Online bin packing with overload cost

Kelin Luo

Eindhoven University of Technology

This is joint work with Frits C. R. Spieksma

June 2, 2021

Overview

1. Problem
   - Review: Online Bin Packing Problem
   - Online Bin Packing Problem with Overload Cost
2. Our Result
   - Lower Bounds
   - Algorithm and Upper Bounds
3. Extension
4. Conclusion

Content

1. Problem
   - Review: Online Bin Packing Problem
   - Online Bin Packing Problem with Overload Cost
2. Our Result
3. Extension
4. Conclusion

Online Bin Packing Problem

Setting

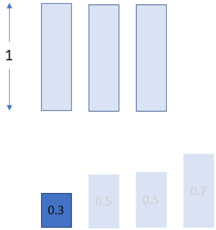
n items with size $p \in (0, 1]$ arrive one by one

Each item must be placed in a bin before the next item arrives

Goal

Pack all items into the minimum number of bins

such that the total size of items packed in each bin is at most 1



Online Bin Packing Problem

Setting

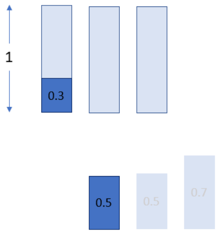
n items with size $p \in (0, 1]$ arrive one by one

Each item must be placed in a bin before the next item arrives

Goal

Pack all items into the minimum number of bins

such that the total size of items packed in each bin is at most 1



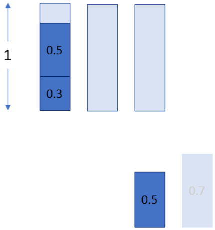
Online Bin Packing Problem

Setting

n items with size $p \in (0, 1]$ arrive one by one
Each item must be placed in a bin before the next item arrives

Goal

Pack all items into the minimum number of bins
such that the total size of items packed in each bin is at most 1



Online Bin Packing Problem

Setting

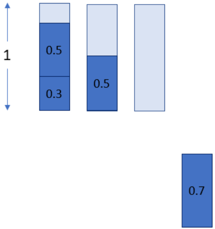
n items with size $p \in (0, 1]$ arrive one by one

Each item must be placed in a bin before the next item arrives

Goal

Pack all items into the minimum number of bins

such that the total size of items packed in each bin is at most 1



Online Bin Packing Problem

Setting

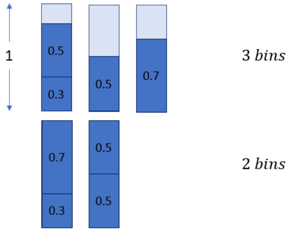
n items with size $p \in (0, 1]$ arrive one by one

Each item must be placed in a bin before the next item arrives

Goal

Pack all items into the minimum number of bins

such that the total size of items packed in each bin is at most 1



Online Bin Packing with Overload Cost (BPOC)

Setting

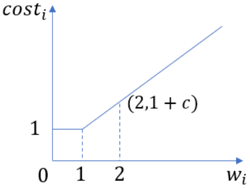
- n items with size $p \in (0, 1]$ arrive one by one
  Each item must be placed into a bin before the next item arrives
- Bin: extensible, unit overload cost c
  infinite number of identical bins

Online Bin Packing with Overload Cost (BPOC)

Setting

- n items with size $p \in (0, 1]$ arrive one by one
  Each item must be placed into a bin before the next item arrives
- Bin: extensible, unit overload cost c
  infinite number of identical bins

Sup. total size of items in bin i is w_i, $cost_i = 1 + c \cdot \max\{w_i - 1, 0\}$



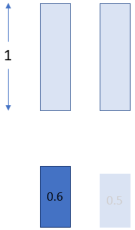
Goal: Pack all items into bins with minimum total cost

Setting

- *Item p* has size $p \in (0,1]$ arrives one by one
- *Bin i* has $cost_i = 1 + c \cdot \max\{w_i - 1, 0\}$

Goal

Pack all items into bin with minimum total cost



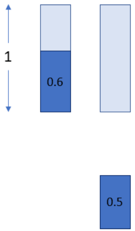
Online Bin Packing with Overload Cost (BPOC)

Setting

- *Item p* has size $p \in (0, 1]$ arrives one by one
- *Bin i* has $cost_i = 1 + c \cdot \max\{w_i - 1, 0\}$

Goal

Pack all items into bin with minimum total cost



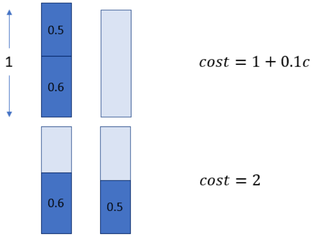
Online Bin Packing with Overload Cost (BPOC)

Setting

- *Item p* has size $p \in (0, 1]$ arrives one by one
- *Bin i* has $cost_i = 1 + c \cdot \max\{w_i - 1, 0\}$

Goal

Pack all items into bin with minimum total cost



Related work: Overload

- Violate one item

Related work: Overload

- Violate one item
  - Open-end Packing
    - Ordered open-end bin packing problem (OOBP): last item (Yang and Leung 2003, Balogh et al., 2020)
    - Strong open-end bin packing problem (SOBP): lightest one item (Epstein and Levin 2008, Epstein 2021)

Related work: Overload

- Violate one item
  - Open-end Packing
  - Bin Packing with Overflow (Perez-Salazar et al., 2021)
    Item X_i is observed after packing; Once overflows, incur cost $C \geq 1$ and no more packing.
    Goal: pack items into bins with minimum cost

Related work: Overload

- Violate one item
  - Open-end Packing
  - Bin packing with overflow (Perez-Salazar et al., 2021)
- Violate more: Online bin stretching
  m bins, stretching factor c
  - LB=7/6, UB=1.228 (Speranza and Tuza 1999)
  - UB= 7/6, 7/6, 19/16 for m=2, 3, 4 (Ye and Zhang 2002)

Related work: Overload

- Violate one item
  - Open-end Packing
  - Bin packing with overflow (Perez-Salazar et al., 2021)
- Violate more: Online bin stretching

BPOC: Infinite number of extensible bins
Min total cost
Absolute competitive ratio

Content

1 Problem

2 Our Result
- Lower Bounds
- Algorithm and Upper Bounds

3 Extension

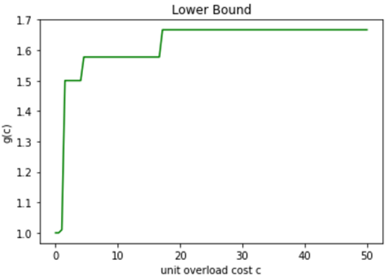
4 Conclusion

Lower bounds

Theorem

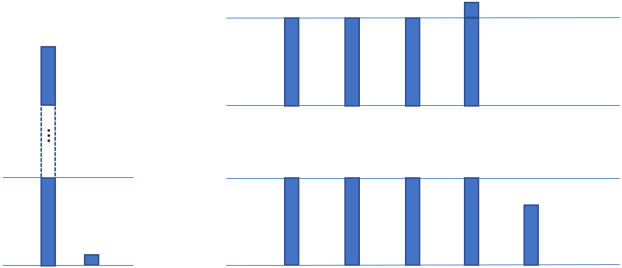
For any $c \geq 0$: no deterministic online algorithm for BPOC can achieve a competitive ratio smaller than $g(c)$.

$$g(c) = \begin{cases} \max(1, c) & \text{if } 0 \leq c < \frac{3}{2} \\ \frac{3}{2} = 1.5 & \text{if } \frac{3}{2} \leq c < 1 + 2\sqrt{3} \\ 1 + \frac{\sqrt{3}}{3} \approx 1.577 & \text{if } 1 + 2\sqrt{3} < c < 17 \\ \frac{5}{3} \approx 1.667 & \text{if } 17 \leq c \end{cases}$$



Lower Bound: $1 < c \leq 17$

Item sequence: Continue to release N^2 items with size $1/N$, or stop when ALG opens a second bin.



$2 + c(W - 1 - 1/N)$

Lower bounds

Theorem

For any $c \geq 0$: no deterministic online algorithm for BPOC can achieve a competitive ratio smaller than $g(c)$.

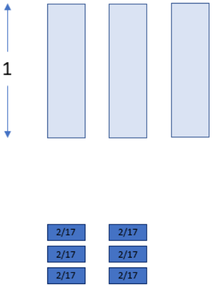
$$g(c) = \begin{cases} \max(1, c) & \text{if } 0 \leq c < \frac{3}{2} \\ \frac{3}{2} = 1.5 & \text{if } \frac{3}{2} \leq c < 1 + 2\sqrt{3} \\ 1 + \frac{\sqrt{3}}{3} \approx 1.577 & \text{if } 1 + 2\sqrt{3} < c < 17 \\ \frac{5}{3} \approx 1.667 & \text{if } 17 \leq c \end{cases}$$

János Balogh, József Békési, György Dósa, Jiří Sgall, Rob van Stee: The optimal absolute ratio for online bin packing, JCSS, 2019

Lower Bound: $5/3$ when $c \geq 17$ $(1/3)$

Item sequence: release 6 items with size $2/17$

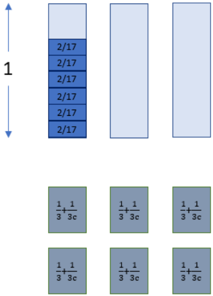
- ALG opens more than one bin: $\frac{cost_{ALG}}{cost_{OPT}} \geq 2$
- ALG opens a single bin:



Lower Bound: 5/3 when $c \geq 17$ (2/3)

Item sequence: release 6 items with size 2/17

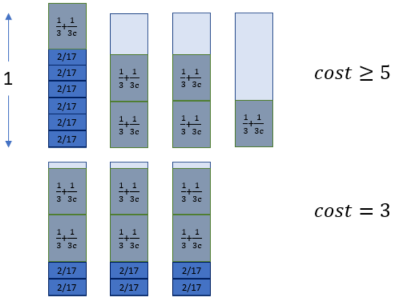
- ALG opens a single bin: release 6 items with size $\frac{1}{3} + \frac{1}{3c}$



Lower Bound: 5/3 when $c \geq 17$ (2/3)

Item sequence: release 6 items with size 2/17

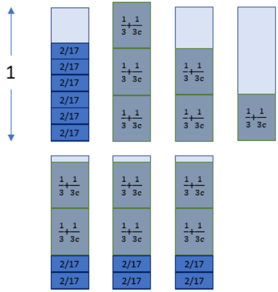
- ALG opens a single bin: release 6 items with size $\frac{1}{3} + \frac{1}{3c}$



Lower Bound: 5/3 when $c \geq 17$ (2/3)

Item sequence: release 6 items with size 2/17

- ALG opens a single bin: release 6 items with size $\frac{1}{3} + \frac{1}{3c}$



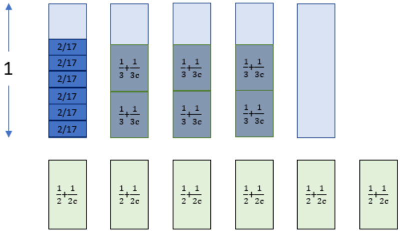
$cost \geq 5$

$cost = 3$

Lower Bound: 5/3 when $c \geq 17$ (3/3)

Item sequence: release 6 items with size 2/17

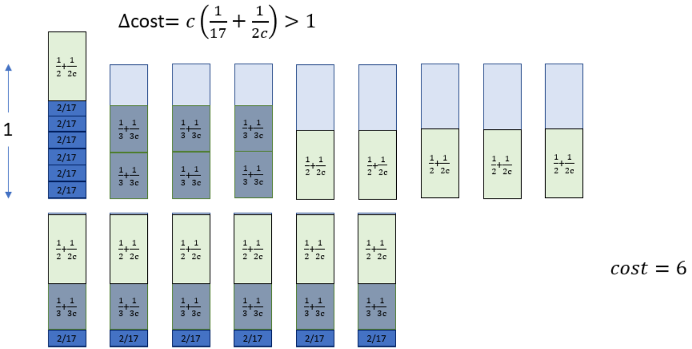
- ALG opens a single bin: release 6 items with size $\frac{1}{3} + \frac{1}{3c}$
  - ALG packs two items into one bin: release 6 items with size $\frac{1}{2} + \frac{1}{2c}$



Lower Bound: $5/3$ when $c \geq 17$ (3/3)

Item sequence: release 6 items with size $2/17$

- ALG opens a single bin: release 6 items with size $\frac{1}{3} + \frac{1}{3c}$
  - ALG packs two items into one bin: release 6 items with size $\frac{1}{2} + \frac{1}{2c}$

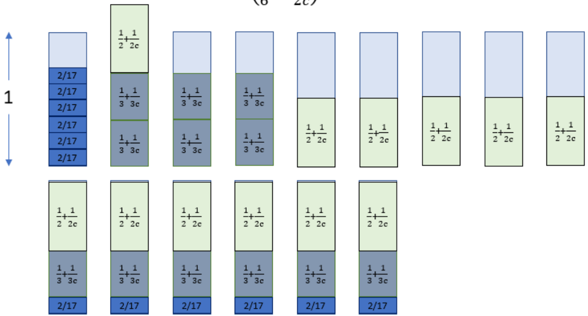


Lower Bound: 5/3 when $c \geq 17$ (3/3)

Item sequence: release 6 items with size $2/17$

- ALG opens a single bin: release 6 items with size $\frac{1}{3} + \frac{1}{3c}$
  - ALG packs two items into one bin: release 6 items with size $\frac{1}{2} + \frac{1}{2c}$

$$\Delta\text{cost} = c\left(\frac{1}{6} + \frac{1}{2c}\right) > 1$$

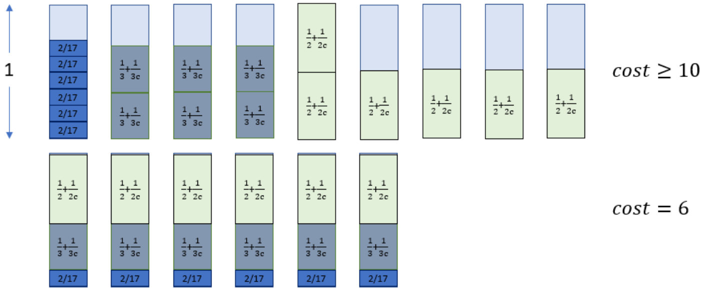


$cost = 6$

Lower Bound: 5/3 when $c \geq 17$ (3/3)

Item sequence: release 6 items with size 2/17

- ALG opens a single bin: release 6 items with size $\frac{1}{3} + \frac{1}{3c}$
  - ALG packs two items into one bin: release 6 items with size $\frac{1}{2} + \frac{1}{2c}$



Upper bounds

- A trivial 2-competitive algorithm
  (Tight for algorithms without overload)
- First-Fit algorithm with fixed overload

A trivial 2-competitive Algorithm

- If $c \leq 1$, then pack all items into one bin.
- If $c > 1$, use Any-Fit algorithm

Proof.

- Obs: any two opened bins has total size > 1
- Obs: $cost_{OPT} \geq \sum_i w_i$ when $c \geq 1$

Suppose Any-Fit opens k bins $cost(AF) = k$
$cost_{OPT} > k/2$ □

A trivial 2-competitive Algorithm

- If $c \leq 1$, then pack all items into one bin.
- If $c > 1$, use Any-Fit algorithm

Proof.

- Obs: any two opened bins has total size > 1
- Obs: $cost_{OPT} \geq \sum_i w_i$ when $c \geq 1$

Suppose Any-Fit opens k bins $cost(AF) = k$
$cost_{OPT} > k/2$ □

Tight case: two items with size $1/2 + \epsilon$

First-Fit Algorithm with Fixed Overload (FFO)

Use First-Fit with a fixed overload, $O(c)$ for each bin

$$O(c) = \begin{cases} \infty, & \text{if } 0 < c \leq \frac{3}{2} \\ \frac{1}{c}, & \text{if } \frac{3}{2} < c \leq \frac{9}{5} \\ \frac{2}{3c}, & \text{if } \frac{9}{5} < c \leq \frac{14}{3} \\ \frac{1}{3c}, & \textit{otherwise} \end{cases}$$

First-Fit Algorithm with Fixed Overload (FFO)

Use First-Fit with a fixed overload, $O(c)$ for each bin

$$O(c) = \begin{cases} \infty, & \text{if } 0 < c \leq \frac{3}{2} \\ \frac{1}{c}, & \text{if } \frac{3}{2} < c \leq \frac{9}{5} \\ \frac{2}{3c}, & \text{if } \frac{9}{5} < c \leq \frac{14}{3} \\ \frac{1}{3c}, & \textit{otherwise} \end{cases}$$

FFO: Pack item into the first opened bin where it fits, total size is not more than $1 + O(c)$, or opens a new bin if the item does not fit into any currently opened bin.

First-Fit Algorithm with Fixed Overload (FFO)

Use First-Fit with a fixed overload, $O(c)$ for each bin

$$O(c) = \begin{cases} \infty, & \text{if } 0 < c \leq \frac{3}{2} \\ \frac{1}{c}, & \text{if } \frac{3}{2} < c \leq \frac{9}{5} \\ \frac{2}{3c}, & \text{if } \frac{9}{5} < c \leq \frac{14}{3} \\ \frac{1}{3c}, & \textit{otherwise} \end{cases}$$

FFO: Pack item into the first opened bin where it fits, total size is not more than $1 + O(c)$, or opens a new bin if the item does not fit into any currently opened bin.

Observation

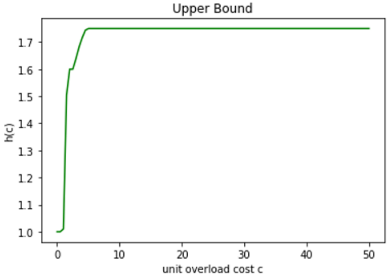
- Obs.1: Total size of two opened bins by FFO $> 1 + O(c)$
- Obs.2: At least $k - 1$ bins have size $> \frac{1+O(c)}{2}$

Upper Bound

Theorem

For any $c \geq 0$: FFO is a $h(c)$-competitive algorithm for BPOC.

$$h(c) = \begin{cases} \max(1, c) & \text{if } 0 \leq c \leq \frac{3}{2} \\ \frac{3+c}{3} & \text{if } \frac{3}{2} < c \leq \frac{9}{5} \\ \frac{8}{5} = 1.6 & \text{if } \frac{9}{5} < c \leq \frac{8}{3} \\ \frac{6c}{3c+2} & \text{if } \frac{8}{3} < c \leq \frac{14}{3} \\ \frac{7}{4} = 1.75 & \text{if } \frac{14}{3} < c \end{cases}$$

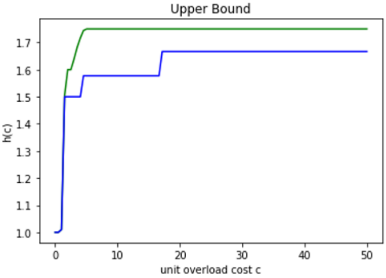


Upper Bound

Theorem

For any $c \geq 0$: FFO is a $h(c)$-competitive algorithm for BPOC.

$$h(c) = \begin{cases} \max(1, c) & \text{if } 0 \leq c \leq \frac{3}{2} \\ \frac{3+c}{3} & \text{if } \frac{3}{2} < c \leq \frac{9}{5} \\ \frac{8}{5} = 1.6 & \text{if } \frac{9}{5} < c \leq \frac{8}{3} \\ \frac{6c}{3c+2} & \text{if } \frac{8}{3} < c \leq \frac{14}{3} \\ \frac{7}{4} = 1.75 & \text{if } \frac{14}{3} < c \end{cases}$$



Upper Bound when $c \leq 3/2$

For $c \leq 3/2$, $O(c) = \infty$, FFO pack all items into one bin

Upper Bound when $c \leq 3/2$

For $c \leq 3/2$, $O(c) = \infty$, FFO pack all items into one bin

Theorem

For any $c \leq 3/2$: FFO is a $\max\{1, c\}$-competitive algorithm for BPOC.

Proof.

Suppose the total size of items is W

- $c \leq 1$, $cost_{FFO} = cost_{OPT}$
- $c > 1$,

$$cost_{FFO} < c \cdot W$$

$$cost_{OPT} \geq W$$

□

Intuition of UB proof when $3/2 < c < 14/3$

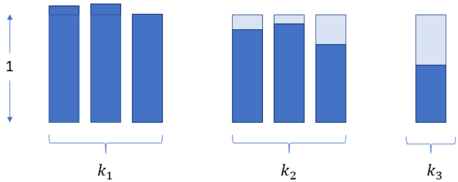
Idea: Estimate the total item size based on the solution of FFO
Bound cost of an optimal solution by $cost_{OPT} \geq$ total item size

Intuition of UB proof when $3/2 < c < 14/3$

Idea: Estimate the total item size based on the solution of FFO
Bound cost of an optimal solution by $cost_{OPT} \geq$ total item size

Partition the k bins used by FFO:

- k_1: bins with total size $[1, 1 + O(c)]$
- k_2: bins with total size $[\frac{1+O(c)}{2}, 1)$
- $k_3 \leq 1$: bins with total size $(0, \frac{1+O(c)}{2})$



Optimal cost: more bins with overload cost, more total item size

Upper Bound when $3/2 < c \leq 9/5$ (1/2)

For $3/2 < c \leq 9/5$, $O(c) = 1/c$

Theorem

For any $3/2 < c \leq 9/5$: FFO is a $\frac{3+c}{3}$-competitive algorithm for BPOC.

Proof.

- o: average overload= overload$/k_1$, $0 \leq o \leq 1/c$

$$cost_{FFO} = k + c \cdot k_1 \cdot o$$

Upper Bound when $3/2 < c \leq 9/5$ (1/2)

For $3/2 < c \leq 9/5$, $O(c) = 1/c$

Theorem

For any $3/2 < c \leq 9/5$: FFO is a $\frac{3+c}{3}$-competitive algorithm for BPOC.

Proof.

- o: average overload= overload$/k_1$, $0 \leq o \leq 1/c$

$$cost_{FFO} = k + c \cdot k_1 \cdot o$$

$$?? \quad \frac{cost_{FFO}}{cost_{OPT}}$$

□

Upper Bound when $3/2 < c \leq 9/5$ (2/2)

Known: $cost_{FFO} = k + c \cdot k_1 \cdot o$

Proof.

- $k_2 \geq 1$: $cost_{OPT} > k_1 + k_1 \cdot o + \frac{1+1/c}{2}(k - k_1)$ (Obs. 2)

$$\frac{cost_{FFO}}{cost_{OPT}} \leq \frac{k + c \cdot k_1 \cdot o}{k_1 + k_1 \cdot o + (1/2 + 1/(2c))(k - k_1)} \leq (3 + c)/3$$

Upper Bound when $3/2 < c \leq 9/5$ (2/2)

Known: $cost_{FFO} = k + c \cdot k_1 \cdot o$

Proof.

- $k_2 \geq 1$: $cost_{OPT} > k_1 + k_1 \cdot o + \frac{1+1/c}{2}(k - k_1)$ (Obs. 2)

$$\frac{cost_{FFO}}{cost_{OPT}} \leq \frac{k + c \cdot k_1 \cdot o}{k_1 + k_1 \cdot o + (1/2 + 1/(2c))(k - k_1)} \leq (3 + c)/3$$

- $k_2 = 0$: $cost_{OPT} > k_1 + \min\{k_1 \cdot o, (k_1 - 1) \cdot o + \frac{1}{c}\}$ (Obs. 1)

$$\frac{cost_{FFO}}{cost_{OPT}} \leq \frac{k_1 + c \cdot k_1 \cdot o + 1}{k_1 + k_1 \cdot o} < (3 + c)/3$$

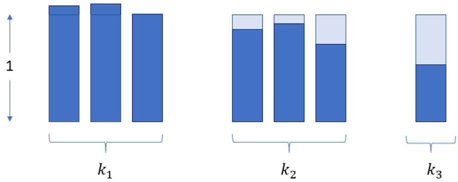
□

(Recall) Intuition of Upper Bound proof

Idea: Estimate the total item size based on the solution of FFO
Bound cost of an optimal solution by $cost_{OPT} \geq$ total item size

Partition the set of bins used by FFO:

- k_1: bins with total size $[1, 1 + O(c)]$
- k_2: bins with total size $[\frac{1+O(c)}{2}, 1)$
- $k_3 \leq 1$: bins with total size $(0, \frac{1+O(c)}{2})$

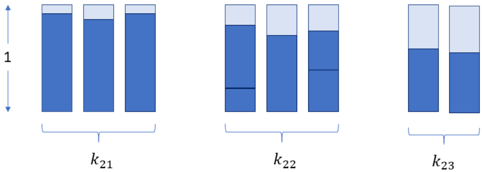


Optimal cost: more bins with overload cost, more total item size

(Further) Intuition of UB proof when c is large

Partition k_2 bins with size $[\frac{1+O(c)}{2}, 1)$ by FFO:

- k_{21}: bins with total size $[\frac{3(1+O(c))}{4}, 1)$
- k_{22}: bins with total size $[\frac{2(1+O(c))}{3}, \frac{3(1+O(c))}{4})$
- k_{23}: bins with total size $[\frac{1+O(c)}{2}, \frac{2(1+O(c))}{3})$

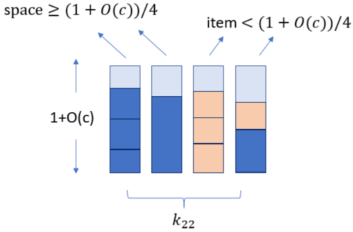


(Further) Intuition of UB proof when c is large

Observation

Except the earliest opened bin of $k_{22} + k_{23}$:

- Obs 3. Bin size $[\frac{2(1+O(c))}{3}, \frac{3(1+O(c))}{4})$: ≤ 2 items, size $> \frac{1+O(c)}{4}$
- Obs 4. Bin size $[\frac{1+O(c)}{2}, \frac{2(1+O(c))}{3})$: 1 item, size $> \frac{1+O(c)}{2}$



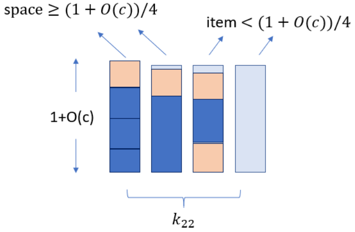
$$[\frac{2(1+O(c))}{3}, \frac{3(1+O(c))}{4})$$

(Further) Intuition of UB proof when c is large

Observation

Except the earliest opened bin of $k_{22} + k_{23}$:

- Obs 3. Bin size $[\frac{2(1+O(c))}{3}, \frac{3(1+O(c))}{4})$: ≤ 2 items, size $> \frac{1+O(c)}{4}$
- Obs 4. Bin size $[\frac{1+O(c)}{2}, \frac{2(1+O(c))}{3})$: 1 item, size $> \frac{1+O(c)}{2}$



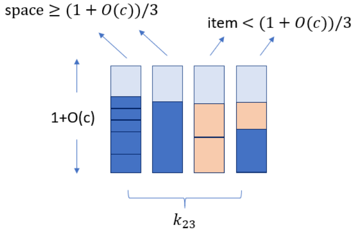
$$[\frac{2(1+O(c))}{3}, \frac{3(1+O(c))}{4})$$

(Further) Intuition of UB proof when c is large

Observation

Except the earliest opened bin of $k_{22} + k_{23}$:

- Obs 3. Bin size $[\frac{2(1+O(c))}{3}, \frac{3(1+O(c))}{4})$: ≤ 2 items, size $> \frac{1+O(c)}{4}$
- Obs 4. Bin size $[\frac{1+O(c)}{2}, \frac{2(1+O(c))}{3})$: 1 item, size $> \frac{1+O(c)}{2}$



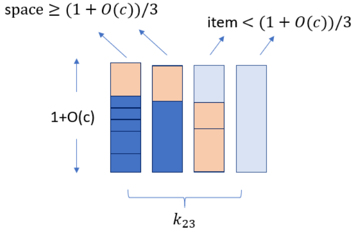
$$[\frac{1+O(c)}{2}, \frac{2(1+O(c))}{3})$$

(Further) Intuition of UB proof when c is large

Observation

Except the earliest opened bin of $k_{22} + k_{23}$:

- Obs 3. Bin size $[\frac{2(1+O(c))}{3}, \frac{3(1+O(c))}{4})$: ≤ 2 items, size $> \frac{1+O(c)}{4}$
- Obs 4. Bin size $[\frac{1+O(c)}{2}, \frac{2(1+O(c))}{3})$: 1 item, size $> \frac{1+O(c)}{2}$



$$[\frac{1+O(c)}{2}, \frac{2(1+O(c))}{3})$$

(Further) Intuition of UB proof when c is large

Observation

Except the earliest opened bin of $k_{22} + k_{23}$:

- Obs 3. Bin size $[\frac{2(1+O(c))}{3}, \frac{3(1+O(c))}{4})$: ≤ 2 items, size $> \frac{1+O(c)}{4}$
- Obs 4. Bin size $[\frac{1+O(c)}{2}, \frac{2(1+O(c))}{3})$: 1 item, size $> \frac{1+O(c)}{2}$

Optimal cost:

- Large item: bound the number of large items in each bin
- Small item: cost $\approx$ size

Upper Bound when $c > 14/3$

For $c > 14/3$, $O(c) = \frac{1}{3c}$

Upper Bound when $c > 14/3$

For $c > 14/3$, $O(c) = \frac{1}{3c}$

Lemma 1

For items in $k_{23} + k_3$ bins, the optimal cost $\geq \frac{2}{3}(k_{23} + k_3)$.

Proof.

- Except the earliest opened bin, each has exactly 1 item
- Any two bin's item has size $> 1 + \frac{1}{3c}$, cost $> 4/3$

$\implies$ Each bin's item has cost $> \frac{2}{3}$

□

Upper Bound when $c > 14/3$

FFO algorithm: when $c > 14/3$, $O(c) = \frac{1}{3c}$

Lemma 2

For items in $k_{22} + k_{23}$ bins, the optimal cost $\geq \frac{2}{3}(k_{22} + k_{23} - 1)$.

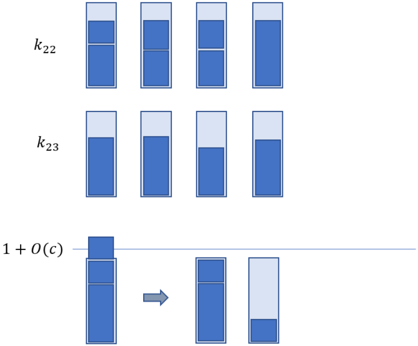
- Except one bin, k_{22} bins each has ≤ 2 items
- k_{23} bins each has 1 item

Upper Bound when $c > 14/3$

FFO algorithm: when $c > 14/3$, $O(c) = \frac{1}{3c}$
For item size $> 1 + O(c)$, cost $> 1 + c \times \frac{1}{3c} = \frac{4}{3}$

Lemma 2

For items in $k_{22} + k_{23}$ bins, the optimal cost $\geq \frac{2}{3}(k_{22} + k_{23} - 1)$.

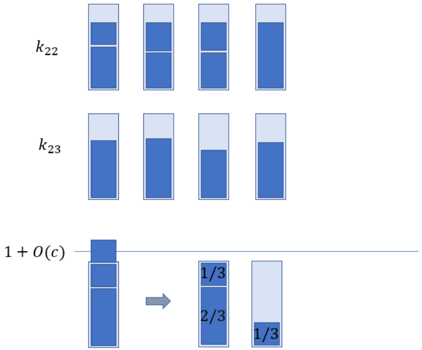


Upper Bound when $c > 14/3$

FFO algorithm: when $c > 14/3$, $O(c) = \frac{1}{3c}$
For item size$> 1 + O(c)$, cost $> 1 + c \times \frac{1}{3c} = \frac{4}{3}$

Lemma 2

For items in $k_{22} + k_{23}$ bins, the optimal cost $\geq \frac{2}{3}(k_{22} + k_{23} - 1)$.

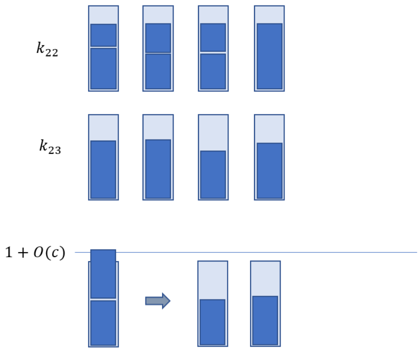


Upper Bound when $c > 14/3$

FFO algorithm: when $c > 14/3$, $O(c) = \frac{1}{3c}$
For item size $> 1 + O(c)$, cost $> 1 + c \times \frac{1}{3c} = \frac{4}{3}$

Lemma 2

For items in $k_{22} + k_{23}$ bins, the optimal cost $\geq \frac{2}{3}(k_{22} + k_{23} - 1)$.

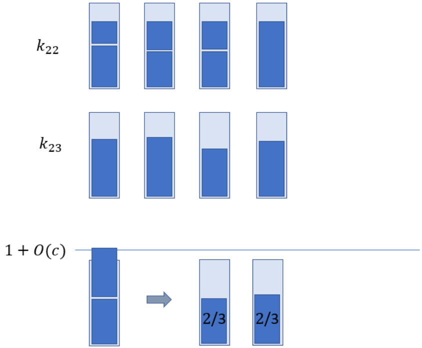


Upper Bound when $c > 14/3$

FFO algorithm: when $c > 14/3$, $O(c) = \frac{1}{3c}$
For item size$> 1 + O(c)$, cost $> 1 + c \times \frac{1}{3c} = \frac{4}{3}$

Lemma 2

For items in $k_{22} + k_{23}$ bins, the optimal cost $\geq \frac{2}{3}(k_{22} + k_{23} - 1)$.

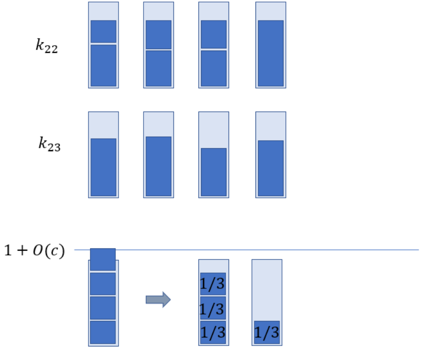


Upper Bound when $c > 14/3$

FFO algorithm: when $c > 14/3$, $O(c) = \frac{1}{3c}$
For item size$> 1 + O(c)$, cost $> 1 + c \times \frac{1}{3c} = \frac{4}{3}$

Lemma 2

For items in $k_{22} + k_{23}$ bins, the optimal cost $\geq \frac{2}{3}(k_{22} + k_{23} - 1)$.

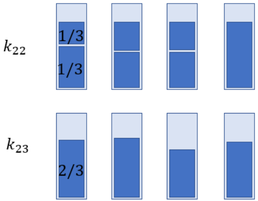


Upper Bound when $c > 14/3$

FFO algorithm: when $c > 14/3$, $O(c) = \frac{1}{3c}$
For item size $> 1 + O(c)$, cost $> 1 + c \times \frac{1}{3c} = \frac{4}{3}$

Lemma 2

For items in $k_{22} + k_{23}$ bins, the optimal cost $\geq \frac{2}{3}(k_{22} + k_{23} - 1)$.



$$OPT \geq \frac{2}{3}(k_{22} + k_{23} - 1)$$

Upper Bound when $c > 14/3$

For $c > 14/3$, $O(c) = \frac{1}{3c}$

Theorem

For any $c > 14/3$: FFO is a $7/4$ -competitive algorithm for BPOC.

Proof.

Wlog, suppose the size of items in $k_1 + k_{21}$ is small.
Size of small items $\geq k_1 + k_1 \cdot o + \frac{3(1+O(c))}{4} k_{21}$

Analyze $cost_{OPT}$ by distinguish $k_{23} + k_3$ and $k_{22} + k_{23}$ using:

- Items in $k_{23} + k_3$, the optimal cost $\geq \frac{2}{3}(k_{23} + k_3)$
- Items in $k_{22} + k_{23}$, the optimal cost $\geq \frac{2}{3}(k_{22} + k_{23} - 1)$

□

Content

1 Problem

2 Our Result

3 Extension

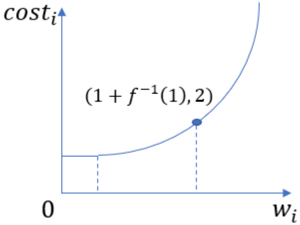
4 Conclusion

Convex Overload Cost

Setting

- *Item p* has size $p \in (0, 1]$ arrives one by one
  Each item must be placed into a bin before the next item arrives
- Bin: extensible, overload cost function $f(x)$
  infinite number of identical bins

Sup. total size of items in bin i is w_i, $cost_i = 1 + f(w_i - 1)$



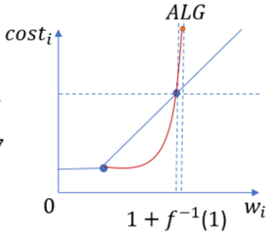
Goal: Pack item into a bin with the minimum cost

Convex Overload Cost: Lower Bounds

Theorem

For any convex cost function f: no deterministic online algorithm for BPOC can achieve a competitive ratio smaller than g.

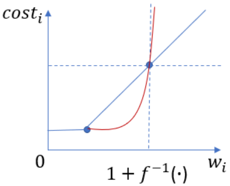
$$g = \begin{cases} \max(1, 1/f^{-1}(1)) & \text{if } 1/f^{-1}(1) \le \frac{3}{2} \\ \frac{3}{2} = 1.5 & \text{if } \frac{3}{2} < 1/f^{-1}(1) \le 1 + 2\sqrt{3} \\ 1 + \frac{\sqrt{3}}{3} \approx 1.577 & \text{if } 1 + 2\sqrt{3} < 1/f^{-1}(1) < 17 \\ \frac{5}{3} \approx 1.667 & \text{if } 17 \le 1/f^{-1}(1) \end{cases}$$



Convex Overload Cost: Upper Bounds

Theorem for convex overload cost

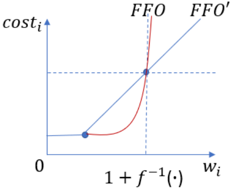
For any convex cost function f with $f^{-1}(1) \leq \frac{2}{3}$: FFO with overload $f^{-1}(\cdot)$ is a $h(f^{-1}(\cdot))(1 + f^{-1}(\cdot))$-competitive algorithm for BPOC.



Convex Overload Cost: Upper Bounds

Theorem for convex overload cost function

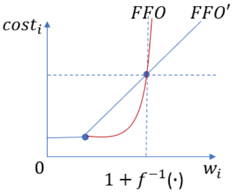
For any convex cost function f with $f^{-1}(1) \leq \frac{2}{3}$: FFO with overload $f^{-1}(\cdot)$ is a $h(f^{-1}(\cdot))(1 + f^{-1}(\cdot))$-competitive algorithm for BPOC.

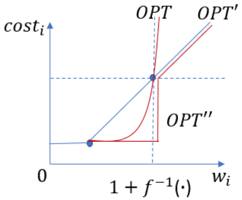


Convex Overload Cost: Upper Bounds

Theorem for convex overload cost function

For any convex cost function f with $f^{-1}(1) \le \frac{2}{3}$: FFO with overload $f^{-1}(\cdot)$ is a $h(f^{-1}(\cdot))(1 + f^{-1}(\cdot))$-competitive algorithm for BPOC.



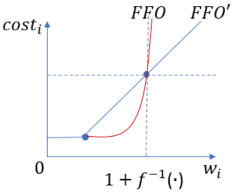


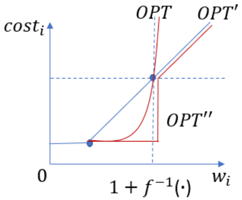
$OPT' \ge h \cdot FFO' \ge h \cdot FFO$ $\quad$ $OPT \ge OPT'' \ge (1 + f^{-1}(\cdot)) \cdot OPT'$

Convex Overload Cost: Upper Bounds

Theorem for convex overload cost function

For any convex cost function f with $f^{-1}(1) \leq \frac{2}{3}$: FFO with overload $f^{-1}(\cdot)$ is a $h(f^{-1}(\cdot))(1 + f^{-1}(\cdot))$-competitive algorithm for BPOC.





$OPT' \geq h \cdot FFO' \geq h \cdot FFO$ $\qquad$ $OPT \geq OPT'' \geq (1 + f^{-1}(\cdot)) \cdot OPT'$

$$\rightarrow OPT \geq h \cdot (1 + f^{-1}(\cdot)) \cdot FFO$$

Convex Overload Cost: Upper Bounds

Theorem for convex overload cost

For any convex cost function f with $f^{-1}(1) \leq \frac{2}{3}$: FFO with overload $f^{-1}(\cdot)$ is a $h(f^{-1}(\cdot))(1 + f^{-1}(\cdot))$-competitive algorithm for BPOC.

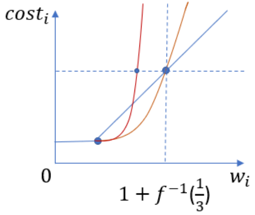
For $f^{-1}(1/3) < \frac{1}{14}$, we have $h(f^{-1}(\cdot)) = \frac{7}{4}$
Competitive ratio: $\frac{7}{4} \cdot (1 + f^{-1}(\frac{1}{3})) < \frac{15}{8} < 2$

Convex Overload Cost: Upper Bounds

Theorem for convex overload cost

For any convex cost function f with $f^{-1}(1) \leq \frac{2}{3}$: FFO with overload $f^{-1}(\cdot)$ is a $h(f^{-1}(\cdot))(1 + f^{-1}(\cdot))$-competitive algorithm for BPOC.

For $f^{-1}(1/3) < \frac{1}{14}$, we have $h(f^{-1}(\cdot)) = \frac{7}{4}$
Competitive ratio: $\frac{7}{4} \cdot (1 + f^{-1}(\frac{1}{3})) < \frac{15}{8} < 2$



Content

1. Problem
2. Our Result
3. Extension
4. Conclusion

Conclusion

Absolute competitive analysis for BPOC
Apply LBs and UBs to more general cost functions

Future direction:

- Gap between lower bound and upper bound
- Asymptotic competitive analysis

Thank you!