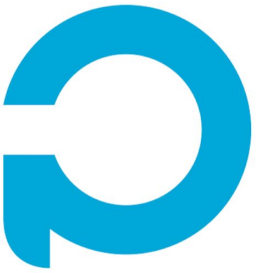

Rapport

Mini Projet 2

DALLA-NORA Enzo

DEGAND Sébastien

GRÉAUX Thomas

LARA Jeremy

2017 - 2018

I. Introduction

Pour ce projet, nous devons analyser 5 algorithmes pour résoudre le problème du bin-packing: Next Fit, First Fit, Best Fit, Worst Fit et Almost Worst Fit. Nous avons donc implémenté ces algorithmes dans le langage C++.
En ce qui concerne notre exemple exemple au choix, nous avons choisis une capacité des boîtes de 100 avec des objets de taille 51 suivis d'objets de taille 49 afin de montrer l'efficacité de l'algorithme First Fit dans ce cas.

II. Analyse

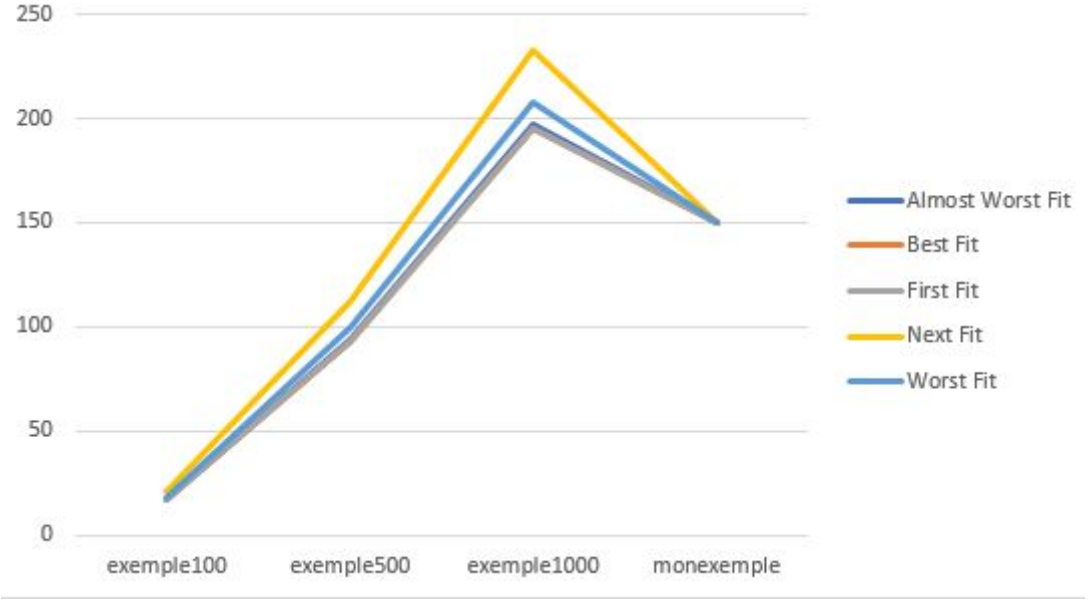

Figure 1. graphique représentant les résultats des algorithmes sur les exemples

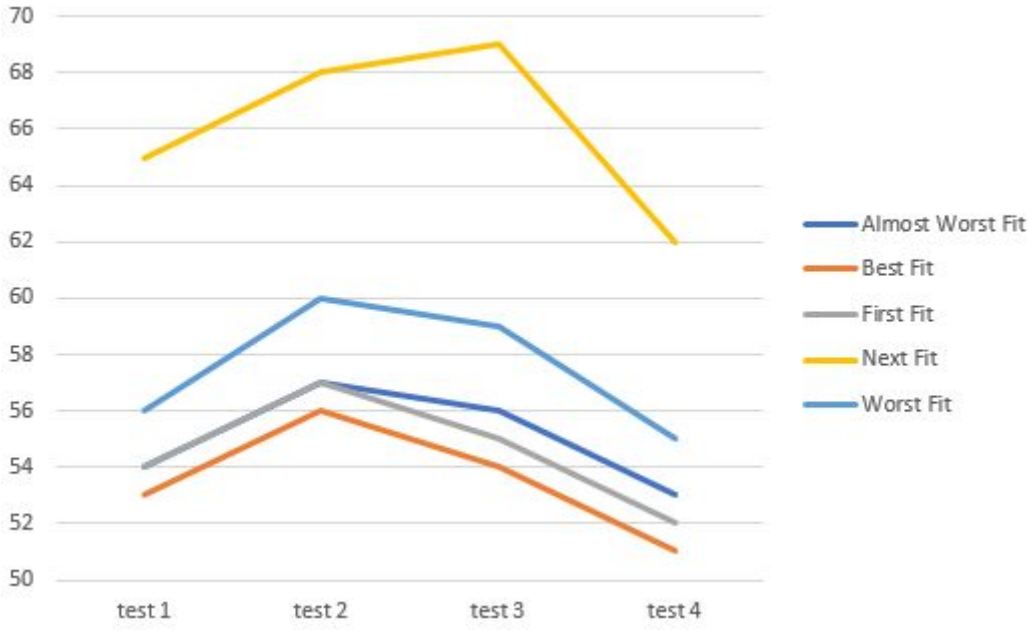

Figure 2. graphique représentant les résultats des algorithmes sur des tests aléatoires de boîte de taille 50000 avec 100 objets à placer.

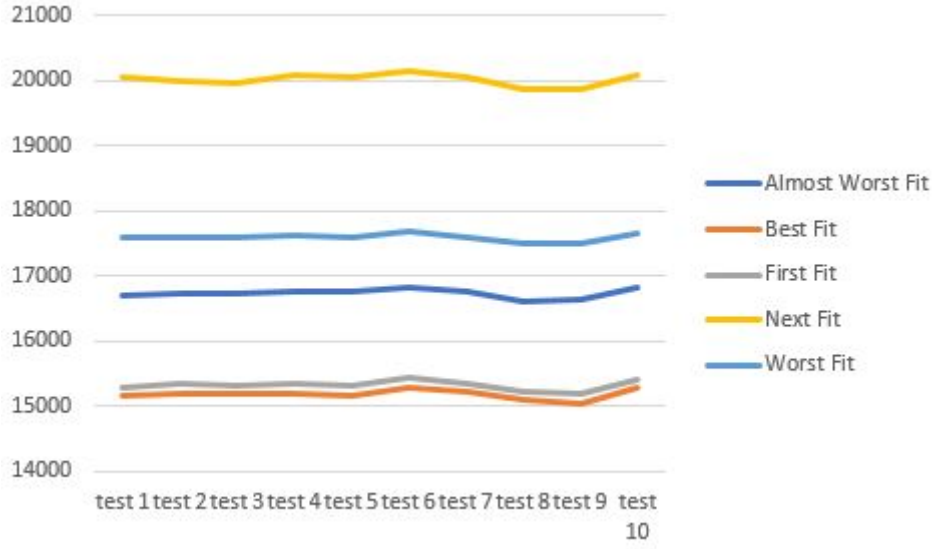

Figure 3. graphique représentant les résultats des algorithmes sur les tests aléatoires de boîte de taille 50000 avec 30000 objets à placer

a. Next Fit

Cet algorithme consiste à prendre les objets un par un et les placer soit dans la dernière boîte s'il y a la place soit dans une nouvelle boîte.
Cet algorithme est donc le plus rapide car on ne parcourt pas toutes les boîtes pour placer les objet mais juste la dernière. Comme on le voit sur les graphique précédent, il est le moins efficace car il consomme le plus de boîtes. En effet, l'algorithme tend à laisser de grands espaces vide non utilisés dans les boites qui pourrait accueillir d'autres objets.

b. First Fit

Cet algorithme consiste à prendre les objets un par un et les placer dans la boîte la première boîte pouvant l'accueillir. Dans le pire des cas, pour un objet on va parcourir toutes les boîtes. Si aucune des boîtes ne peut accueillir l'objet, on en créer une nouvelle.
Sur les exemples, les résultats sont confondues avec l'algorithme Best Fit. Avec notre exemple choisis, on voit que cet algorithme est inefficace car tous les objets vont être stocké dans des boîtes différentes laissant de gros espaces vide non utilisé. Néanmoins avec une taille plus importante des boîtes, cet algorithme est moins efficace.
Cet algorithme est utilisé dans l'allocation de mémoire en informatique car il a le meilleur rapport efficacité/temps de calcul.

Cependant, pour notre industriel le critère du temps de calcul n'est pas recherché contrairement à l'efficacité.

c. Best Fit

Cet algorithme consiste à prendre les objets un par un et les placer dans la boîte la plus remplie pouvant l'accueillir. Si aucune boîte ne peut accueillir l'objet, on en créer une nouvelle.
Sur tous les tests effectués, cet algorithme est le plus efficace en terme de rangement. Cependant, pour chaque objet on va parcourir toutes les boîtes et est donc plus lent que First Fit qui s'arrête lorsqu'une boîte peut accueillir l'objet. La différence importante avec First Fit est que cet algorithme va chercher à maximiser le contenu des boîtes existantes pour éviter le "gaspillage"

d. Worst Fit

Cet algorithme consiste à prendre les objets un par un et les placer dans la boîte la moins remplie pouvant l'accueillir. Si aucune boîte ne peut accueillir l'objet, on en crée une nouvelle.
Celui-ci est moins efficace que First Fit et Best Fit sur tous les tests effectués. Il est de la même complexité que First Fit.
Il est donc l'inverse de ce que fait First Fit mais est moins efficace que ça soit en gestion de mémoire en informatique ou pour du transport d'objet comme ici.

e. Almost Worst Fit

Cet algorithme consiste à prendre les objets un par un et les placer dans la seconde boîte la moins remplie pouvant l'accueillir. Si aucune boîte ne peut accueillir l'objet, on en crée une nouvelle. Dans le pire des cas, pour un objet, on va également parcourir toutes les boîtes.
Sur les tests effectués, cet algorithme est plus efficace que Worst Fit mais toujours pas autant que First Fit et Best Fit.

III. Conclusion

Dans un premier temps, voici le classement des différentes solutions, du plus performant au moins performant :

1. Best Fit
2. First Fit
3. Almost Worst Fit
4. Worst Fit
5. Next Fit

Ainsi, la solution préconisée est d’utiliser l’algorithme Best Fit pour optimiser le transport des objets à traiter.

Ceci étant dit, il faut quand même souligné que si jamais la taille des boîtes utilisées venaient à ne plus être constante, nous serions alors dans un tout nouveau cas d’étude. Les résultats seraient bien différents et il n’y aurait pas de meilleure solution apparente, puisque chaque algorithme présenterait des avantages et des inconvénients selon l’exemple dans lequel il serait utilisé.

Typiquement ce sont dans les problèmes d'allocation mémoire que nous pouvons observer des boîtes aux tailles variables qui correspondent aux blocs mémoires non alloués.