More General

Let $\mathrm{OPT}(n_1,\dots,n_A)$ be the number of machines that are required to schedule input vector $(n_1,\dots,n_A)$ with Makespan at most T (A: number of different sizes).

If $\mathrm{OPT}(n_1,\dots,n_A) \le m$ we can schedule the input.

$$\mathrm{OPT}(n_1,\dots,n_A) = \begin{cases} 0 & (n_1,\dots,n_A) = 0 \\ 1 + \min\limits_{(s_1,\dots,s_A)\in C} \mathrm{OPT}(n_1 - s_1,\dots,n_A - s_A) & (n_1,\dots,n_A) \gneq 0 \\ \infty & \text{otw.} \end{cases}$$

where C is the set of all configurations.

$|C| \le (B+1)^A$, where B is the number of jobs that possibly can fit on the same machine.

The running time is then $O((B+1)^A n^A)$ because the dynamic programming table has just n^A entries.

Bin Packing

Given n items with sizes $s_1,\dots,s_n$ where

$$1 > s_1 \ge \cdots \ge s_n > 0 \ .$$

Pack items into a minimum number of bins where each bin can hold items of total size at most 1.

Theorem 5
There is no ρ-approximation for Bin Packing with $\rho < 3/2$ unless $\mathrm{P} = \mathrm{NP}$.

Bin Packing

Proof

- In the partition problem we are given positive integers $b_1,\dots,b_n$ with $B = \sum_i b_i$ even. Can we partition the integers into two sets S and T s.t.

$$\sum_{i\in S} b_i = \sum_{i\in T} b_i \quad ?$$

- We can solve this problem by setting $s_i := 2b_i/B$ and asking whether we can pack the resulting items into 2 bins or not.
- A ρ-approximation algorithm with $\rho < 3/2$ cannot output 3 or more bins when 2 are optimal.
- Hence, such an algorithm can solve Partition.

Bin Packing

Definition 6
An asymptotic polynomial-time approximation scheme (APTAS) is a family of algorithms $\{A_\epsilon\}$ along with a constant c such that A_ϵ returns a solution of value at most $(1+\epsilon)\mathrm{OPT} + c$ for minimization problems.

- Note that for Set Cover or for Knapsack it makes no sense to differentiate between the notion of a PTAS or an APTAS because of scaling.
- However, we will develop an APTAS for Bin Packing.

Bin Packing

Again we can differentiate between small and large items.

Lemma 7

Any packing of items of size at most γ into ℓ bins can be extended to a packing of all items into $\max\{\ell, \frac{1}{1-\gamma}\text{SIZE}(I) + 1\}$ bins, where $\text{SIZE}(I) = \sum_i s_i$ is the sum of all item sizes.

- If after Greedy we use more than ℓ bins, all bins (apart from the last) must be full to at least $1 - \gamma$.
- Hence, $r(1-\gamma) \leq \text{SIZE}(I)$ where r is the number of nearly-full bins.
- This gives the lemma.

Choose $\gamma = \epsilon/2$. Then we either use ℓ bins or at most

$$\frac{1}{1-\epsilon/2} \cdot \text{OPT} + 1 \leq (1+\epsilon) \cdot \text{OPT} + 1$$

bins.

It remains to find an algorithm for the large items.

Bin Packing

Linear Grouping:

Generate an instance I' (for large items) as follows.

- Order large items according to size.
- Let the first k items belong to group 1; the following k items belong to group 2; etc.
- Delete items in the first group;
- Round items in the remaining groups to the size of the largest item in the group.

Lemma 8

$\text{OPT}(I') \leq \text{OPT}(I) \leq \text{OPT}(I') + k$

Proof 1:

- Any bin packing for I gives a bin packing for I' as follows.
- Pack the items of group 2, where in the packing for I the items for group 1 have been packed;
- Pack the items of groups 3, where in the packing for I the items for group 2 have been packed;
- ...

Lemma 9

$\mathrm{OPT}(I') \le \mathrm{OPT}(I) \le \mathrm{OPT}(I') + k$

Proof 2:

- Any bin packing for I' gives a bin packing for I as follows.
- Pack the items of group 1 into k new bins;
- Pack the items of groups 2, where in the packing for I' the items for group 2 have been packed;
- ...

Assume that our instance does not contain pieces smaller than $\epsilon/2$. Then $\mathrm{SIZE}(I) \ge \epsilon n/2$.

We set $k = \lfloor \epsilon \mathrm{SIZE}(I) \rfloor$.

Then $n/k \le 2n/\lfloor \epsilon^2 n/2 \rfloor \le 4/\epsilon^2$ (here we used $\lfloor \alpha \rfloor \ge \alpha/2$ for $\alpha \ge 1$).

Hence, after grouping we have a constant number of piece sizes $(4/\epsilon^2)$ and at most a constant number $(2/\epsilon)$ can fit into any bin.

We can find an optimal packing for such instances by the previous Dynamic Programming approach.

- cost (for large items) at most

$$\mathrm{OPT}(I') + k \le \mathrm{OPT}(I) + \epsilon \mathrm{SIZE}(I) \le (1+\epsilon)\mathrm{OPT}(I)$$

- running time $\mathcal{O}((\frac{2}{\epsilon} n)^{4/\epsilon^2})$.

Can we do better?

In the following we show how to obtain a solution where the number of bins is only

$$\mathrm{OPT}(I) + \mathcal{O}(\log^2(\mathrm{SIZE}(I))) \ .$$

Note that this is usually better than a guarantee of

$$(1+\epsilon)\mathrm{OPT}(I) + 1 \ .$$

Configuration LP

Change of Notation:

- Group pieces of identical size.
- Let s_1 denote the largest size, and let b_1 denote the number of pieces of size s_1.
- s_2 is second largest size and b_2 number of pieces of size s_2;
- ...
- s_m smallest size and b_m number of pieces of size s_m.