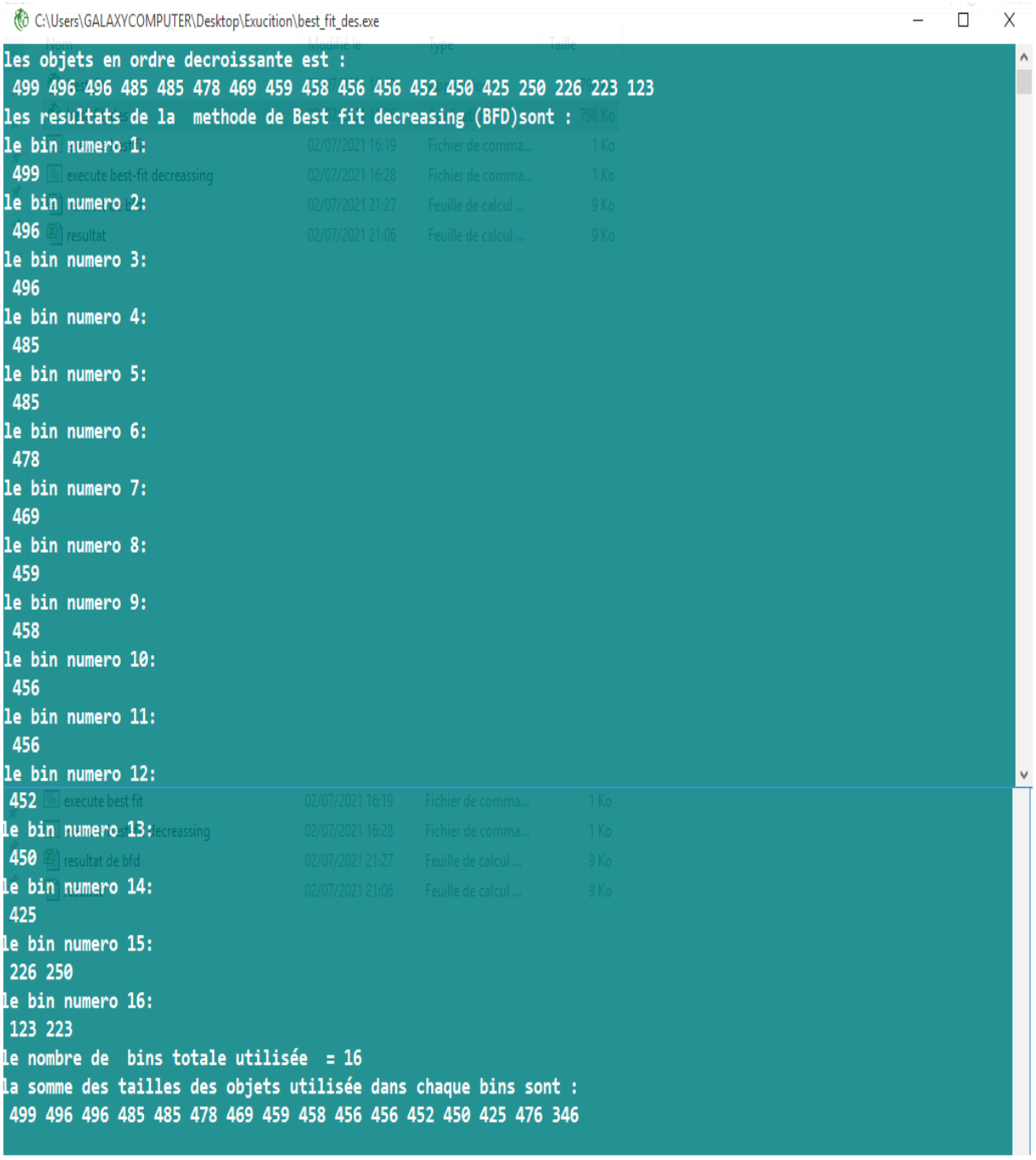


Figure 20:4.5.Les résultats obtenus avec BFD

La figure donne :

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

Bins	Bin1	Bin 2	Bin 3	Bin 4	Bin 5	Bin 6	Bin 7	Bin 8	Bin 9	Bin 10	Bin 11	Bin 12	Bin 13	Bin 14	Bin 15	Bin 16
Objets	499	496	496	485	485	478	469	459	458	456	456	452	450	425	226	129
															250	223

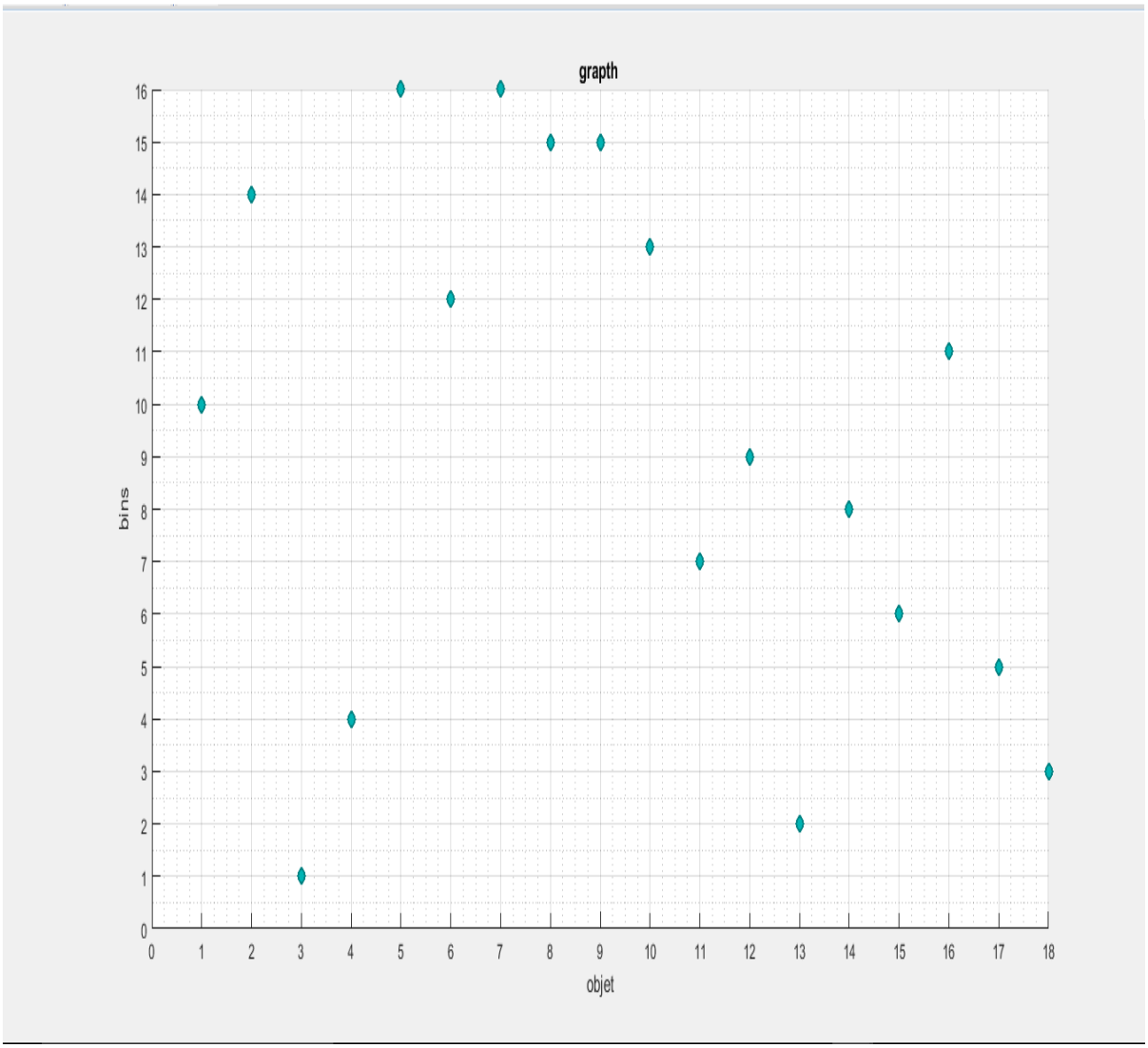


Figure 21:4.6.la représentation graphique du résultat avec BFD

4.2. Exemple d'application 2 :

Je dispose de données informatique de différents tailles que je veux mettre sur CD de taille 1024mo. J'essaie donc d'optimiser la place afin d'utiliser le moins de CD possible.

Données :

Objet n°	1	2	3	4	5	6	7	8	9
Taille (mo)	129	259	556	932	756	852	794	835	158

Objet n°	10	11	12	13	14	15	16	17	18
Taille (mo)	294	225	562	256	523	961	352	522	722

Objet n°	19	20	21	22	23	24	25	26	27
Taille (mo)	888	940	137	164	253	364	372	388	460

Objet n°	28	29	30	31	32	33	34	35	36
Taille (mo)	432	461	132	234	345	456	566	678	578

Objet n°	37	38	39	40	41	42	43	44	45
Taille (mo)	895	258	369	753	258	852	654	562	895

Table 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 MATLAB fonctionne sans problème.

4.2.1. Résolution par l'heuristique BF :

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

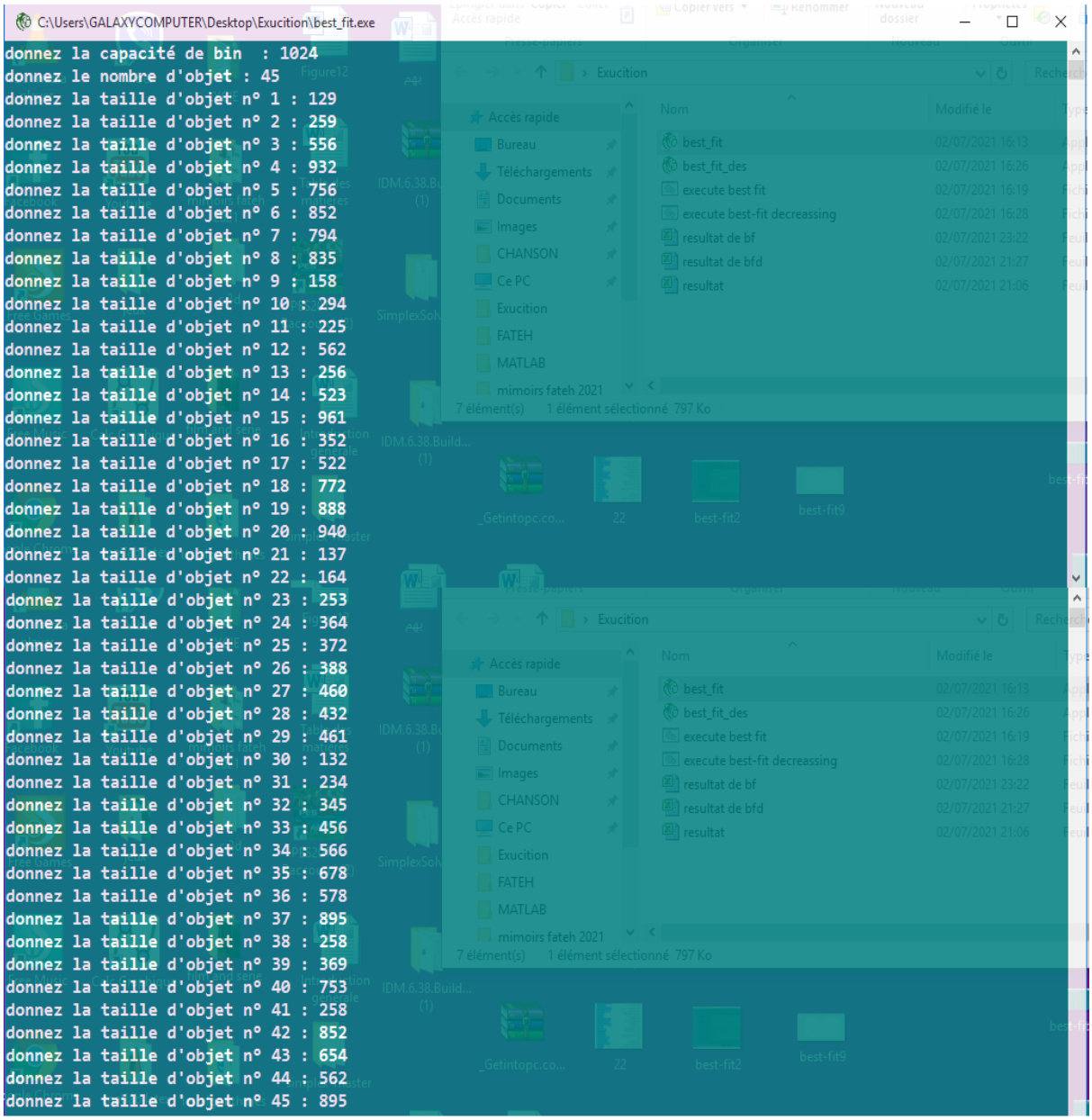


Figure 22:4.7.création d'objets pour méthode BF

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

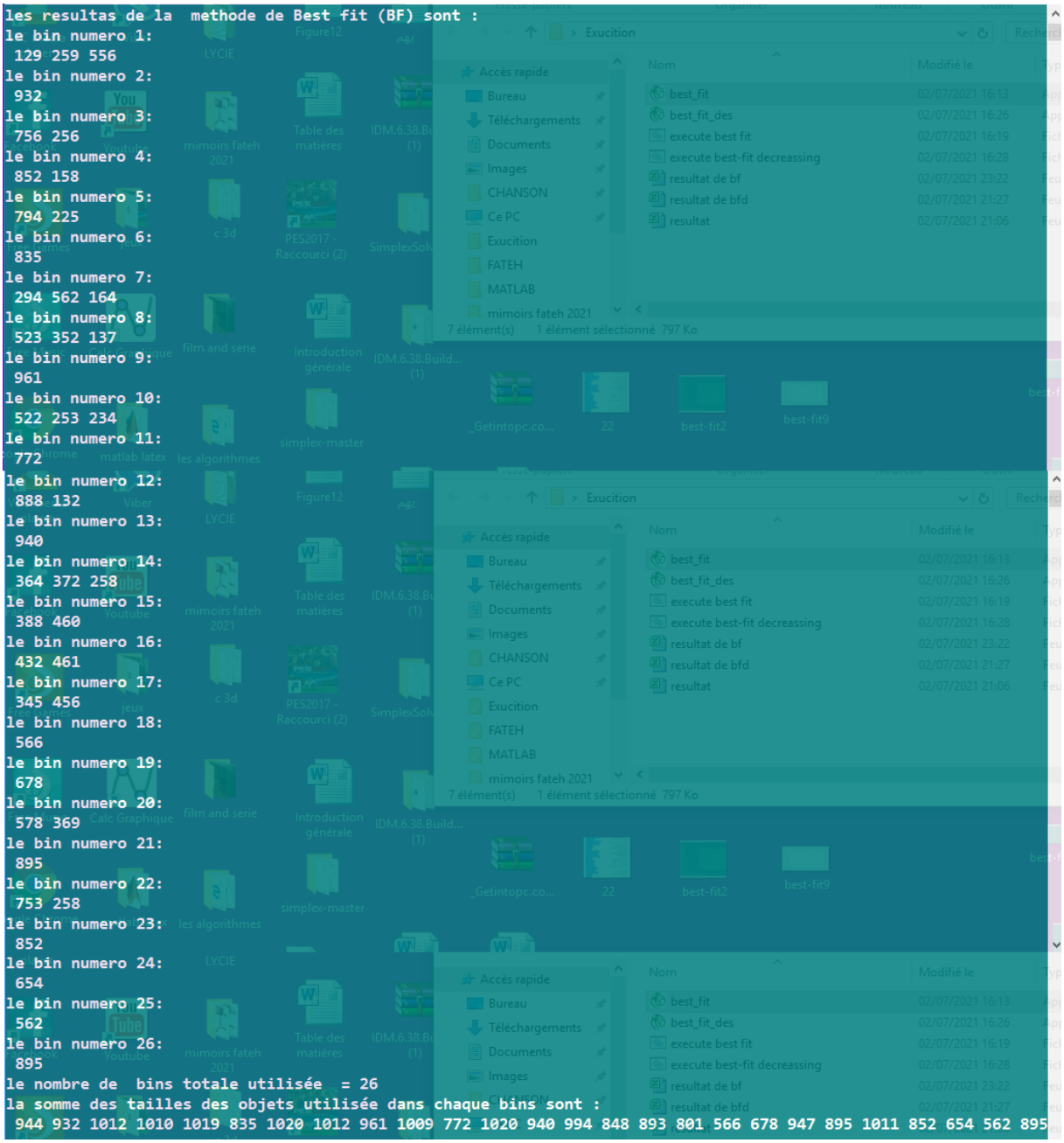


Figure 23:4.8.Les résultats obtenus avec BF

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

bins	objet		
bin 1	129	259	556
bin 2	932		
bin 3	756	256	
bin 4	852	158	
bin 5	794	225	
bin 6	835		
bin 7	294	562	164
bin 8	523	352	137
bin 9	961		
bin 10	522	253	234
bin 11	772		
bin 12	888	132	
bin 13	940		
bin 14	364	372	258
bin 15	388	460	
bin 16	432	461	
bin 17	345	456	
bin 18	566		
bin 19	678		
bin 20	578	369	
bin 21	895		
bin 22	753	258	
bin 23	852		
bin 24	654		
bin 25	562		
bin 26	895		

Table 4:4.3.Les résultats obtenus avec BF

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

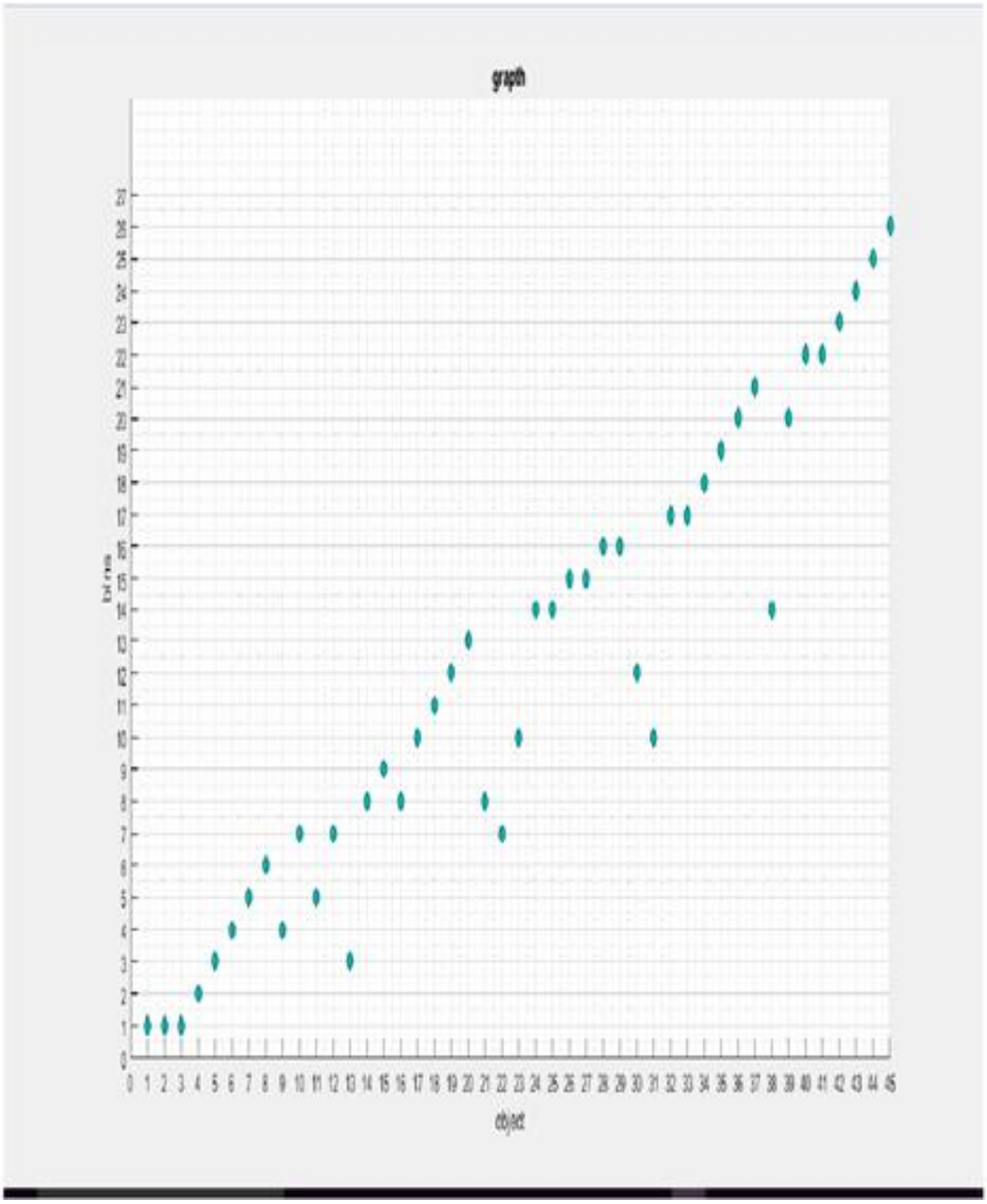


Figure 24:4.9.la représentation graphique du résultat avec BF

4.2.2. La Résolution de problème par la méthode BFD

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

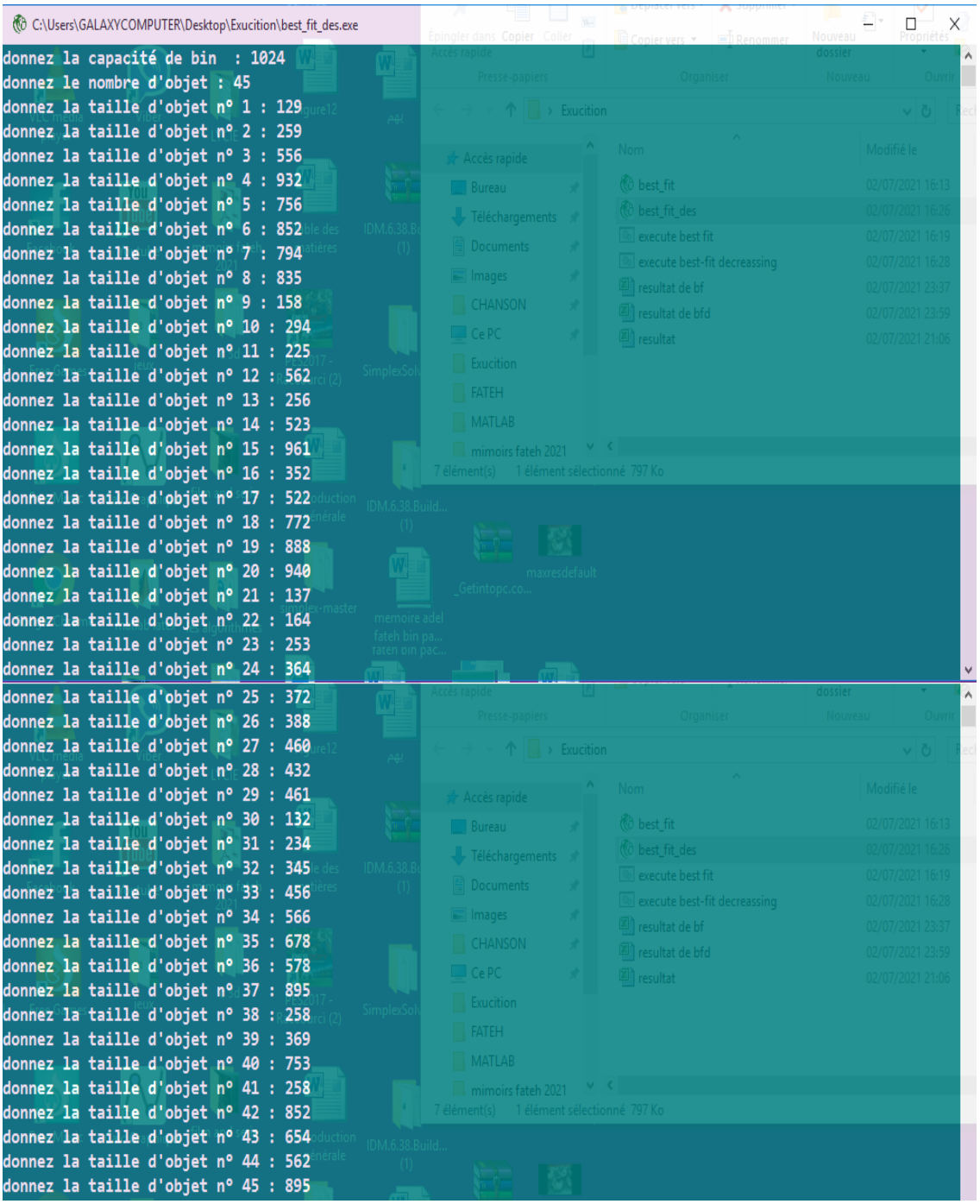


Figure 25:4.10.création d'objets pour méthode BFD

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

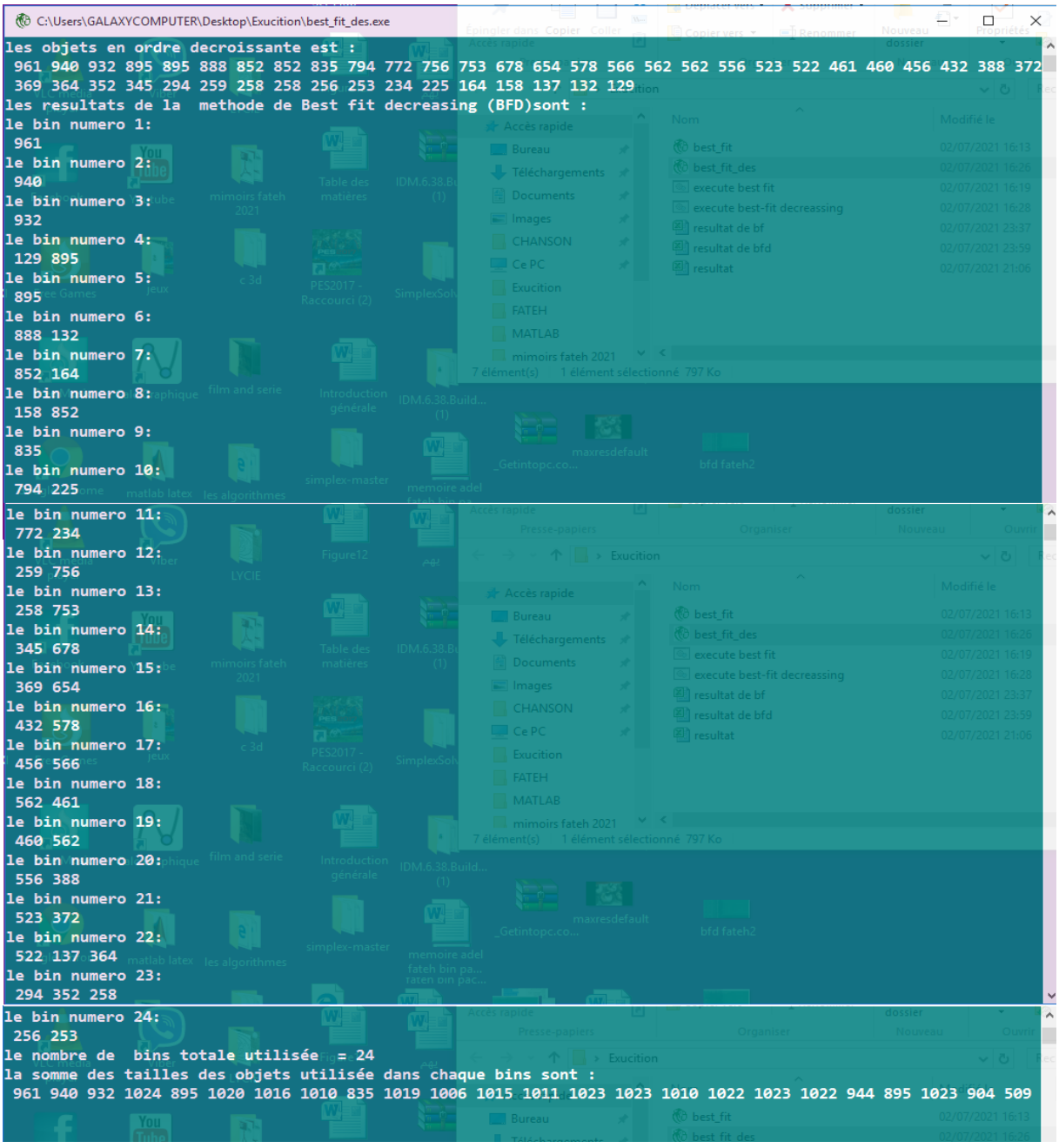


Figure 26:4.11.Les résultats obtenus avec BFD

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

bins	objet		
bin 1	961		
bin 2	940		
bin 3	932		
bin 4	129	895	
bin 5	895		
bin 6	888	132	
bin 7	852	164	
bin 8	158	852	
bin 9	835		
bin 10	794	225	
bin 11	772	234	
bin 12	259	756	
bin 13	258	753	
bin 14	345	678	
bin 15	369	654	
bin 16	432	578	
bin 17	456	566	
bin 18	562	461	
bin 19	460	562	
bin 20	556	388	
bin 21	523	372	
bin 22	522	137	364
bin 23	294	352	258
bin 24	256	253	
bin 25			
bin 26			

Table 5:4.4.Les résultats obtenus avec BFD

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

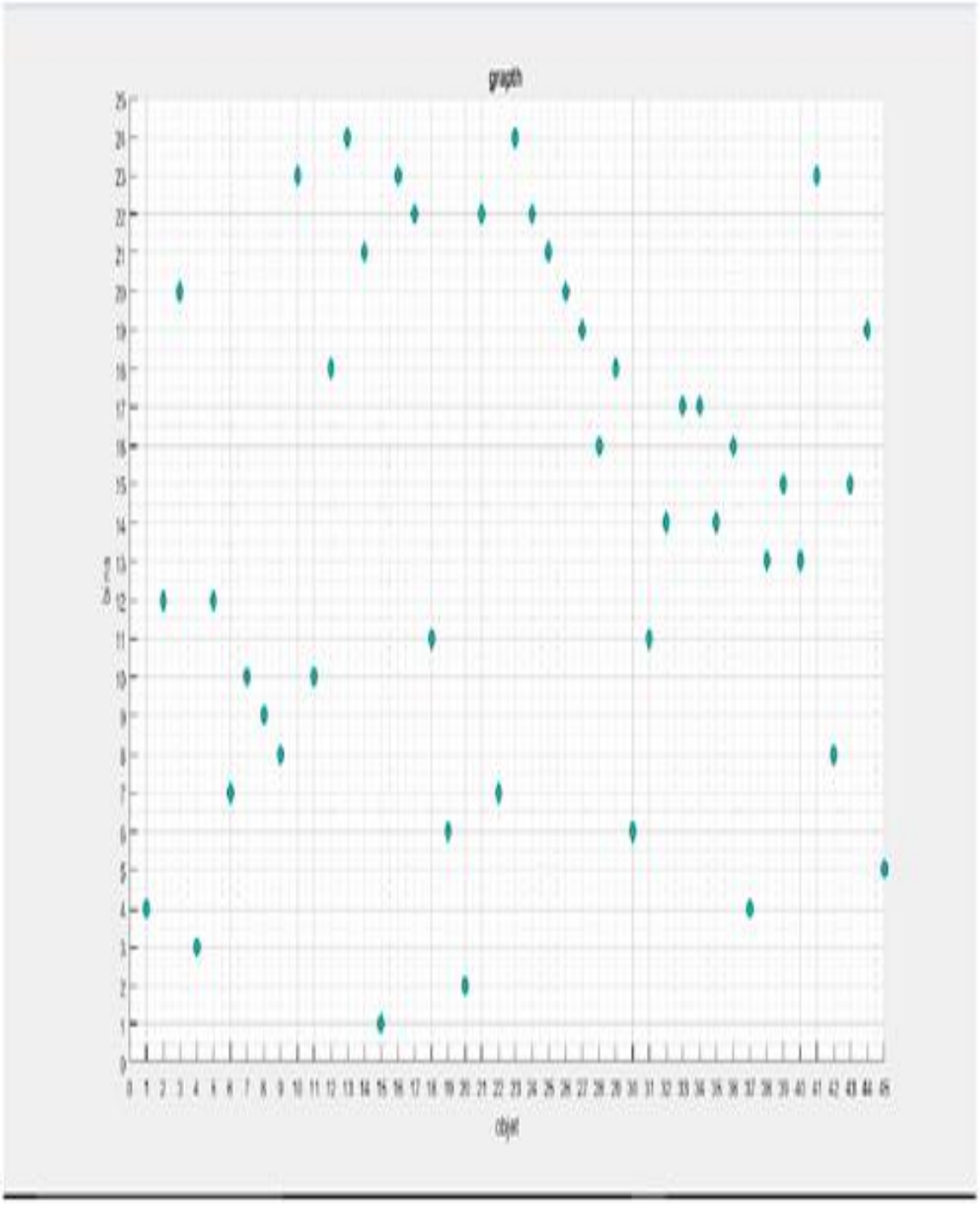


Figure 27:4.12.la représentation graphique du résultat avec BFD

5. Simulation

Dans cette partie on essayera de voir le comportement de l'heuristique BF et BFD, 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 BF	53	253	503	1003	1503	2003	2503	5003
N° du bin avec BFD	49	245	490	978	1468	1956	2446	4890

Table 6:4.5.les résultats de simulation

Conclusion générale

Nous avons traité dans ce mémoire le problème de placement unidimensionnel, 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 Best Fit et Best Fi Decreasing, 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 BFD meilleure que l'heuristique de BF.

Références bibliographiques

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

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

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

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

[5] **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

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

[7] **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

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

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

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

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

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

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

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

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

[16] **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.

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

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

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

Annexe

Code Matlab de l'algorithme BFD

```
% Algorithme de best-fit decreassing pour le probléme de placement 1D
% u(1:n): size of n objects,
% C: capacity of each bin
% w(n, n): assignment matrix. w(i,j)= 1 if u(i) is assigned to b(j)
% r(1:n): residue capacity, initially, r = C*ones(1,n);
clear all
clc
system('@title Algorithme de best-fit decreassing');
% création des objets
C=input('donnez la capacité de bin  : ');
n=input('donnez le nombre d''objet : ');
% C = 1024;
% u = [523 961 352 522 772 888 940 137 164 253 364 372 388 460 432 461 132
234 345 456 566 678 578 895 258 369 147 159 753 258 852 654];  n = length(u);
for iii=1:n
 u(iii)=input(sprintf('donnez la taille d''objet n° %d : ',iii));
end
r = C*ones(1,n);
w = zeros(n);
% this part of statements will be used by both FFD and BFD algorithms
[ud1,idec]=sort(-u);% sort obj. in decreasing size order
ud=-ud1;
[tmp,irec]=sort(idec);
%disp(irec);% irec is the recovery ordering
wbfd=w; rbfd=r; % initialize for best fit method
disp('les objets en ordre decroissante est : ');
fprintf('%s\n',sprintf(' %d',ud))
```

```
%disp(ud);
for i=1:n,
   % disp(['u(' int2str(i) ') = ' int2str(ud(i))]);
   % disp('r = '); disp(r);
 %disp([ud(i)*ones(1,n) <= rbfd]);
   idx1=find([ud(i)*ones(1,n) <= rbfd]); % indices of bins still have spaces left
   %disp(idx1);
  % disp('contune ..');pause;
   [tmp,idx2]=min(rbfd(idx1)-ud(i));    % find the index within idx1 that best fits
ud(i)
   wbfd(i,idx1(idx2))=1; rbfd(idx1(idx2))=tmp; % update the state
 % disp(tmp);
end
wbfd=wbfd(irec,:); % sort it back to the original assignment order
disp('les resultats de la  methode de Best fit decreasing (BFD)sont : ')
%disp('the assignment are:')
%disp(wbfd)
wwww=u*wbfd;
sizo=sum([wwww > 0]);
%disp(dddddd);
tt=zeros(1,n);
X=[];
Y=[];
k=1;
matrix=zeros(n,sizo);
awww={};
for i=1:sizo

    str1='le bin numero ';
   str2=int2str(i);
   str3=': ';
   s=[str1 str2 str3];
```

```
   disp(s);
    awww{1,i}=s;
   k=1;
    tt=zeros(1,n);
    for j=1:n

    if (wbfd(j,i)==1)
tt(k)=u(j);
    matrix(k,i+1)=u(j);
    X=[X,j];
    Y=[Y, i];
    awww{k,i}=u(j);
     k=k+1;
    end
    end
    tt(tt==0) = [];
  fprintf('%s\n',sprintf(' %d',tt))
end
if exist('resultat de bfd.xlsx', 'file')==2
  delete('resultat de bfd.xlsx');
end
filename = 'resultat de bfd.xlsx';
if ~isempty(awww)
xlswrite(filename,awww);
end
%ww='';
%for h=1:sizo
 %   ww=strcat(ww,'b°',int2str(h),{'  '});
%end
%disp(ww);
%disp(matrix);
```

```
occpbfd=u*wbfd;
nbinbfd=sum([occpbfd > 0]);
disp(['le nombre de  bins totale utilisée  = ' int2str(nbinbfd)]);
disp('la somme des tailles des objets utilisée dans chaque bins sont :');
fprintf('%s\n',sprintf(' %d',occpbfd(1:nbinbfd)))
%disp()
sz = 40;
scatter(X,Y,sz,'MarkerEdgeColor',[0 .5 .5],...
            'MarkerFaceColor',[0 .7 .7],...
            'LineWidth',1.5);
  grid on;
grid minor;
set(gca,'XTick',[0:1:n+3]);
set(gca,'YTick',[0:1:sizo+3]);
x0=10;
y0=10;
width=900;
height=900;
set(gcf,'position',[x0,y0,width,height]);
xlabel('objet');
ylabel('bins');
 title('grapth ');
 pause();
```

Code Matlab de l'algorithme BF

```
clear all
system('title  Algorithme de best-fit');
clc
% load test data
C=input('donnez la capacité de bin  : ');
n=input('donnez le nombre d''objet : ');
%  u = [523 961 352 522 772 888 940 137 164 253 364 372 388 460 432 461 132 234
345 456 566 678 578 895 258 369 147 159 753 258 852 654];  n = length(u);
```

```
%  C = 1024;
for iii=1:n
 u(iii)=input(sprintf('donnez la taille d''objet n° %d : ',iii));
end
r = C*ones(1,n);
w = zeros(n);
% this part of statements will be used by both FFD and BFD algorithms
[ud1,idec]=sort(-u); % sort obj. in decreasing size order
ud=-ud1;
% disp(ud);
[tmp,irec]=sort(idec); % irec is the recovery ordering
% note that ud=u(idec); and u = ud(irec);
wbf=w; rbf=r; % initialize for best fit method
for i=1:n,
   % disp(['u(' int2str(i) ') = ' int2str(u(i))]);
   % disp('r = '); disp(r);
   idx1=find([u(i)*ones(1,n) <= rbf]); % indices of bins still have spaces left
   [tmp,idx2]=min(rbf(idx1)-u(i));    % find the index within idx1 that best fits u(i)
   wbf(i,idx1(idx2))=1; rbf(idx1(idx2))=tmp; % update the state
   % disp('Press any key to continue ...'); pause
end
disp('les resultas de la  methode de Best fit (BF) sont : ')
%disp('the assignment are:')
%disp(wbf)
wwww=u*wbf;
sizo=sum([wwww > 0]);
tt=zeros(1,n);
X=[];
Y=[];
k=1;
matrix=zeros(n,sizo)
%disp(dddddd);
```

```
awww={};
for i=1:sizo
     str1='le bin numero ';
   str2=int2str(i);
   str3=': ';
   s=[str1 str2 str3];
   disp(s);
    awww{1,i}=s;
   k=1;
   tt=zeros(1,n);
    for j=1:n
    if (wbf(j,i)==1)
   tt(k)=u(j);
    matrix(k,i)=u(j);
    X=[X,j];
    Y=[Y, i];
     awww{k,i}=u(j);
     k=k+1;
    end
    end
    tt(tt==0) = [];
    fprintf('%s\n',sprintf(' %d',tt))
end
if exist('resultat de bf.xlsx', 'file')==2
  delete('resultat de bf.xlsx');
end
filename = 'resultat de bf.xlsx';
if ~isempty(awww)
xlswrite(filename,awww);
end
occpbf=u*wbf;
nbinbf=sum([occpbf > 0]);
```

```
disp(['le nombre de  bins totale utilisée  = ' int2str(nbinbf)]);
disp('la somme des tailles des objets utilisée dans chaque bins sont :');
fprintf('%s\n',sprintf(' %d',occpbf(1:nbinbf)))

% disp('Press any key to continue to Best fit method ...'); pause
sz = 40;
scatter(X,Y,sz,'MarkerEdgeColor',[0 .5 .5],...
            'MarkerFaceColor',[0 .7 .7],...
            'LineWidth',1.5);
  grid on;
grid minor;
set(gca,'XTick',[0:1:n+1]);
set(gca,'YTick',[0:1:sizo+1]);
x0=10;
y0=10;
width=900;
height=900;
set(gcf,'position',[x0,y0,width,height]);
xlabel('object');
ylabel('bins');
 title('grapth ');
pause();
```

Liste des Abréviations

Lettre	abréviation	signification
B	BP	Bin acking
	BPP-1D	Bin Packing uni-dimensionnel
	BPP-2D	Bin Packing bi-dimensionnel
	BPP-3D	Bin Packing tri-dimensionnel
	BF	Best-Fit
	BFD	Best-Fit Decreasing
F	FF	First-Fit
	FFD	First-Fit Decreasing
N	NF	Next-Fit
	NFD	Next-Fit Decreasing
O	OC	Optimisation Combinatoire
	OPT	Optimale
P	PSE	Procédure par Séparation et évaluation
R	RO	Recherche Opérationnel
W	WF	Worst-Fit
	WFD	Worst-FitDecreasing