# 装箱算法

## 1. 问题定义

物流公司在流通过程中，需要将打包完毕的箱子装入到一个货车的车厢中，为了提高物流效率，需要将车厢尽量填满，显然，车厢如果能被100%填满是最优的，但通常认为，车厢能够填满85%，可认为装箱是比较优化的。

设车厢为长方形，其长宽高分别为L，W，H；共有n个箱子，箱子也为长方形，第i个箱子的长宽高为li，wi，hi（n个箱子的体积总和是要远远大于车厢的体积），做以下假设和要求：

长方形的车厢共有8个角，并设靠近驾驶室并位于下端的一个角的坐标为（0,0,0），车厢共6个面，其中长的4个面，以及靠近驾驶室的面是封闭的，只有一个面是开着的，用于工人搬运箱子；

需要计算出每个箱子在车厢中的坐标，即每个箱子摆放后，其和车厢坐标为（0,0,0）的角相对应的角在车厢中的坐标，并计算车厢的填充率。

问题分为两部分：

基础部分：

1.所有的参数为整数；

2.静态装箱，即从n个箱子中选取m个箱子，并实现m个箱子在车厢中的摆放（无需考虑装箱的顺序，即不需要考虑箱子从内向外，从下向上这种在车厢中的装箱顺序）；

3.所有的箱子全部平放，即箱子的最大面朝下摆放；

4.算法时间不做严格要求，只要1天内得出结果都可。

高级部分：

1.参数考虑小数点后两位；

2.实现在线算法，也就是箱子是按照随机顺序到达，先到达先摆放；

3.需要考虑箱子的摆放顺序，即箱子是从内到外，从下向上的摆放顺序；

4.因箱子共有3个不同的面，所有每个箱子有3种不同的摆放状态；

5.算法需要实时得出结果，即算法时间小于等于2秒。

## 2 建模

### 2.1车厢

使用二维数组存储每一个单位空间的剩余可放置高度，例如，当车厢为空时，二维数组表示如下：

其中表示车厢的高度，当车厢不为空时，二维数组表示如下：

其中，表示该位置上放置的第个箱子的高度，表示位置(i,j)还可以使用的剩余高度。

### 2.2箱子

每个箱子包含三个参数l,w,h分别表示箱子的长宽高，基于贪心算法，我们需要对每个箱子求出最大面、最小面，并排序。

### 2.3将箱子放入车厢

每次将箱子放入车厢的过程就是一个遍历车厢剩余高度二维数组的过程，遍历窗口即箱子的底面，，获得该窗口剩余高度的最小值作为当前窗口可以放置箱子的高度，如果箱子高度h小于，那么该箱子可以放置在当前位置，并更新当前剩余高度。

### 2.4车厢尺寸

在实际应用中，一般集装箱内尺寸大概为13.58x2.34x2.68米，因此我们将各项数据规约到厘米（即小数点后两位），即集装箱大小为：1500\*250\*250。

### 2.5箱子尺寸

在实际应用中，箱子的长宽高之比通常会在一定的范围内，小型包裹可以小到0.1m\*0.1m\*0.1m，大型包裹也可以很大（但不能超过集装箱），例如轿车3\*1.8\*1.4m，但是箱子尺寸并非完全随机，而是相对成比例的，因此本次实验中不考虑长宽高中某两个参数比例相差悬殊的情况。考虑到在线算法要求精确到小数点后两位，因此以厘米为单位，生成箱子时取范围内随机整数，箱子类型比例如下：

(1) 40%的小箱子。

(2) 40%的中等箱子。

(3) 10%的大箱子。

(4) 10%的超大箱子。

箱子体积期望为1,513,925，大约62个箱子可以填满整个车厢，为了生成足够数量的箱子，生成500个。

## 3 装箱策略

### 3.1离线算法

离线算法为贪心算法，首先将箱子按照体积从大到小进行排序，随后按顺序进行装箱；每次放置货物时，都将货物的最大面作为底面，底面作为一个窗口在空间截面滑动，如果遇到高度满足的区域，就放置进去，减少空间剩余高度，并记录当前空闲体积。这是基于一种启发式思想，大盒子放下面可以让小盒子垫在上面，但是这样窗口滑动次数会较多，导致算法运行时间较长。

我们有以下优化技巧：（1）如果下一个箱子大于空闲体积，直接放弃；（2）当货物的总体积远大于车厢体积时，由于已经将箱子按照体积从大到小进行排序，因此可以跳过k个体积近似的箱子，加快装箱的搜索速度，对填充率影响不大，但可以减少算法运行时间。

### 3.2在线算法

在线算法无法对箱子进行排序，只能按照箱子的生成顺序进行填装。根据经验，如果货物到达顺序为大小大，会形成一块比例失衡的空间，在后续中很难找到合适的箱子进行填充；即使后续存在许多可以填充的小箱子，也会由于大箱子的摆放使得这块空间成为封闭空间，降低填充率。

因此，在装箱前，额外对每一个箱子的大小进行判断，小箱子和大箱子分别从两侧开始填装，相当于对小箱子进行堆叠，实验证明，相比于大小混装，此种装箱策略可以小幅度提升填充率。

此外，由于箱子是随机到达的，无法根据体积贪心，如果依然考虑大面或小面作为底面的想法。此时的问题是，由于扩充了100倍，遍历底面空间很耗时，所以考虑小面作为底面，最长边为高。

同时由于充分利用高度空间，设置一个begin参数表示当前车厢占用率-某个参数，作为每次外层循环开始的位置，减少遍历次数，因为充分利用了高度，可以认为只要占用的空间，在高度上基本都占满了，因此从begin位置开始遍历是合理的。

## 4 结果分析一

基于上述问题建模，我们做了大量实验得出以下结论。

### 4.1离线算法

离线算法由于可以优先装大箱子，将空间割裂为不可利用的碎片的情况较少，还可以充分利用小箱子进行填充，因此在随机生成的数据集上，平均填充率可以稳定达到90%以上。

### 4.2在线算法

在考虑到将大小箱子分别填装，实现了小箱子的堆叠，减少了碎片空间的产生。在随机生成的数据集上，平均填充率可以稳定在85%左右，略高于85%。同时每个箱子的计算时间在1s以内。

## 5 结果分析二

以上的参数设置为我们小组为了符合实际而进行的假设，针对老师给出的箱子设定，我们也进行了实验，效果相较以上设置会差一些，我们认为是由于车厢尺寸设置不合理导致的，长宽高的比例不符合长方体，且箱子尺寸大多为大箱子，仅仅放入少数箱子就可以占满车厢的某个维度，导致剩余箱子无法放入，实验结果如下：

### 5.1离线算法

|  |  |
| --- | --- |
| 箱子类型 | 装箱率 |
| 3 | 0.7158 |
| 5 | 0.819 |
| 8 | 0.7586 |
| 10 | 0.7856 |
| 15 | 0.7789 |

### 5.2在线算法

|  |  |
| --- | --- |
| 箱子类型 | 装箱率 |
| 3 | 0.7584 |
| 5 | 0.7507 |
| 8 | 0.7584 |
| 10 | 0.7198 |
| 15 | 0.7215 |

每个箱子的计算时间在1s以内。