Original notes by Chung-Yeung Lee and Gisli Hjaltason.

12 Bin Packing

Problem Statement: Given n items $s_1, s_2, ..., s_n$, where each s_i is a rational number, $0 < s_i \leq 1$, our goal is to minimize the number of bins of size 1 such that all the items can be packed into them.

Remarks:

1. It is known that the problem is NP-Hard.
2. A Simple Greedy Approach (First-Fit) can yield an approximation algorithm with a performance ratio of 2.

12.1 Approximate Schemes for bin-packing problems

In the 1980's, two approximate schemes were proposed. They are

1. (Fernandez de la Vega and Lueker) $\forall \epsilon > 0$, there exists an Algorithm A_ϵ such that

$$A_\epsilon(I) \leq (1+\epsilon)OPT(I) + 1$$

   where A_ϵ runs in time polynomial in n, but exponential in $1/\epsilon$ (n=total number of items).

2. (Karmarkar and Karp) $\forall \epsilon > 0$, there exists an Algorithm A_ϵ such that

$$A_\epsilon(I) \leq OPT(I) + O(\log^2(OPT(I))$$

   where A_ϵ runs in time polynomial in n and $1/\epsilon$ (n=total number of items). They also guarantee that $A_\epsilon(I) \leq (1+\epsilon)OPT(I) + O(\frac{1}{\epsilon^2})$.

We shall now discuss the proof of the first result. Roughly speaking, it relies on two ideas:

- Small items does not create a problem.
- Grouping together items of similar sizes can simplify the problem.

12.1.1 Restricted Bin Packing

We consider the following restricted version of the bin packing problem (RBP). We require that

1. Each item has size $\geq \delta$.
2. The size of each item takes only one of m distinct values $v_1, v_2, ..., v_m$. That is we have n_i items of size v_i $(1 \leq i \leq m)$, with $\sum_{i=1}^{m} n_i = n$.

For constant δ and m, the above can be solved in polynomial time (actually in $O(n + f(m, \delta))$). Our overall strategy is therefore to reduce BP to RBP (by throwing away items of size $< \delta$ and grouping items carefully), solve it optimally and use $RBP(\delta, m)$ to compute a soluton to the original BP.

Theorem 12.1 *Let J be the instance of BP obtained from throwing away the items of size less than δ from instance I. If J requires β bins then I needs only $max(\beta, (1+2\delta)OPT(I)+1)$ bins.*

Proof:

We observe that from the solution of J, we can add the items of size less than δ to the bins until the empty space is less than δ. Let S be the total size of the items, then we may assume the number of items with size $< \delta$ is large enough (otherwise I needs only β bins) so that we use β' bins.

$$S \geq (1-\delta)(\beta'-1)$$

$$\beta' \leq 1 + \frac{S}{1-\delta}$$

$$\beta' \leq 1 + \frac{OPT(I)}{1-\delta}$$

$$\beta' \leq 1 + (1+2\delta)OPT(I)$$

as $(1-\delta)^{-1} \leq 1 + 2\delta$ for $0 < \delta < \frac{1}{2}$. □

Next, we shall introduce the grouping scheme for RBP. Assume that the items are sorted in descending order. Let n' be the total number of items. Define G_1=the group of the largest k items, G_2=the group that contains the next k items, and so on. We choose

$$k = \lfloor \frac{\epsilon^2 n'}{2} \rfloor.$$

Then, we have m+1 groups $G_1, .., G_{m+1}$, where

$$m = \lfloor \frac{n'}{k} \rfloor.$$

Further, we consider groups H_i = group obtained from G_i by setting all items sizes in G_i equal to the largest one in G_i. Note that

- size of any item in $H_i \geq$ size of any items in G_i.
- size of any item in $G_i \geq$ size of any items in H_{i+1}.

The following diagram (Fig 1) illustrates the ideas:

We then define J_{low} be the instance consisting of items in $H_2, .., H_{m+1}$. Our goal is to show

$$\text{OPT}(J_{\text{low}}) \leq \text{OPT}(J) \leq \text{OPT}(J_{\text{low}}) + k,$$

The first inequality is trivial, since from OPT(J) we can always get a solution for J_{low}.

Using OPT(J_{low}) we can pack all the items in $G_2, \ldots, G_{m+1}$ (since we over allocated space for these by converting them to H_i). In particular, group G_1, the group left out in J_{low}, contains k items, so that no more than k extra bins are needed to accommodate those items.

Since $(H_1 \cup J_{\text{low}})$ is an instance of a Restricted Bin Packing Problem we can solve it optimally, and then replace the items in H_i with items in G_i to get a solution for J.

Directly from this inequality, and using the definition of k, we have

$$Soln(J) \leq OPT(H_1 \cup \ldots \cup H_{m+1}) \leq \text{OPT}(J_{\text{low}}) + k \leq \text{OPT}(J) + k \leq \text{OPT}(J) + \frac{\epsilon^2 n'}{2}.$$

Choosing $\delta = \epsilon/2$, we get that

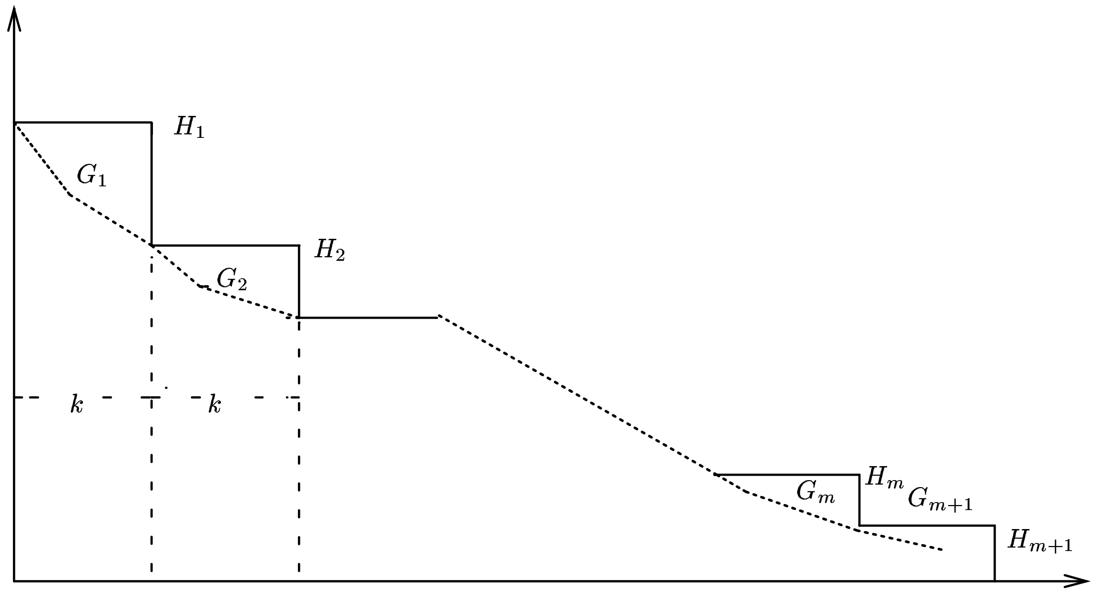
$$\text{OPT}(J) \geq \sum_{i=1}^{n'} s_i \geq n'\frac{\epsilon}{2},$$

so we have

$$Soln(J) \leq \text{OPT}(J) + \frac{\epsilon^2 n'}{2} \leq \text{OPT}(J) + \epsilon\text{OPT}(J) = (1+\epsilon)\text{OPT}(J).$$

By applying Theorem 12.1, using $\beta \leq (1+\epsilon)\text{OPT}(J)$ and the fact that $2\delta = \epsilon$, we know that the number of bins needed for the items of I is at most

$$\max\{(1+\epsilon)\text{OPT}(J), (1+\epsilon)\text{OPT}(I) + 1\} \leq (1+\epsilon)\text{OPT}(I) + 1.$$



Grouping Scheme for RBP

Figure 2: Grouping scheme

The last inequality follows since $OPT(J) \leq OPT(I)$.

We will turn to the problem of finding an optimal solution to RBP. Recall that an instance of RBP(δ, m) has items of sizes $v_1, v_2, \ldots, v_m$, with $1 \geq v_1 \geq v_2 \geq \cdots \geq v_m \geq \delta$, where n_i items have size v_i, $1 \leq i \leq m$. Summing up the n_i's gives the total number of items, n. A bin is completely described by a vector $(T_1, T_2, \ldots, T_m)$, where T_i is the number of items of size v_i in the bin. How many different different (valid) bin types are there? From the bin size restriction of 1 and the fact that $v_i \geq \delta$ we get

$$1 \geq \sum_i T_i v_i \geq \sum_i T_i \delta = \delta \sum_i T_i \Rightarrow \sum_i T_i \leq \frac{1}{\delta}.$$

As $\frac{1}{\delta}$ is a constant, we see that the number of bin types is constant, say p.

Let $T^{(1)}, T^{(2)}, \ldots, T^{(p)}$ be an enumeration of the p (valid) different bin types. A solution to the RBP can now be stated as having x_i bins of type $T^{(i)}$. Let $T_j^{(i)}$ denote the number of items of size v_j in $T^{(i)}$. The problem of finding the optimal solution can be posed as an integer linear programming problem:

$$\min \sum_{i=1}^{p} x_i,$$

such that

$$\sum_{i=1}^{p} x_i T_j^{(i)} = n_j \quad \forall j = 1, \ldots, m.$$

$$x_i \geq 0, x_i \text{ integer} \quad \forall i = 1, \ldots, p.$$

This is a constant size problem, since both p and m are constants, independent of n, so it can be solved in time independent of n. This result is captured in the following theorem, where $f(\delta, m)$ is a constant that depends only on δ and m.

Theorem 12.2 *An instance of RBP(δ, m) can be solved in time $O(n, f(\delta, m))$.*

An approximation scheme for BP may be based on this method. An algorithm A_ϵ for solving an instance I of BP would proceed as follows:

Step 1: Get an instance J of $\mathrm{BP}(\delta, n')$ by getting rid of all elements in I smaller than $\delta = \epsilon/2$.

Step 2: Obtain H_i from J, using the parameters k and m established earlier.

Step 3: Find an optimal packing for $H_1 \cup \ldots \cup H_{m+1}$ by solving the corresponding integer linear programming problem.

Step 4: Pack the items of G_i in place of the corresponding (larger) items of H_i as packed in step 3.

Step 5: Pack the small items in $I \setminus J$ using First-Fit.

This algorithm finds a packing for I using at most $(1+\epsilon)\mathrm{OPT}(I) + 1$ bins. All steps are at most linear in n, except step 2, which is $O(n \log n)$, as it basically amounts to sorting J. The fact that step 3 is linear in n was established in the previous algorithm, but note that while $f(\delta, m)$ is independent of n, it is exponential in $\frac{1}{\delta}$ and m and thus $\frac{1}{\epsilon}$. Therefore, this approximation scheme is polynomial but not fully polynomial. (An approximation scheme is fully polynomial if the running time is a polynomial in n and $\frac{1}{\epsilon}$.