Rapport du TP OPTIMISATION Pt.2: Methodes Heuristiques

BACHI Yasmine (CdE) SAADI Fatma Zohra Khaoula
NOUALI Sarah MOUSSAOUI Meroua
MIHOUBI Lamia Zohra

13 mai 2020

Table des matières

I Worse case Analysis **1**

II Méthodes Heuristiques **3**

1 Introduction **3**

2 Next Fit (NF) **4**
2.1 principe 4
2.2 Pseudo-Code 4

3 Next Fit Decreasing (NFD) **4**
3.1 principe 4
3.2 Pseudo-Code 4

4 First Fit (FF) **5**
4.1 principe 5
4.2 Pseudo-Code 6

5 First Fit Descreasing (FFD) **6**
5.1 principe 6
5.2 Pseudo-Code 6

6 Best Fit (BF) **7**
6.1 principe 7
6.2 Pseudo-Code 8

7 Best Fit Decreasing (BFD) **8**
7.1 principe 8
7.2 Pseudo-Code 8

8 Résultats des tests **10**
8.1 Analyse des résultats par rapport au temps d'exécution 10
8.2 Analyse des résultats par rapport à la qualité de la solution . . 14
8.3 Conclusion 16

Première partie

Worse case Analysis

Les méthodes approchées cherchent à trouver une solution la plus proche possible de la solution optimale, pour mesurer la qualité de cette solution obtenue, nous utiliserons l'analyse du pire des cas *"worst case analysis"*.

Dans cette analyse, les performances d'un algorithme sont mesurées par l'écart de la solution du pire cas (l'instance où l'algorithme donne la pire solution) à la solution optimale. L'une des métriques les plus utilisées dans l'analyse du pire des cas est le *Worse case Ratio*.

Worse case Ratio

Ce rapport mesure la déviation maximal de la solution obtenue par l'heuristique, par rapport à la solution optimale.

Soit :

- **L** :une instance du bin packing.
- **A(L)** :le nombre de boîtes utilisées en appliquant l'heuristique A sur L.
- **OPT(L)** :le nombre de boîtes optimal.

Le rapport est donné par la formule suivante :

$$Ra \equiv \{r \geq 1 : \frac{A(L)}{OPT(L)} \geq r \, pour \, toutes \, les \, instances \, L\}$$

Pour calculer le ratio, on calcul le rapport $A(L)/OPT(L)$ pour chaque instance, ensuite on prend le plus petit des majorants de ces rapports. Il est claire que la valeur idéale c'est un "1" qui correspond au cas où la solution obtenue est égale à la solution optimale pour toutes les instances, et dès que la solution obtenue s'éloigne de la solution optimale le rapport va augmenter.

Deuxième partie

Méthodes Heuristiques

1 Introduction

Les heuristiques sont des méthodes spécifiques qui exploitent au mieux la structure du problème dans le but de trouver une solution raisonnable (non nécessairement optimale) en un temps réduit. L'utilisation de ce type d'algorithmes s'impose car les méthodes de résolution exactes sont de complexité exponentielle, et échouent à trouver la solution pour des instances de tailles moyennes voir petites, comme on la constater lors du chapitre précédant . L'usage des heuristiques est donc pertinent pour surmonter ces limites.

Dans ce chapitre, nous allons présenter la conception détaillée des heuristiques sur lesquelles notre choix d'implémentation s'est porté et qui sont :

1. Next Fit (NF)
2. Next Fit Decreasing (NFD)
3. First Fit (FF)
4. First Fit Descreasing (FFD)
5. Best Fit (BF)
6. Best Fit Decreasing (BFD)

Dans le but d'explorer ces méthodes, comparer leurs performances, montrer leurs avantages et découvrir leurs limites , nous effectuerons des tests empiriques et comparatifs sur les mêmes benchmarks utilisés pour les tests des méthodes exactes.(Benchmark Scholl)

2 Next Fit (NF)

2.1 principe

Si l'article tient dans la même boite que l'article précédent, il est placé avec ce dernier. Sinon, on ouvre une nouvelle boite et le mettre là-dedans.

— NF est un algorithme simple d'une complexité de O(n).

2.2 Pseudo-Code

Algorithm 1: Next Fit

for Tous les articles i = 1, 2,. . . , n **do**
  if l'article i s'inscrit dans la boîte actuelle **then**
    Ranger l'article i dans la boîte actuelle
  else
    Créer une nouvelle boîte, en faire la boîte actuelle et ranger l'article i dedant.
  end if
end for

3 Next Fit Decreasing (NFD)

3.1 principe

Le NFD est une amélioration de l'algorithme Next-Fit. Cet algorithme ordonne es articles par ordre décroissant des poids,puis applique l'algorithme NF.

3.2 Pseudo-Code

Algorithm 2: Next Fit Decreasing

Triez les articles par ordre décroissant
Appliquer Next-Fit à la liste triée

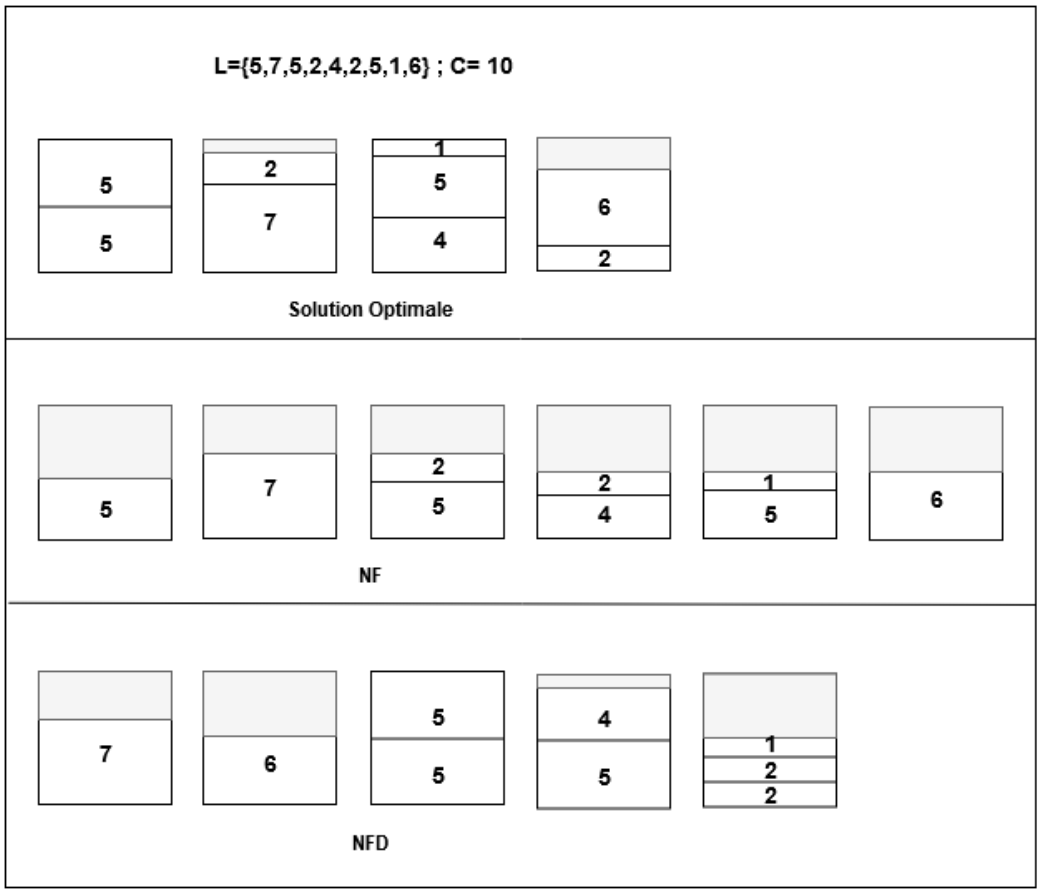

FIGURE 1 – Exemple NF et NFD

4 First Fit (FF)

4.1 principe

Ranger chaque article courant dans la première boîte, entre celles déjà ouvertes, qui peut le contenir sinon ouvrir une nouvelle boîte et on le range dedans.

— L'algorithme First Fit implémenté a une complexité de $O(n^2)$.

4.2 Pseudo-Code

Algorithm 3: First Fit

```
for Tous les articles i = 1, 2,. . . , n do
  for Tous les boîtes j = 1, 2,. . . m do
    if l'article i s'inscrit dans la boîte j then
      Ranger l'article i dans la boîte j
      Quitter la boucle ( passer à l'article suivant)
    end if
  end for
  if l'article i ne rentre dans aucune boîte disponible then
    Créer une nouvelle boîte et ranger l'article i dedans
  end if
end for
```

5 First Fit Descreasing (FFD)

5.1 principe

Le FFD est une amélioration de l'algorithme First-Fit . Cet algorithme ordonne les poids dans le sens décroissant puis lui applique l'algorithme FF.

— L'algorithme First Fit peut être implémenté en O(nlog n) en utilisant les arbres de recherche binaires

5.2 Pseudo-Code

Algorithm 4: First Fit Decreasing

```
Triez les articles par ordre décroissant
Appliquer First-Fit à la liste triée
```

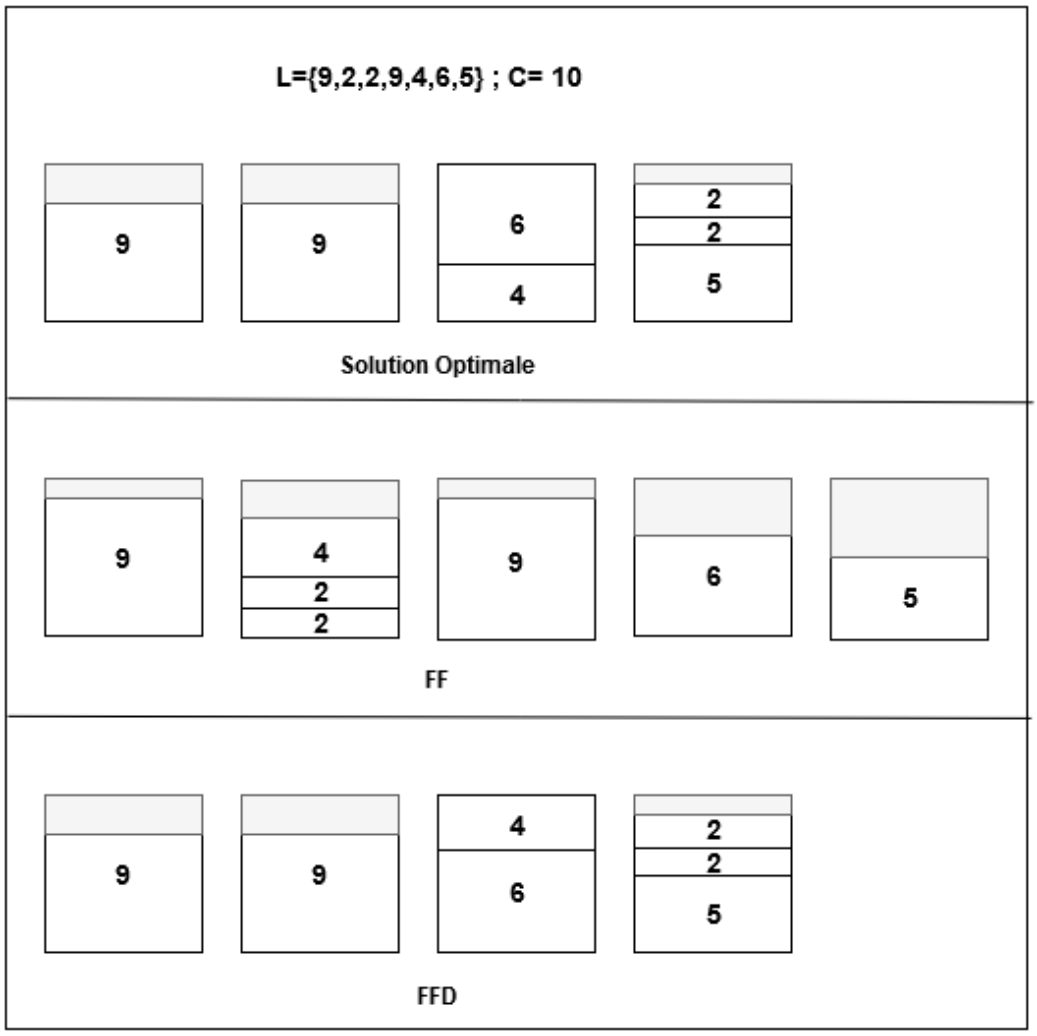

FIGURE 2 – Exemple FF et FFD

6 Best Fit (BF)

6.1 principe

Ranger chaque article courant dans la boîte la mieux remplie, entre celles déjà ouvertes, qui peut le contenir sinon ouvrir une nouvelle boîte et on le range dedans.

— L'algorithme Best Fit implémenté a une complexité de $O(n^2)$.

6.2 Pseudo-Code

Algorithm 5: Best Fit

for Tous les articles i = 1, 2,. . . , n **do**
  for Tous les boîtes j = 1, 2,. . . m **do**
    if l'article i s'inscrit dans la boîte j **then**
      Calculer la capacité restante dans la boîte j une fois l'article
    end if
  end for
  Ranger l'article i dans la boîte j, où j est la boîte ayant la capacité restante minimale après avoir ajouté l'article(c'est-à-dire que "l'article convient le mieux").
  if une telle boîte n'existe pas (l'article ne peut être rangé dans aucune boîte) **then**
    Créer une nouvelle boîte et ranger l'article i dedans
  end if
end for

7 Best Fit Decreasing (BFD)

7.1 principe

Le BFD est une amélioration de l'algorithme Best-Fit . Cet algorithme ordonne les poids dans le sens décroissant puis applique l'algorithme BF.

7.2 Pseudo-Code

Algorithm 6: Best Fit Decreasing

Triez les articles par ordre décroissant
Appliquer Best-Fit à la liste triée

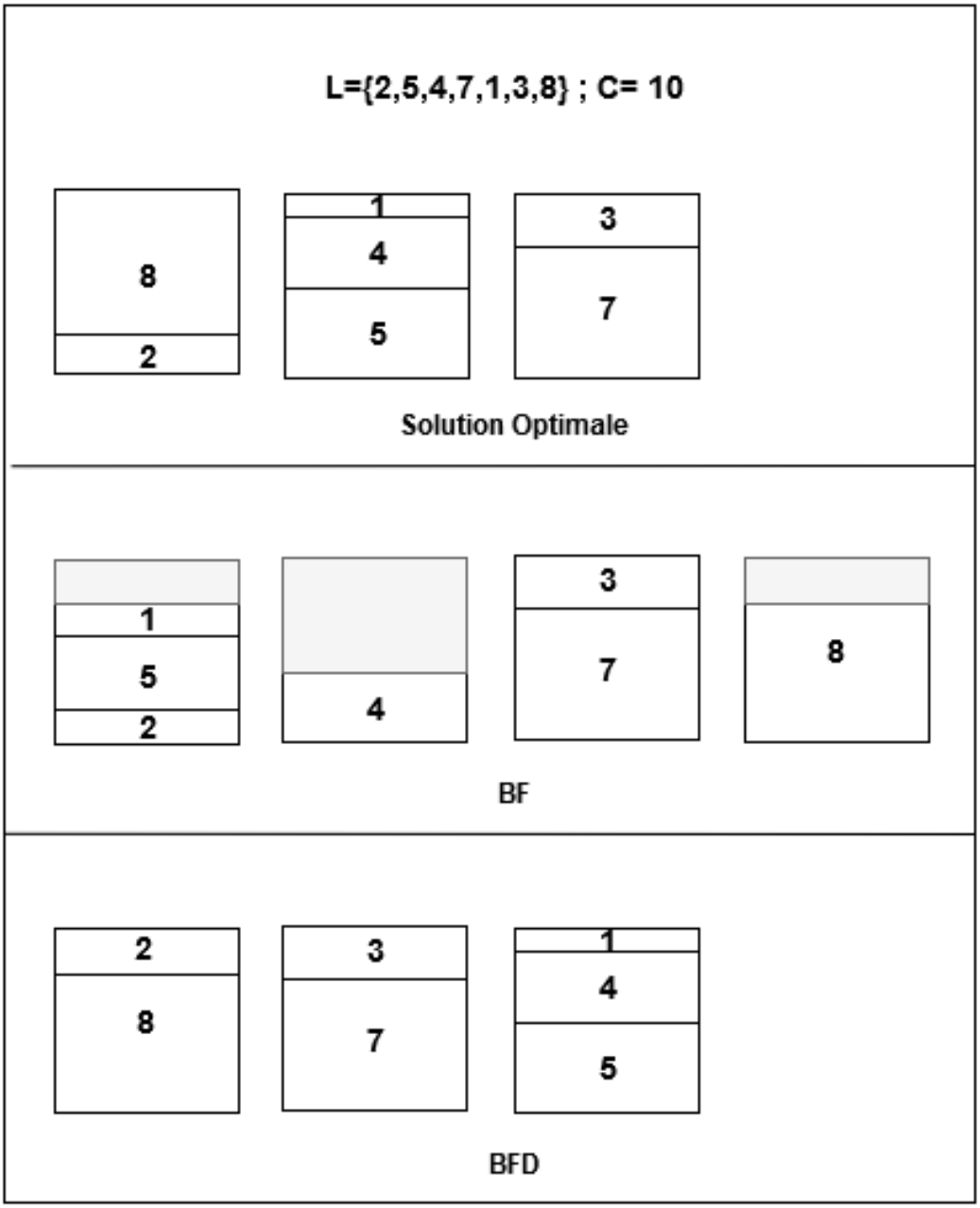

FIGURE 3 – Exemple BF et BFD

8 Résultats des tests

Dans cette partie, nous allons comparer les performances de nos algorithmes implémentés : *Next-Fit(NF), Next-Fit Decreasing(NFD), First-Fit(FF), First-Fit Decreasing(*FF*), Best-Fit(BF) et Best-Fit Decreasing (BFD).*

Pour pouvoir faire une bonne comparaison avec les autres méthodes de résolution du problème du Bin Packing, on a trouvé judicieux de prendre les même instances du benchmark Scholl.

Remarque : les méthodes ont été développées en utilisant le langage de programmation **python**, et exécutées sur un **DELL Inspiron15 [Intel® Core™ i7-8550U CPU @ 1.80GHz×8, 8Go]**

Pour pouvoir comparer entre les performances des différentes méthodes heuristiques, notre étude se portera sur 2 axes :

— Le temps d'exécution.

— La qualité de la solution.

8.1 Analyse des résultats par rapport au temps d'exécution

les résultats en temps d'exécution sont présentés dans le tableau suivant :

N	N=50					
C	NF	NFD	FF	FFD	BF	BFD
100	0.000144	0.000127	0.000190	0.000184	0.000241	0.000215
120	0.000124	0.000130	0.000160	0.000167	0.000198	0.000213
150	9.665489	9.436607	0.000126	0.000124	0.000150	0.000149
1000	0.000140	0.000116	0.000269	0.000216	0.000214	0.000225

N	N=100					
C	NF	NFD	FF	FFD	BF	BFD
100	0.000464	0.000470	0.000713	0.000715	0.000793	0.000787
120	0.000494	0.000480	0.000644	0.000574	0.000695	0.000771
150	0.000329	0.000362	0.000402	0.000397	0.000483	0.000494
1000	0.000427	0.000361	0.000690	0.000760	0.000945	0.000751

N	N=200					
C	NF	NFD	FF	FFD	BF	BFD
100	0.001325	0.001391	0.001916	0.001862	0.002066	0.002173
120	0.001197	0.001158	0.001533	0.001542	0.001815	0.001871
150	0.000745	0.000760	0.001045	0.001219	0.001976	0.001678
1000	0.000800	0.000801	0.001478	0.001676	0.001954	0.001746
100000	0.000739	0.000702	0.001153	0.001635	0.001162	0.001505

N	N=500					
C	NF	NFD	FF	FFD	BF	BFD
100	0.007068	0.007047	0.010820	0.010633	0.011144	0.010667
120	0.006306	0.006812	0.008905	0.008931	0.010936	0.010151
150	0.003895	0.003845	0.006202	0.005959	0.006320	0.005925
1000	0.002363	0.002021	0.003468	0.003332	0.003123	0.003095

FIGURE 4 – Tableau des temps d'exécution des heuristiques

Pour faciliter la lecture des résultats, l'utilisation d'un graphique s'impose. Ci-dessous un histogramme représentant les temps d'exécution en fonction des instances pour chaque heuristique :

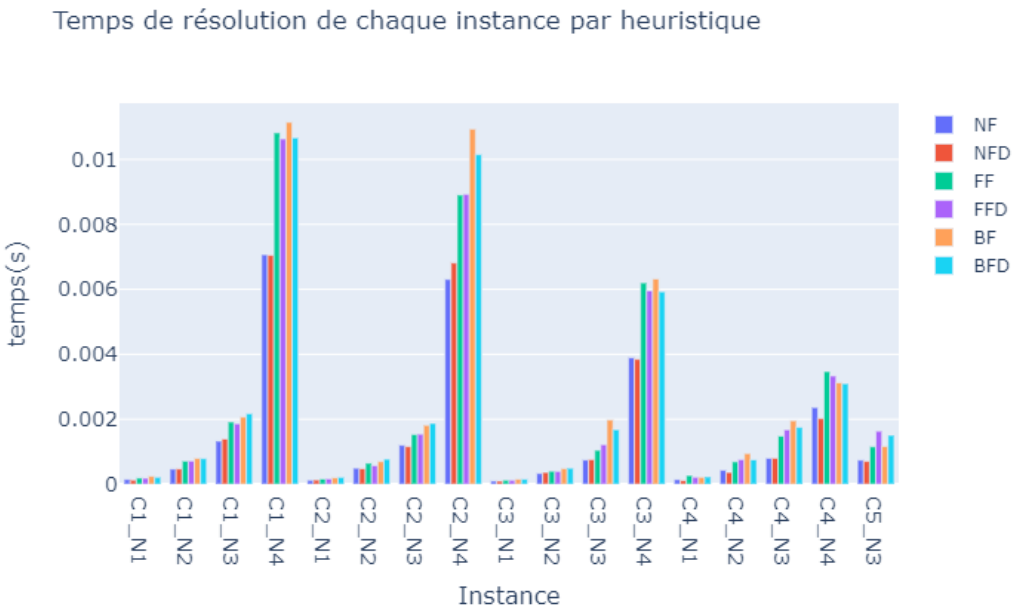

FIGURE 5 – Histogramme des temps d'exécution des heuristiques

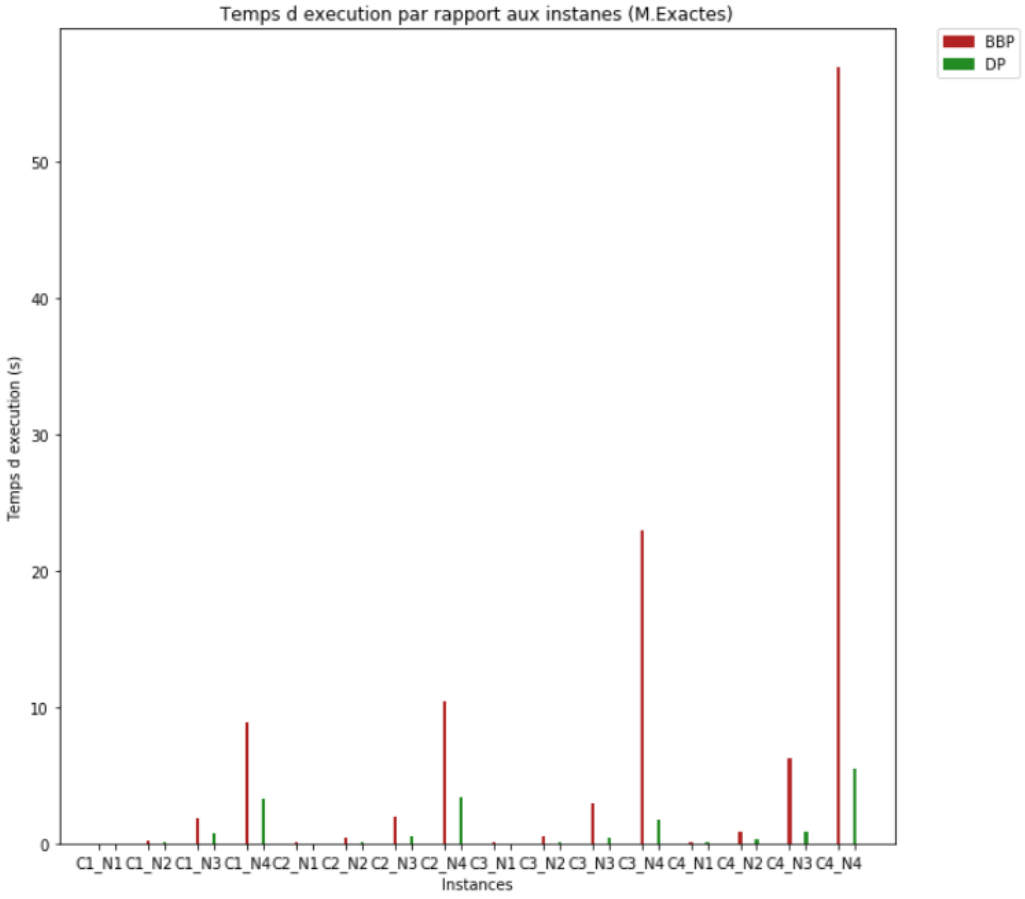

FIGURE 6 – Comparaison des temps d'exécution des heuristiques avec les méthodes exactes

8.1.1 Analyse des résultats

- En augmentant la complexité du problème *(N et C)*, le temps d'exécution des heuristiques augmente, mais tout en restant incomparable avec celui des méthodes exactes ***[figure 06]***.
- Toutes les méthodes heuristiques arrivent rapidement à trouver une solution aux instances du problème pour les trois classes d'instances du Benchmark Scholl (moins de 0.012s).
- Les performances de NF et NFD sont meilleures que celles des autres méthodes, avec le BF et BFD qui consomment le plus de temps, dans la plupart du temps, pour trouver une solution.
- Les heuristiques *FF* et *FFD* s'exécutent en des temps légèrement meilleurs que BF et BFD mais moins rapides que NF et NFD.

8.1.2 Interprétation des résultats

- Les algorithmes BF et BFD nécessitent plus de temps car le principe de BF repose sur le fait qu'il faut d'abord parcourir toutes les boîtes déjà ouvertes avant de prendre une décision (ranger un article).
- Les algorithmes NF et NFD sont les plus rapide car le principe de NF repose sur le fait que la décision où mettre l'article concerne seulement la dernière boîte ouverte, donc on n'a pas à parcourir l'ensemble des boîtes pour chaque article.
- Les algorithmes *FF* (resp *FFD*) impose de parcourir partiellement la liste des boîtes ouvertes jusqu'à trouver la 1ère boîte qui convient, ce qui justifie le temps d'exécution moyen (entre celui de BF et NF).

8.2 Analyse des résultats par rapport à la qualité de la solution

Pour cela, on utilisera la métrique Worse case Ratio **[voir Partie 01]**

Remarque : Vu que les instances du benchmark Scholl contiennent des articles déjà ordonnés, les versions *online (NF,FF,BF)* et *offline (NFD,* FF,*BFD)* des heuristiques donnent exactement les même résultats (car la différence entre les deux c'est l'étape d'ordonnancement des articles). Dans cette partie nous allons nous contenter d'étudier la qualité de la solution des algorithmes *onlines*.

Ci-dessous un tableau récapitulatif des ratios obtenus pour chaque heuristique sur l'ensemble des instances du benchmark ***[figure 07]*** , ainsi qu'une représentation graphique (en histogramme) de ces résultats ***[figure 08]*** :

Instance	Ratio		
	BF	FF	NF
C1_N1	1.0625	1.0625	1.4375
C1_N2	1.0	1.0	1.428571
C1_N3	1.011494	1.011494	1.396825
C1_N4	1.005988	1.005988	1.396449
C2_N1	1.0	1.0	1.125
C2_N2	1.071428	1.071428	1.190476
C2_N3	1.0	1.0	1.137931
C2_N4	1.0	1.0	1.098591
C3_N3	1.072727	1.072727	1.178571

FIGURE 7 – Tableau des ratios obtenues par les heuristiques

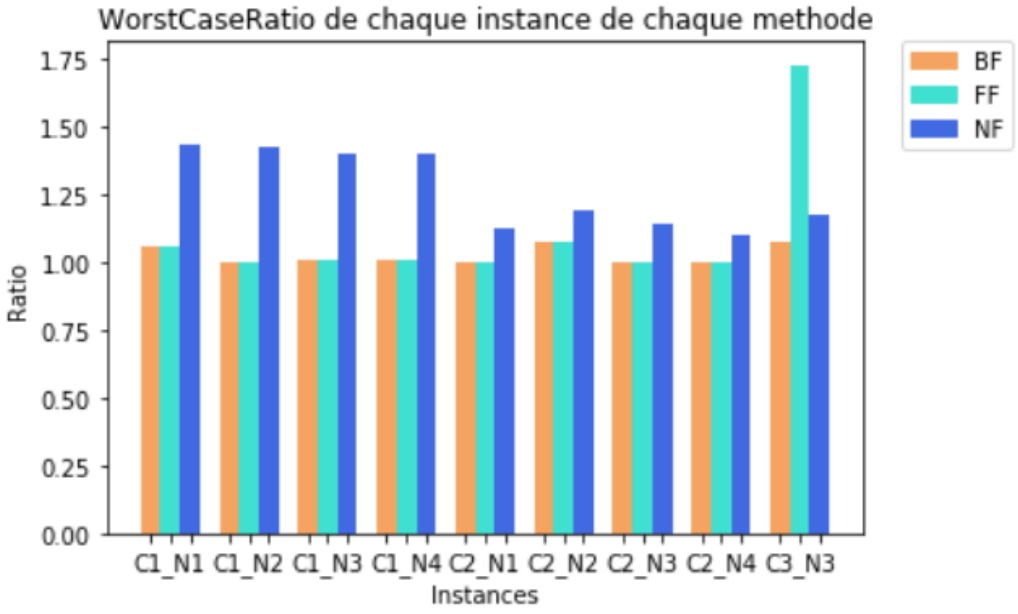

FIGURE 8 – Histogramme des ratios des heuristiques en fonction des instances

Ci-dessous une représentation graphique (en histogramme) qui représente le ratio de toutes les instances du Benchmark *Scholl*, toutes classes confondues par heuristique (BF,FF et NF) :

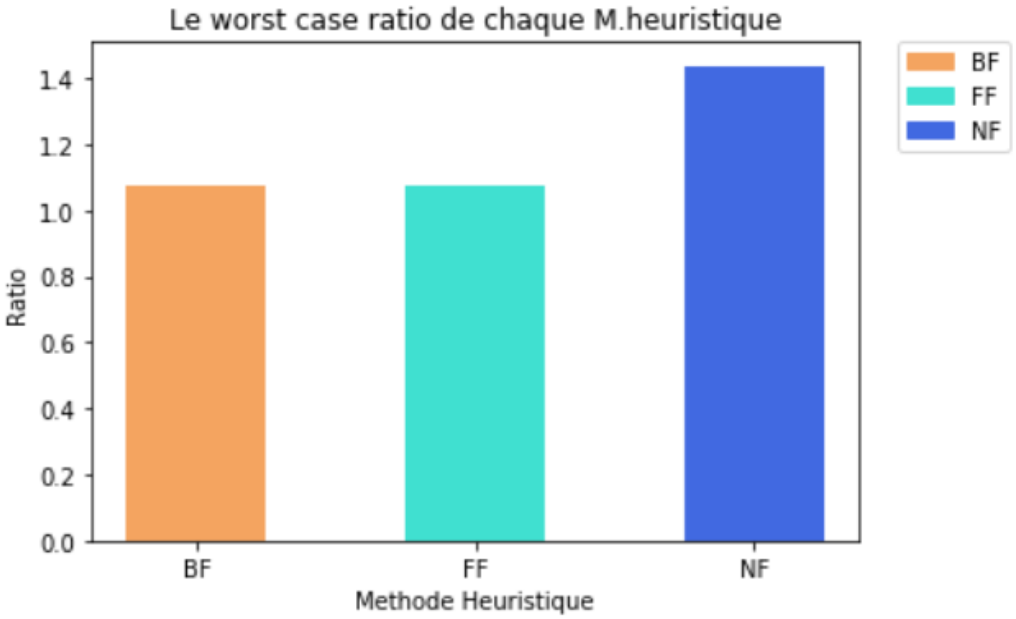

FIGURE 9 – Histogramme des ratios des heuristiques pour tout le benchmark Scholl

8.2.1 Analyse des résultats

— Les ratios obtenus pour *BF* et *FF* (*BFD* et *FF* resp) sont identiques et diffèrent de *NF (NFD)*.
— Les heuristiques *BF, FF* (et respectivement *BFD*, *FF*) arrivent pour certains types d'instances à trouver la valeur optimales du problème (ratio égale 1), contrairement à *NF* (respectivement *NFD*) qui ne trouve pas assez souvent la solution (ratio supérieure à 1).

8.2.2 Interprétation des résultats

La différence dans la qualité de la solution obtenue est due aux nombres de boîtes considérés pour prendre une décision qui est plus large dans First Fit et Best Fit (toutes les boîtes ouvertes peuvent accueillir l'article), par contre dans Next Fit seulement la dernière boîte peut accueillir l'article.

8.3 Conclusion

En exécutant les heuristiques étudiées (*FF, NF ,BF* et leurs versions *offline*) sur les instances du benchmark Scholl, on a trouvé que l'heuristique NF est la plus rapide à s'exécuter, mais elle donne la plus mauvaise qualité

de solution. Par contre les heuristiques *FF* et *BF* sont moins rapides (avec *BF* légèrement moins rapide que *FF*) mais offrent une meilleure qualité.

Comme on a pu le constater durant ce chapitre, les méthodes heuristiques de type *online (FF, NF,BF)* et de type *offline (FF, NFD, BFD)* donnent de très bon résultats par rapport au temps d'exécution. Mais l'un des inconvénient avec les méthodes heuristiques c'est qu'elles n'assurent pas la qualité de la solution.

Ces algorithmes sont connu sous le nom d'algorithmes gloutons, c'est à dire qu'ils cherchent à trouver une solution dans un temps très réduit, mais ne donne pas d'assurance sur la qualité de cette solution. C'est pour cela que ces méthodes sont généralement utilisées pour initialiser d'autres méthodes plus sophistiquées comme les métaheuristiques.