Advice for Packing Items into Bins

Kayman Brusse

CSC2421: Topics in Algorithms

December 2019

Why Advice?

In the online setting, algorithms have **zero** knowledge about the incoming input.

Algorithms *with advice model* the scenario where some a priori knowledge is available, and can be leveraged during online computation.

Model

Tape Model: An omniscient oracle populates a tape with b bits of advice, which the algorithm can read from during execution.

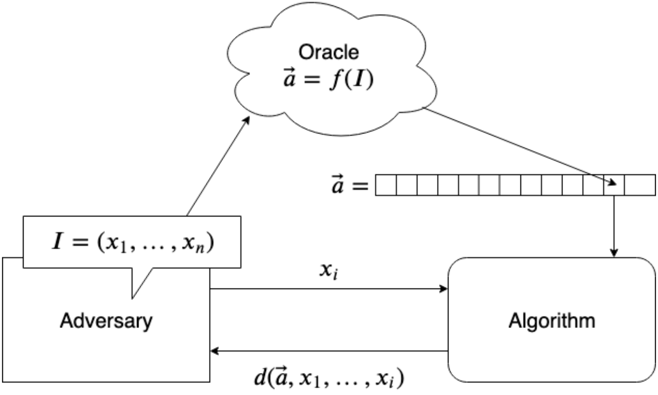

Figure 1: An algorithm making decisions using advice

Talk Plan

Introduce three problems in the online model:

- Bin Packing
- Bin Covering
- Dual Bin Packing

For each problem, we present a recent result that highlights the difference between these problems when advice is taken into account.

Bin Packing

Input: A sequence $\sigma = x_1, \ldots, x_n$ of n items, each of size $x_i \in (0, 1]$.

In this problem each bin has a capacity of 1. The objective is to place *every* item into a bin in such a way that **minimizes** the number of bins used.

Bin Covering

Input: A sequence $\sigma = x_1, \ldots, x_n$ of n items, each of size $x_i \in (0, 1]$.

Bins have unlimited (or 2) capacity, but we say a bin is *covered* if the total size of items inside is at least 1. The objective is to place the items into bins in order to **maximize** the number of bins covered.

Example

Take $\sigma = 0.9, 0.5, 0.4, x_4, x_5, x_6$.

- If $x_4 = 0.1$ and $x_5 = 0.5$, and $x_6 = 0.6$ then $OPT(\sigma) = 3$.
- If $x_4 = x_5 = x_6 = \epsilon$ then $OPT(\sigma) = 1$.

An online algorithm can't distinguish between these two cases!

Dual-Bin Packing

Input: A sequence $\sigma = x_1, \ldots, x_n$ of n items, each of size $x_i \in (0, 1]$, and m bins of capacity 1.

The objective is to place *some subset* of the items into bins in order to **maximize** the number of **items** packed. We denote the maximum bit size of any x_i by s.

Example

Take $m = 2$ and $\sigma = 0.9, 0.9, x_1, x_2$.

- If $x_1 = x_2 = 0.1$, then $OPT(\sigma) = 4$.
- If $x_1 = x_2 = 0.5$ then $OPT(\sigma) = 3$.

Results without Advice

All of these problems are well studied.

Problem	Upper Bound CR	Lower Bound CR
Bin Packing	1.5783	1.54278
Bin Covering	0.5	0.5
Dual Bin Packing[1]	-	∞

Table 1: Bounds on the competitive ratio without advice.

[1] More assumptions on the input need to be made to get a constant ratio.

Bin Packing with Constant Advice

Theorem 1 (Angelopoulos et al. 2018)

For any $k \geq 4$, there is an online algorithm for bin packing with k bits of advice and a competitive ratio of $1.5 + \frac{15}{2^{k/2+1}}$

In particular, when $k = 16$, the ratio is < 15.3, better than any strictly online algorithm.

Bin Packing Item Sizes

$$(0,1] = \underbrace{\underbrace{(0,\frac{1}{3}]}_{\textit{Tiny}} \cup \underbrace{(\frac{1}{3},\frac{1}{2}]}_{\textit{Mini}}}_{\textit{Small}} \cup \underbrace{\underbrace{(\frac{1}{2},\frac{2}{3}]}_{\textit{Critical}} \cup \underbrace{(\frac{2}{3},1]}_{\textit{Huge}}}_{\textit{Large}}$$

The ReserveCritical Algorithm

Use $\log n$ bits of advice to encode the number of critical items in σ, and reserve space.

1. **Huge:** Open a new bin and pack in separately.
2. **Critical:** Pack into a bin with reserved $\frac{2}{3}$ space.
3. **Mini:** Pack beside an previous mini item, or open a new bin.
4. **Tiny:** Pack FirstFit into non-reserved space, or pack into a dedicated tiny bin.

Lemma

ReserveCritical has a competitive ratio of 1.5.

The RedBlue Algorithm

Goals: Approximate ReserveCritical using $O(1)$ bits of advice

Let X and Y be the number of bins ReserveCritical opens for critical and tiny items (resp) on input σ.

The oracle for RedBlue encodes a value i using k bits of advice such that

$$\frac{i}{2^k} \leq \frac{X}{X+Y} < \frac{i+1}{2^k}$$

The RedBlue Algorithm cont'd

RedBlue packs huge and mini items the same as before:

1. **Huge:** Open a new bin and pack in separately.
2. **Mini:** Pack beside a previous mini item, or open a new bin.

Maintain a set of Blue bins with reserved $\frac{2}{3}$ space. Pack **critical** items in a blue bin, and open a new one if none exist.

Maintain a set of Red bins for **tiny** items and use FirstFit. If a new bin is required, label it Red or Blue depending on i.

The RedBlue Algorithm cont'd

Set $\beta = \frac{i}{2^k}$. When opening a new bin for a **tiny** item it is labeled as follows:

1. If $\beta > 1 - \frac{1}{2^{k/2}}$, label it blue.
2. If $\beta < \frac{1}{2^{k/2}}$, label it red.
3. If $\frac{1}{2^{k/2}} \leq \beta \leq 1 - \frac{1}{2^{k/2}}$, consider R_{i-1} and B_{i-1} the current number of red and blue bins. If

$$\beta \leq \frac{B_{i-1} + 1}{B_{i-1} + R_{i-1} + 1}$$

label it blue, otherwise red.

RedBlue Analysis

Lemma (Case 2)

If $\beta < \frac{1}{2^{k/2}}$ then the average level of all red and blue bins (excluding at most two red bins) is at least $\frac{3}{4}(1 - \frac{1}{2^{k-1}})$.

All bins with **huge** and **mini** items (apart from maybe the last) have value $\geq \frac{2}{3}$, we have

$$\frac{OPT(\sigma)}{\textsc{RedBlue}(\sigma)} < \frac{1}{\frac{2}{3}(1 - \frac{1}{2^{k-1}})} \leq 1.5 + \frac{3}{2^k - 2}$$

RedBlue Analysis cont'd

Let R and B be the number of red and blue bins in the final packing by RedBlue.

Lemma(Case 1)

If $\beta > 1 - \frac{1}{2^{k/2}}$, then $B \leq (1 + \frac{5}{2^{k/2}})(X + Y) + 1$, and $R = 0$.

Lemma(Case 3)

If $\frac{1}{2^{k/2}} \leq \beta \leq 1 - \frac{1}{2^{k/2}}$ then $B + R < (X + Y)(1 + \frac{2}{2^{k/2-2}}) + 2^{k/2}$

Proof of Theorem 1

In both cases, for some constants r and c (in terms of k):

$$R + B \leq r(X + Y) + c$$

Let H and M be the number of huge and mini items:

$$\begin{aligned}
\textsc{RedBlue}(\sigma) &\leq H + \lceil \frac{M}{2} \rceil + R + B \\
&\leq H + \lceil \frac{M}{2} \rceil + r(X + Y) + c \\
&\leq (r)\textsc{ReserveCritical}(\sigma) + c \\
&\leq (r)1.5\mathrm{OPT}(\sigma) + c' \\
&\leq (1.5 + \frac{15}{2^{k/2+1}})\mathrm{OPT}(\sigma) + c' \qquad \text{for } k \geq 4
\end{aligned}$$

Where the last inequality holds over *all* cases.

A Lower Bound for Bin Covering

Theorem 2 (Boyar et al. 2019)

There is no algorithm for bin covering with only $o(\log \log n)$ bits of advice that achieves a competitive ratio better than $\frac{1}{2}$.

Family of Inputs

Consider the family of input sequences $\{\sigma_j\}$ for $1 \leq j \leq n$

$$\sigma_j = \langle \underbrace{\epsilon, \ldots, \epsilon}_{n \text{ items}}, \underbrace{1 - j\epsilon, \ldots, 1 - j\epsilon}_{\frac{n}{j} \text{ items}} \rangle$$

Observe that $OPT(\sigma_j) = \frac{n}{j}$ by placing j copies of ϵ in each bin.

Proof of Theorem 2

Proceed by contradiction, and suppose A has competitive ratio $\frac{1}{2} + \mu$ and uses $o(\log \log n)$ bits of advice, where $\mu > 0$. It follows that:

$$A(\sigma_j) \geq (\frac{1}{2} + \mu) OPT(\sigma_j) - d = \frac{n}{2j} + \frac{\mu n}{j} - d$$

Proof of Theorem 2 cont'd

We say that two sequences are part of the same *sub-family* if they receive the same advice. There are $o(\log n)$ such sub-families.

If $S = \{\sigma_{\alpha_1}, \ldots, \sigma_{\alpha_k}\} \subseteq \{\sigma_j\}$ is a sub-family, then A places the first n items of σ_{α_i} the same way.

Let $m_i(S) = m_i$ denote the number of bins that get at least i items in such a placement of ϵ's, and observe that $\sum_{i=1}^{n} m_i = n$.

Proof of Theorem 2 cont'd

On input $\sigma_j \in S$, if a bin has j copies of ϵ in it, it can be covered in a single item. Hence:

$$A(\sigma_j) \leq m_j + \frac{(\frac{n}{j} - m_j)}{2} = \frac{n}{2j} + \frac{m_j}{2}$$

Since $A(\sigma_j) \geq \frac{n}{2j} + \frac{\mu n}{j} - d$ from earlier, we have

$$\mu \frac{n}{j} \leq \frac{m_j}{2} + d$$

Proof of Theorem 2 cont'd

If we sum over a sub-family, that is $j \in \{\alpha_1, \ldots, \alpha_k\}$, we have

$$\mu n(\frac{1}{\alpha_1} + \cdots + \frac{1}{\alpha_k}) \leq \frac{1}{2}(m_{\alpha_1} + \cdots + m_{\alpha_k}) + kd \leq (\frac{1}{2} + d)n$$

Since d is fixed, $\frac{1}{\alpha_1} + \cdots + \frac{1}{\alpha_k} \in O(1)$

Proof of Theorem 2 cont'd

Summing over all $o(\log n)$ sub-families each sequence is included exactly once, we have

$$\sum_{i=1}^{n} \frac{1}{i} = o(\log n) \times O(1) \in o(\log n)$$

However, $\sum_{i=1}^{n} \frac{1}{i} = H_n$ is well-known to be $\Theta(\log n)$, a contradiction.

$\square$

Dual Bin Packing and Input Bit Size

There exists and algorithm with $O(\frac{s+\log(n)}{\epsilon^2})$ bits of advice that achieves a $(1+\epsilon)$ competitive ratio.

Theorem 3 (Borodin, Pankratov, Salehi-Abari 2018)

An online algorithm for the dual bin packing problem with **unrestricted input bit size** that has competitive ratio $1+\epsilon$ requires $(1-O(\epsilon\log\epsilon))n = \Omega_\epsilon(n)$ bits of advice.

The Binary Separation Problem

Input: A sequence $\langle n_1, y_1, \ldots, y_n \rangle$ of $n = n_1 + n_2$ positive numbers, with n_1 being **"large"** and n_2 being **"small"**.

As each number y_i is revealed, the algorithm must guess if it belongs to the large or small group. The correct answer is revealed after the guess.

Theorem (Boyar et al. 2016)

Assume an algorithm for this problem where each y_i has n bits makes $r(n)$ mistakes using at most $b(n)$ bits of advice. Set $\alpha = (n - r(n))/n$. If $\alpha \in [\frac{1}{2}, 1)$, then $b(n) \geq (1 - O(\alpha \log \alpha))n$.

Proof by Reduction

Let ALG be an algorithm that solves the dual bin-packing problem with competitive ratio c and uses $b(n)$ bits of advice.

We will describe ALG' which solves the binary separation problem by running ALG on an instance with $2n$ numbers and n bins.

Proof of Theorem 3

Let $\delta_{\max} > \delta_{\min} > 0$ and let $f : \mathbb{R}_{\geq 0} \to (\delta_{\min}, \delta_{\max})$ be a strictly decreasing function.

ALG' processes input $I = \langle n_1, y_1, \ldots, y_n \rangle$ as follows:

1. The first n_1 items given to ALG are set to $\frac{1}{2} + \delta_{\min}$.
2. To decide how to guess y_i, send the item $\frac{1}{2} - f(y_i)$ to ALG. If ALG packs it in a bin with $\frac{1}{2} + \delta_{\min}$, then ALG' guesses y_i is large, and small otherwise.
3. After processing I, for each y_i that was revealed to be *truly* small, give a "complement" item of weight $\frac{1}{2} + f(y_i)$.

Proof of Theorem 3 cont'd

- p_1: number of items not packed in Phase 1).
- s_2: number of **small** items not packed in Phase 2).
- l_2: number of **large** items not packed in Phase 2).
- p_3: number of items not packed in Phase 3).

Observe OPT packs all $2n$ items. The number of items ALG does not pack is

$$p_1 + s_2 + l_2 + p_3 \leq 2n - \frac{2n}{c} \leq \frac{c-1}{c}2n$$

Proof of Theorem 3 cont'd

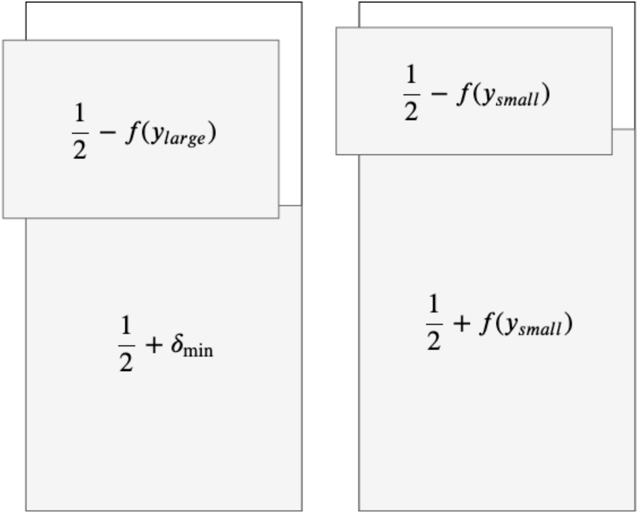

Figure 2: How ALG can packs items.

Proof of Theorem 3 cont'd

If ALG guesses that a **large** item is small:

- It is unpacked, or placed in one of $p_1 + p_3$ leftover bins.
- ALG guesses correct for large items at least

$$g_1 = (n_1 - p_1) - 2(p_1 + p_3) - l_2$$

If ALG guesses that a **small** item is large

- It places the small item with $\frac{1}{2} + \delta_{\text{min}}$ from Phase 1.
- ALG guesses correct for small items at least

$$g_2 = n_2 - s_2 - 2(p_1 + p_3) - l_2$$

Proof of Theorem 3 cont'd

Thus, ALG' makes correct guesses at least:

$$\begin{aligned} g_1 + g_2 &= n_1 + n_2 - s_2 - p_1 - 4(p_1 + p_3) - 2l_2 \\ &\geq n_1 + n_2 - 5(p_1 + p_3 + s_2 + l_2) \\ &\geq n - 10\frac{c-1}{c}n \end{aligned}$$

The ratio of good guesses is $\frac{10-9c}{c}$, which is greater than $\frac{1}{2}$ for $c < 1 + \frac{1}{19}$.

Proof of Theorem 3 cont'd

Observe

$$\frac{10 - 9(1+\epsilon)}{1+\epsilon} \in O(\epsilon)$$

If $\epsilon < \frac{1}{19}$ then by the theorem for Binary Seperation, in order for ALG to obtain a ratio of $1+\epsilon$, it requires

$$(1 - O(\epsilon \log \epsilon)n = \Omega_\epsilon(n)$$

bits of advice.

Remarks

- The last result implies a separation of advice complexity classes EAC and $WEAC$, by showing dependence on s is necessary.
- Efficiently computing advice has been left (mostly) untouched in this talk, but is still relevant.
- In general, improving bounds on competitive ratio with varying degrees of advice is still open for all of these problems.