COMP 7720 - Online Algorithms

Online Bin Packing

Shahin Kamali

Lecture 17 - Nov. 1st, 2018

University of Manitoba
Review & Plan
Today’s objectives

- A review of weighting argument and competitive ratio of Best Fit/First Fit
- Worst-case vs Average case: practical algorithms
- Average-case analysis of Best Fit and other algorithms
Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]\).

- Place members of each class separately from others.
Harmonic Algorithm classes: $(\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]$.

Place members of each class separately from others.

- Harmonic $\ K = 4$

<br>

\[
\begin{align*}
&\ < 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ \ldots >
\end{align*}
\]
Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K - 1}], (0, \frac{1}{K}]\).

- Place members of each class separately from others.

**Harmonic** \(K = 4\)

\[< 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ldots >\]
Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]\).

- Place members of each class separately from others.

Harmonic Algorithm

\[ K = 4 \]

\[
\begin{array}{c}
< 0.9 0.3 0.8 0.5 0.1 0.1 0.3 0.2 0.4 0.2 0.4 0.5 0.5 0.8 0.6 0.4 0.5 ... > \\
\end{array}
\]
Harmonic Algorithm classes: \( \left( \frac{1}{2}, 1 \right], \left( \frac{1}{3}, \frac{1}{2} \right], \ldots, \left( \frac{1}{K}, \frac{1}{K-1} \right], (0, \frac{1}{K}] \). Place members of each class separately from others.

\[Harmonic\quad K = 4\]

\(< 0.9\ 0.3\ 0.8\ 0.5\ 0.1\ 0.1\ 0.3\ 0.2\ 0.4\ 0.2\ 0.4\ 0.5\ 0.5\ 0.8\ 0.6\ 0.4\ 0.5\ ... >\]
Harmonic Algorithm

- Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]\).
- Place members of each class separately from others.

\[\text{Harmonic} \quad K = 4\]

\[
\begin{align*}
< 0.9 & \quad 0.3 & \quad 0.8 & \quad 0.5 & \quad 0.1 & \quad 0.1 & \quad 0.3 & \quad 0.2 & \quad 0.4 & \quad 0.2 & \quad 0.4 & \quad 0.5 & \quad 0.5 & \quad 0.8 & \quad 0.6 & \quad 0.4 & \quad 0.5 & \ldots >
\end{align*}
\]

\[
\begin{align*}
x > \frac{1}{2} \quad & \quad 0.9 \quad 0.8 \\
\frac{1}{3} < x \leq \frac{1}{2} \quad & \quad 0.5 \\
\frac{1}{4} < x \leq \frac{1}{3} \quad & \quad 0.3 \\
x \leq \frac{1}{4} \quad & 
\end{align*}
\]
Harmonic Algorithm

Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]\).

Place members of each class separately from others.

*Harmonic* \(K = 4\)

\(< 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ ... >

\[
\begin{align*}
x > \frac{1}{2} \\
\frac{1}{3} < x \leq \frac{1}{2} \\
\frac{1}{4} < x \leq \frac{1}{3} \\
x \leq \frac{1}{4}
\end{align*}
\]
Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K - 1}], (0, \frac{1}{K}].\)

Place members of each class separately from others.

Harmonic \(K = 4\)

\(< 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ldots >\)
Harmonic Algorithm

- Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]\).
- Place members of each class separately from others.

Harmonic Algorithm

- For \(K = 4\):

\[
< 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ 0.3 \ 0.3 \ 0.1 \ 0.1 \ ...
\]
Harmonic Algorithm

Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]\).

- Place members of each class separately from others.

Harmonic \(K = 4\)

\[
\begin{align*}
< & \ 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ ... > \\
\end{align*}
\]
Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}].\)

- Place members of each class separately from others.

\[
\text{Harmonic } K = 4
\]

\[
< 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ldots >
\]
Harmonic Algorithm

- Harmonic Algorithm classes: \( (\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}] \).
- Place members of each class separately from others.

Harmonic Algorithm

\[ K = 4 \]

\[ <0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ ... > \]
Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]\).

Place members of each class separately from others.

\[
\text{Harmonic } K = 4
\]

\[
\begin{align*}
&< 0.9\ 0.3\ 0.8\ 0.5\ 0.1\ 0.1\ 0.3\ 0.2\ 0.4\ 0.2\ 0.4\ 0.5\ 0.5\ 0.8\ 0.6\ 0.4\ 0.5 \ldots > \\
\end{align*}
\]
Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]\).

Place members of each class separately from others.

---

**Harmonic**  \(K = 4\)

\(< 0.9 \, 0.3 \, 0.8 \, 0.5 \, 0.1 \, 0.1 \, 0.3 \, 0.2 \, 0.4 \, 0.2 \, 0.4 \, 0.5 \, 0.5 \, 0.8 \, 0.6 \, 0.4 \, 0.5 \ldots >\)

---

\(x > \frac{1}{2}\)

\(\frac{1}{3} < x \leq \frac{1}{2}\)

\(\frac{1}{4} < x \leq \frac{1}{3}\)

\(x \leq \frac{1}{4}\)
Harmonic Algorithm

- Harmonic Algorithm classes: \( (\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}] \).
- Place members of each class separately from others.

\[
\begin{align*}
\text{Harmonic} & \quad K = 4 \\
< 0.9 & 0.3 0.8 0.5 0.1 0.1 0.3 0.2 0.4 0.2 0.4 0.5 0.5 0.8 0.6 0.4 0.5 \ldots >
\end{align*}
\]
Harmonic Algorithm

Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]\).

Place members of each class separately from others.

Harmonic \(K = 4\)

\[< 0.9, 0.3, 0.8, 0.5, 0.1, 0.1, 0.3, 0.2, 0.4, 0.2, 0.4, 0.5, 0.5, 0.8, 0.6, 0.4, 0.5, \ldots >\]
Harmonic Algorithm

Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]\).

- Place members of each class separately from others.

Harmonic  \(K = 4\)

\(< 0.9 0.3 0.8 0.5 0.1 0.1 0.3 0.2 0.4 0.2 0.4 0.5 0.5 0.8 0.6 0.4 0.5 \ldots >\)
Harmonic Algorithm

- Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]\).
- Place members of each class separately from others.

\[\text{Harmonic } K = 4\]

\(< 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ ... >\]
Harmonic Algorithm

Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]\).

Place members of each class separately from others.

Harmonic \( K = 4 \)

\(< 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ldots >\)
Weighting Technique in a Nutshell

- Step I: Define a weight function \( w(x) \geq x \) for an item of size \( x \)
- Step II: Prove that any bin of the online algorithm has weight 1.
- Step III: Prove that it is not possible to place a total weight more than \( J \) in any empty bin
- The competitive ratio will be \( J \)
We define a weight for each item based on its size. The weight of an item in class $i$ is $1/i$ when $i < k$. The weight of an item of size $x$ in class $k$ is $k \frac{k}{k-1} x$.

<table>
<thead>
<tr>
<th>Harmonic</th>
<th>$K = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt; 0.9$</td>
<td>$0.3$</td>
</tr>
<tr>
<td>$0.5$</td>
<td>$0.1$</td>
</tr>
<tr>
<td>$0.3$</td>
<td>$0.2$</td>
</tr>
<tr>
<td>$0.2$</td>
<td>$0.4$</td>
</tr>
<tr>
<td>$0.5$</td>
<td>$0.8$</td>
</tr>
<tr>
<td>$0.4$</td>
<td>$0.4$</td>
</tr>
<tr>
<td>$0.5$</td>
<td>$0.4$</td>
</tr>
<tr>
<td>$0.5$</td>
<td>$0.4$</td>
</tr>
<tr>
<td>$0.5$</td>
<td>$0.4$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x &gt; 1/2$</th>
<th>$1/3 &lt; x \leq 1/2$</th>
<th>$1/4 &lt; x \leq 1/3$</th>
<th>$x \leq 1/4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight = 1</td>
<td>weight = $1/2$</td>
<td>weight = $1/3$</td>
<td>weight = $4/3x$</td>
</tr>
</tbody>
</table>

COMP 7720 - Online Algorithms   Online Bin Packing
We define a weight of an item of class $i < k$ to be $1/i$ and the weight of an item of class $k$ to be $\frac{k}{k-1} \cdot x$.

We showed the weight of all bins (except at most $k$ of them) is at least 1 in Harmonic’s packing.

We showed the the maximum weight of any bin is at most $J = 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{42} + \ldots \approx 1.691$ when $k$ is large enough.

We often assume $k$ is a constant around 20.

The competitive ratio of the algorithm will be at most $J$. 
Competitive Analysis Of First Fit

- Competitive ratio of First Fit is 1.7
  - More precisely, for any sequence $\sigma$, we have $\text{FF}(\sigma) \leq \lceil 1.7 \text{OPT}(\sigma) \rceil$.

- Use a weighting method!

$$W(\alpha) = \begin{cases} \frac{6}{5} \alpha & \text{for } 0 \leq \alpha \leq \frac{1}{6}, \\ \frac{9}{5} \alpha - \frac{1}{10} & \text{for } \frac{1}{6} < \alpha \leq \frac{1}{3}, \\ \frac{6}{5} \alpha + \frac{1}{10} & \text{for } \frac{1}{3} < \alpha \leq \frac{1}{2}, \\ \frac{6}{5} \alpha + \frac{4}{10} & \text{for } \frac{1}{2} < \alpha \leq 1. \end{cases}$$

- Use case analysis to prove:
  - Total weight of all items in a bin of FF is at least 1
  - Total weight of items in any bin is at most 1.7
Any-Fit family of algorithms

- **Any Almost Any Fit** strategy has a competitive ratio of 1.7
  - Any Fit algorithm: algorithm which avoid opening new bin when one of the currently open bins have enough space
  - Almost Any Fit algorithm: an AnyFit algorithm which avoid Worst-Fit strategy (i.e., avoid placing item in the least full bin)

Proof is similar to First Fit Best Fit has a competitive ratio of 1.7.
Any-Fit family of algorithms

- Any **Almost Any Fit** strategy has a competitive ratio of 1.7
  - Any Fit algorithm: algorithm which avoid opening new bin when one of the currently open bins have enough space
  - Almost Any Fit algorithm: an AnyFit algorithm which avoid Worst-Fit strategy (i.e., avoid placing item in the least full bin)

- Proof is similar to First Fit
Any-Fit family of algorithms

- Any **Almost Any Fit** strategy has a competitive ratio of 1.7
  - Any Fit algorithm: algorithm which avoid opening new bin when one of the currently open bins have enough space
  - Almost Any Fit algorithm: an AnyFit algorithm which avoid Worst-Fit strategy (i.e., avoid placing item in the least full bin)

- Proof is similar to First Fit
- Best Fit has a competitive ratio of 1.7.
Analysis Measures

Compare the performance of an online algorithm $A$ with an optimal offline algorithm $OPT$:

- $OPT$ knows the whole sequence in the beginning.
- $OPT$ can change its packing at any time.
Analysis Measures

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

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

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

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

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

Average performance ratio, expected waste, and competitive ratios for different bin packing algorithms.

<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 (NF)</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best Fit (BF)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Fit (FF)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary of Bin Packing Algorithms

- Average performance ratio, expected waste, and competitive ratios for different bin packing algorithms.

<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 (NF)</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best Fit (BF)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Fit (FF)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refined First Fit</td>
<td>1.6 Yao80A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harmonic (HA)</td>
<td>$T_\infty \approx 1.691$ Lee Lee85</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary of Bin Packing Algorithms

- Average performance ratio, expected waste, and competitive ratios for different bin packing algorithms.

<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 (Nf)</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best Fit (Bf)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Fit (Ff)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refined First Fit</td>
<td>1.6 Yao80A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harmonic (Ha)</td>
<td>$\rightarrow T_\infty \approx 1.691$ Lee Lee85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refined Harmonic</td>
<td>1.635 LeeLee85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modified Harmonic</td>
<td>1.615 BrowLeeLee 89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harmonic++</td>
<td>1.5888 Seid02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extreme Harmonic</td>
<td>1.5817 Heydrich, van Stee 15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary of Bin Packing Algorithms

- Average performance ratio, expected waste, and competitive ratios for different bin packing algorithms.
- Competitive ratio of any algorithm is at least 1.54037 BalBek12

<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 (NF)</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best Fit (BF)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Fit (FF)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refined First Fit</td>
<td>1.6 Yao80A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harmonic (HA)</td>
<td>$T_{\infty} \approx 1.691$ LeeLee85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refined Harmonic</td>
<td>1.635 LeeLee85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modified Harmonic</td>
<td>1.615 BrowLeeLee89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harmonic++</td>
<td>1.5888 Seid02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extreme Harmonic</td>
<td>1.5817 Heydrich, van Stee 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced Harmonic</td>
<td>1.5783 Balogh et al 18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary of Bin Packing Algorithms

- Average performance ratio, expected waste, and competitive ratios for different bin packing algorithms.

- Competitive ratio of any algorithm is at least 1.54037

<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 (NF)</td>
<td>2</td>
<td>1.3 CoHoSY80</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Best Fit (BF)</td>
<td>1.7 Johnso73</td>
<td></td>
<td>Θ(√n log^{3/4} n)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BeJLMM84</td>
<td>Shor86</td>
</tr>
<tr>
<td>First Fit (FF)</td>
<td>1.7 Johnso73</td>
<td>1 LeiSho89</td>
<td>Θ(n^{2/3}) CoJoSW95</td>
</tr>
<tr>
<td>Refined First Fit</td>
<td>1.6 Yao80A</td>
<td>&gt; 1</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Harmonic (HA)</td>
<td>→ T∞ ≈ 1.691</td>
<td>1.2899 LeeLee85</td>
<td>Ω(n)</td>
</tr>
<tr>
<td></td>
<td>Lee Lee85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refined Harmonic</td>
<td>1.635 LeeLee85</td>
<td>1.2824 GuChXu02</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Modified Harmonic</td>
<td>1.615</td>
<td>1.189 RamaTsuga89</td>
<td>Ω(n)</td>
</tr>
<tr>
<td></td>
<td>BrowLeeLee 89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harmonic++</td>
<td>1.5888 Seid02</td>
<td>&gt; 1</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Extreme Harmonic</td>
<td>1.5817</td>
<td>&gt; 1</td>
<td>Ω(n)</td>
</tr>
<tr>
<td></td>
<td>Heydrich, van Stee15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced Harmonic</td>
<td>1.5783</td>
<td>&gt; 1</td>
<td>Ω(n)</td>
</tr>
<tr>
<td></td>
<td>Balogh et al 18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Compromise between Competitive Ratio and Average-case Ratio

Is there an algorithm that performs as well as Best Fit while having better competitive ratio?

<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 (NF)</td>
<td>2</td>
<td>1.3 CoHoSY80</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Best Fit (BF)</td>
<td>1.7 Johnso73</td>
<td>1 BeJLMM84</td>
<td>Θ(√n log^{3/4} n) Shor86</td>
</tr>
<tr>
<td>First Fit (FF)</td>
<td>1.7 Johnso73</td>
<td>1 LeiSho89</td>
<td>Θ(n^{2/3}) Shor86 CoJoSW95</td>
</tr>
<tr>
<td>Refined First Fit</td>
<td>1.6 Yao80A</td>
<td>&gt; 1</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Harmonic (HA)</td>
<td>→ T∞ ≈ 1.691</td>
<td>1.2899 LeeLee85</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Refined Harmonic</td>
<td>1.635 LeeLee85</td>
<td>1.2824 GuChXu02</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Modified Harmonic</td>
<td>1.615</td>
<td>1.189 RamaTsuga89</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Harmonic++</td>
<td>1.588 Seid02</td>
<td>&gt; 1</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Extreme Harmonic</td>
<td>1.5817 Van15</td>
<td>&gt; 1</td>
<td>Ω(n)</td>
</tr>
</tbody>
</table>
Review & Plan

Compromise between Competitive Ratio and Average-case Ratio

Is there an algorithm that performs as well as Best Fit while having better competitive ratio?

<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 (NF)</td>
<td>2</td>
<td>1.3 CoHoSY80</td>
<td>$\Omega(n)$</td>
</tr>
<tr>
<td>Best Fit (BF)</td>
<td>1.7 Johnso73</td>
<td>1 BeJLMM84</td>
<td>$\Theta(\sqrt{n \log^{3/4} n})$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Shor86</td>
</tr>
<tr>
<td>First Fit (FF)</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)</td>
<td>$\rightarrow T_\infty \approx 1.691$</td>
<td>1.2899 LeeLee85</td>
<td>$\Omega(n)$</td>
</tr>
<tr>
<td>LeeLee85</td>
<td></td>
<td></td>
<td></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 RamBrowLeeLee89</td>
<td>1.189 RamaTsuga89</td>
<td>$\Omega(n)$</td>
</tr>
<tr>
<td>Harmonic++</td>
<td>1.588 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>
<tr>
<td>Harmonic Match</td>
<td>$\rightarrow T_\infty \approx 1.691$</td>
<td>1</td>
<td>$\Theta(\sqrt{n \log^{3/4} n})$</td>
</tr>
<tr>
<td>Refined Harmonic Match</td>
<td>1.635</td>
<td>1</td>
<td>$\Theta(\sqrt{n \log^{3/4} n})$</td>
</tr>
</tbody>
</table>
Harmonic Match

Harmonic Match:
- An extension of the classes of Harmonic algorithm.

\[ \frac{1}{3} < x \leq \frac{1}{2} \]
\[ \frac{1}{4} < x \leq \frac{1}{3} \]
\[ \frac{1}{5} < x \leq \frac{1}{4} \]
\[ \frac{1}{k+1} < x \leq \frac{1}{k} \]
Harmonic Match Algorithm

Harmonic Match

- Harmonic Match:
  - An extension of the classes of Harmonic algorithm.

\[
\begin{align*}
  i = 1 & \quad \frac{1}{3} < x \leq \frac{1}{2} \quad \longleftrightarrow \quad \frac{1}{2} < x \leq \frac{2}{3} \\
  i = 2 & \quad \frac{1}{4} < x \leq \frac{1}{3} \quad \longleftrightarrow \quad \frac{2}{3} < x \leq \frac{3}{4} \\
  i = 3 & \quad \frac{1}{5} < x \leq \frac{1}{4} \quad \longleftrightarrow \quad \frac{3}{4} < x \leq \frac{4}{5} \\
  \vdots & \\
  i = k - 1 & \quad \frac{1}{k + 1} < x \leq \frac{1}{k} \quad \longleftrightarrow \quad \frac{k - 1}{k} < x \leq \frac{k}{k + 1} \\
  i = k & \quad x \leq \frac{1}{k + 1} \quad \longleftrightarrow \quad x > \frac{k}{k + 1}
\end{align*}
\]
Harmonic Match Algorithm

Harmonic Match

- Harmonic Match:
  - An extension of the classes of Harmonic algorithm.
  - Apply a relaxed variant of Best Fit on items of each class.

\[
\begin{align*}
i = 1 & \quad \frac{1}{3} < x \leq \frac{1}{2} \quad \leftrightarrow \quad \frac{1}{2} < x \leq \frac{2}{3} \\
i = 2 & \quad \frac{1}{4} < x \leq \frac{1}{3} \quad \leftrightarrow \quad \frac{2}{3} < x \leq \frac{3}{4} \\
i = 3 & \quad \frac{1}{5} < x \leq \frac{1}{4} \quad \leftrightarrow \quad \frac{3}{4} < x \leq \frac{4}{5} \\
\vdots & \\
i = k-1 & \quad \frac{1}{k+1} < x \leq \frac{1}{k} \quad \leftrightarrow \quad \frac{k-1}{k} < x \leq \frac{k}{k+1} \\
i = k & \quad x \leq \frac{1}{k+1} \quad \leftrightarrow \quad x > \frac{k}{k+1}
\end{align*}
\]
Harmonic Match Algorithm

For placing an item of size $x$:
- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.
Harmonic Match Algorithm

For placing an item of size $x$:

- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.

< 0.62 0.28 0.3 0.4 0.79 0.34 0.71 0.21 0.42 0.33 0.22 0.23 0.27 0.21 0.24 >
Harmonic Match Algorithm

For placing an item of size $x$:

- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.
Harmonic Match Algorithm

For placing an item of size $x$:

- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.

< 0.62 0.28 0.3 0.4 0.79 0.34 0.71 0.21 0.42 0.33 0.22 0.23 0.27 0.21 0.24 >

### Example

- Bin with items $0.62$, $0.28$
Harmonic Match Algorithm

For placing an item of size \( x \):
- If \( x > 0.5 \), open a new bin.
- If \( x \leq 0.5 \):
  - Use Best Fit strategy to place \( x \) together with an item \( y > 0.5 \) of the same class.
  - If no such \( y \) exists, place \( x \) together with items of the same class using Next Fit strategy.

\[
< 0.62 \ 0.28 \ 0.3 \ 0.4 \ 0.79 \ 0.34 \ 0.71 \ 0.21 \ 0.42 \ 0.33 \ 0.22 \ 0.23 \ 0.27 \ 0.21 \ 0.24 \ >
\]
Harmonic Match Algorithm

For placing an item of size $x$:
- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.

\[
\begin{align*}
< 0.62 & \quad 0.28 & \quad 0.3 & \quad 0.4 & \quad 0.79 & \quad 0.34 & \quad 0.71 & \quad 0.21 & \quad 0.42 & \quad 0.33 & \quad 0.22 & \quad 0.23 & \quad 0.27 & \quad 0.21 & \quad 0.24 > \\
\end{align*}
\]

```
0.62

0.3

0.28
```
Harmonic Match Algorithm

For placing an item of size $x$:

- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.
Harmonic Match Algorithm

For placing an item of size $x$:

- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.

\[
\begin{array}{c}
< 0.62 \ 0.28 \ 0.3 \ 0.4 \ 0.79 \ 0.34 \ 0.71 \ 0.21 \ 0.42 \ 0.33 \ 0.22 \ 0.23 \ 0.27 \ 0.21 \ 0.24 > \\
\end{array}
\]
Harmonic Match Algorithm

For placing an item of size $x$:

- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.

\begin{align*}
< 0.62 & \ 0.28 & 0.3 & 0.4 & 0.79 & 0.34 & 0.71 & 0.21 & 0.42 & 0.33 & 0.22 & 0.23 & 0.27 & 0.21 & 0.24 > \\
0.62 & & 0.4 & & & & & & & & & & & & \\
0.34 & & & & & & & & & & & & & & \\
0.79 & & & & & & & & & & & & & & \\
0.3 & & 0.28 & & & & & & & & & & & & \\
0.71 & & & & & & & & & & & & & & \\
0.79 & & & & & & & & & & & & & & 
\end{align*}
Harmonic Match Algorithm

For placing an item of size $x$:

- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.

< 0.62  0.28  0.3  0.4  0.79  0.34  0.71  0.21  0.42  0.33  0.22  0.23  0.27  0.21  0.24 >
Harmonic Match Algorithm

For placing an item of size $x$:

- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.

< 0.62  0.28  0.3  0.4  0.79  0.34  0.71  0.21  0.42  0.33  0.22  0.23  0.27  0.21  0.24 >
Harmonic Match Algorithm

For placing an item of size $x$:
- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.

\[
< 0.62 \ 0.28 \ 0.3 \ 0.4 \ 0.79 \ 0.34 \ 0.71 \ 0.21 \ 0.42 \ 0.33 \ 0.22 \ 0.23 \ 0.27 \ 0.21 \ 0.24 >
\]
Harmonic Match Algorithm

For placing an item of size $x$:
- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.
For placing an item of size $x$:
- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.

\[ < 0.62 \ 0.28 \ 0.3 \ 0.4 \ 0.79 \ 0.34 \ 0.71 \ 0.21 \ 0.42 \ 0.33 \ 0.22 \ 0.23 \ 0.27 \ 0.21 \ 0.24 > \]
Harmonic Match Algorithm

For placing an item of size $x$:

- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.

\[
< 0.62 \ 0.28 \ 0.3 \ 0.4 \ 0.79 \ 0.34 \ 0.71 \ 0.21 \ 0.42 \ 0.33 \ 0.22 \ 0.23 \ 0.27 \ 0.21 \ 0.24 >
\]
Harmonic Match Algorithm

For placing an item of size $x$:
- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.

< 0.62  0.28  0.3  0.4  0.79  0.34  0.71  0.21  0.42  0.33  0.22  0.27  0.21  0.24 >
Harmonic Match Algorithm

For placing an item of size $x$:

- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.

< 0.62  0.28  0.3  0.4  0.79  0.34  0.71  0.21  0.42  0.33  0.22  0.23  0.27  0.21  0.24 >
Harmonic Match Algorithm

Harmonic Match vs Harmonic

Packing of Harmonic Match is the same as Harmonic except that some items are ‘removed’ from Harmonic packing.

< 0.62 0.28 0.3 0.4 0.79 0.34 0.71 0.21 0.42 0.33 0.22 0.23 0.27 0.21 0.24 >
Harmonic Match Algorithm

Harmonic Match vs Harmonic

Packing of Harmonic Match is the same as Harmonic except that some items are ‘removed’ from Harmonic packing.
Harmonic Match Algorithm

Competitive Analysis

- Harmonic is a **monotone** algorithm.
  - Removing an item does not increase the number of bins opened by Harmonic.
Harmonic Match Algorithm

Competitive Analysis

- Harmonic is a monotone algorithm.
  - Removing an item does not increase the number of bins opened by Harmonic.

Theorem

For any sequence, the number of bins opened by Harmonic Match is no more than that of Harmonic.
Harmonic is a **monotone** algorithm.

- Removing an item does not increase the number of bins opened by Harmonic.

**Theorem**

*For any sequence, the number of bins opened by Harmonic Match is no more than that of Harmonic.*

- Competitive ratio of Harmonic Match is the same as Harmonic, i.e., $T_\infty \approx 1.691$.

- Unlike Harmonic, First Fit and Best Fit are **anomalous** in the sense that removing items might increase the cost of these algorithms.
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.
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.
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 an expected number of $\Theta(\sqrt{n \log^{3/4} n})$ points.
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 an expected number of $\Theta(\sqrt{n} \log^{3/4} n)$ points.
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 an expected number of \( \Theta(\sqrt{n} \log^{3/4} n) \) points.
Harmonic Match Algorithm

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 an expected number of $\Theta(\sqrt{n} \log^{3/4} n)$ points.
Harmonic Match Algorithm

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 an expected number of \( \Theta(\sqrt{n \log^{3/4} n}) \) points.
Harmonic Match Algorithm

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 $\lfloor i/n \rfloor, \lceil i/n \rceil$.

E.g., $\sigma = \langle 0.53, 0.69, 0.21, 0.78, 0.4 \rangle$
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 \( \lfloor i/n \rfloor, \lceil i/n \rceil \)

E.g., \( \sigma = \langle 0.53, 0.69, 0.21, 0.78, 0.4 \rangle \)
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 $\lfloor i/n \rfloor, \lceil i/n \rceil$

E.g., $\sigma = \langle 0.53, 0.69, 0.21, 0.78, 0.4 \rangle$
Harmonic Match Algorithm

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., $\sigma = \langle 0.53, 0.69, 0.21, 0.78, 0.4 \rangle$
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 $\lfloor i/n \rfloor, \lceil i/n \rceil$.

E.g., $\sigma = \langle 0.53, 0.69, 0.21, 0.78, 0.4 \rangle$
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 $\lfloor i/n \rfloor, \lceil i/n \rceil$

E.g., $\sigma = \langle 0.53, 0.69, 0.21, 0.78, 0.4 \rangle$
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 $\lfloor i/n \rfloor, \lceil i/n \rceil$.

E.g., $\sigma = \langle 0.53, 0.69, 0.21, 0.78, 0.4 \rangle$
Harmonic Match Algorithm

Reduction of bin packing to upright matching

- Points receive random labels (with a chance of 0.5 an item is larger than 0.5 (⊕) and with a chance of 0.5 it is ≤ 0.5 (⊖)).
Harmonic Match Algorithm

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 $2x$ is random in $U(0, 1]$
Harmonic Match Algorithm

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 $2x$ is random in $U(0, 1]$  

- Points $y$-coordinates are random
  - Exactly one point is distributed randomly in the interval $U[i/n, (i + 1)/n)$ on the $y$-axis.
Harmonic Match Algorithm

Reduction of bin packing to upright matching

- So, an instance of bin packing can be reduced to upright matching
- What is the equivalent of greedy algorithm?
Harmonic Match Algorithm

Reduction of bin packing to upright matching

- So, an instance of bin packing can be reduced 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 $x \rightarrow 2(1 - y) \geq 2x \rightarrow x + y \leq 1$
Harmonic Match Algorithm

Reduction of bin packing to upright matching

- So, an instance of bin packing can be reduced 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., $x + y \leq 1$).
Harmonic Match Algorithm

Reduction of bin packing to upright matching

- Greedy is equivalent to **Almost Best Fit**: 

  - If \( x > \frac{1}{2} \), open a new bin for \( x \).
  - If \( x \leq \frac{1}{2} \), place \( x \) with an item \( y \geq 0 \) 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 right after it is opened if the bin is opened by an item of size \( \leq \frac{1}{2} \).
- It closes a bin as soon as two items are placed in it.

For any sequence, the cost of Best Fit is at most equal to Almost-Best-Fit.
Harmonic Match Algorithm

Reduction of bin packing to 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 \( x \).
Harmonic Match Algorithm

Reduction of bin packing to 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 right after it is opened if the bin is opened by an item of size $\leq 1/2$.
  - It closes a bin as soon as two items are placed in it.
Harmonic Match Algorithm

Reduction of bin packing to 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 right after it is opened if the bin is opened by an item of size $\leq 1/2$.
  - It closes a bin as soon as two items are placed in it.

- **For any sequence, the cost of Best Fit is at most equal to Almost-Best-Fit**
Number of unmatched point by greedy is expected to be \( \Theta(\sqrt{n} \log^{3/4} n) \).
Average-case analysis of Best Fit

- Number of unmatched points by greedy is expected to be $\Theta(\sqrt{n} \log^{3/4} n)$.
- So, the number of bins with 1 item in Almost Best Fit (ABF) is at most $\Theta(\sqrt{n} \log^{3/4} n)$ on expectation.
Number of unmatched point by greedy is expected to be \( \Theta(\sqrt{n} \log^{3/4} n) \).

So, the number of bins with 1 item in Almost Best Fit (ABF) is at most \( \Theta(\sqrt{n} \log^{3/4} n) \) on expectation.

The cost of ABF is less than \( \frac{n}{2} + \Theta(\sqrt{n} \log^{3/4} n) \) for a sequence of length \( n \) on expectation.
Harmonic Match Algorithm

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 with 1 item in Almost Best Fit (ABF) is at most \( \Theta(\sqrt{n \log^{3/4} n}) \) on expectation.

- The cost of ABF is less than \( \frac{n}{2} + \Theta(\sqrt{n \log^{3/4} n}) \) for a sequence of length \( n \) on expectation.

- The cost of \( \text{OPT} \) is expected to be at least \( \frac{n}{2} \) (since half items are expected to be larger than 0.5).
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 with 1 item in Almost Best Fit (ABF) is at most \( \Theta(\sqrt{n}\log^{3/4}n) \) on expectation.

- The cost of ABF is less than \( n/2 + \Theta(\sqrt{n}\log^{3/4}n) \) 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
  \[
  \frac{n/2 + \Theta(\sqrt{n}\log^{3/4}n)}{n/2} \approx 1 \text{ for large values of } n
  \]

- Expected waste of ABF (and hence BF) is at most
  \[
  E(ABF(\sigma) - \text{OPT}(\sigma)) = n/2 + \Theta(\sqrt{n}\log^{3/4}n) - n/2 = \Theta(\sqrt{n}\log^{3/4}n)
  \]
Harmonic Match Algorithm

Average-case analysis of Best Fit

- 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 (NF)</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best Fit (BF)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Fit (FF)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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></td>
<td></td>
</tr>
<tr>
<td>Best Fit ((B_F))</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Fit ((F_F))</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refined First Fit</td>
<td>1.6 Yao80A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harmonic ((H_A))</td>
<td>(T_\infty \approx 1.691) LeeLee85</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Harmonic Match Algorithm

Average-case analysis of Best Fit

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 (NF)</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best Fit (BF)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Fit (FF)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refined First Fit</td>
<td>1.6 Yao80A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harmonic (HA)</td>
<td>$T_\infty \approx 1.691$ LeeLee85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refined Harmonic</td>
<td>1.635 LeeLee85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modified Harmonic</td>
<td>1.615 RamBrowLeeLee89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harmonic++</td>
<td>1.5888 Seid02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extreme Harmonic</td>
<td><strong>1.5817</strong> Van15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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></td>
<td></td>
</tr>
<tr>
<td>Best Fit ($B_f$)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Fit ($F_f$)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refined First Fit</td>
<td>1.6 Yao80A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harmonic ($H_A$)</td>
<td>$T_\infty \approx 1.691$ LeeLee85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refined Harmonic</td>
<td>1.635 LeeLee85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modified Harmonic</td>
<td>1.615 RamBrowLeeLee89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harmonic++</td>
<td>1.5888 Seid02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extreme Harmonic</td>
<td>1.5817 Van15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 (NF)</td>
<td>2</td>
<td>1.3 CoHoSY80</td>
<td>Ω((n))</td>
</tr>
<tr>
<td>Best Fit (BF)</td>
<td>1.7 Johnso73</td>
<td>1 BeJLMM84</td>
<td>Θ((\sqrt{n \log^{3/4} n})) Shor86</td>
</tr>
<tr>
<td>First Fit (FF)</td>
<td>1.7 Johnso73</td>
<td>1 LeiSho89</td>
<td>Θ((n^{2/3})) Shor86 CoJoSW95</td>
</tr>
<tr>
<td>Refined First Fit</td>
<td>1.6 Yao80A</td>
<td>&gt; 1</td>
<td>Ω((n))</td>
</tr>
<tr>
<td>Harmonic (HA)</td>
<td>(T_\infty \approx 1.691) LeeLee85</td>
<td>1.2899 LeeLee85</td>
<td>Ω((n))</td>
</tr>
<tr>
<td>Refined Harmonic</td>
<td>1.635 LeeLee85</td>
<td>1.2824 GuChXu02</td>
<td>Ω((n))</td>
</tr>
<tr>
<td>Modified Harmonic</td>
<td>1.615</td>
<td>1.189</td>
<td>Ω((n))</td>
</tr>
<tr>
<td></td>
<td>RamBrowLeeLee89</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RamaTsuga89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Harmonic++</td>
<td>1.5888 Seid02</td>
<td>&gt; 1</td>
<td>Ω((n))</td>
</tr>
<tr>
<td>Extreme Harmonic</td>
<td>1.5817 Van15</td>
<td>&gt; 1</td>
<td>Ω((n))</td>
</tr>
</tbody>
</table>
Experimental Evaluation

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