HOUDEBINE Matthieu G22
RIVIERE Jean-Matthieu G21

Projet N°4 : Heuristiques pour le problème du bin-packing

Dossier de spécifications et de conception

Année universitaire 2012-2013 ISIMA – Université Blaise Pascal

Sommaire

Présentation générale du projet

Introduction au sujet

Le but de ce projet est d'optimiser le rangement d'objets de tailles différentes dans des conteneurs de capacités différentes, afin de réduire au maximum le nombre de conteneur utilisés. Ce problème est-dit « n-p complet » : il ne possède pas une unique solution optimale qui puisse fonctionner pour tous les cas de figure.

Ce dossier contient les spécifications de l'application (à savoir l'interface entre la partie exécutive de ce projet et l'utilisateur) qui sera réalisé durant ce projet.

Enoncé du besoin

Ce projet permettra à l'utilisateur de l'application finale de pouvoir évaluer facilement les performances de différents algorithmes de bin-packing (tels que « Next Fit », « Best Fit »…) sur des jeux de données diverses et variés tels que des jeux d'essais pré-saisis en tant qu'exemples, ou également des jeux d'essais personnalisés saisis par l'utilisateur lui-même.

Objectifs

Le programme conçu devra disposer d'une interface claire, simplifiée et répondant aux besoins du cahier des charges. L'utilisateur devra pouvoir sélectionner la provenance des jeux d'essais utilisés pour les tests, ainsi que les différents algorithmes à utiliser sur ces jeux d'essais. Le programme devra également retourner à l'issue des calculs le nombre d'itérations utilisées par chaque algorithme pour arriver à la solution, ainsi que le temps de calcul, dans un objectif de comparaison de performance des algorithmes choisis.

Le principal objectif de ce projet est de pouvoir comparer les résultats des différents algorithmes sur différents jeux d'essais divers et variés afin de pouvoir en tirer des conclusions d'ordre plus général sur les avantages et défauts de chacun.

Informations concernant la réalisation technique

Le langage utilisé pour la réalisation sera le langage C. L'interface de l'application sera accessible par l'utilisateur au travers un terminal. L'application sera compatible avec les systèmes Unix dans un premier temps, idéalement compatible tous systèmes courants (Linux, Windows, Mac OS).

Pour chaque algorithme implémenté dans l'application réalisé, un algorithme de principe correspondant sera fourni au travers de ce dossier, ainsi que son algorithme détaillé (en langage courant, globalement une description non technique de ce que fait le programme et de son fonctionnement).

Organisation

a) Intervenants

Responsable de projet : GOURGAND Michel

Equipe de développement :

- HOUDEBINE Matthieu – ISIMA G22
- RIVIERE Jean-Matthieu – ISIMA G21

b) Calendrier

Date de remise du projet : Semaine du 24 Juin 2013

Description des cas d'utilisation

Ecran N°1 : Sélection des données

L'utilisateur sélectionnera la provenance du jeu d'essai utilisé pour les tests. Deux choix s'offrent à lui :

a. Sélectionner un jeu d'essai parmi ceux préenregistrés dans le programme, à savoir des jeux d'essais de nombre d'objets et capacité différents, triés ou non, dont les données ont été pré-saisies dans des fichiers.
   Pour cela le programme lui affiche une liste de jeux d'essais. L'utilisateur sélectionne un seul jeu d'essai.
b. Générer un jeu d'essai aléatoire, en fournissant au programme le nombre d'objets et la capacité des conteneurs.
c. Sélectionner un jeu d'essai contenu dans un fichier que l'utilisateur aura lui-même rempli, au format indiqué afin que l'application puisse interpréter correctement les données. Pour cela le programme lui affiche une boite de dialogue ou l'utilisateur saisira le chemin d'accès du fichier au format texte contenant les données.

Modèle de présentation

1)

Source du jeu d'essai — A partir de la bibliothèque / Généré aléatoirement / A partir d'un fichier

Valider

---

1) a.

Sélection du jeu d'essai — Exemple 1 / Exemple 2 / ...

Valider

---

1) b.

Nombre d'objets (n) 1000

Capacité des conteneurs (K) 100

Valider

---

1) c.

Chemin d'accès C:\Données\data1.txt

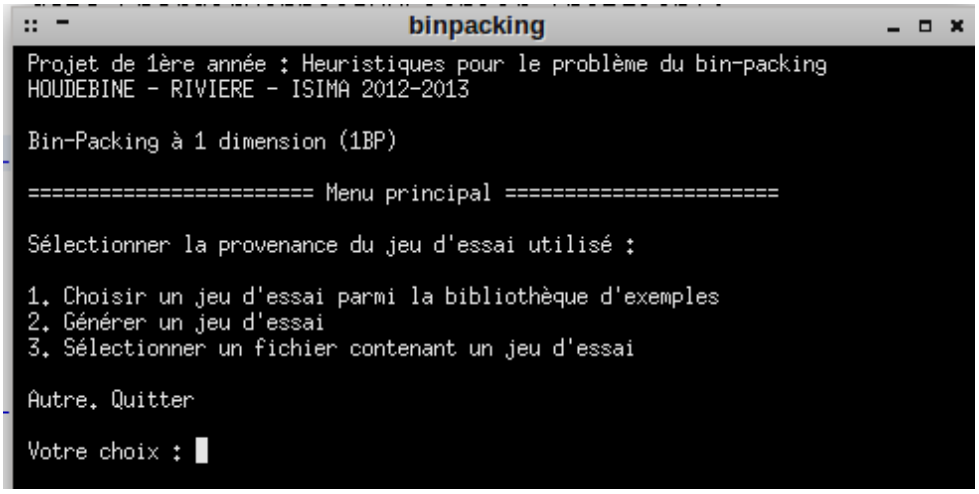
Valider

Le jeu d'essai a bien été enregistré

Le jeu d'essai contenu dans ce fichier n'est pas au bon format

Captures d'écran

1)



1) a.


Votre choix : 1
.
25-10
10000-1000
10000-1000-desc
10000-1000-asc
100-10
..
Entrer le nom de l'exemple à utiliser :

1) b.


Votre choix : 2
Nombre d'objets à créer (n) : 1000
Capacité des conteneurs (K) : 10

1) c.

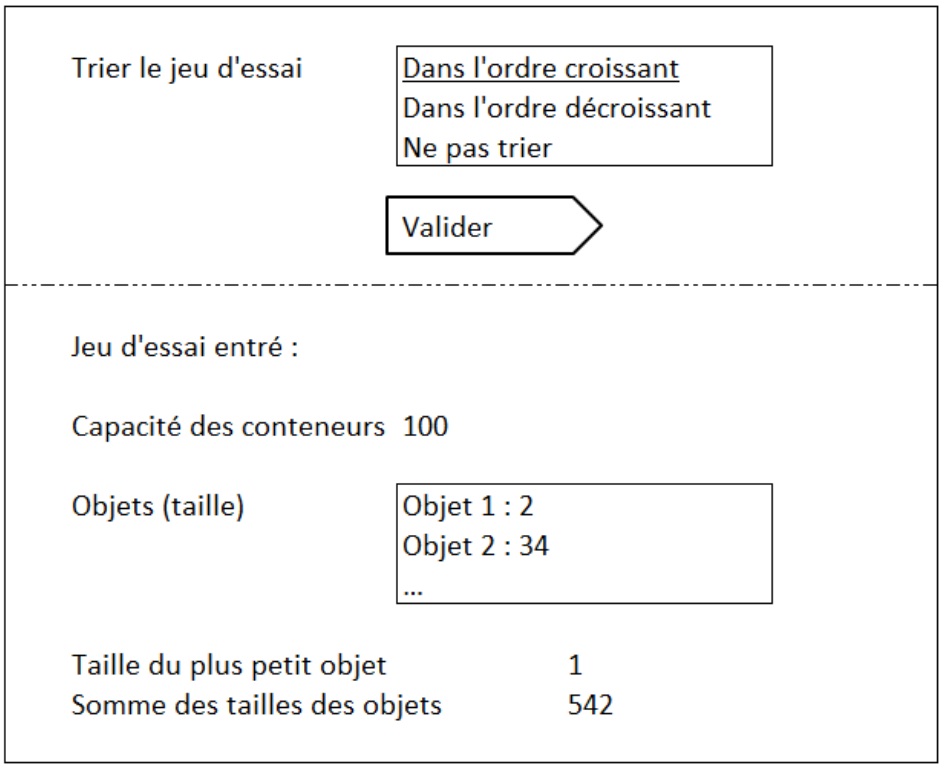

Votre choix : 3
Entrer le chemin d'accès : /data/BinPacking/Jeu_d_essai_125

Ecran N°2 : Tri et affichage des données saisies

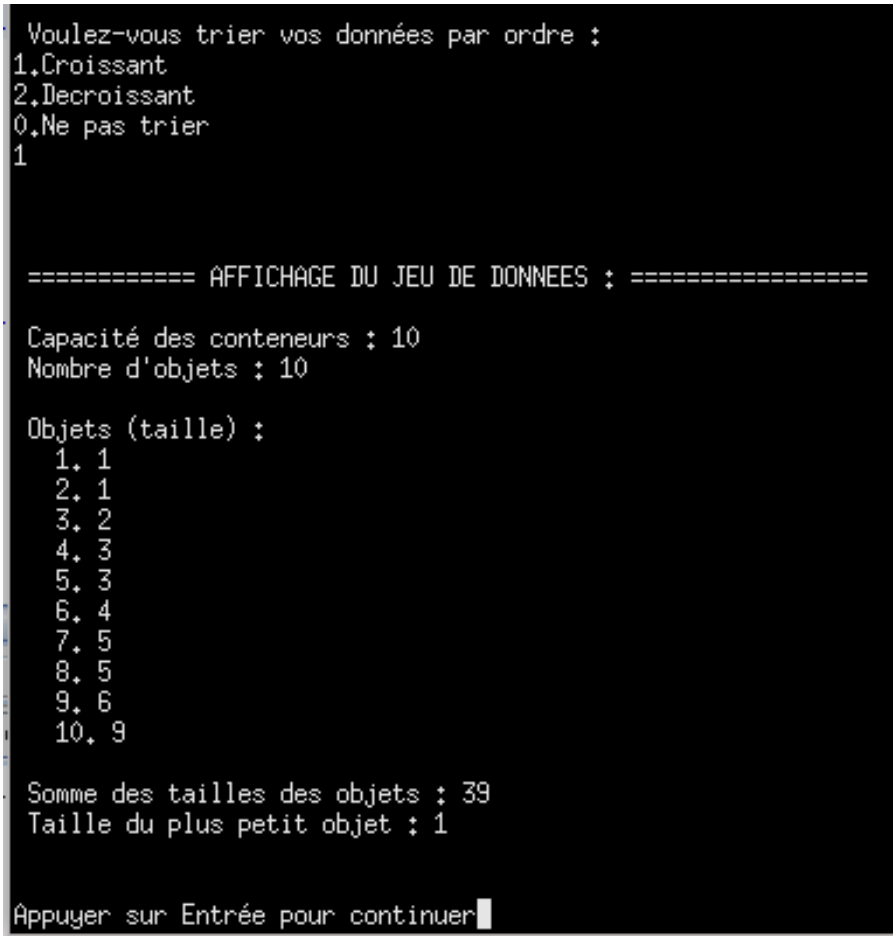
Une fois que les données ont bien été chargées par le programme, l'utilisateur peut les trier par ordre croissant de taille d'objets, par ordre décroissant, ou ne pas les trier.

Ensuite, et afin que l'utilisateur puisse vérifier le jeu d'essai choisi, le programme affichera ce jeu d'essai à travers une fenêtre récapitulative qui précèdera le lancement des calculs. Le programme affiche également, à titre d'information, la taille du plus petit objet ainsi que la somme des tailles des objets.

Modèle de présentation



Captures d'écran



Ecran N°3 : Récapitulatif des résultats

A la fin des calculs, le programme affiche pour chaque algorithme choisi la solution de cet algorithme (à savoir le nombre de boites utilisées) et le temps qui aura été nécessaire pour trouver cette solution. Le programme affiche également, à titre d'information, la borne minimale du jeu d'essai.

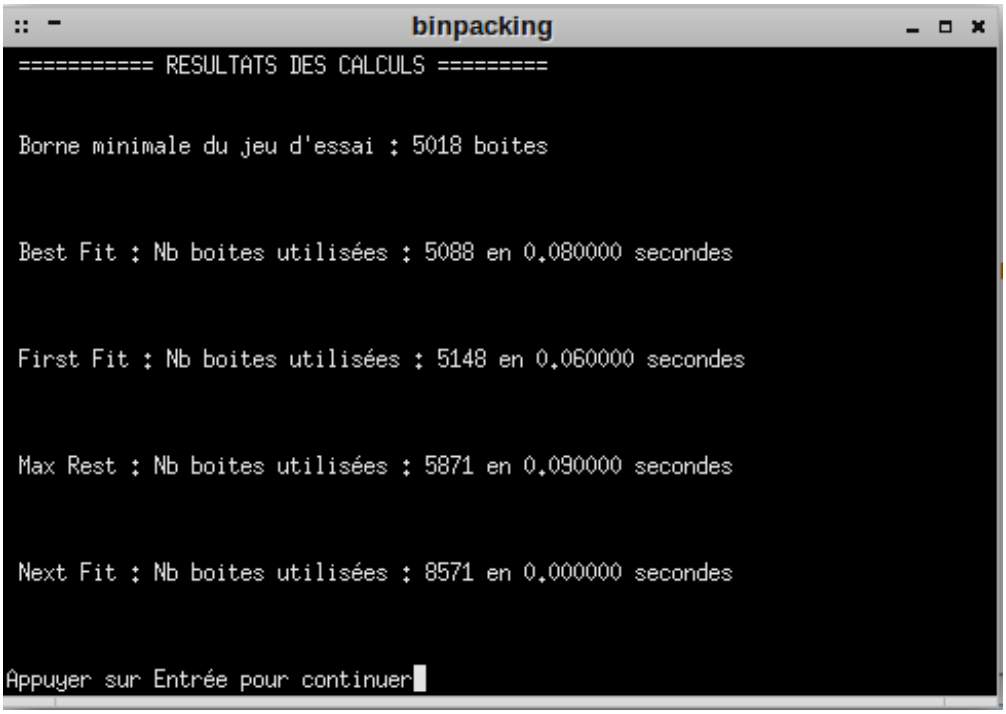
Modèle de présentation

Borne minimale du jeu d'essai	120 boites

Affichage des résultats :

Algorithme 1	
Temps de calcul :	2.3 secondes
Nombre de boites utilisées	145 boites
Algorithme 2	
...	

Captures d'écran

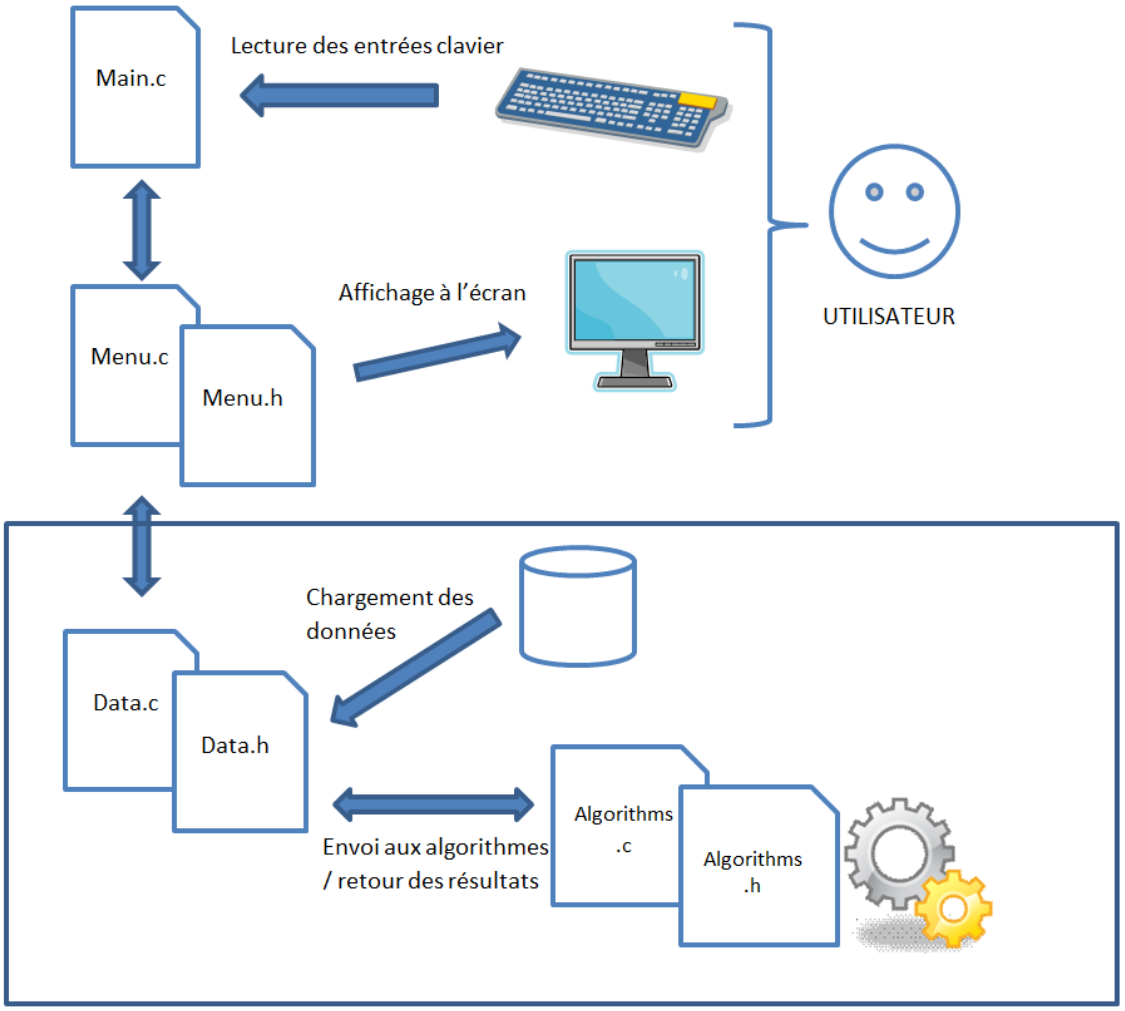


Conception du projet

Organisation du code

Dans une optique de généricité du code afin d'obtenir un programme plus modulaire et dont certaines parties telles que les algorithmes pourront éventuellement être réutilisées, les différentes fonctions qui composent ce programme ont ainsi été séparées en différents fichiers en fonction de leur rôle dans ce projet. Cette séparation a également permis de mettre en place une encapsulation des données au sein du programme, afin que les données ne soient accessibles que par les couches basses (couches de calculs et de traitements) et non par les couches utilisateurs.

Voici un schéma simplifié résumant l'organisation du code :



Détail des algorithmes utilisés

Le but de ce projet étant de pouvoir créer un programme capable de comparer sur un même jeu d'essai les performances de différents algorithmes, quatre algorithmes ont donc été codés et adaptés afin de fournir les résultats demandés (à savoir nombre de boites utilisées et temps d'exécution).

Algorithme Next-Fit

Le plus basique de tous, cet algorithme fonctionne sur le principe suivant : à chaque instant t est gérée au plus une boite. Si l'objet courant ne rentre pas dans la boite actuellement gérée, une autre boite est créée et tous les objets suivants seront insérés dedans jusqu'à ce qu'un objet n'y rentre plus, etc.

Principe :

```
Pour tous les objets i de 1 à n faire :
    Si l'objet courant i peut rentrer dans la boite courante alors
        Stocker l'objet i dans la boite courante
    Sinon :
        Créer une nouvelle boite, en faire la boite courante, y
        ranger l'objet i
    Fin si
Fin pour
```

Algorithme First Fit

Cet algorithme fonctionne selon le principe suivant : Pour chaque nouvel objet traité, l'algorithme parcourt les boites existantes et stocke l'objet i dans la première boite dans laquelle il peut tenir. Si aucune boite n'a pu être trouvée, une nouvelle boite est créée et l'objet est rangé dedans.

Principe :

```
Pour tous les objets i de 1 à n faire :
    Pour toutes les boites j = 1, 2, … faire :
        Si l'objet courant i peut rentrer dans la boite courante
        j alors :
            Ranger l'objet i dans la boite j
            Sortir de la boucle de parcours des boites et passer
            à l'objet suivant
        Fin si
    Fin pour
    Si l'objet i n'a pas été stocké alors :
        Créer une nouvelle boite et y ranger l'objet i
    Fin si
Fin pour
```

Algorithme Max-Rest

Pour chaque objet, l'algorithme cherche la boite dans laquelle il reste le plus de place disponible et essaye d'y ranger l'objet. Si l'objet ne rentre pas dans cette boite, l'algorithme crée une nouvelle boite et y range l'objet.

Principe :

```
Pour tous les objets i de 1 à n faire :
      Déterminer la boite j avec la capacité restante la plus élevée
      Si l'objet i peut rentrer dans la boite j alors :
            Ranger l'objet i dans cette boite
      Sinon :
            Créer une nouvelle boite et y ranger l'objet i
      Fin si
Fin Pour
```

Algorithme Best-Fit

Cet algorithme, plus complexe, fonctionne de la manière suivante : il range chaque objet dans la boite pouvant le contenir avec la capacité restante la moins élevée.

Principe :

```
Pour tous les objets i de 1 à n faire :
      Pour toutes les boites j = 1, 2, … faire :
            Si l'objet i peut rentrer dans la boite j alors :
                  Calculer la capacité restante dans la boite j après
                  ajout de l'objet i
            Fin si
      Fin pour
      Ranger l'objet i dans la boite j qui aura la capacité restante
      la moins élevée après ajout de l'objet.
      Si l'objet i n'a pas été stocké alors :
            Créer une nouvelle boite et y ranger l'objet i
      Fin si
Fin pour
```

Format des jeux d'essai

Le programme ayant été développé dans le but de pouvoir être réutilisé avec n'importe quel jeu d'essai fourni par l'utilisateur, il a donc fallu déterminer une convention pour les formats de données contenus dans les fichiers lus par le programme.

Ainsi, le programme acceptera tous les jeux d'essai sous la forme suivante :

Nombre d'objets sur la 1ère ligne
Taille des conteneurs sur la 2ème ligne
Taille des objets (une taille par ligne)

Soit par exemple, pour un jeu d'essai de 3 objets de taille 1 et des conteneurs de taille 2 :

```
3
2
1
1
1
```

Le nom du fichier n'est pas important, toutefois il vaut mieux éviter les espaces, caractères accentués ou symboles.

Tout jeu d'essai peut être enregistré dans la base de données d'exemples de ce projet : il suffit de copier le fichier contenant le jeu d'essai au format ci-dessus dans le répertoire « examples » situé à la racine des sources du projet (dans le cas d'un environnement de développement), ou dans le même dossier que l'exécutable (pour un projet compilé).

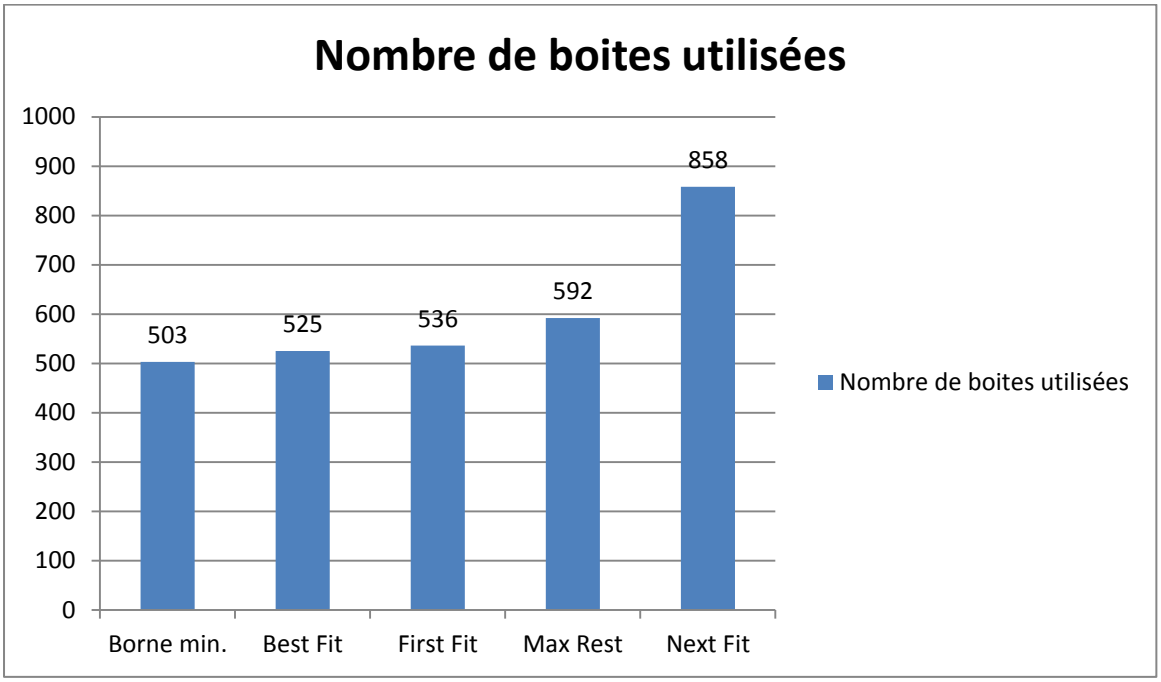
Résultats d'exécution

Différents jeux d'essais ont été utilisés afin de tester les algorithmes :

1. *Jeu d'essai aléatoire non trié, n = 1000 objets avec des boites de taille K = 100*

Borne minimale du jeu d'essai : 503 boites

Résultats en nombre de boites utilisées :

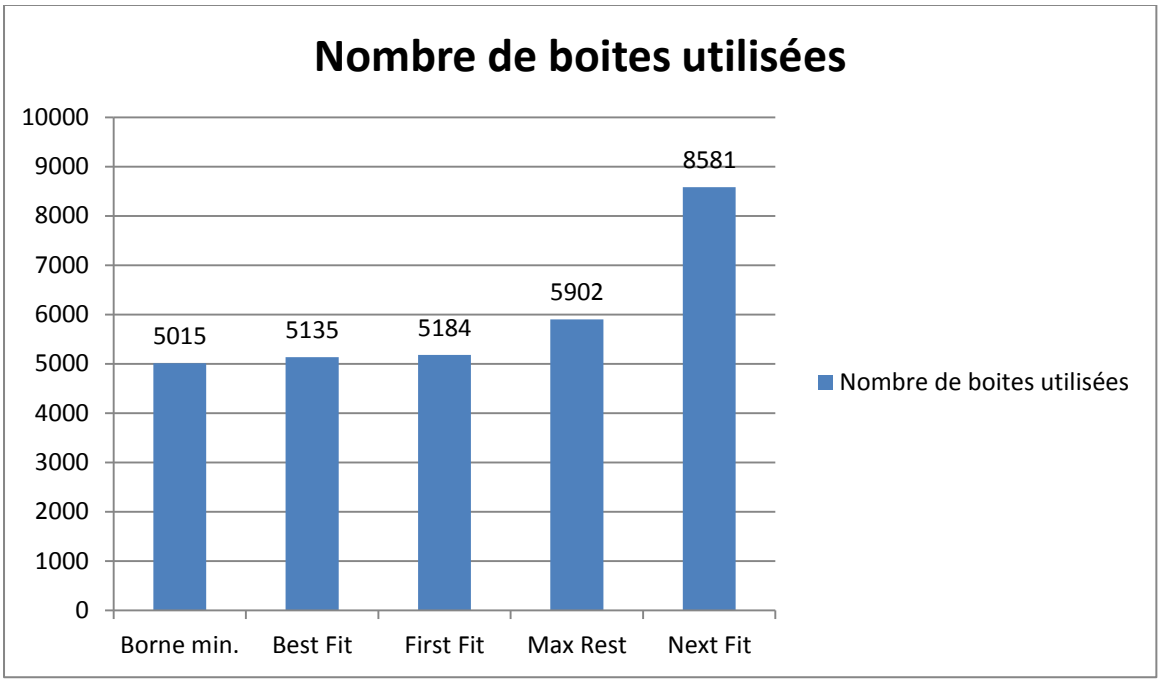


Pour un jeu d'essai aussi petit, les temps d'exécution sont tous proches de 0,000000 secondes

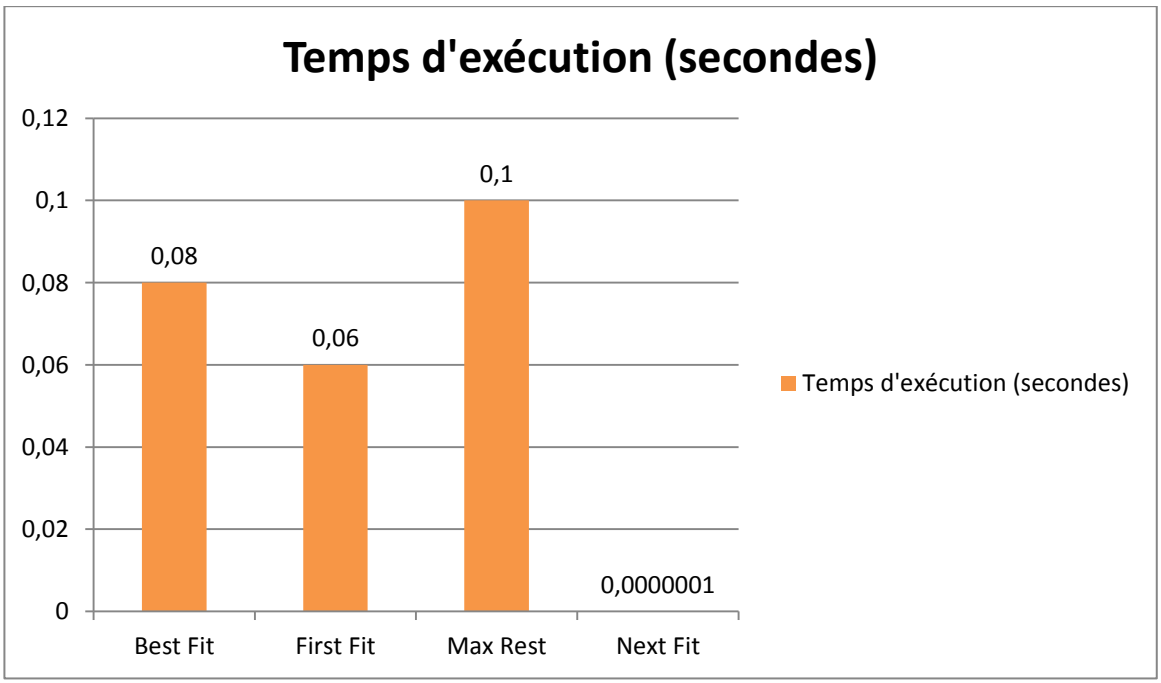
2. *Jeu d'essai aléatoire non trié, n = 10000 objets, boites de taille K = 1000*

Borne minimale du jeu d'essai : 5015 boites

Résultats en nombre de boites utilisées :



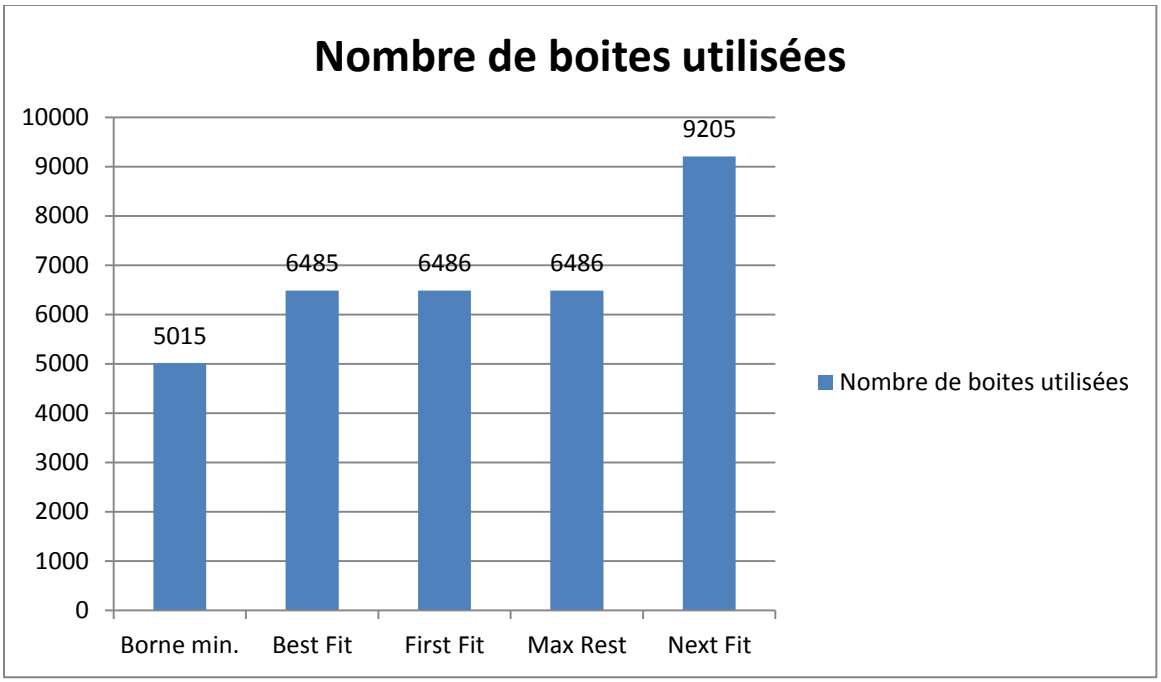
Résultats en temps d'exécution :



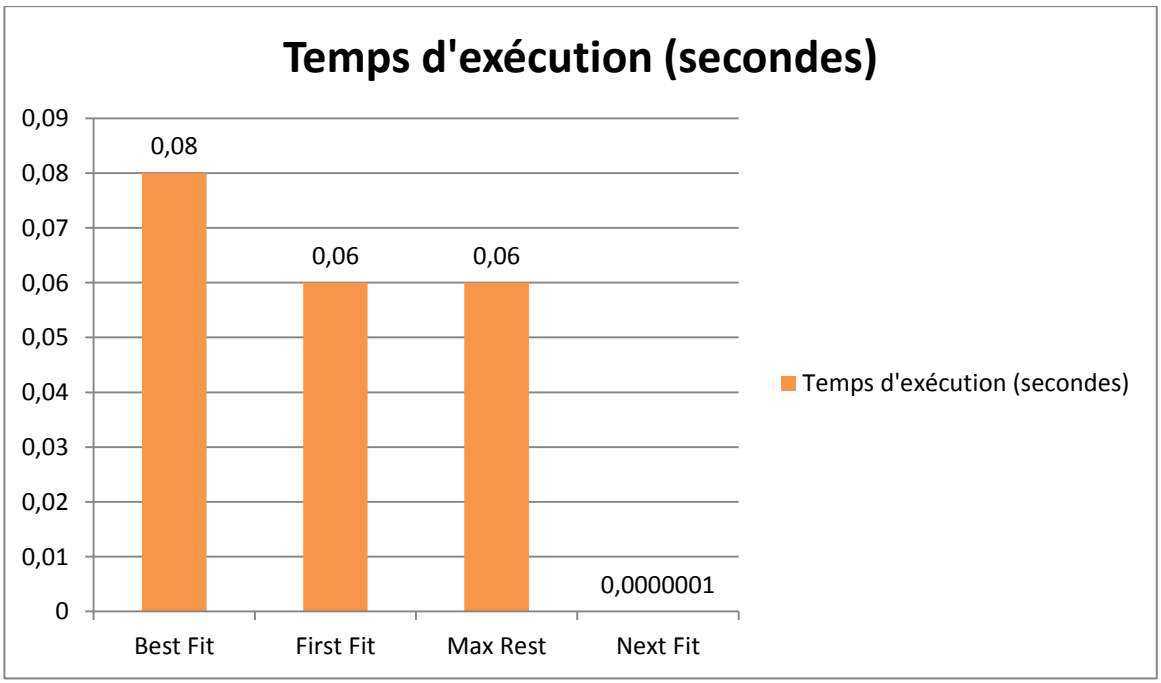
3. *Même jeu d'essai (n=10000, K=1000) trié dans l'ordre croissant de taille d'objets*

Borne minimale du jeu d'essai : 5015 boites

Résultats en nombre de boites utilisées :



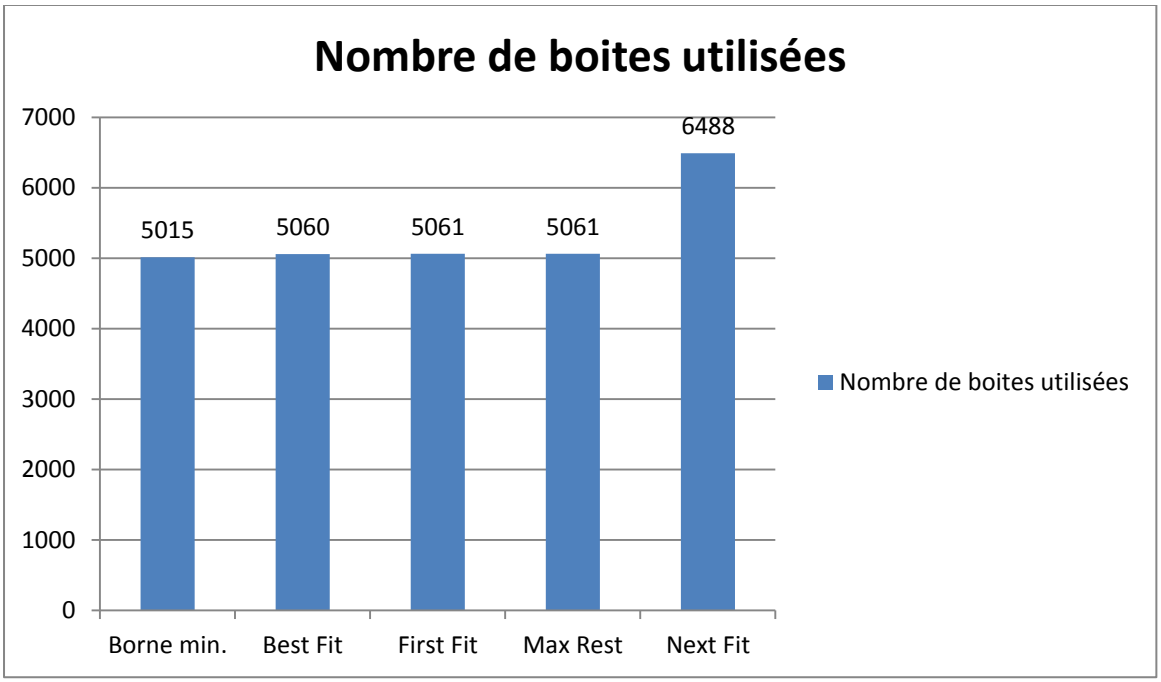
Résultats en temps d'exécution :



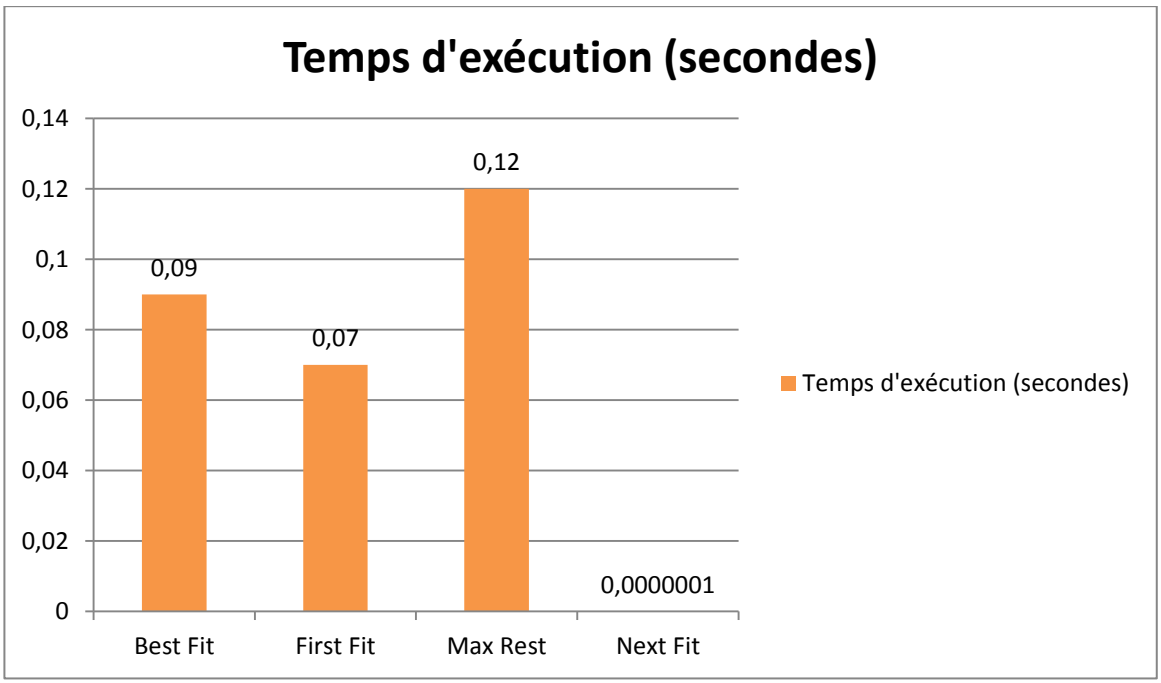
4. *Même jeu d'essai (n=10000, K=1000) trié dans l'ordre décroissant de taille d'objets*

Borne minimale du jeu d'essai : 5015 boites

Résultats en nombre de boites utilisées :



Résultats en temps d'exécution :



Conclusion

Bilan des résultats

Au regard des résultats des tests, on peut confirmer les points forts et faibles de chacun des algorithmes, pour certains évidents :

L'algorithme Next Fit est très rapide peu importe que le jeu d'essai soit ou non trié, du fait de sa seule boucle et parce qu'il ne gère qu'une seule boite à la fois. Il pourra donc être utilisé dans des cas de figure ou une solution doit être rapidement apportée à un problème, au détriment de l'efficacité de cette solution.

L'algorithme Max Rest est assez long en temps d'exécution sur des jeux d'essais non triés ou triés par ordre décroissant, ceci étant dû à la méthode utilisée pour rechercher la boite avec la capacité restante la plus importante. Cet algorithme pourra donc être utilisé de préférence sur des jeux d'essais triés par ordre croissant, avec une méthode de recherche optimisée.

Les algorithmes First Fit et Best Fit sont ceux qui donnent les meilleurs résultats : Best Fit permet d'obtenir à chaque fois un nombre de boites utilisées le plus réduit possible, cependant il est généralement plus long que Next Fit qui donne des résultats assez proches. Il conviendra donc d'utiliser l'un ou l'autre selon les besoins afin d'obtenir de très bons résultats avec un temps de calcul raisonnable.

D'ordre général, on remarque également que pour des jeux d'essais triés par ordre croissant, les algorithmes sont globalement moins efficaces en termes de nombre de boites utilisées.

Bilan Personnel de l'équipe de développement :

Matthieu HOUDEBINE :

La réalisation de ce projet a été très intéressante d'un point de vue autant technique que scientifique car, celui-ci étant basé sur la modélisation de problèmes mathématiques et d'heuristiques de résolution en langage C, j'ai pu consolider mes acquis à la fois en algorithmique, mathématique et en programmation et également acquérir de nouvelles compétences en effectuant différentes recherches notamment sur les cas d'utilisation concrets d'heuristiques de résolution de problèmes de Bin Packing sur des exemples de la vie courante (chargement d'un camion…).

Au niveau du développement de ce projet, l'utilisation d'un environnement de développement (NetBeans IDE 7.3) a grandement facilité l'écriture du programme, notamment par la création d'un Makefile automatique ainsi que l'auto-complétion et le débuggeur intégré.

J'ai pu mettre à contribution mes connaissances acquises à l'ISIMA ainsi que lors de mes précédentes années d'études notamment en cours de Langage C, Programmation Numérique, Analyse Numérique et également Systèmes d'Exploitation, connaissances qui m'ont beaucoup aidé à avancer dans la réalisation de ce projet.

Jean-Matthieu RIVIERE :
(Absent lors de l'écriture de ce rapport)

Les sources de ce projet (sous forme de projet NetBeans 7.3) peuvent être téléchargées sur
http://fc.isima.fr/~houdebin/BinPacking.zip

Bibliographie :

http://fr.wikipedia.org/wiki/Probl%C3%A8me_de_bin_packing
http://en.wikipedia.org/wiki/Bin_packing_problem (plus complète)
http://www.cs.sunysb.edu/~algorith/files/bin-packing.shtml
http://www.cs.ucr.edu/~neal/2006/cs260/piyush.pdf