Dynamic Bin Packing of Unit Fractions Items

Prudence Wong

Joint work with
Joseph Wun-Tat Chan, Tak-Wah Lam, Fencol Yung
(HKU)
Outline

- Background & Problem definition
  - Previous work
  - Our results
    - Lower bound for Worst-fit / Best-fit
    - Upper bound for Any-fit
    - Upper bound for First-fit
  - Ongoing work
(Offline) Bin Packing

Given a set of items, each with size in \((0,1]\), pack all items using the \textbf{minimum} number of size-1 bins.
(Offline) Bin Packing

Given a set of items, each with size in \((0,1]\), pack all items using the minimum number of size-1 bins.
Online Bin Packing

The items are given one by one, need to pack the current item before knowing the next item.

items: [colors and sizes]

bins: [spaces for items to be placed]
Online Bin Packing

- The items are given one by one, need to pack the current item before knowing the next item.
Online algorithms

- Online algorithm A: NO future knowledge
- Optimal offline algorithm O: complete information

- Competitive analysis:
  A is \textit{c-competitive} if
  \[ \text{numBins}(A, \sigma) \leq c \text{numBins}(O, \sigma) \]
  \( \forall \) input \( \sigma \)
Dynamic Bin Packing

- Items may **depart** at any time
- Input is a sequence of item arrivals and item departures
Competitive Analysis

- Assume at each time step there is either an item arrives or an item departs.

- Online algorithm A is \textit{c-competitive} if
  \[ \max_t \text{numBins}(A, \sigma, t) \leq c \max_t \text{numBins}(O, \sigma, t) \]
  \( \forall \) input \( \sigma \)
The size of each item is in the form of \( \frac{1}{i} \), where \( i \) is an integer.

I.e., the sizes of the items can only be one of \{ 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \ldots \}.

Motivated by the windows scheduling problem.
Outline

- Background & Problem definition

- Previous work

- Our results
  - Lower bound for Worst-fit / Best-fit
  - Upper bound for Any-fit
  - Upper bound for First-fit

- Ongoing work
Previous Work: Arbitrary Size Items

- Offline bin packing is NP-hard
- Online bin packing
  - no algorithm is 1.54014-competitive [Vliet 92]
  - HARMONIC++ 1.58889-competitive [Seiden 02]

**Basic ideas:** classify item sizes into a set of predefined *intervals*, each interval is then scheduled separately
Previous Work: Unit Fraction Items

[Bar-Noy & Ladner & Tamir 04]

- **Offline**
  - Lower bound: \( \lceil h(\sigma) \rceil \) bins
  - Any-fit decreasing uses at most \( \lceil h(\sigma) \rceil + 1 \) bin

- **Online**
  - Any-fit is 6/5-competitive
  - Lower bound: \( h(\sigma) + \Omega(\ln h(\sigma)) \) bins
  - Algorithm \( B^*_\text{dyn} \) uses at most \( h(\sigma) + O(h(\sigma)^{1/2}) \) bins

- Optimal offline algorithm uses at least \( \lceil h(\sigma) \rceil \) bins.
- \( B^*_\text{dyn} \) is asymptotically optimal
Previous Work: Dynamic BP

- Dynamic bin packing of arbitrary size items
  [Coffman & Garey & Johnson 83]
  - First-fit has a competitive ratio between 2.75 and 2.897
  - Lower bound of the competitive ratio: 2.388
    - If offline can repack, lower bound: 2.5
Outline

- Background & Problem definition
- Previous work
- Our results
  - Lower bound for Worst-fit / Best-fit
  - Upper bound for Any-fit
  - Upper bound for First-fit
- Ongoing work
### Our Results

#### Dynamic bin packing of unit fractions items

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Competitive ratio</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Upper bound</td>
<td>Lower bound</td>
</tr>
<tr>
<td>First-fit</td>
<td>2.4985</td>
<td>2.45</td>
</tr>
<tr>
<td>Best-fit &amp; Worst-fit</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Any-fit</td>
<td>3</td>
<td>2.428</td>
</tr>
</tbody>
</table>

#### No online algorithm is better than 2.428-competitive

- this result improves the previous 2.388 lower bound on arbitrary size items with no repacking
Any-fit Algorithms

- **Any-fit**
  - Pack a new item to an arbitrary occupied bin if the item can fit in
  - Otherwise pack it to a new bin

- **Rule in choosing occupied bin**
  - **First-fit**: The “oldest” existing occupied bin that can fit
  - **Best-fit**: The “heaviest” existing occupied bin that can fit
  - **Worst-fit**: The “lightest” existing occupied bin that can fit
Worst-fit is no better than 3-competitive

The adversary (the item sizes are shown)

Worst-fit:
3k bins

Optimal:
k+2 bins

The competitive ratio = $3 - \frac{6}{k+2}$
Best-fit is no better than 3-competitive

- The adversary (the item sizes are shown)

Best-fit:
- 3k bins

Optimal:
- k+2 bins

The competitive ratio = 3 - 6/(k+2)
Any-fit is 3-competitive

1. \( t_1 \): when any-fit uses \( n \) bins
   - \( m \): \# bins containing size-1 items at \( t_1 \)
   - OPT uses at least \( m \) bins at \( t_1 \)

2. \( k \): largest integer s.t. any-fit packs an item of size \( \leq \frac{1}{2} \) into empty bin \( k \), say at \( t_2 \)
   - at \( t_2 \), all existing \( k-1 \) bins have load > \( \frac{1}{2} \)
   - OPT uses at least \( k/2 \) bins at time \( t_2 \)

\[ \text{OPT uses} \quad \max\{ m, k/2 \} \quad \text{bins} \]

\[ \text{Load: sum of the items sizes packed into a bin} \]

CTAG Seminar Dec 2005
Any-fit is 3-competitive

1. $t_1$: when any-fit uses $n$ bins
   - $m$: # bins containing size-1 items at $t_1$

2. $k$: largest integer s.t. any-fit packs an item of size $\leq \frac{1}{2}$ into empty bin $k$, say at $t_2$

Any-fit uses at most $m+k$ bins

OPT uses $\max\{m, k/2\}$ bins
Any-fit is 3-competitive

1. \( \text{Any-fit uses at most } n \text{ bins} \)

2. Competitive ratio of any-fit:
\[
\leq \frac{(m+k)}{\max\{m, k/2\}} \leq 3
\]

\[\text{any-fit uses at most } m+k \text{ bins}\]

\[\text{OPT uses } \max\{m, k/2\} \text{ bins}\]
First-fit (upper bound)

- generalize the proof for any-fit
  - consider some specific bins used by first-fit
  - bound # bins used by OPT for the items first-fit packs into those specific bins
  - case analysis
First-fit (Sketch)

To define a sequence of integer pairs \((b_i, r_i)\)

- \(b_1\): max # bins used by first-fit
  \(1/r_1\): smallest item first-fit ever packs into \(b_1\)

- \(b_i\): largest integer < \(b_{i-1}\) s.t. first-fit ever packs an item of size < \(1/r_{i-1}\) into \(b_i\)
  \(1/r_i\): smallest item first-fit ever packs into \(b_i\)

- \(k\): largest value of \(i\) s.t. \(b_i\) and \(r_i\) can be defined

Example: \(k=2, r_1=1, r_2=3\)

- only size-1 item is packed in bins \(b_2+1, \ldots, b_1\)
- smallest item ever packed into bins 1, \(\ldots, b_2\) is \(1/3\)
First-fit (Sketch)

when FF packs an item of size $\leq 1/3$ into bin $b_2$

OPT uses $2(b_2-1)/3 + 1/3$ bins

-load $> 2/3$

when FF packs an item of size 1 into bin $b_1$

OPT uses $(b_2-1)/3 + b_2 - b_1$ bins

-load $\geq 1/3$

$\text{FF/OPT} \leq 2$

-Example: $k=2, r_1=1, r_2=3$

- only size-1 item is packed in bins $b_2+1, \ldots, b_1$

- smallest item ever packed into bins 1, $\ldots$, $b_2$ is $1/3$

CTAG Seminar Dec 2005
First-fit (Case Analysis)

- If $r_1 \geq 2$, then $b_1 < 2L + 1$
- If $r_1 = 1$ and $r_2 \geq 3$, then $b_1 < 2.4445L + 1$
- If $r_1 = 1$, $r_2 = 2$ and $r_3 \geq 4$, then $b_1 < 2.4792L + 1.25$
- If $r_1 = 1$, $r_2 = 2$, $r_3 = 3$ and $r_4 \geq 5$, then $b_1 < 2.4942L + 1.3167$
- If $r_1 = 1$, $r_2 = 2$, $r_3 = 3$, $r_4 = 4$ and $r_5 \geq 6$, then $b_1 < 2.49345L + 1.3325$
- If $r_1 = 1$, $r_2 = 2$, $r_3 = 3$, $r_4 = 4$ and $r_5 = 5$, then $b_1 < \mathbf{2.4985L + 1.337}$
Outline

➢ Background & Problem definition
➢ Previous work
➢ Our results
  ❖ Lower bound for Worst-fit / Best-fit
  ❖ Upper bound for Any-fit
  ❖ Upper bound for First-fit
➡ Ongoing work
Ongoing Work

- **Resource augmentation**
  - pioneered by [Kalyanasundaram and Pruhs 95]
  - Online algorithm is given bins of size $b > 1$ while OPT uses size-1 bin

- **Preliminary results**
  - Any-fit using size-$2$ bins is 1-competitive (against OPT using size-1 bins)
  - No online algorithm is 1-competitive if using bins of smaller sizes
  - Some tight bounds on competitive ratio of BF, WF for general $b \leq 2$ and not tight bounds for FF

- **Work in progress**
  - derive general lower bounds for general values of $b$
  - design algorithm with close competitive ratio
Any-fit using size-2 bins

- Suppose any-fit uses n bins
- When any-fit first pack an item into bin n

\[
\text{total load} > n-1+x \\
\text{OPT uses n bins}
\]

\[
\begin{align*}
\text{load} & > 1 \\
\text{n-1 bins}
\end{align*}
\]

\[
AF \leq OPT
\]
Given size-x bins, with $x = 2(1 - 1/k)$

Input $1/k$ ... $k^3$ items

Any online algorithm

Size $= 2 - 2/k$ ≤ $k^2$ bins

Input $1$ ... $1$

$k^2 - k + 1$ items

OPT packs $1/k$ items into $(k - 1)$ bins, total = $k^2$

Online needs to use $k^2 - k + 1$ more bins, total = $k^2 + 1$
Other Direction

- Offline version of Dynamic Bin Packing
  - NP-hard for arbitrary size items
  - NP-hard? for unit fraction items
    - This is unknown even for offline bin packing of unit fraction items
  - Approximation algorithm
The End