Configuration Integer Program for Bin Packing

- let $s_1 > s_2 > \cdots > s_m$ denote the different item sizes;
- for $i = 1, \ldots, m$ let b_i denote the number of items of size s_i;
- an m-tuple $(t_1, \ldots, t_m) \in \mathbb{Z}^m_{\geq 0}$ is a configuration if $\sum_{i=1}^m t_i \cdot s_i \leq 1$;
- let $T_1, \ldots, T_N$ be a complete enumeration of all configurations and $T_j = (t_{1j}, \ldots, t_{mj})$;
- for $j = 1, \ldots, N$ the integer variable x_j denotes the number of bins that shall be packed according to configuration T_j:

$$\begin{aligned} \min \quad & \sum_{j=1}^{N} x_j & \\ \text{s.t.} \quad & \sum_{j=1}^{N} t_{ij} \cdot x_j \geq b_i & \text{for all } i = 1, \ldots, m, \\ & x_j \in \mathbb{Z}_{\geq 0} & \text{for all } j = 1, \ldots, N. \end{aligned}$$

Configuration LP and its Dual

Primal:

$$\begin{aligned} \min \quad & \sum_{j=1}^{N} x_j & \\ \text{s.t.} \quad & \sum_{j=1}^{N} t_{ij} \cdot x_j \geq b_i & \text{for all } i = 1, \ldots, m, \\ & x_j \geq 0 & \text{for all } j = 1, \ldots, N. \end{aligned}$$

Dual:

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} b_i \cdot y_i & \\ \text{s.t.} \quad & \sum_{i=1}^{m} t_{ij} \cdot y_i \leq 1 & \text{for all } j = 1, \ldots, N, \\ & y_i \geq 0 & \text{for all } i = 1, \ldots, m. \end{aligned}$$

Notice: $\text{SIZE}(I) \leq \text{OPT}_{LP}(I) \leq \text{OPT}(I)$

Solving the Configuration LP Approximately

- Configuration LP suffers from exponentially many variables.
- Dual separation problem is Knapsack Problem and thus *NP*-hard.
- Remember: optimization and separation are equally difficult.
- Therefore, it is *NP*-hard to solve the Configuration LP to optimality.

Theorem 4.14.

An LP solution of value at most $\mathrm{OPT}_{LP}(I) + 1$ can be computed in polynomial time.

Proof:... □

Harmonic Grouping Scheme and Rounding

Grouping

- consider items in order of non-increasing size;
- open a group and start putting items in current group, one at a time;
- close current group if its total size is at least 2 and start new group;

Let $r :=$ number of groups; let G_i denote ith group; $n_i := |G_i|$.

Notice that $n_i \geq n_{i-1}$ and $r \leq \lceil \mathrm{SIZE}(I)/2 \rceil$.

Rounding: Construct new instance I' as follows:

- discard items in G_1 and G_r;
- for $i = 2, \ldots, r-1$ discard the $n_i - n_{i-1}$ smallest items in G_i;
- for $i = 2, \ldots, r-1$ round sizes of remaining items in G_i to largest one.

Lemma 4.15.

There are at most $\mathrm{SIZE}(I)/2$ distinct item sizes in I'; the total size of all discarded items is $O(\log \mathrm{SIZE}(I))$. □

Recursive Bin Packing Algorithm

BinPack(I)

1. if SIZE(I) < 10 then pack remaining items using First-Fit and stop;
2. apply harmonic grouping scheme to create instance I';
3. pack discarded items in $O(\log \text{SIZE}(I))$ bins using First-Fit;
4. compute near-optimal solution x to Configuration LP for instance I';
5. for $j = 1, \ldots, N$ pack $\lfloor x_j \rfloor$ bins in configuration T_j;
6. call the packed items instance I_1 and the remaining items I_2;
7. pack I_2 via BinPack(I_2);

Analysis of Algorithm BinPack

Lemma 4.16.

$\text{OPT}_{LP}(I_1) + \text{OPT}_{LP}(I_2) \leq \text{OPT}_{LP}(I') \leq \text{OPT}_{LP}(I)$.

Proof:. . . □

Theorem 4.17.

Algorithm BinPack runs in polynomial time and finds a solution using at most $\text{OPT}(I) + O\big(\log^2 \text{OPT}(I)\big)$ bins.

Proof:. . . □