Bin-Packing

K. Subramani[1]

[1] Lane Department of Computer Science and Electrical Engineering
West Virginia University

March 17, 2014

Outline

Outline

Outline

Outline

Topics

Topics

Outline

Topics

Outline

1. Problem definition (Offline and Online versions).

Topics

Outline

1. Problem definition (Offline and Online versions).
2. Lower bounds on online performance.

Topics

Outline

1. Problem definition (Offline and Online versions).
2. Lower bounds on online performance.
3. The Next Fit (NF) algorithm and analysis.

Topics

Outline

1. Problem definition (Offline and Online versions).
2. Lower bounds on online performance.
3. The Next Fit (NF) algorithm and analysis.
4. The First Fit (FF) algorithm and analysis.

Topics

Outline

1. Problem definition (Offline and Online versions).
2. Lower bounds on online performance.
3. The Next Fit (NF) algorithm and analysis.
4. The First Fit (FF) algorithm and analysis.
5. The Best Fit (FF) algorithm and analysis.

Topics

Outline

1. Problem definition (Offline and Online versions).
2. Lower bounds on online performance.
3. The Next Fit (NF) algorithm and analysis.
4. The First Fit (FF) algorithm and analysis.
5. The Best Fit (FF) algorithm and analysis.
6. The First Fit Decreasing (FFD) algorithm and analysis.

Topics

Outline

1. Problem definition (Offline and Online versions).
2. Lower bounds on online performance.
3. The Next Fit (NF) algorithm and analysis.
4. The First Fit (FF) algorithm and analysis.
5. The Best Fit (FF) algorithm and analysis.
6. The First Fit Decreasing (FFD) algorithm and analysis.
7. An inapproximability result.

Problem definition

Problem Statement

Problem definition

Problem Statement

We are given n objects $\{s_1, s_2, \ldots s_n\}$, such that $0 < s_i \leq 1$ and an unlimited supply of unit sized bins. The goal is to pack the objects into bins, minimizing the number of bins used.

Problem definition

Problem Statement

We are given n objects $\{s_1, s_2, \ldots s_n\}$, such that $0 < s_i \leq 1$ and an unlimited supply of unit sized bins. The goal is to pack the objects into bins, minimizing the number of bins used.

Note

There are two versions of this problem, viz., offline and online.

Problem definition

Problem Statement

We are given n objects $\{s_1, s_2, \ldots s_n\}$, such that $0 < s_i \leq 1$ and an unlimited supply of unit sized bins. The goal is to pack the objects into bins, minimizing the number of bins used.

Note

There are two versions of this problem, viz., offline and online. In the former, the complete input is presented before the algorithm commences.

Problem definition

Problem Statement

We are given n objects $\{s_1, s_2, \ldots s_n\}$, such that $0 < s_i \leq 1$ and an unlimited supply of unit sized bins. The goal is to pack the objects into bins, minimizing the number of bins used.

Note

There are two versions of this problem, viz., offline and online. In the former, the complete input is presented before the algorithm commences. In the latter, the input is presented one object at a time.

Performance bounds on the online version

Performance bounds on the online version

Intuition

Performance bounds on the online version

Intuition

M "small" items of size $\frac{1}{2} - \varepsilon$, followed by M "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.

Performance bounds on the online version

Intuition

M "small" items of size $\frac{1}{2} - \varepsilon$, followed by M "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.

Theorem

There exist inputs that can force ANY online bin-packing algorithm to use at least $\frac{4}{3}$ times the optimal number of bins.

Performance bounds on the online version

Intuition

M "small" items of size $\frac{1}{2} - \varepsilon$, followed by M "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.

Theorem

There exist inputs that can force ANY online bin-packing algorithm to use at least $\frac{4}{3}$ times the optimal number of bins.

Observation

Since we (the adversary) can truncate the input whenever we like, the algorithm must maintain its guaranteed ratio **at all** *points during its course.*

Limits of online algorithms

Limits of online algorithms

Inapproximability in online algorithms

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously:

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously:
   $I1$: M "small" items of size $\frac{1}{2} - \varepsilon$, followed by $I2$: M "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously: $I1\ :\ M$ "small" items of size $\frac{1}{2} - \varepsilon$, followed by $I2\ :\ M$ "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.
2. To handle I_1, the optimal algorithm has used $\frac{M}{2}$ bins.

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously:
   $I1$: M "small" items of size $\frac{1}{2} - \varepsilon$, followed by $I2$: M "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.
2. To handle I_1, the optimal algorithm has used $\frac{M}{2}$ bins. Let the online algorithm have used b bins.

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously:
   $I1$: M "small" items of size $\frac{1}{2} - \varepsilon$, followed by $I2$: M "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.
2. To handle I_1, the optimal algorithm has used $\frac{M}{2}$ bins. Let the online algorithm have used b bins.
3. In order to beat the $\frac{4}{3}$ ratio, we must have

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously: $I1 : M$ “small” items of size $\frac{1}{2} - \varepsilon$, followed by $I2 : M$ “large” items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.
2. To handle I_1, the optimal algorithm has used $\frac{M}{2}$ bins. Let the online algorithm have used b bins.
3. In order to beat the $\frac{4}{3}$ ratio, we must have $\frac{b}{\frac{M}{2}} < \frac{4}{3}$

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously:
   $I1\ :\ M$ "small" items of size $\frac{1}{2} - \varepsilon$, followed by $I2\ :\ M$ "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.
2. To handle I_1, the optimal algorithm has used $\frac{M}{2}$ bins. Let the online algorithm have used b bins.
3. In order to beat the $\frac{4}{3}$ ratio, we must have $\frac{b}{\frac{M}{2}} < \frac{4}{3} \Rightarrow \frac{b}{M} < \frac{2}{3}$

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously:
   $I1\ :\ M$ "small" items of size $\frac{1}{2} - \varepsilon$, followed by $I2\ :\ M$ "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.
2. To handle I_1, the optimal algorithm has used $\frac{M}{2}$ bins. Let the online algorithm have used b bins.
3. In order to beat the $\frac{4}{3}$ ratio, we must have $\frac{b}{\frac{M}{2}} < \frac{4}{3} \Rightarrow \frac{b}{M} < \frac{2}{3}$
4. Now consider the state of the online and optimal algorithms after processing I_2.

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously: $I1\ :\ M$ "small" items of size $\frac{1}{2} - \varepsilon$, followed by $I2\ :\ M$ "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.
2. To handle I_1, the optimal algorithm has used $\frac{M}{2}$ bins. Let the online algorithm have used b bins.
3. In order to beat the $\frac{4}{3}$ ratio, we must have $\frac{b}{\frac{M}{2}} < \frac{4}{3} \Rightarrow \frac{b}{M} < \frac{2}{3}$
4. Now consider the state of the online and optimal algorithms after processing I_2. The optimal uses M bins total.

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously:
   $I1$: M "small" items of size $\frac{1}{2} - \varepsilon$, followed by $I2$: M "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.
2. To handle I_1, the optimal algorithm has used $\frac{M}{2}$ bins. Let the online algorithm have used b bins.
3. In order to beat the $\frac{4}{3}$ ratio, we must have $\frac{b}{\frac{M}{2}} < \frac{4}{3} \Rightarrow \frac{b}{M} < \frac{2}{3}$
4. Now consider the state of the online and optimal algorithms after processing I_2. The optimal uses M bins total.
5. Every **new** bin that the online algorithm opens after the first b bins, has at most 1 item in it.

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously: $I1\ :\ M$ "small" items of size $\frac{1}{2} - \varepsilon$, followed by $I2\ :\ M$ "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.
2. To handle I_1, the optimal algorithm has used $\frac{M}{2}$ bins. Let the online algorithm have used b bins.
3. In order to beat the $\frac{4}{3}$ ratio, we must have $\frac{b}{\frac{M}{2}} < \frac{4}{3} \Rightarrow \frac{b}{M} < \frac{2}{3}$
4. Now consider the state of the online and optimal algorithms after processing I_2. The optimal uses M bins total.
5. Every **new** bin that the online algorithm opens after the first b bins, has at most 1 item in it.
6. Since only the first b bins can have 2 items and the remaining bins have 1 item each, packing $2 \cdot M$ items will require at least $(2 \cdot M - b)$ bins.

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously: $I1\ :\ M$ "small" items of size $\frac{1}{2} - \varepsilon$, followed by $I2\ :\ M$ "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.
2. To handle I_1, the optimal algorithm has used $\frac{M}{2}$ bins. Let the online algorithm have used b bins.
3. In order to beat the $\frac{4}{3}$ ratio, we must have $\frac{b}{\frac{M}{2}} < \frac{4}{3} \Rightarrow \frac{b}{M} < \frac{2}{3}$
4. Now consider the state of the online and optimal algorithms after processing I_2. The optimal uses M bins total.
5. Every **new** bin that the online algorithm opens after the first b bins, has at most 1 item in it.
6. Since only the first b bins can have 2 items and the remaining bins have 1 item each, packing $2 \cdot M$ items will require at least $(2 \cdot M - b)$ bins.
7. Therefore, we must have, $2 \cdot M - b < \frac{4}{3} \cdot M$

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously:
   $I1$: M "small" items of size $\frac{1}{2} - \varepsilon$, followed by $I2$: M "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.
2. To handle I_1, the optimal algorithm has used $\frac{M}{2}$ bins. Let the online algorithm have used b bins.
3. In order to beat the $\frac{4}{3}$ ratio, we must have $\frac{b}{\frac{M}{2}} < \frac{4}{3} \Rightarrow \frac{b}{M} < \frac{2}{3}$
4. Now consider the state of the online and optimal algorithms after processing I_2. The optimal uses M bins total.
5. Every **new** bin that the online algorithm opens after the first b bins, has at most 1 item in it.
6. Since only the first b bins can have 2 items and the remaining bins have 1 item each, packing $2 \cdot M$ items will require at least $(2 \cdot M - b)$ bins.
7. Therefore, we must have, $2 \cdot M - b < \frac{4}{3} \cdot M \Rightarrow \frac{b}{M} > \frac{2}{3}$.

Limits of online algorithms

Inapproximability in online algorithms

1. Consider the sequence discussed previously:
   $I1 : M$ "small" items of size $\frac{1}{2} - \varepsilon$, followed by $I2 : M$ "large" items of size $\frac{1}{2} + \varepsilon$, where $0 < \varepsilon \leq 0.001$.
2. To handle I_1, the optimal algorithm has used $\frac{M}{2}$ bins. Let the online algorithm have used b bins.
3. In order to beat the $\frac{4}{3}$ ratio, we must have $\frac{b}{\frac{M}{2}} < \frac{4}{3} \Rightarrow \frac{b}{M} < \frac{2}{3}$
4. Now consider the state of the online and optimal algorithms after processing I_2. The optimal uses M bins total.
5. Every **new** bin that the online algorithm opens after the first b bins, has at most 1 item in it.
6. Since only the first b bins can have 2 items and the remaining bins have 1 item each, packing $2 \cdot M$ items will require at least $(2 \cdot M - b)$ bins.
7. Therefore, we must have, $2 \cdot M - b < \frac{4}{3} \cdot M \Rightarrow \frac{b}{M} > \frac{2}{3}$. A contradiction.

Next-Fit (NF)

Next-Fit (NF)

Approach

Next-Fit (NF)

Approach

1. Open a new bin. Set $curr - bin$ to this bin.

Next-Fit (NF)

Approach

1. Open a new bin. Set $curr - bin$ to this bin.
2. **for**($i = 1$ **to** n)

Next-Fit (NF)

Approach

1. Open a new bin. Set $curr - bin$ to this bin.
2. **for**($i = 1$ **to** n)
3.   **if** (s_i fits in $curr - bin$.)

Next-Fit (NF)

Approach

1. Open a new bin. Set $curr - bin$ to this bin.
2. **for**($i = 1$ **to** n)
3.   **if** (s_i fits in $curr - bin$.)
4.   Assign it to $curr - bin$.

Next-Fit (NF)

Approach

1. Open a new bin. Set $curr - bin$ to this bin.
2. **for**($i = 1$ **to** n)
3.   **if** (s_i fits in $curr - bin$.)
4.   Assign it to $curr - bin$.
5.   **else**

Next-Fit (NF)

Approach

1. Open a new bin. Set $curr - bin$ to this bin.
2. **for**($i = 1$ **to** n)
3. **if** (s_i fits in $curr - bin$.)
4. Assign it to $curr - bin$.
5. **else**
6. Open a new bin and assign s_i to it.

Next-Fit (NF)

Approach

1. Open a new bin. Set $curr - bin$ to this bin.
2. **for**($i = 1$ **to** n)
3. **if** (s_i fits in $curr - bin$.)
4. Assign it to $curr - bin$.
5. **else**
6. Open a new bin and assign s_i to it.
7. Update $curr - bin$ to the newly opened bin.

Next-Fit (NF)

Approach

1. Open a new bin. Set $curr - bin$ to this bin.
2. **for**($i = 1$ **to** n)
3. **if** (s_i fits in $curr - bin$.)
4. Assign it to $curr - bin$.
5. **else**
6. Open a new bin and assign s_i to it.
7. Update $curr - bin$ to the newly opened bin.
8. **endif**

Next-Fit (NF)

Approach

1. Open a new bin. Set $curr - bin$ to this bin.
2. **for**($i = 1$ **to** n)
3.   **if** (s_i fits in $curr - bin$.)
4.   Assign it to $curr - bin$.
5.   **else**
6.   Open a new bin and assign s_i to it.
7.   Update $curr - bin$ to the newly opened bin.
8.   **endif**
9. **endfor**

Next-Fit (NF)

Approach

1. Open a new bin. Set $curr - bin$ to this bin.
2. **for**($i = 1$ **to** n)
3. **if** (s_i fits in $curr - bin$.)
4. Assign it to $curr - bin$.
5. **else**
6. Open a new bin and assign s_i to it.
7. Update $curr - bin$ to the newly opened bin.
8. **endif**
9. **endfor**

Note

You never go back in Next-Fit!

Analysis of Next-Fit

Analysis of Next-Fit

Theorem

If OPT is the number of bins in the optimal solution, then Next-Fit never uses more than $2 \cdot OPT$ bins. There exist sequences that force Next-Fit to use $(2 \cdot OPT - 2)$ bins.

Analysis of Next-Fit

Theorem

If OPT is the number of bins in the optimal solution, then Next-Fit never uses more than $2 \cdot OPT$ bins. There exist sequences that force Next-Fit to use $(2 \cdot OPT - 2)$ bins.

Proof.

Analysis of Next-Fit

Theorem

If OPT is the number of bins in the optimal solution, then Next-Fit never uses more than $2 \cdot OPT$ bins. There exist sequences that force Next-Fit to use $(2 \cdot OPT - 2)$ bins.

Proof.

Let the bins used by Next-Fit be denoted by $B_1, B_2, \ldots, B_k$.

Analysis of Next-Fit

Theorem

If OPT is the number of bins in the optimal solution, then Next-Fit never uses more than $2 \cdot OPT$ bins. There exist sequences that force Next-Fit to use $(2 \cdot OPT - 2)$ bins.

Proof.

Let the bins used by Next-Fit be denoted by $B_1, B_2, \ldots, B_k$. Let $|B_i|$ denote the space used in bin B_i.

Analysis of Next-Fit

Theorem

If OPT is the number of bins in the optimal solution, then Next-Fit never uses more than $2 \cdot OPT$ bins. There exist sequences that force Next-Fit to use $(2 \cdot OPT - 2)$ bins.

Proof.

Let the bins used by Next-Fit be denoted by $B_1, B_2, \ldots, B_k$. Let $|B_i|$ denote the space used in bin B_i. Observe that $(|B_{2 \cdot i-1}| + |B_{2 \cdot i}|) > 1$, $\forall i = 1, 2, \ldots \lfloor \frac{k}{2} \rfloor$.

Analysis of Next-Fit

Theorem

If OPT is the number of bins in the optimal solution, then Next-Fit never uses more than $2 \cdot OPT$ bins. There exist sequences that force Next-Fit to use $(2 \cdot OPT - 2)$ bins.

Proof.

Let the bins used by Next-Fit be denoted by $B_1, B_2, \ldots, B_k$. Let $|B_i|$ denote the space used in bin B_i. Observe that $(|B_{2 \cdot i-1}| + |B_{2 \cdot i}|) > 1$, $\forall i = 1, 2, \ldots \lfloor \frac{k}{2} \rfloor$. Adding up all the inequalities,

$$\lfloor \frac{k}{2} \rfloor \quad <$$

Analysis of Next-Fit

Theorem

If OPT is the number of bins in the optimal solution, then Next-Fit never uses more than $2 \cdot OPT$ bins. There exist sequences that force Next-Fit to use $(2 \cdot OPT - 2)$ bins.

Proof.

Let the bins used by Next-Fit be denoted by $B_1, B_2, \ldots, B_k$. Let $|B_i|$ denote the space used in bin B_i. Observe that $(|B_{2 \cdot i-1}| + |B_{2 \cdot i}|) > 1$, $\forall i = 1, 2, \ldots \lfloor \frac{k}{2} \rfloor$. Adding up all the inequalities,

$$\lfloor \frac{k}{2} \rfloor \quad < \quad \sum_{i=1}^{\lfloor \frac{k}{2} \rfloor} (|B_{2 \cdot i-1}| + |B_{2 \cdot i}|)$$

Analysis of Next-Fit

Theorem

If OPT is the number of bins in the optimal solution, then Next-Fit never uses more than $2 \cdot OPT$ bins. There exist sequences that force Next-Fit to use $(2 \cdot OPT - 2)$ bins.

Proof.

Let the bins used by Next-Fit be denoted by $B_1, B_2, \ldots, B_k$. Let $|B_i|$ denote the space used in bin B_i. Observe that $(|B_{2 \cdot i-1}| + |B_{2 \cdot i}|) > 1$, $\forall i = 1, 2, \ldots \lfloor \frac{k}{2} \rfloor$. Adding up all the inequalities,

$$\begin{aligned} \lfloor \frac{k}{2} \rfloor &< \sum_{i=1}^{\lfloor \frac{k}{2} \rfloor} (|B_{2 \cdot i-1}| + |B_{2 \cdot i}|) \\ &< \sum_{i=1}^{n} s_i \end{aligned}$$

Analysis of Next-Fit

Theorem

If OPT is the number of bins in the optimal solution, then Next-Fit never uses more than $2 \cdot OPT$ bins. There exist sequences that force Next-Fit to use $(2 \cdot OPT - 2)$ bins.

Proof.

Let the bins used by Next-Fit be denoted by $B_1, B_2, \ldots, B_k$. Let $|B_i|$ denote the space used in bin B_i. Observe that $(|B_{2 \cdot i-1}| + |B_{2 \cdot i}|) > 1, \forall i = 1, 2, \ldots \lfloor \frac{k}{2} \rfloor$. Adding up all the inequalities,

$$\begin{aligned} \lfloor \frac{k}{2} \rfloor &< \sum_{i=1}^{\lfloor \frac{k}{2} \rfloor} (|B_{2 \cdot i-1}| + |B_{2 \cdot i}|) \\ &< \sum_{i=1}^{n} s_i \end{aligned}$$

From the above relation, we can conclude that,

Analysis of Next-Fit

Theorem

If OPT is the number of bins in the optimal solution, then Next-Fit never uses more than $2 \cdot OPT$ bins. There exist sequences that force Next-Fit to use $(2 \cdot OPT - 2)$ bins.

Proof.

Let the bins used by Next-Fit be denoted by $B_1, B_2, \ldots, B_k$. Let $|B_i|$ denote the space used in bin B_i. Observe that $(|B_{2 \cdot i-1}| + |B_{2 \cdot i}|) > 1$, $\forall i = 1, 2, \ldots \lfloor \frac{k}{2} \rfloor$. Adding up all the inequalities,

$$\begin{aligned} \lfloor \frac{k}{2} \rfloor &< \sum_{i=1}^{\lfloor \frac{k}{2} \rfloor} (|B_{2 \cdot i-1}| + |B_{2 \cdot i}|) \\ &< \sum_{i=1}^{n} s_i \end{aligned}$$

From the above relation, we can conclude that,

$$\frac{k-1}{2}$$

Analysis of Next-Fit

Theorem

If OPT is the number of bins in the optimal solution, then Next-Fit never uses more than $2 \cdot OPT$ bins. There exist sequences that force Next-Fit to use $(2 \cdot OPT - 2)$ bins.

Proof.

Let the bins used by Next-Fit be denoted by $B_1, B_2, \ldots, B_k$. Let $|B_i|$ denote the space used in bin B_i. Observe that $(|B_{2 \cdot i-1}| + |B_{2 \cdot i}|) > 1$, $\forall i = 1, 2, \ldots \lfloor \frac{k}{2} \rfloor$. Adding up all the inequalities,

$$\begin{aligned} \lfloor \frac{k}{2} \rfloor &< \sum_{i=1}^{\lfloor \frac{k}{2} \rfloor} (|B_{2 \cdot i-1}| + |B_{2 \cdot i}|) \\ &< \sum_{i=1}^{n} s_i \end{aligned}$$

From the above relation, we can conclude that,

$$\frac{k-1}{2} \leq \lfloor \frac{k}{2} \rfloor$$

Analysis of Next-Fit

Theorem

If OPT is the number of bins in the optimal solution, then Next-Fit never uses more than $2 \cdot OPT$ bins. There exist sequences that force Next-Fit to use $(2 \cdot OPT - 2)$ bins.

Proof.

Let the bins used by Next-Fit be denoted by $B_1, B_2, \ldots, B_k$. Let $|B_i|$ denote the space used in bin B_i. Observe that $(|B_{2 \cdot i-1}| + |B_{2 \cdot i}|) > 1$, $\forall i = 1, 2, \ldots \lfloor \frac{k}{2} \rfloor$. Adding up all the inequalities,

$$\begin{aligned} \lfloor \frac{k}{2} \rfloor &< \sum_{i=1}^{\lfloor \frac{k}{2} \rfloor} (|B_{2 \cdot i-1}| + |B_{2 \cdot i}|) \\ &< \sum_{i=1}^{n} s_i \end{aligned}$$

From the above relation, we can conclude that,

$$\frac{k-1}{2} \leq \lfloor \frac{k}{2} \rfloor \leq \lceil \sum_{i=1}^{n} s_i \rceil - 1$$

□

Analysis (contd.)

Analysis (contd.)

Proof.

Observe that $OPT \geq \lceil \sum_{i=1}^{n} s_i \rceil$.

Analysis (contd.)

Proof.

Observe that $OPT \geq \lceil \sum_{i=1}^{n} s_i \rceil$. Hence,

Analysis (contd.)

Proof.

Observe that $OPT \geq \lceil \sum_{i=1}^{n} s_i \rceil$. Hence,

$$\frac{k-1}{2} \leq OPT - 1$$

Analysis (contd.)

Proof.

Observe that $OPT \geq \lceil \sum_{i=1}^{n} s_i \rceil$. Hence,

$$\begin{aligned} \frac{k-1}{2} &\leq OPT - 1 \\ \Rightarrow (k-1) &\leq 2 \cdot OPT - 2 \end{aligned}$$

Analysis (contd.)

Proof.

Observe that $OPT \geq \lceil \sum_{i=1}^{n} s_i \rceil$. Hence,

$$\begin{aligned} \frac{k-1}{2} &\leq OPT - 1 \\ \Rightarrow (k-1) &\leq 2 \cdot OPT - 2 \\ \Rightarrow k &\leq 2 \cdot OPT - 1 \end{aligned}$$

□

First-Fit (FF)

First-Fit (FF)

Approach

First-Fit (FF)

Approach

1. Open a new bin.

First-Fit (FF)

Approach

1. Open a new bin.
2. **for**($i = 1$ **to** n)

First-Fit (FF)

Approach

1. Open a new bin.
2. **for**($i = 1$ **to** n)
3.   **if** (s_i can be assigned to any open bin)

First-Fit (FF)

Approach

1. Open a new bin.
2. **for**($i = 1$ **to** n)
3. **if** (s_i can be assigned to any open bin)
4. Assign s_i to the first feasible bin.

First-Fit (FF)

Approach

1. Open a new bin.
2. **for**($i = 1$ **to** n)
3.   **if** (s_i can be assigned to any open bin)
4.   Assign s_i to the first feasible bin.
5.   **else**

First-Fit (FF)

Approach

1. Open a new bin.
2. **for**($i = 1$ **to** n)
3.   **if** (s_i can be assigned to any open bin)
4.   Assign s_i to the first feasible bin.
5.   **else**
6.   Open a new bin and assign s_i to it.

First-Fit (FF)

Approach

1. Open a new bin.
2. **for**($i = 1$ **to** n)
3. **if** (s_i can be assigned to any open bin)
4. Assign s_i to the first feasible bin.
5. **else**
6. Open a new bin and assign s_i to it.
7. **endif**

First-Fit (FF)

Approach

1. Open a new bin.
2. **for**($i = 1$ **to** n)
3.   **if** (s_i can be assigned to any open bin)
4.   Assign s_i to the first feasible bin.
5.   **else**
6.   Open a new bin and assign s_i to it.
7.   **endif**
8. **endfor**

Analysis of First-Fit

Analysis of First-Fit

Theorem

First-Fit uses at most $2 \cdot OPT$ bins.

Analysis of First-Fit

Theorem

First-Fit uses at most $2 \cdot OPT$ bins.

Proof.

Analysis of First-Fit

Theorem

First-Fit uses at most $2 \cdot OPT$ bins.

Proof.

Let k denote the number of bins used by Next-Fit. How many bins can be more than half-empty?

Analysis of First-Fit

Theorem

First-Fit uses at most $2 \cdot OPT$ bins.

Proof.

Let k denote the number of bins used by Next-Fit. How many bins can be more than half-empty? At most one!

Analysis of First-Fit

Theorem

First-Fit uses at most $2 \cdot OPT$ bins.

Proof.

Let k denote the number of bins used by Next-Fit. How many bins can be more than half-empty? At most one! It follows that,

Analysis of First-Fit

Theorem

First-Fit uses at most $2 \cdot OPT$ bins.

Proof.

Let k denote the number of bins used by Next-Fit. How many bins can be more than half-empty? At most one! It follows that,

$$\sum_{i=1}^{n} s_i \quad > \quad \frac{k-1}{2}$$

Analysis of First-Fit

Theorem

First-Fit uses at most $2 \cdot OPT$ bins.

Proof.

Let k denote the number of bins used by Next-Fit. How many bins can be more than half-empty? At most one! It follows that,

$$\sum_{i=1}^{n} s_i > \frac{k-1}{2}$$

$$\Rightarrow k < 2 \cdot \sum_{i=1}^{n} s_i + 1$$

Analysis of First-Fit

Theorem

First-Fit uses at most $2 \cdot OPT$ bins.

Proof.

Let k denote the number of bins used by Next-Fit. How many bins can be more than half-empty? At most one! It follows that,

$$\begin{aligned} \sum_{i=1}^{n} s_i &> \frac{k-1}{2} \\ \Rightarrow k &< 2 \cdot \sum_{i=1}^{n} s_i + 1 \\ \Rightarrow k &< 2 \cdot OPT + 1 \end{aligned}$$

Analysis of First-Fit

Theorem

First-Fit uses at most $2 \cdot OPT$ bins.

Proof.

Let k denote the number of bins used by Next-Fit. How many bins can be more than half-empty? At most one! It follows that,

$$\begin{aligned} \sum_{i=1}^{n} s_i &> \frac{k-1}{2} \\ \Rightarrow k &< 2 \cdot \sum_{i=1}^{n} s_i + 1 \\ \Rightarrow k &< 2 \cdot OPT + 1 \\ \Rightarrow k &\leq 2 \cdot OPT \end{aligned}$$

□

Tighter bounds

Tighter bounds

Theorem

If OPT is the optimal number of bins, then First-Fit never uses more than $1.7 \cdot OPT$ bins. On the other hand, there are sequences that force it to use at least $\frac{17}{10} \cdot (OPT - 1)$ bins.

Tighter bounds

Theorem

If OPT is the optimal number of bins, then First-Fit never uses more than $1.7 \cdot OPT$ bins. On the other hand, there are sequences that force it to use at least $\frac{17}{10} \cdot (OPT - 1)$ bins.

Proof.

Homework. □

Best-Fit (BF)

Best-Fit (BF)

Approach

Best-Fit (BF)

Approach

Place each item in the bin that provides the *tightest* fit, i.e., in the bin that results in the smallest empty space.

Best-Fit (BF)

Approach

Place each item in the bin that provides the *tightest* fit, i.e., in the bin that results in the smallest empty space.

Note

The generic positive and negative results of First-Fit apply.

First-Fit Decreasing (FFD)

First-Fit Decreasing (FFD)

Approach

First-Fit Decreasing (FFD)

Approach

1. Sort the elements so that $s_1 \geq s_2 \geq \ldots \geq s_n$.

First-Fit Decreasing (FFD)

Approach

1. Sort the elements so that $s_1 \geq s_2 \geq \ldots \geq s_n$.
2. Use First-Fit.

First-Fit Decreasing (FFD)

Approach

1. Sort the elements so that $s_1 \geq s_2 \geq \ldots \geq s_n$.
2. Use First-Fit.

Note

FFD is the offline analog of FF.

First bound

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.
   - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
    - $A = \{s_i : s_i > \frac{2}{3}\}$.
    - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.
    - $C = \{s_i : \frac{1}{2} \geq s_i > \frac{1}{3}\}$.

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.
   - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.
   - $C = \{s_i : \frac{1}{2} \geq s_i > \frac{1}{3}\}$.
   - $D = \{s_i : \frac{1}{3} \geq s_i\}$.

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.
   - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.
   - $C = \{s_i : \frac{1}{2} \geq s_i > \frac{1}{3}\}$.
   - $D = \{s_i : \frac{1}{3} \geq s_i\}$.
2. Consider the following two cases:

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.
   - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.
   - $C = \{s_i : \frac{1}{2} \geq s_i > \frac{1}{3}\}$.
   - $D = \{s_i : \frac{1}{3} \geq s_i\}$.
2. Consider the following two cases:
   (a) There is at least one bin that contains only elements from D -

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.
   - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.
   - $C = \{s_i : \frac{1}{2} \geq s_i > \frac{1}{3}\}$.
   - $D = \{s_i : \frac{1}{3} \geq s_i\}$.
2. Consider the following two cases:
   (a) There is at least one bin that contains only elements from D - In this case, for all but one bin (the last one), the total occupancy is at least $\frac{2}{3}$.

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.
   - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.
   - $C = \{s_i : \frac{1}{2} \geq s_i > \frac{1}{3}\}$.
   - $D = \{s_i : \frac{1}{3} \geq s_i\}$.
2. Consider the following two cases:
   (a) There is at least one bin that contains only elements from D - In this case, for all but one bin (the last one), the total occupancy is at least $\frac{2}{3}$. In other words $\sum_{i=1}^{n} s_i \geq \frac{2}{3} \cdot (k-1)$.

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.
   - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.
   - $C = \{s_i : \frac{1}{2} \geq s_i > \frac{1}{3}\}$.
   - $D = \{s_i : \frac{1}{3} \geq s_i\}$.
2. Consider the following two cases:
   (a) There is at least one bin that contains only elements from D - In this case, for all but one bin (the last one), the total occupancy is at least $\frac{2}{3}$. In other words $\sum_{i=1}^{n} s_i \geq \frac{2}{3} \cdot (k-1)$. It follows that the theorem holds.

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.
   - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.
   - $C = \{s_i : \frac{1}{2} \geq s_i > \frac{1}{3}\}$.
   - $D = \{s_i : \frac{1}{3} \geq s_i\}$.
2. Consider the following two cases:
   (a) There is at least one bin that contains only elements from D - In this case, for all but one bin (the last one), the total occupancy is at least $\frac{2}{3}$. In other words $\sum_{i=1}^{n} s_i \geq \frac{2}{3} \cdot (k-1)$. It follows that the theorem holds.
   (b) There is no bin that contains only elements from D -

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.
   - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.
   - $C = \{s_i : \frac{1}{2} \geq s_i > \frac{1}{3}\}$.
   - $D = \{s_i : \frac{1}{3} \geq s_i\}$.
2. Consider the following two cases:
   (a) There is at least one bin that contains only elements from D - In this case, for all but one bin (the last one), the total occupancy is at least $\frac{2}{3}$. In other words $\sum_{i=1}^{n} s_i \geq \frac{2}{3} \cdot (k-1)$. It follows that the theorem holds.
   (b) There is no bin that contains only elements from D - In this case, we can focus on the solution returned by FFD, assuming that all the elements of bucket D are thrown out.

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.
   - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.
   - $C = \{s_i : \frac{1}{2} \geq s_i > \frac{1}{3}\}$.
   - $D = \{s_i : \frac{1}{3} \geq s_i\}$.
2. Consider the following two cases:
   (a) There is at least one bin that contains only elements from D - In this case, for all but one bin (the last one), the total occupancy is at least $\frac{2}{3}$. In other words $\sum_{i=1}^{n} s_i \geq \frac{2}{3} \cdot (k-1)$. It follows that the theorem holds.
   (b) There is no bin that contains only elements from D - In this case, we can focus on the solution returned by FFD, assuming that all the elements of bucket D are thrown out. In this revised input instance, elements of A are loners,

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.
   - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.
   - $C = \{s_i : \frac{1}{2} \geq s_i > \frac{1}{3}\}$.
   - $D = \{s_i : \frac{1}{3} \geq s_i\}$.
2. Consider the following two cases:
   (a) There is at least one bin that contains only elements from D - In this case, for all but one bin (the last one), the total occupancy is at least $\frac{2}{3}$. In other words $\sum_{i=1}^{n} s_i \geq \frac{2}{3} \cdot (k-1)$. It follows that the theorem holds.
   (b) There is no bin that contains only elements from D - In this case, we can focus on the solution returned by FFD, assuming that all the elements of bucket D are thrown out. In this revised input instance, elements of A are loners, every bin contains at most two elements,

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.
   - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.
   - $C = \{s_i : \frac{1}{2} \geq s_i > \frac{1}{3}\}$.
   - $D = \{s_i : \frac{1}{3} \geq s_i\}$.
2. Consider the following two cases:
   (a) There is at least one bin that contains only elements from D - In this case, for all but one bin (the last one), the total occupancy is at least $\frac{2}{3}$. In other words $\sum_{i=1}^{n} s_i \geq \frac{2}{3} \cdot (k-1)$. It follows that the theorem holds.
   (b) There is no bin that contains only elements from D - In this case, we can focus on the solution returned by FFD, assuming that all the elements of bucket D are thrown out. In this revised input instance, elements of A are loners, every bin contains at most two elements, and at most one of these two elements can be from bucket B.

First bound

Theorem

Let k denote the number of bins used by FFD. Then $k \leq 1.5 \cdot OPT + 1$.

Proof.

1. Let us partition the objects into 4 buckets:
   - $A = \{s_i : s_i > \frac{2}{3}\}$.
   - $B = \{s_i : \frac{2}{3} \geq s_i > \frac{1}{2}\}$.
   - $C = \{s_i : \frac{1}{2} \geq s_i > \frac{1}{3}\}$.
   - $D = \{s_i : \frac{1}{3} \geq s_i\}$.
2. Consider the following two cases:
   (a) There is at least one bin that contains only elements from D - In this case, for all but one bin (the last one), the total occupancy is at least $\frac{2}{3}$. In other words $\sum_{i=1}^{n} s_i \geq \frac{2}{3} \cdot (k-1)$. It follows that the theorem holds.
   (b) There is no bin that contains only elements from D - In this case, we can focus on the solution returned by FFD, assuming that all the elements of bucket D are thrown out. In this revised input instance, elements of A are loners, every bin contains at most two elements, and at most one of these two elements can be from bucket B. It is not hard to see that FFD gives an optimal packing!

□

Tighter bounds

Tighter bounds

Theorem

Let the bins used by FFD be $B_1, B_2, \ldots B_{OPT}, B_{OPT+1}, \ldots B_r$. *Then* $r \leq (\frac{4 \cdot OPT + 2}{3})$.

Analysis

Analysis

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ *have size at most* $\frac{1}{3}$.

Analysis

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ *have size at most* $\frac{1}{3}$.

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Analysis

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ have size at most $\frac{1}{3}$.

Lemma (Lemnumber)

The number *of items that FFD puts in the bins $\{B_{OPT+1}, \ldots B_r\}$ is at most $(OPT - 1)$.*

Note

If the above two lemmas are proved, the theorem easily follows.

Analysis

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ *have size at most* $\frac{1}{3}$.

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Note

If the above two lemmas are proved, the theorem easily follows. To see this, observe that as per the two lemmas above, there are $(OPT - 1)$ *extra items, each having size at most* $\frac{1}{3}$ *and hence the number of extra bins required is at most*

Analysis

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ have size at most $\frac{1}{3}$.

Lemma (Lemnumber)

The number *of items that FFD puts in the bins $\{B_{OPT+1}, \ldots B_r\}$ is at most $(OPT - 1)$.*

Note

If the above two lemmas are proved, the theorem easily follows. To see this, observe that as per the two lemmas above, there are $(OPT - 1)$ extra items, each having size at most $\frac{1}{3}$ and hence the number of extra bins required is at most $\lceil \frac{OPT-1}{3} \rceil$.

Analysis

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ *have size at most* $\frac{1}{3}$.

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Note

If the above two lemmas are proved, the theorem easily follows. To see this, observe that as per the two lemmas above, there are $(OPT - 1)$ *extra items, each having size at most* $\frac{1}{3}$ *and hence the number of extra bins required is at most* $\lceil \frac{OPT-1}{3} \rceil$. *It follows that the total number of bins required by FFD is at most :*

Analysis

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ have size at most $\frac{1}{3}$.

Lemma (Lemnumber)

The number *of items that FFD puts in the bins $\{B_{OPT+1}, \ldots B_r\}$ is at most $(OPT - 1)$.*

Note

If the above two lemmas are proved, the theorem easily follows. To see this, observe that as per the two lemmas above, there are $(OPT - 1)$ extra items, each having size at most $\frac{1}{3}$ and hence the number of extra bins required is at most $\lceil \frac{OPT-1}{3} \rceil$. It follows that the total number of bins required by FFD is at most :

$$OPT + \lceil \frac{OPT - 1}{3} \rceil \quad \leq \quad OPT + \frac{OPT - 1}{3} + 1$$

Analysis

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ have size at most $\frac{1}{3}$.

Lemma (Lemnumber)

The number *of items that FFD puts in the bins $\{B_{OPT+1}, \ldots B_r\}$ is at most $(OPT - 1)$.*

Note

If the above two lemmas are proved, the theorem easily follows. To see this, observe that as per the two lemmas above, there are $(OPT - 1)$ extra items, each having size at most $\frac{1}{3}$ and hence the number of extra bins required is at most $\lceil \frac{OPT-1}{3} \rceil$. It follows that the total number of bins required by FFD is at most :

$$\begin{aligned} OPT + \lceil \frac{OPT - 1}{3} \rceil &\leq OPT + \frac{OPT - 1}{3} + 1 \\ &= \frac{4 \cdot OPT + 2}{3} \end{aligned}$$

Analysis (contd.)

Analysis (contd.)

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ *have size at most* $\frac{1}{3}$.

Analysis (contd.)

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ have size at most $\frac{1}{3}$.

Proof of Lemsize.

Analysis (contd.)

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ have size at most $\frac{1}{3}$.

Proof of Lemsize.

We proved it in the previous analysis!

Analysis (contd.)

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ have size at most $\frac{1}{3}$.

Proof of Lemsize.

We proved it in the previous analysis!

Analysis (contd.)

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ have size at most $\frac{1}{3}$.

Proof of Lemsize.

We proved it in the previous analysis!

- As before, consider the partition of the four items into 4 buckets.

Analysis (contd.)

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ *have size at most* $\frac{1}{3}$.

Proof of Lemsize.

We proved it in the previous analysis!

- As before, consider the partition of the four items into 4 buckets.
- If we ignore the D bucket, FFD gives the optimal number of bins, say OPT_1 for the truncated instance.

Analysis (contd.)

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ have size at most $\frac{1}{3}$.

Proof of Lemsize.

We proved it in the previous analysis!

- As before, consider the partition of the four items into 4 buckets.
- If we ignore the D bucket, FFD gives the optimal number of bins, say OPT_1 for the truncated instance.
- Clearly, the optimal solution for the complete instance, OPT is at least as large as OPT_1, i.e, $OPT \geq OPT_1$.

Analysis (contd.)

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ have size at most $\frac{1}{3}$.

Proof of Lemsize.

We proved it in the previous analysis!

- As before, consider the partition of the four items into 4 buckets.
- If we ignore the D bucket, FFD gives the optimal number of bins, say OPT_1 for the truncated instance.
- Clearly, the optimal solution for the complete instance, OPT is at least as large as OPT_1, i.e, $OPT \geq OPT_1$.
- What we know is that every bin opened after OPT_1 and hence OPT must contain only items from bucket D, i.e., each of those items will have size at most $\frac{1}{3}$.

□

Analysis (contd.)

Lemma (Lemsize)

In the FFD bin sequence, all items in the bins $\{B_{OPT+1}, \ldots B_r\}$ *have size at most* $\frac{1}{3}$.

Proof of Lemsize.

We proved it in the previous analysis!

- As before, consider the partition of the four items into 4 buckets.
- If we ignore the D bucket, FFD gives the optimal number of bins, say OPT_1 for the truncated instance.
- Clearly, the optimal solution for the complete instance, OPT is at least as large as OPT_1, i.e, $OPT \geq OPT_1$.
- What we know is that every bin opened after OPT_1 and hence OPT must contain only items from bucket D, i.e., each of those items will have size at most $\frac{1}{3}$.

□

Analysis (contd.)

Analysis (contd.)

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Analysis (contd.)

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Proof of Lemnumber.

Analysis (contd.)

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Proof of Lemnumber.

Analysis (contd.)

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Proof of Lemnumber.

1. Assume at least OPT objects were put in the bins after B_{OPT}.

Analysis (contd.)

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Proof of Lemnumber.

1. Assume at least OPT objects were put in the bins after B_{OPT}.
2. Recall that we must have $\sum_{i=1}^{n} s_i \leq OPT$.

Analysis (contd.)

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Proof of Lemnumber.

1. Assume at least OPT objects were put in the bins after B_{OPT}.
2. Recall that we must have $\sum_{i=1}^{n} s_i \leq OPT$.
3. Let weight associated with bin B_j be W_j and let $x_1, x_2, \ldots x_{OPT}$ denote the weights of the first OPT objects in the extra bins.

Analysis (contd.)

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Proof of Lemnumber.

1. Assume at least OPT objects were put in the bins after B_{OPT}.
2. Recall that we must have $\sum_{i=1}^{n} s_i \leq OPT$.
3. Let weight associated with bin B_j be W_j and let $x_1, x_2, \ldots x_{OPT}$ denote the weights of the first OPT objects in the extra bins.
4. Clearly, we must have,

Analysis (contd.)

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Proof of Lemnumber.

1. Assume at least OPT objects were put in the bins after B_{OPT}.
2. Recall that we must have $\sum_{i=1}^{n} s_i \leq OPT$.
3. Let weight associated with bin B_j be W_j and let $x_1, x_2, \ldots x_{OPT}$ denote the weights of the first OPT objects in the extra bins.
4. Clearly, we must have, $\sum_{i=1}^{n} s_i \geq \sum_{j=1}^{OPT} W_j + \sum_{j=1}^{OPT} x_j$, which implies that,

Analysis (contd.)

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Proof of Lemnumber.

1. Assume at least OPT objects were put in the bins after B_{OPT}.
2. Recall that we must have $\sum_{i=1}^{n} s_i \leq OPT$.
3. Let weight associated with bin B_j be W_j and let $x_1, x_2, \ldots x_{OPT}$ denote the weights of the first OPT objects in the extra bins.
4. Clearly, we must have, $\sum_{i=1}^{n} s_i \geq \sum_{j=1}^{OPT} W_j + \sum_{j=1}^{OPT} x_j$, which implies that, $\sum_{i=1}^{n} s_i \geq \sum_{j=1}^{OPT} (W_j + x_j)$.

Analysis (contd.)

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Proof of Lemnumber.

1. Assume at least OPT objects were put in the bins after B_{OPT}.
2. Recall that we must have $\sum_{i=1}^{n} s_i \leq OPT$.
3. Let weight associated with bin B_j be W_j and let $x_1, x_2, \ldots x_{OPT}$ denote the weights of the first OPT objects in the extra bins.
4. Clearly, we must have, $\sum_{i=1}^{n} s_i \geq \sum_{j=1}^{OPT} W_j + \sum_{j=1}^{OPT} x_j$, which implies that, $\sum_{i=1}^{n} s_i \geq \sum_{j=1}^{OPT} (W_j + x_j)$.
5. However, $(W_j + x_j) > 1$.

Analysis (contd.)

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Proof of Lemnumber.

1. Assume at least OPT objects were put in the bins after B_{OPT}.
2. Recall that we must have $\sum_{i=1}^{n} s_i \leq OPT$.
3. Let weight associated with bin B_j be W_j and let $x_1, x_2, \ldots x_{OPT}$ denote the weights of the first OPT objects in the extra bins.
4. Clearly, we must have, $\sum_{i=1}^{n} s_i \geq \sum_{j=1}^{OPT} W_j + \sum_{j=1}^{OPT} x_j$, which implies that, $\sum_{i=1}^{n} s_i \geq \sum_{j=1}^{OPT} (W_j + x_j)$.
5. However, $(W_j + x_j) > 1$. (Why?)

Analysis (contd.)

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1},\ldots B_r\}$ *is at most* $(OPT-1)$.

Proof of Lemnumber.

1. Assume at least OPT objects were put in the bins after B_{OPT}.
2. Recall that we must have $\sum_{i=1}^{n} s_i \leq OPT$.
3. Let weight associated with bin B_j be W_j and let $x_1, x_2, \ldots x_{OPT}$ denote the weights of the first OPT objects in the extra bins.
4. Clearly, we must have, $\sum_{i=1}^{n} s_i \geq \sum_{j=1}^{OPT} W_j + \sum_{j=1}^{OPT} x_j$, which implies that, $\sum_{i=1}^{n} s_i \geq \sum_{j=1}^{OPT} (W_j + x_j)$.
5. However, $(W_j + x_j) > 1$. (Why?) This means that $\sum_{i=1}^{n} s_i > OPT$,

Analysis (contd.)

Lemma (Lemnumber)

The number *of items that FFD puts in the bins* $\{B_{OPT+1}, \ldots B_r\}$ *is at most* $(OPT - 1)$.

Proof of Lemnumber.

1. Assume at least OPT objects were put in the bins after B_{OPT}.
2. Recall that we must have $\sum_{i=1}^{n} s_i \leq OPT$.
3. Let weight associated with bin B_j be W_j and let $x_1, x_2, \ldots x_{OPT}$ denote the weights of the first OPT objects in the extra bins.
4. Clearly, we must have, $\sum_{i=1}^{n} s_i \geq \sum_{j=1}^{OPT} W_j + \sum_{j=1}^{OPT} x_j$, which implies that, $\sum_{i=1}^{n} s_i \geq \sum_{j=1}^{OPT} (W_j + x_j)$.
5. However, $(W_j + x_j) > 1$. (Why?) This means that $\sum_{i=1}^{n} s_i > OPT$, which is a contradiction!

□

Still tighter bounds

Still tighter bounds

Theorem

FFD uses at most $\left(\frac{11 \cdot OPT}{9} + 4\right)$ *bins.*

Still tighter bounds

Theorem

FFD uses at most $(\frac{11 \cdot OPT}{9} + 4)$ *bins. There are sequences for which FFD uses* $\frac{11 \cdot M}{9}$ *bins.*

Inapproximability

Inapproximability

Theorem

There does not exist a polynomial time algorithm for the Bin-Packing problem with approximation factor $(\frac{3}{2} - \varepsilon)$, *for any* $\varepsilon > 0$, *unless* **P=NP**.

Inapproximability

Theorem

There does not exist a polynomial time algorithm for the Bin-Packing problem with approximation factor $(\frac{3}{2} - \varepsilon)$*, for any* $\varepsilon > 0$*, unless* **P=NP**.

Proof.

Inapproximability

Theorem

There does not exist a polynomial time algorithm for the Bin-Packing problem with approximation factor $(\frac{3}{2} - \varepsilon)$*, for any* $\varepsilon > 0$*, unless* **P=NP**.

Proof.

1. Assume that there exists a $(\frac{3}{2} - \varepsilon)$ algorithm, $\mathscr{A}$, for Bin-Packing, for some $\varepsilon > 0$.

Inapproximability

Theorem

There does not exist a polynomial time algorithm for the Bin-Packing problem with approximation factor $(\frac{3}{2}-\varepsilon)$, for any $\varepsilon > 0$, unless **P=NP**.

Proof.

1. Assume that there exists a $(\frac{3}{2}-\varepsilon)$ algorithm, $\mathscr{A}$, for Bin-Packing, for some $\varepsilon > 0$.
2. Recall the 2-partition problem.

Inapproximability

Theorem

There does not exist a polynomial time algorithm for the Bin-Packing problem with approximation factor $(\frac{3}{2} - \varepsilon)$*, for any* $\varepsilon > 0$*, unless* **P=NP**.

Proof.

1. Assume that there exists a $(\frac{3}{2} - \varepsilon)$ algorithm, $\mathscr{A}$, for Bin-Packing, for some $\varepsilon > 0$.
2. Recall the 2-partition problem. In this problem, you are are given a set of numbers $\{a_1, a_2, \dots a_n\}$ and asked if they can be partitioned into two sets, each adding up to $\frac{1}{2}\sum_i a_i$.

Inapproximability

Theorem

There does not exist a polynomial time algorithm for the Bin-Packing problem with approximation factor $(\frac{3}{2} - \epsilon)$, for any $\epsilon > 0$, unless **P=NP**.

Proof.

1. Assume that there exists a $(\frac{3}{2} - \epsilon)$ algorithm, $\mathscr{A}$, for Bin-Packing, for some $\epsilon > 0$.
2. Recall the 2-partition problem. In this problem, you are are given a set of numbers $\{a_1, a_2, \ldots a_n\}$ and asked if they can be partitioned into two sets, each adding up to $\frac{1}{2}\sum_i a_i$.
3. Give the instance of 2-partition to algorithm $\mathscr{A}$.

Inapproximability

Theorem

There does not exist a polynomial time algorithm for the Bin-Packing problem with approximation factor $(\frac{3}{2} - \varepsilon)$, for any $\varepsilon > 0$, unless **P=NP**.

Proof.

1. Assume that there exists a $(\frac{3}{2} - \varepsilon)$ algorithm, $\mathscr{A}$, for Bin-Packing, for some $\varepsilon > 0$.
2. Recall the 2-partition problem. In this problem, you are are given a set of numbers $\{a_1, a_2, \ldots a_n\}$ and asked if they can be partitioned into two sets, each adding up to $\frac{1}{2}\sum_i a_i$.
3. Give the instance of 2-partition to algorithm $\mathscr{A}$.
4. The answer to the instance is "yes", if and only if the n items can be packed into two bins having size $\frac{1}{2}\sum_i a_i$.

Inapproximability

Theorem

There does not exist a polynomial time algorithm for the Bin-Packing problem with approximation factor $(\frac{3}{2} - \varepsilon)$, for any $\varepsilon > 0$, unless **P=NP**.

Proof.

1. Assume that there exists a $(\frac{3}{2} - \varepsilon)$ algorithm, $\mathscr{A}$, for Bin-Packing, for some $\varepsilon > 0$.
2. Recall the 2-partition problem. In this problem, you are are given a set of numbers $\{a_1, a_2, \ldots a_n\}$ and asked if they can be partitioned into two sets, each adding up to $\frac{1}{2}\sum_i a_i$.
3. Give the instance of 2-partition to algorithm $\mathscr{A}$.
4. The answer to the instance is "yes", if and only if the n items can be packed into two bins having size $\frac{1}{2}\sum_i a_i$.
5. Observe that if the input is a "yes" instance, then $\mathscr{A}$ would have to return with an optimal answer!

□