Chapter 10

Bin Packing

Here we consider the classical BIN PACKING problem: We are given a set $I = \{1, \ldots, n\}$ of *items*, where item $i \in I$ has *size* $s_i \in (0, 1]$ and a set $B = \{1, \ldots, n\}$ of *bins* with *capacity* one. Find an assignment $a : I \to B$ such that the number of non-empty bins is minimal. As a shorthand, we write $s(J) = \sum_{j \in J} s_j$ for any $J \subseteq I$.

10.1 Hardness of Approximation

The BIN PACKING problem is NP-complete. More specifically:

Theorem 10.1. *It is* NP*-complete to decide if an instance of* BIN PACKING *admits a solution with two bins.*

Proof. We reduce from PARTITION, which we know is NP-complete. Recall that in the PARTITION problem, we are given n numbers $c_1, \ldots, c_n \in \mathbb{N}$ and are asked to decide if there is a set $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} c_i = \sum_{i \notin S} c_i$. Given a PARTITION instance, we create an instance for BIN PACKING by setting $s_i = 2c_i / (\sum_{j=1}^{n} c_j) \in (0, 1]$ for $i = 1, \ldots, n$. Obviously two bins suffice if and only if there is a $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} c_i = \sum_{i \notin S} c_i$. $\square$

This allows us to derive a lower bound on the approximabilty of BIN PACKING.

Corollary 10.2. *There is no ρ-approximation algorithm with $\rho < 3/2$ for* BIN PACKING *unless* $\mathsf{P} = \mathsf{NP}$.

10.2 Heuristics

We will show that there are constant factor approximations for BIN PACKING. Firstly we consider the probably most simple NEXT FIT algorithm, which can be shown to be 2-approximate. Secondly, we give the FIRST FIT DECREASING algorithm and show that it is 3/2-approximate. Thus, with the above hardness result, this is best-possible, unless $\mathsf{P} = \mathsf{NP}$.

Next Fit

The NEXT FIT algorithm works as follows: Initially all bins are empty and we start with bin $j = 1$ and item $i = 1$. If bin j has residual capacity for item i, assign item i to bin j, i.e., $a(i) = j$, and consider item $i + 1$. Otherwise consider bin $j + 1$ and item i. Repeat until item n is assigned.

Theorem 10.3. Next Fit *is a* 2*-approximation for* Bin Packing*. The algorithm runs in* $O(n)$ *time.*

Proof. Let k be the number of non-empty bins in the assignment a found by Next Fit. Let k^* be the optimal number of bins. We show the slightly stronger statement that

$$k \leq 2 \cdot k^* - 1.$$

Firstly we observe the lower bound $k^* \geq \lceil s(I) \rceil$. Secondly, for bins $j = 1, \ldots, \lfloor k/2 \rfloor$ we have

$$\sum_{i:a(i) \in \{2j-1, 2j\}} s_i > 1.$$

Adding these inequalities we get

$$\left\lfloor \frac{k}{2} \right\rfloor < s(I).$$

Since the left hand side is an integer we have that

$$\frac{k-1}{2} \leq \left\lfloor \frac{k}{2} \right\rfloor \leq \lceil s(I) \rceil - 1.$$

This proves $k \leq 2 \cdot \lceil s(I) \rceil - 1 \leq 2 \cdot k^* - 1$ and hence the claim. □

The analysis is tight for the algorithm, which can be seen with the following instance with $2n$ items. For some $\varepsilon > 0$ let $s_{2i-1} = 2 \cdot \varepsilon$, $s_{2i} = 1 - \varepsilon$ for $i = 1, \ldots, n$.

First Fit Decreasing

The algorithm Next Fit never considers bins again that have been left behind. Thus the wasted capacity therein leaves room for improvement. A natural way is First Fit: Initially all bins are empty and we start with current number of bins $k = 0$ and item $i = 1$. Consider all bins $j = 1, \ldots, k$ and place item i in the first bin that has sufficient residual capacity, i.e., $a(i) = j$. If there is no such bin increment k and repeat until item n is assigned. One can prove that First Fit uses at most $k \leq \lceil 17/10 \cdot k^* \rceil$ many bins, where k^* is the optimal number.

There is a further natural heuristic improvement of First Fit, called First Fit Decreasing: Reorder the items such that $s_1 \geq \cdots \geq s_n$ and apply First Fit. The intuition behind considering large items first is the following: "Large" items do not fit into the same bin anyway, so we already use unavoidable bins and try to place "small" items into the residual space.

Theorem 10.4. First Fit Decreasing *is a* $3/2$*-approximation for* Bin Packing*. The algorithm runs in* $O(n^2)$ *time.*

Proof. Let k be the number of non-empty bins of the assignment a found by First Fit Decreasing and let k^* be the optimal number.

Consider bin number $j = \lceil 2/3k \rceil$. If it contains an item i with $s_i > 1/2$, then each bin $j' < j$ did not have space for item i. Thus j' was assigned an item i' with $i' < i$. As the items are considered in non-increasing order of size we have $s_{i'} \geq s_i > 1/2$. That is, there are at least j items of size larger than $1/2$. These items need to be placed in individual bins. This implies

$$k^* \geq j \geq \frac{2}{3}k.$$

Otherwise, bin j and any bin $j' > j$ does not contain an item with size larger than 1/2. Hence the bins $j, j+1, \ldots, k$ contain at least $2(k-j)+1$ items, none of which fits into the bins $1, \ldots, j-1$. Thus we have

$$\begin{aligned} s(I) &> \min\{j-1, 2(k-j)+1\} \\ &\geq \min\{\lceil 2/3k \rceil - 1, 2(k-(2/3k+2/3))+1\} \\ &= \lceil 2/3k \rceil - 1 \end{aligned}$$

and $k^* \geq s(I) > \lceil 2/3k \rceil - 1$. This even implies

$$k^* \geq \left\lceil \frac{2}{3}k \right\rceil \geq \frac{2}{3}k$$

and hence the claim. □

10.3 Asymptotic Polynomial Time Approximation Scheme

With the hardness result that there is no approximation algorithm for Bin Packing with guarantee better than 3/2, unless $\mathsf{P} = \mathsf{NP}$, we do not have to search for a PTAS (or even an FPTAS). However, notice that the reduction used that the optimal number of bins is "small", such as 2 or 3. It is plausible that, in "practical" instances, the optimal number k^* of bins grows as the number of items grows. Maybe we can do better for those instances.

This leads us to define: An *asymptotic polynomial time approximation scheme* (APTAS) is a familiy of algorithms, such that for any $\varepsilon > 0$ there is a number k' and a $(1+\varepsilon)$-approximation algorithm, whenever $k^* \geq k'$. For Bin Packing such a family exists. However, the involved running times are rather high, even though polynomial in n.

Theorem 10.5. *For any $0 < \varepsilon \leq 1/2$ there is an algorithm that runs in time polynomial in n and finds an assignment having at most $k \leq (1+\varepsilon) \cdot k^* + 1$ many bins.*

Lemma 10.6. *Let $\varepsilon > 0$ and $d \in \mathbb{N}$ be constants. For any instance of* Bin Packing *where $s_i \geq \varepsilon$ and $|\{s_1, \ldots, s_n\}| \leq d$, there is a polynomial time algorithm that solves it optimally.*

Proof. The number of items in a bin is bounded by $m := \lfloor 1/\varepsilon \rfloor$. Therefore, the number of different assignments for one bin is bounded by $r = \binom{m+d}{m}$, which is a (large) constant. There are at most n bins used and therefore, the number of feasible assignments is bounded by $p = \binom{n+r}{r}$. This is a polynomial in n. Thus we can enumerate all assignments and choose the best one to give an optimum solution. □

Lemma 10.7. *Let $\varepsilon > 0$ be a constant. For any instance of* Bin Packing *where $s_i \geq \varepsilon$, there is a $(1+\varepsilon)$-approximation algorithm.*

Proof. Let I be the given instance. Sort the n items by increasing size and partition them into $g = \lceil 1/\varepsilon^2 \rceil$ many groups each having at most $q = \lfloor n\varepsilon^2 \rfloor$ many items. Notice that two groups may contain items of the same size.

Construct an instance J by rounding up the size of each item to the size of the largest item in its group. Instance J has at most g many different item sizes. Therefore, we can find an optimal assignment for J by invoking Lemma 10.6. This is clearly a feasible assignment for the original item sizes.

Now we show that $k^*(J) \leq (1+\varepsilon)k^*(I)$: We construct another instance J' by rounding down the size of each item to the smallest item size in its group. Clearly $k^*(J') \leq k^*(J)$.

The crucial observation is that an assignment for instance J' yields an assignment for all but the largest q items of the instance J. Therefore

$$k^*(J) \leq k^*(J') + q \leq k^*(I) + q.$$

To finalize the proof, since each item has size at least ε, we have $k^*(I) \geq n \cdot \varepsilon$ and $q = \lfloor n\varepsilon^2 \rfloor \leq \varepsilon \cdot k^*(I)$. Hence

$$k^*(J) \leq (1+\varepsilon) \cdot k^*(I)$$

and the claim is established. □

Proof of Theorem 10.5. Let I denote the given instance and I' the instance after discarding the items with size less than ε from I. We can invoke Lemma 10.7 and find an assignment which uses at most $k(I') \leq (1+\varepsilon) \cdot k^*(I')$ many bins. By using First Fit, we assign the items with sizes less than ε into the solution found for instance I'. We use additional bins if an item does not fit into any of the bins used so far.

If no additional bins are needed, then our assignment uses $k(I) \leq (1+\varepsilon) \cdot k^*(I') \leq (1+\varepsilon) \cdot k^*(I)$ many bins. Otherwise, all but the last bin have residual capacity less than ε. Thus $s(I) \geq (k(I)-1)(1-\varepsilon)$, which is a lower bound for $k^*(I)$. Thus we have

$$k(I) \leq \frac{k^*(I)}{1-\varepsilon} + 1 \leq (1+2\varepsilon) \cdot k^*(I) + 1,$$

where we have used $0 < \varepsilon \leq 1/2$. □