# 应用运筹学基础：组合优化 (5) - 近似算法选讲 (3)

这节课首先介绍了 bin packing 的另一种线性规划模型：configuration LP，之后提出了一种渐进比为 1 的 bin packing 近似算法。

## Configuration LP

我们将 $n$ 个物品按体积分类，设共有 $k$ 种不同的体积，记为 $c_1, c_2, \ldots, c_k$，每种体积有物品 $b_1, b_2, \ldots, b_k$ 个。

对于一个 bin，我们枚举体积为 $c_1$ 的物品放几个，$c_2$ 的放几个，...，$c_k$ 的放几个。

假设一共有 $N$ 种不同的方案（又称为 pattern），记 $t_{i,j}$ 表示第 $i$ 种方案中，体积为 $c_j$ 的物品放了几个。我们设 $x_i$ 表示对一个 bin packing 问题得到的答案中，第 $i$ 种方案的 bin 用了几个。

那么，我们能写出 bin packing 的线性规划问题：

$$\begin{aligned} \min_x \quad & \sum_{i=1}^N x_i \\ \text{s.t.} \quad & \sum_{i=1}^N t_{i,j} x_i \geq b_j \quad \forall j \in \{1, 2, \ldots, k\} \\ & x \geq 0 \end{aligned}$$

这个线性规划看起来变量很多，但是可以通过某种神奇的方法在比较好的多项式时间内求解。

## 近似算法

下面将提出一种渐进比为 1 的近似算法。记 $N(I)$ 表示该算法对 bin packing 的实例 $I$ 算出来的答案，我们将要证明 $N(I) \leq \mathrm{OPT}_{LP}(I) + O(\log^2 \mathrm{OPT}_{LP}(I))$。算法步骤如下：

\1. 记 $c(I)$ 表示实例 $I$ 中物品的体积总和，先将体积小于 $\frac{1}{c(I)}$ 的物品拿走。等其它物品用迭代算法分配完后，再把小的物品用 first fit 安排进去。

\2. 使用迭代算法分配剩下的物品。

为什么最后分配“小”的物品不影响结论呢？下面先证明一个引理。

**引理**：设 $I$ 为 bin packing 问题的实例，$g$ 是一个 0~1 之间的实数。我们称体积至少为 $\frac{g}{2}$ 的物品为“大”物品，其它物品是“小”物品。我们首先分配大物品，设一共用了 $A$ 个 bin。此时再用 first fit 把小物品也分配进去，则总共使用的 bin 数至多为 $\max(A, (1+g)c(I)+1)$。

**证明**：设总共使用的 bin 数量为 $B$。如果最后的 first fit 没有再开新的 bin，那么 $B=A$；否则至多有一个 bin 体积小于 $1-\frac{g}{2}$，那么我们可以推出 $c(I) \ge (1-\frac{g}{2})(B-1)$，再结合 $0 \le g \le 1$，就有 $B \le \frac{2}{2-g}c(I)+1 \le (1+g)c(I)+1$。

回到我们的算法，算法的第 1 步其实就是让 $\frac{1}{c(I)} = \frac{g}{2}$，即 $c(I) = \frac{2}{g}$。只要有 $c(I) \ge 2$（$c(I) < 2$ 用 first fit 就行了，反正要证的是渐进比嘛...），那么用 first fit 分配小物品后，使用的总 bin 数就是 $(1+\frac{2}{c(I)})c(I)+1 = c(I)+3$（然后和 $A$ 取个 max），不改变我们渐进比为 1 的结论。下面我们只要证明 $A$ 也符合结论就行了。

## 迭代算法与证明

下面说明要用到的迭代算法，并证明算法的结果仍然符合渐进比为 1 的结论。每一次迭代考虑当前的实例 $I$，进行以下步骤：

### 第一步

求出 configuration LP 的解。设第 $i$ 个 pattern 用了 $x_i$ 次，由于 LP 只有 $k$ 个限制，那么非零量至多只有 $k$ 个（所以不用担心因为非零量太多直接变成指数级算法）。

### 第二步

把实例 $I$ 拆成两部分，$I_1$ 包含了解的整数部分（即 $\lfloor x \rfloor$），$I_2$ 包含了解的小数部分（即 $x - \lfloor x \rfloor$）。由于我们使用的 pattern 种数至多为 $k$，那么小数部分物品的体积总和不超过 $k$。

很容易发现，$I_1$ 的最优解就是恰好放满 $\mathrm{OPT}_{LP}(I_1)$ 个 bin，那么容易有 $\mathrm{OPT}_{LP}(I_1) + \mathrm{OPT}_{LP}(I_2) = \mathrm{OPT}_{LP}(I)$。

举个例子：

假如有 2 种物品 $c_1, c_2$ 和 2 种 pattern。

第 1 个 pattern 有 3 个 $c_1$ 与 1 个 $c_2$；

第 2 个 pattern 有 1 个 $c_1$ 与 3 个 $c_2$。

Configuration LP 的解为 2.5 个 pattern 1 与 1.5 个 pattern 2。

那么我们把 $I$ 拆成两个部分，

$I_1$ 里有 $3 \times 2 + 1 \times 1 = 7$ 个 $c_1$ 和 $1 \times 2 + 3 \times 1 = 5$ 个 $c_2$，

$I_2$ 里有 $3 \times 0.5 + 1 \times 0.5 = 2$ 个 $c_1$ 和 $1 \times 0.5 + 3 \times 0.5 = 2$ 个 $c_2$。

## 第三步

把 $I_2$ 中的所有物品按体积总大到小排序，设为 $d_1 \geq d_2 \geq \ldots$。

我们把这些物品分堆：首先找到最小的 $n_1$，把 $d_1$ 到 $d_{n_1}$ 分成第一堆，且堆内物品体积总和至少为 2；再找到最小的 $n_2$，把 $d_{n_1+1}$ 到 $d_{n_2}$ 分成第二堆，且堆内体积总和至少为 2；...这样分成 $p$ 堆，显然只有最后一堆的体积总和可能小于 2。

由于每个物品的体积都至多为 1，每一堆的体积总和肯定小于 3。还容易发现，由于物品是按体积从大到小排序的，有 $n_1 \leq n_2 \leq \cdots \leq n_{p-1}$。

接下来，我们去掉第一堆和第 $p$ 堆。在第 2 到 $p-1$ 堆中，对于第 $i$ 堆，我们只留下最大的 $n_{i-1}$ 个物品（去掉剩下的 $n_i - n_{i-1}$ 个物品），并且把这些物品的体积都放大到第 $i$ 堆里最大的体积。将我们留下来的物品构成实例 $I'$。

不难注意到，如果我们把 $I'$ 也进行分堆，那么 $I'$ 里第一堆的物品数和 $I_2$ 里第一堆的物品数相同，但体积都没有 $I_2$ 第一堆的大，其它堆也是如此。所以我们有 $\mathrm{OPT}(I') \leq \mathrm{OPT}(I_2)$。我们只要把 $I'$ 作为新一轮迭代的实例进行迭代即可。

不过，我们要迭代多少轮呢？注意到 $I_2$ 分堆后，每一堆的体积至少为 2，那么 $I'$ 中体积不同的物品种数至多为 $\frac{c(I_2)}{2}$。别忘了我们在第二步中发现的结论：小数部分物品的体积总和不超过体积不同的物品种数。所以每一轮 $I_2$ 的体积都会折半，那么 $\log c(I)$ 轮之后迭代就会结束。

最后我们再来看看被我们去掉的物品总体积是多少。

首先，我们去掉了第一堆和第 $p$ 堆，这两堆的体积之和至多为 6。

再来看第 2 堆到第 $p-1$ 堆。对于第 $i$ 堆，我们去掉了 $n_i - n_{i-1}$ 个物品，它们的体积均值不会超过 $\frac{3}{n_i}$（只考虑小的值，均值会变小）。我们来求个和

$$\begin{aligned} & 6+\sum_{i=2}^{p-1}(n_i-n_{i-1})\frac{3}{n_i} \\ = \quad & 6+3\sum_{i=2}^{p-1}\left(\frac{1}{n_i}+\frac{1}{n_i}+\cdots+\frac{1}{n_i}\right) \\ \le \quad & 6+3\sum_{i=2}^{p-1}\left(\frac{1}{n_{i-1}+1}+\frac{1}{n_{i-1}+2}+\cdots+\frac{1}{n_i}\right) \\ \le \quad & 6+3\sum_{i=1}^{n_{p-1}}\frac{1}{i} \end{aligned}$$

别忘了我们一开始就把体积小于 $\frac{1}{c(I)}$ 的物品拿走了，那么 $n_{p-1} \le 3\boldsymbol{\nabla}\cdot\frac{1}{c(I)} = 3c(I)$，那么去掉的物品总体积就是 $O(\log c(I))$ 的了。

我们最后再把这些去掉的物品 first fit 一下就好了。我们知道 first fit 近似比是 1.7 的，不会改变大 O 的结论。

## 回顾一下

每次迭代都会把当前实例 $I$ 分成 $I_1$ 和 $I_2$。$I_1$ 里的物品由于恰好装满箱子，肯定是最优解的一部分，那么比最优解差的部分就来自于 $I_2$ 中被去掉的物品。而 $I_2$ 中被去掉的物品总体积为 $O(\log c(I))$，迭代最多进行 $\log c(I)$ 次，所以算法的结果就是 $\mathrm{OPT}_{LP}(I)+O(\log^2 C(I)) \le \mathrm{OPT}_{LP}(I)+O(\log^2 \mathrm{OPT}_{LP}(I))$，这就完成了证明。