PACKING 2D :

Professeur : **GRANDCOLAS Stephane**

GUEIT Lionel
EKOMO Pierre Martin

SOMMAIRE :

I. PRESENTATION DU PROJET :

Dans ce projet il est question d'implémenter le remplissage d'un conteneur pour se faire on s'est proposé de représenter le conteneur par une structure de forme rectangulaire et les objets, aussi de forme de la même forme ont la particularité d'être tous de dimensions différentes. Ce projet a été divisé en deux parties dans le première partie il était question de remplir le conteneur par la méthode glouton qui consistait à insérer les premiers objets sélectionnés et de dimensions valides directement dans le conteneur sans véritable soucis d'optimisation. Maintenant afin d'obtenir un remplissage optimale on s'est proposé d'implémenter le remplissage avec l'algorithme de sac à dos le travail demandé dans cette partie est le suivant :

-calcul d'une solution
  -sac à dos pour le remplissage des bandes,
  -reconstruction et sauvegarde des solutions,
-visualisation des solutions avec EZ-Draw .

Ce document sera axé sur trois grandes parties, d'abord nous présenterons le problème du remplissage par sac à dos, nous dirons ensuite de quelle manière nous l'avons abordé et enfin nous parlerons des différentes améliorations apportées et leurs effets sur l'efficacité et la qualité des solutions.

II. PRESENTATION DU PROBLEME DU SAC A DOS

Le problème de remplissage par sac à dos est un problème d'optimisation;Il modélise une situation analogue au remplissage d'un sac à dos ayant une capacité limitée et dans lequel on doit introduire des objets avec un poids et une certaine valeur, un meilleur remplissage de ce sac reviendrait à avoir des objets d'une valeur maximale n'excédant pas le poids supporté par le sac .

Une manière simple et efficace de le programmer serait de construire une solution à i variables à partir d'un sous problème à i-1 variables d'où l'idée de programmation dynamique du problème.

Ce problème est NP-complet, ce qui signifie en d'autres termes qu'il n'existe pas d'algorithme polynomial pour le résoudre. Il existe toutefois un algorithme pseudo-polynomial . On utilise une table de programmation dynamique A à deux dimensions.

A(i, p) représente le profit maximal que l'on peut obtenir en prenant un sous-ensemble d'objets de {1, . . . , i} dont le poids total soit inférieur ou égal à p. A(i, p) vaut 0 si il n'existe aucun sous-ensemble de {1, . . . , i} de poids

au plus égal à p :

$A(i + 1, p) = \max(A(i, p), A(i, p - Pi+1) + Vi+1)$ si $Pi+1 \leq p$

$A(i + 1, p) = A(i, p)$ sinon .

Maintenant qu'on comprend mieux le problème de sac à dos nous pouvons revenir dans le cas du remplissage du conteneur qui est le but de notre projet.

III. APPROCHE DU PROBLEME DE REMPLISSAGE DU CONTENEUR PAR L'ALGORITHME DE SAC A DOS

Pour résoudre le problème du remplissage d'un conteneur avec l'algorithme de sac à dos il est important de faire une bonne adaptation des données et de la situation .

Pour avoir une meilleure approche dynamique du problème nous n'avons pas directement considéré le conteneur comme le sac mais plutôt un ensemble de sacs, ces sacs étant les bandes de celui_ci cet à dire des partitions du conteneur suivant différentes largeurs.

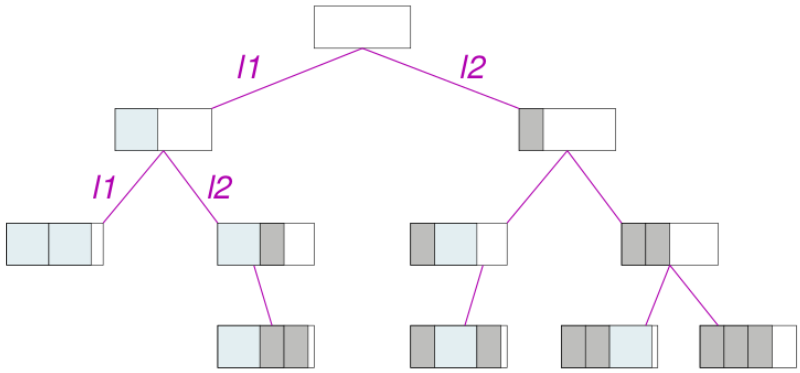
En résumé un sac correspond à une largeur de bande du conteneur son poids est représenté par sa hauteur car nous partons du principe que tous les objets à insérer dans la bande seront bien-sur de largeur inférieure ou égale à celle_ci ; ceci est assuré par une sélection préalable des objets enfin la valeur des objets n'est rien d'autre que leur surface.

Le remplissage dynamique est fait selon le schéma suivant :

Recherche d'un remplissage.

Recherche énumérative sur k largeurs de bandes.

(on considère des largeurs correspondant aux dimensions des objets)



Exemple avec $k = 2$ et deux dimensions différentes en tout et pour tout.

Beaucoup de modifications ont été apportées dans certaines fonctions qui étaient déjà présentes dans la première partie ceci afin d'améliorer considérablement la vitesse d'exécution du programme et de garantir un véritable test en quantité des données passées en paramètre dans le programme.
Voici une brève présentation des modules utilisés :

Au début il nous faut générer des objets pour cela on utilise le module suivant :

-**genere**: Qui permet la génération aléatoire du problème;L'utilisateur appelle son exécutable en lui passant trois 5 arguments:la largeur du conteneur,sa hauteur, le nombre d'objets candidats, le largeur et la hauteur à partir desquels ceux_ci sont générés aléatoirement.
-**Lecture**: Qui permet la lecture des données générées par genere.c
-**structure**: Qui permet de faire différentes opérations sur les structures allocations en fonction du nombre et de la taille des objets lus par lecture.c , dans ce module y sont déclarées aussi les diverses structures utilisées pour la résolution du problème à savoir la structure « Meilleur » pour conserver le meilleur choix d'objets ;
-**remplir_conteneur**:Qui contient toutes les fonctions (outils) utilisées pour le remplissage du conteneur ;
-**main** : C'est le module qui permet d'établir le lien entre les différents modules du programme il commence avec la lecture des données générées par genere.c et se termine par l'appel de la fonction afficherSolution du module remplir_conteneur.

IV .EXPLICATION DU MÉCANISME DE FONCTIONNEMENT DU REMPLISSAGE

Ici nous expliciterons surtout le mécanisme de fonctionnement de la fonction « remplir_conteneur » car c'est elle qui illustre mieux le travail demandé dans cette deuxième partie l'algorithme de base est le suivant :

```
begin
C := selectionner_k_largeurs_valides (l, O, k);
if C = ∅ then
mettre_a_jour_meilleur_remplissage (O);

return;
while C = ∅ do
extraire e de C ;
S := remplir_bande (e, O, H);
remplir_conteneur (l − e, O \ S , H , k );
end
```

Dans la fonction « remplir_conteneur » on commence par faire une recherche sur les objets afin de trouver ceux qui sont disponibles et dont les largeurs ne dépassent pas celle restante du conteneur ceci est réalisé par la fonction « selectionner_k_largeur » qui va retourner la largeur d'exploration valeur que nous appelleront ici « Select » .

Selon la valeur Select retournée on distingue deux cas :

-Le cas ou Select est supérieur à zéro : Dans ce cas on boucle sur Select afin de remplir chaque bande du conteneur à la fin du remplissage d'une bande on fait un appel récursif sur la fonction de remplissage du conteneur afin de descendre dans la l'arbre d'exploration et enfin juste avant la fin de la boucle on libère les objets utilisés pour remplir la bande.

Le rôle de l'algorithme de sac à dos se trouve dans la fonction utilisée pour le remplissage de bande ,parmi les paramètres de cette fonction il y'a l'une des k largeurs sélectionnées ; C'est à partir de cette largeur que la fonction fait sa propre sélection d'objets (les objets devant être disponibles et largeur inférieure ou égale à la largeur en paramètre), pour ces objets pour revenir à l'algorithme de sac à dos la valeur est représentée par leur surface, le poids ,leur hauteur et le poids max n'est rien d'autre que la hauteur de la bande donc celle du conteneur.

A la fin du remplissage de la bande on rend les objets utilisés indisponibles et on sort de la fonction.

-Dans le cas où Select est égal à zéro : on calcule la surface des objets utilisés pour remplir le conteneur, puis on la compare à celle du meilleur remplissage précédant, si elle est supérieure on se charge de sauvegarder les objets utilisés et on remonte dans l'arbre de récursion pour trouver d'autres meilleurs remplissages.

Beaucoup de problèmes ont été rencontrés à l'exécution du programme notamment au niveau de la vitesse d'exécution , le programme mettait du temps à trouver un résultat pour un nombre raisonnable d'objets et pour très peu d'objets,il produisait une erreur de segmentation, des solutions ont été apportées à cet effet.

V. AMÉLIORATION DU PROGRAMME

1. Vitesse d'exécution

Afin d'alléger le programme les différentes allocations ont été réalisées qu'une seule fois dans le main avant l'appel de la fonction de remplissage du conteneur, ceci à aussi permis d'éviter des erreurs de segmentation lors du traitement d'un grand nombre d'objets.

Un calcul des occurrences des largeurs sélectionnées à été ajouté, afin d'avoir un maximum d'objets de grande taille et de même largeur dans les premières bandes, cette sélection est assurée par une fonction

max qui calcule la plus grande largeur ayant la plus grande occurrence à partir du tableau d'occurrences de largeurs.

Différentes propriétés de coupure ont été ajoutées :
-la propriété des 100% qui intervient si on a une surface courante égale à la surface totale du conteneur on s'arrête ;
-Si tous les objets sont présents dans le conteneur on s'arrête ;
-Si la surface courante déjà occupée additionnée à la surface de la partie non explorée du conteneur est supérieure à la meilleure surface restante on s'arrête.

2. Cas d'un très petit nombre d'objets

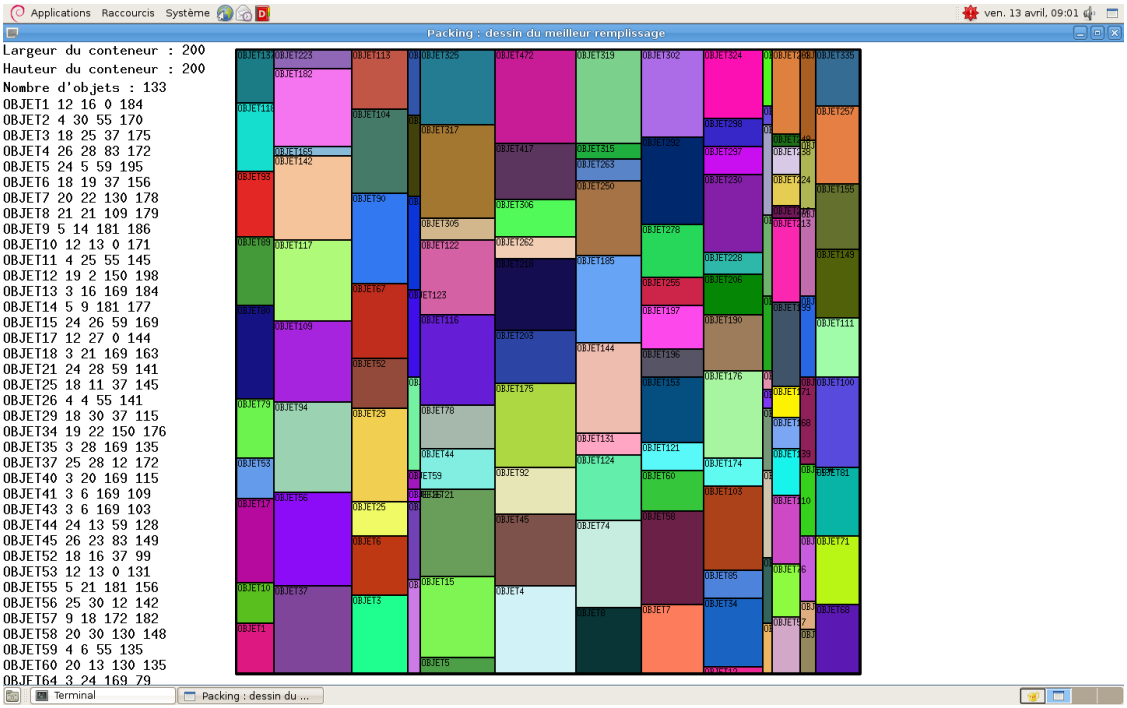
Pour éviter les erreurs de segmentation en cas d'un très petit nombre d'objets une condition à été ajoutée selon laquelle on ne fait aucune sélection de largeur si la profondeur d'exploration est supérieure ou égale au nombre d'objets.
Malgré toutes ces améliorations on note toujours un long temps d'exécution dans le cas où la dimension du conteneur est trop grande par rapport aux dimensions des objets générés.

VI . EXEMPLES D'EXECUTION DU PROGRAMME

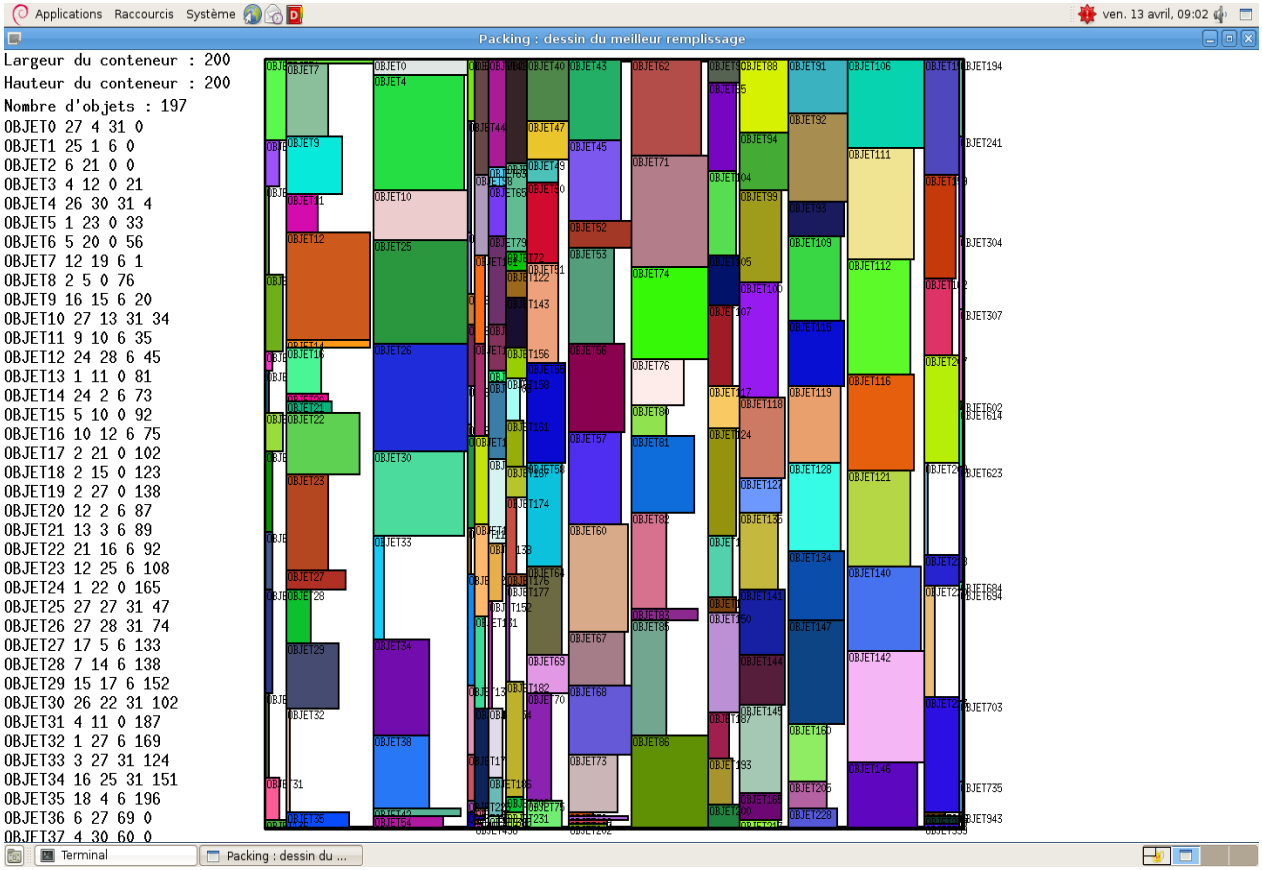
1. Illustrations

Pour de un conteneur de largeur : 200 hauteur : 200
Pour 1000 objets de largeur <=30 et hauteur <=30

avec SAC A DOS :



Avec glouton :



2. TABLEAU

	Glouton		Sac à Dos	
	Temps	Pourcentage	Temps	Pourcentage
Conteneur 200x200 Nb_objets:250 objet : 50x50	0m0.017s	82,77%	0m1.862s	98,35%
Conteneur 200x200 Nb_objets:500 objet : 50x50	0m0.057s	84,12%	0m0.900s	99,11%
Conteneur 200x200 Nb_objets:1000 objet : 50x50	0m0.063s	86,00%	0m0.188s	100,00%
Conteneur 300x300 Nb_objets:1000 objet : 50x50	0m0.203s	82,55%	0m4.824s	99,56%

CONCLUSION

En conclusion le projet de packing 2D a été très enrichissant en matière d'algorithmique et de programmation. D'après les résultats obtenus on remarque qu'une amélioration pouvait être faite au niveau du remplissage glouton mais on a pas jugé nécessaire de la faire on a plutôt consacré notre temps à optimiser le sac à dos.