IN4301 Advanced Algorithm - Homework Assignment 2

> Lu Dai (4506677)   L.Dai-1@student.tudelft.nl

Question 1

1.
According to the algorithms, a new bin is added only when the the current used bins can't load the current item. Let $L_1, L_2, \ldots, L_{k-1}, L_k$ denote the total load in bin $b_1, b_2, \ldots, b_{k-1}, b_k$ repsectively. Therefore,

> For b_1, b_2, we know $l_1 + l_2 > 1$
> For b_1, b_2, b_3,, we know $l_1 + l_3 > 1, l_2 + l_3 > 1, l_1 + l_2 > 1$
> ...
> For $b_1, b_2, b_3, \ldots, b_k$, the sum of total load of any pair of bins is larger than 1. This means $l_i + l_j > 1$ $_{(i \neq j,\ 1 \leq i \leq k,\ 1 \leq j \leq k)}$

- if k is odd:
  $\sum_{i=1}^{n} w_i = (l_1 + l_2) + (l_3 + l_4) + \ldots + (l_{k-2} + l_{k-1}) + l_k \geq \frac{k-1}{2} + l_k \geq \frac{k-1}{2}$
- if k is even:
  $\sum_{i=1}^{n} w_i = (l_1 + l_2) + (l_3 + l_4) + \ldots + (l_{k-1} + l_k) \geq \frac{k}{2}$

Above all, $\sum_{i=1}^{n} w_i \geq \lfloor \frac{k}{2} \rfloor$. The total load of the k bins used by A is at least $\lfloor \frac{k}{2} \rfloor$.

2.
From previous results, we know $\sum_{i=1}^{n} w_i \geq \lfloor \frac{A(x)}{2} \rfloor \geq \frac{A(x)-1}{2}$.
As the optimal number of bins is opt(x), so $\sum_{i=1}^{n} w_i \leq opt(x) * 1$.
Therefore,

$$\frac{A(x) - 1}{2} \leq \sum_{i=1}^{n} w_i \leq opt(x) * 1$$

$$A(x) \leq 2 * opt(x) + 1$$

3.
There are only three sizes $\frac{1}{7} + \epsilon$, $\frac{1}{3} + \epsilon$ and $\frac{1}{2} + \epsilon$ of m items respectively and sequentially. Consider ϵ is very small, we discuss all the possible situations of items which a bin can load and the space wasted:

No.	Items	Waste space	No.	Items	Waste space
1	6 ($\frac{1}{7} + \epsilon$)	$\sim \frac{1}{7}$	4	4 ($\frac{1}{7} + \epsilon$) + 1 ($\frac{1}{3} + \epsilon$)	$\sim \frac{2}{21}$
2	2 ($\frac{1}{3} + \epsilon$)	$\sim \frac{1}{3}$	5	2 ($\frac{1}{7} + \epsilon$) + 2 ($\frac{1}{3} + \epsilon$)	$\sim \frac{1}{21}$
3	1 ($\frac{1}{2} + \epsilon$)	$\sim \frac{1}{2}$	6	3 ($\frac{1}{7} + \epsilon$) + 1 ($\frac{1}{2} + \epsilon$)	$\sim \frac{1}{14}$
7	1 ($\frac{1}{3} + \epsilon$) + 1 ($\frac{1}{2} + \epsilon$)	$\sim \frac{1}{6}$	8	1 ($\frac{1}{7} + \epsilon$) + 1 ($\frac{1}{3} + \epsilon$) + 1 ($\frac{1}{2} + \epsilon$)	$\sim \frac{1}{42}$

From the table, we know that use **any combination is always better than packing items with the same size**. So the worst case is that a bin is always loaded with items of the same size, while the best case is the 8th combination which minimize the total waste space.

- Worst case:
  $A_{worst}(x) = \frac{m}{6} + \frac{m}{2} + \frac{m}{1} = \frac{5}{3}\ m$
- Best Case:
  $opt(x) = \frac{3m}{3} = m$

Thus, $A(x) \le \frac{5}{3}\ opt(x)$ means the performance ratio on such instances is at least $\frac{5}{3}$.

4.
Let L_i denotes the total weight in bin b_i. Next-Fit gets the results $(L_1, L_2, \dots, L_{k-1}, L_k)$. According to its definition, we can easily deduce that the sum of weight of neighboring bins is always larger than 1, which is $L_i + L_{i+1} > 1$. This happens in only two situations,

- (1) $L_i > \frac{1}{2}, L_{i+1} > \frac{1}{2}$
- (2) $(L_i > \frac{1}{2}, L_{i+1} \le \frac{1}{2})\ \ or\ \ (L_i \le \frac{1}{2}, L_{i+1} > \frac{1}{2})$

For situation (1), no bins can be merged anymore, thus the result stays k.
For situation (2). Intuitively, in order to have more bins merged, we should have more bins whose total load $< \frac{1}{2}$.
At most, there could be $\lceil \frac{k}{2} \rceil$ bins which has total load less than $\frac{1}{2}$. And at least $\lfloor \frac{k}{2} \rfloor$ bins larger than $\frac{1}{2}$. The best case is to merge these$\lceil \frac{k}{2} \rceil$ bins togethers into 1 bin, which reaches a $\lceil \frac{k}{2} \rceil - 1$ decrease in total number of bins. Then the optimized result become $\lfloor \frac{k}{2} \rfloor + 1$ bins. Thus,

$$\frac{k}{2} + 1 \ge \lfloor \frac{k}{2} \rfloor + 1 \ge OPT$$

$$k \ge 2 * (OPT - 1) = (2 - \frac{2}{OPT}) * OPT$$

Thus, the performance ratio of Next-Fit can't be better than $2 - \frac{2}{OPT}$.

Question 2

As X is an NP-hard minimization problem, thus for every instance x, $c(x) \ge opt(x)$.
Suppose there exist an FPTAS algorithm with approximation ratio $1 + \epsilon\ \ (\epsilon \ge 0)$, thus $c(x) = (1 + \epsilon) * opt(x)$. As it is given that $opt(x) \le p(|x|)$, thus

$$opt(x) \le c(x) = opt(x) + \epsilon * opt(x) \le opt\ (\ x) + \epsilon * p(|x|)$$

.
As there exists some instances holding $\epsilon^{-1} > p(|x|)$, which is $\epsilon * p(|x|) < 1$.
If for some small instances, . Then $opt(x) \le c(x) \le \epsilon * p(|x|) < 1$. Thus,

$$opt(x) <= c(x) < opt(x) + 1.$$

As it is given c(x) is always an interger, thus it is obviously $c(x) = opt(x)$.