COMP 7720 - Online Algorithms

Online Bin Packing

Shahin Kamali

Lecture 18 - Nov. 6th, 2018

University of Manitoba
Review & Plan
Today’s objectives

- Average-case analysis of Best Fit and other algorithms
- An application of bin packing in Cloud
Analysis Measures

- Compare the performance of an online algorithm $A$ with an optimal offline algorithm $\text{OPT}$:
  - $\text{OPT}$ knows the whole sequence in the beginning.
  - $\text{OPT}$ can change its packing at any time.

- Competitive ratio of $A$ is the maximum value of $A(\sigma)/\text{OPT}(\sigma)$ among all sequences $\sigma$.
  - We are interested in the asymptomatic competitive ratio where $\text{OPT}(\sigma)$ is arbitrary large.

- Average case ratio of $A$ is the expected value of $A(\sigma)/\text{OPT}(\sigma)$.
  - Item sizes are generated randomly and independently, from an identical distribution (typically uniform distribution).

- Expected waste of $A$ is the expected value of $A(\sigma) - \text{OPT}(\sigma)$. 
Average-Case Analysis

- Consider **upright matching** problem.
  - We are given \( n \) points in a \( 1 \times 1 \) coordinate
  - The goal is to match a maximum number of \( \ominus \) with \( \oplus \) points
  - Each \( \ominus \) point can be matched only to \( \oplus \) points on its upright position.
  - Labels and positions of points are i.i.d. random variables.

- **Greedy algorithm**: process *ominus* points one by one from top to bottom
  - Match each \( \ominus \) item with the left-most unmatched \( \oplus \) item above it.

- It is known that Greedy matches all points except and expected number of \( \Theta(\sqrt{n} \log^{3/4} n) \) points.
Reduction of bin packing to upright matching

- Consider a bin packing sequence of length \( n \) with item sizes randomly distributed in \((0, 1]\).

- Create an instance of upright matching:
  - Items are mapped to points in the square.
  - An item of size \( \alpha > 0.5 \) gets an \( \oplus \) label and \( x \)-coordinate \( 2(1 - \alpha) \).
  - An item of size \( \alpha \leq 0.5 \) gets an \( \ominus \) label and \( x \)-coordinate \( 2\alpha \).
  - \( y \)-coordinate of the item at index \( i \) is set randomly in \([i/n], [i/n] \).
  - E.g., \( \sigma = \langle 0.53, 0.69, 0.21, 0.78, 0.4 \rangle \)
Reduction of bin packing to upright matching

- Points receive random labels (with a chance of 0.5 an item is larger than 0.5 ($\oplus$) and with a chance of 0.5 it is $\leq 0.5$ ($\ominus$)).

- Points $x$-coordinates are random
  - for an $\oplus$ point, item size $x$ is random in $U(0.5, 1]$ and hence $2(1 - x)$ is random in $U[0, 1)$
  - for an $\ominus$ point, item size $x$ is random in $U(0, 0.5]$ and hence $2(x)$ is random in $U(0, 1]$

- Points $y$-coordinates are random
  - Exactly one point is located randomly in $U[i/n, (i + 1)/n)$
Reduction of bin packing to upright matching

What is the equivalent of greedy algorithm?

- An \( \oplus \) point \( y \) appears on the right of \( x \) if sum of items \( x \) and \( y \) is less than 1.
  - \( y \) is on right of \( x \) \( \rightarrow \)
    \[ 2(1 - y) \geq 2x \rightarrow x + y \leq 1 \]

Greedy matches each \( \ominus \) point \( p \) (item \( x \leq 0.5 \)) with the leftmost \( \oplus \) point (largest item \( y \) so that \( y > 0.5 \)) that appears above (i.e., \( y \) is before \( x \) in the sequence) and on the right of \( p \) (i.e., \( x + y \leq 1 \)).
Best Fit & upright matching

- Greedy is equivalent to **Almost Best Fit**:  
  - If \( x > 1/2 \), open a new bin for \( x \).
  - If \( x \leq 1/2 \), place \( x \) with an item \( y \geq 0.5 \) which best fits \( x \) (i.e., largest such \( y \) so that \( x + y \leq 1 \)).
  - If no such \( y \) exists, open a new bin for \( x \).

- **Almost Best Fit** is similar to Best Fit except that:
  - It closes a bin as soon as an item of size \( \leq 1/2 \) is placed in it.

- **any sequence, the cost of Best Fit is at most equal to Almost-Best-Fit**
Average-case analysis of Best Fit

- Number of unmatched point by greedy is expected to be $\Theta(\sqrt{n} \log^{3/4} n)$.
- So, the number of bins in Almost Best Fit (ABF) is expected to be $(n - \Theta(\sqrt{n} \log^{3/4} n))/2 + \Theta(\sqrt{n} \log^{3/4} n) = n/2 + \Theta(\sqrt{n} \log^{3/4} n)$.
- The cost of Best Fit is at most $n/2 + \Theta(\sqrt{n} \log^{3/4} n)$ for a sequence of length $n$ on expectation.
- The cost of $\text{OPT}$ is expected to be at least $n/2$ (since half items are expected to be larger than 0.5).
- Average case ratio of ABF (and hence BF) is at most $\frac{n/2 + \Theta(\sqrt{n} \log^{3/4} n)}{n/2} \approx 1$ for large values of $n$.
- Expected waste of ABF (and hence BF) is at most $E(\text{ABF}(\sigma) - \text{OPT}(\sigma)) = n/2 + \Theta(\sqrt{n} \log^{3/4} n) - n/2 = \Theta(\sqrt{n} \log^{3/4} n)$.
The average-case analysis for Harmonic Match is similar to Best Fit; we repeat the same analysis for each class separately.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Competitive Ratio</th>
<th>Average Ratio</th>
<th>Expected waste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Fit ((N_F))</td>
<td>2</td>
<td>1.3 CoHoSY80</td>
<td>(\Omega(n))</td>
</tr>
<tr>
<td>Best Fit ((B_F))</td>
<td>1.7 Johnso73</td>
<td>1 BeJLMM84</td>
<td>(\Theta(\sqrt{n}\log^{3/4} n)) Shor86</td>
</tr>
<tr>
<td>First Fit ((F_F))</td>
<td>1.7 Johnso73</td>
<td>1 LeiSho89</td>
<td>(\Theta(n^{2/3})) Shor86 CoJoSW95</td>
</tr>
<tr>
<td>Refined First Fit</td>
<td>1.6 Yao80A</td>
<td>&gt; 1</td>
<td>(\Omega(n))</td>
</tr>
<tr>
<td>Harmonic ((H_A))</td>
<td>(T_\infty \approx 1.691) LeeLee85</td>
<td>1.2899 LeeLee85</td>
<td>(\Omega(n))</td>
</tr>
<tr>
<td>Refined Harmonic</td>
<td>1.635 LeeLee85</td>
<td>1.2824 GuChXu02</td>
<td>(\Omega(n))</td>
</tr>
<tr>
<td>Modified Harmonic</td>
<td>1.615</td>
<td>1.189</td>
<td>(\Omega(n))</td>
</tr>
<tr>
<td>Harmonic++</td>
<td>1.5888 Seid02</td>
<td>&gt; 1</td>
<td>(\Omega(n))</td>
</tr>
<tr>
<td>Extreme Harmonic</td>
<td>1.5817 Van15</td>
<td>&gt; 1</td>
<td>(\Omega(n))</td>
</tr>
</tbody>
</table>
Experimental Evaluation

- Experimental average-case performance of online algorithms for different distributions.

![Graph showing performance of online algorithms for different distributions](image.png)
In practical scenarios, we should have an eye on both worst-case and average-case performance.

Harmonic algorithms do well in the worst-case (competitive ratio) but have poor average-case performance.

Another family of algorithms, e.g., Sum-of-Square algorithm, have a good average-case performance (better than Best Fit) but have a poor competitive ratio.

There is not necessarily a trade-off between worst-case and average-case performance in bin packing.

We can devise algorithms that are good in both senses → Harmonic-match.
An application of Bin Packing: Fault-tolerant Server Consolidation

"As far as we can tell, the system went down because someone stepped on a crack in the sidewalk."

image: Andrew Toos via CartoonStock
Fault-tolerant Bin Packing
(Server Consolidation in the Cloud)

- Bins represent servers and items are clients (e.g., databases tenants on Amazon or movies on NetFlix).
- Server might fail and it should not interrupt the service (clients should always available).
- Given a sequence of items, place two replicas of each item in different servers
  - Each replica of an item with load $x$ has a load of $x/2$.
  - Think of load as the number of people who watch a NetFlix movie; so each replica requires half bandwidth
- In case of a server’s failure, the load of each replica is redirected to the server that hosts its partner.
Valid Solutions

- Consider sequence \( \langle a = 0.6, b = 0.3, c = 0.6, d = 0.8, e = 0.1, f = 0.4 \rangle \).

- A valid packing:

- An invalid packing:

![Diagram showing valid and invalid packings with items a, b, c, d, e, and f in different containers]
Mirroring Algorithms

- Consider two types of replicas (blue and red), and apply Best Fit for each type separately.
- The level of a bin is never more than 0.5 (otherwise there will be an overflow in case of a bin failure).
- Consider sequence $\langle a = 0.6, b = 0.3, c = 0.6, d = 0.8, e = 0.1, f = 0.4 \rangle$. 

![Diagram showing bin packing with blue and red replicas]
Mirroring Algorithm

- Mirroring algorithms are not better than 2-competitive
- Consider sequence $\langle 2\epsilon_1, 2\epsilon_2, \ldots, 2\epsilon_n \rangle$
- $OPT$ can place all items so that all bins are almost full
  - Each two bin share at most one item!
Horizontal Harmonic (HH) Algorithm

- Like Harmonic, define *classes* for replicas.
  - \( \left( \frac{1}{3}, \frac{1}{2} \right], \left( \frac{1}{4}, \frac{1}{3} \right], \ldots, \left( \frac{1}{K}, \frac{1}{K-1} \right], (0, \frac{1}{K}] \) (E.g., \( K = 30 \)).
- Treat members of each class separately.
  - No two bins share more than one replica.
Horizontal Harmonic (HH) Algorithm

- Consider sequence \( \langle a_1, a_2, \ldots, a_m \rangle \) of replicas of the same class (E.g., for class 3, replicas lie in the range \((1/5, 1/4]\)).

- Place \( i \) blue replicas of class \( i < K \) in the same bin.

- Place red replicas whose partners are in the same bin in different bins.

  - This ensures a valid packing.
Horizontal Harmonic (HH) Algorithm

- Real-world implementation of Horizontal-Harmonic shows promising performance (ongoing research).
- The algorithms works well in both worst-case and average-case.
- In the next class, we use a weighting function to show Horizontal Harmonic has a competitive ratio of at most 1.59.