国家网络安全学院

《课程报告》

课　　程：　高级算法分析与设计

成　　员：　曾逸飞、2022202210112

成　　员：　杨洲、2022202210138

一、设计题目

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：

（1）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；

（2）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。

二、设计目标

（1）**在线算法**，箱子按随机顺序到达

（2）参数**可以考虑到小数点后两位**

（3）装箱时考虑箱子**6种不同的摆放状态**

（4）箱子放入时的位置**与其最终位置保持一致**

（5）**单次装箱的算法执行时间小于1秒**，对于所选的数据，总执行时间小于1分钟

（6）禁止箱子出现悬空

三、方案设计

本实验设计了一种基于可用点筛选与贪心算法的装箱方案，初始状态下（车厢空时），可用点只有墙角的坐标点O(0, 0, 0)。当长宽高分别为x,y,z的箱子放置在某一可用点时，会因为其自身的占用，导致该可用点失效，因此需要将该可用点移除；同时，由于箱子均为贴边紧密放置，因此会在顶、正、侧三面（贴边）的角产生三个新的可用点A、B、C，作为其他箱子放置的备选坐标，该过程如图3.1所示：

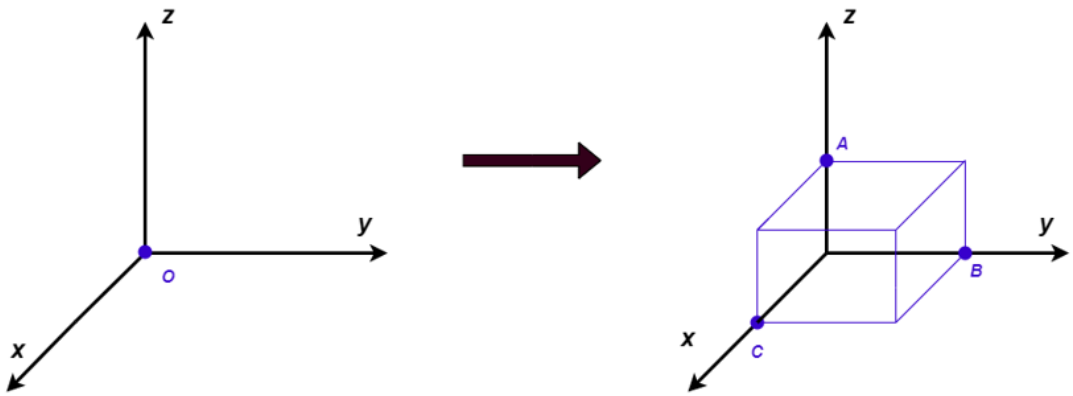

图 3.1 可用点调整过程

在每一次摆放新的箱子时，我们在每一个可用摆放点处尝试 6 种不同的摆放姿态，从中筛选出所有不会与其他箱子发生碰撞且不会悬空与溢出的摆放点与摆放姿态，然后我们在所有的合法摆放点与摆放姿态下计算此时的占用得分值，将该值记为 score，并且选择最小的 score 所对应的摆放点与摆放状态作为当前最优的摆放点与摆放状态。在该箱子被成功放置后，与之对应的放置点失效，本算法根据该放置点的坐标以及被放置箱子的大小以及摆放姿态再增加 3 个可用放置点。

在算法设计的过程中，我们会考虑两个因素：摆放合法性和摆放最优性。针对摆放合法性，最关键的步骤包括箱子姿态的调整、箱子的碰撞检测、箱子的溢出检测、箱子的悬空检测、放置点的更新。针对摆放最优性，最重要的步骤是优化目标的建立以及搜索与摆放过程，下文对这几个过程进行重点介绍。

3.1. 箱子姿态的调整

为了提高箱子的整体利用率，我们需要合理地调整箱子的摆放姿态。如图 3.2 所示，同一个箱子共有 6 种摆放姿态。

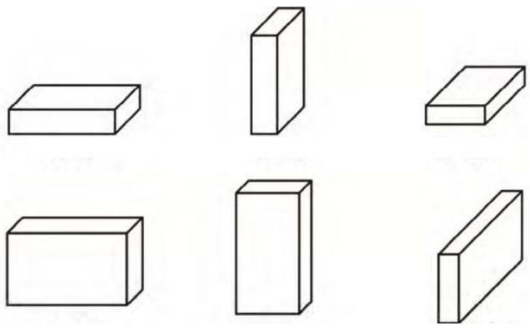

图 3.2 同一箱子的 6 种不同摆放状态

我们结合箱子的原始长、宽、高，以及在空间坐标系x，y，z轴上的摆放方式，定义了表 3.1 所示的 6 种摆放姿态，在此处$long$，$width$，$height$代表原始物体的长、宽、高，x，y，z代表三个坐标轴的正方向所摆放的内容。姿态调整完成后，我们会将该摆放姿态对应的编号存储在箱子类的 pose 属性中

表 3.1　箱子 6 种摆放姿态的建模

姿态编号	摆放方式
1	$x = long, y = width, z = height$
2	$x = long, y = height, z = width$
3	$x = width, y = long, z = height$
4	$x = width, y = height, z = long$
5	$x = height, y = long, z = width$
6	$x = height, y = width, z = long$

3.2. 箱子的碰撞检测

箱子的碰撞检测决定了该物品能否以某种姿态放置在某个特定的可用点上。检测过程分别从主视图、俯视图、左视图三个方向进行。如图 3.3 所示，当两个箱子在两个及以上的视图中发生重叠时，则判断两箱子产生碰撞，该过程可描述为：

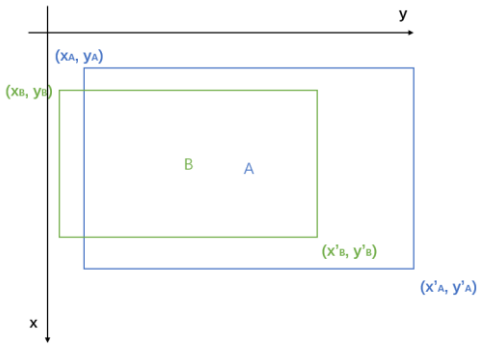

图 3.3　两箱子在俯视图下产生重叠示意图

Step1:将当前箱子记为 A，放置点为(x_1, y_1, z_1)，当前摆放状态所对应的长、宽、高分别记为$(long_1, width_1, height_1)$，从已放置列表$settled_list$中选择一个箱子记为 B，其放置点记为$(x_2, y_2, z_2)$，当前排放状态所对应的长、宽、高分别记为$(long_2, width_2, height_2)$，初始化一个计数器变量$counter$记为 0

Step2:从箱子 A 的俯视图中获取两点$a_1(x_1,y_1)$、$a_2(x_1+long_1,y_1+width_1)$，从箱子 B 的俯视图中获取两点$b_1(x_2,y_2)$、$b_2(x_2+long_2,y_2+width_2)$

Step3: 判断$x_1 \geq x_2+long_2$或$x_1+long_1 \leq x_2$或$y_1 \geq y_2+width_2$或$y_1+width_1 \leq y_2$是否成立，若不成立则令$counter=counter+1$

Step4:从箱子 A 的主视图中获取两点$a_3(y_1,z_1)$、$a_4(y_1+width_1,z_1+height_1)$，从箱子 B 的主视图中获取两点$b_3(y_2,z_2)$、$b_4(y_2+width_2,z_2+height_2)$

Step5: 判断$y_1 \geq y_2+width_2$或$y_1+width_1 \leq y_2$或$z_1 \geq z_2+height_2$或$z_1+height_1 \leq z_2$是否成立，若不成立则令$counter=counter+1$

Step6:从箱子 A 的俯视图中获取两点$a_5(x_1,z_1)$、$a_6(x_1+long_1,z_1+height_1)$，从箱子 B 的俯视图中获取两点$b_5(x_2,z_2)$、$b_6(x_2+long_2,z_2+height_2)$

Step7: 判断$x_1 \geq x_2+long_2$或$x_1+long_1 \leq x_2$或$z_1 \geq z_2+height_2$或$z_1+height_1 \leq z_2$是否成立，若不成立则令$counter=counter+1$

Step8:若$counter>1$，则检测到碰撞，返回$True$，不允许该物箱子以该姿态放在该点。反之，则从已放置列表$settled_list$中选择下一个箱子作为 B，然后重新进行 Step1-Step7,当$settled_list$中所有的物品都被遍历完一遍后，均没有检测到碰撞，则返回$False$。

3.3.箱子的溢出检测

根据箱子的放置坐标$A(x,y,z)$和摆放时的长、宽、高（这里分别记为$long,width,height$），计算其放置后与的坐标。若其中有一个角的坐标超出车厢的长、宽、高限制，则说明该箱子的放置产生了溢出。该过程可描述为：

Step1:将货车箱子的长、宽、高,分别记为L, W, H.

Step2:判断$x+long \leq L$, $y+width \leq W$, $z+height \leq H$,三个不等式是否同时成立，若同时成立，则认为没有产生溢出，反之则认为产生溢出。

3.4.箱子的悬空检测

由于本实验中不允许货物箱子出现悬空放置的情况，因此在所有 z 坐标大于 0 的放置点放置货物箱子时，均需要检测该箱子是否悬空，也就是在俯视图下，被放置箱子的投影是否位于支撑箱子的投影中，该过程可描述为：

Step1：取放置点$O_1(x_1,y_1,z_1)$，以及该姿态下长、宽、高分别为

$long_1, width_1, height_1$的被放置在此处的物品$A$

Step2：判断$z_1 > 0$是否成立，若不成立，则无需进行悬空检测，直接结束该过程，反之则继续进行 Step3。

Step3:遍历已放置列表$settled_list$，从中找到一个被放置在$O_2(x_2, y_2, z_2)$且该姿态下的长、宽、高分别为$long_2, width_2, height_2$的被放置物品$B$，若$z_2 + height_2 = z_1$，则进行 Step4，反之则继续寻找。

Step4：判断$x_1 \geq x_2 + long_2$或$x_1 + long_1 \leq x_2$或$y_1 \geq y_2 + width_2$或$y_1 + width_1 \leq y_2$是否成立，若成立，则跳转到 Step3 重新寻找，反之则继续。

Step5：判断$x_1 \geq x_2, y_1 \geq y_2, x_1 + long_1 \leq x_2 + long_2, y_1 + width_1 \leq y_2 + width_2$是否同时成立，若成立，则完全不会悬空，结束此过程。反之，则跳转到 Step3 继续搜索，直至$settled_list$遍历结束。

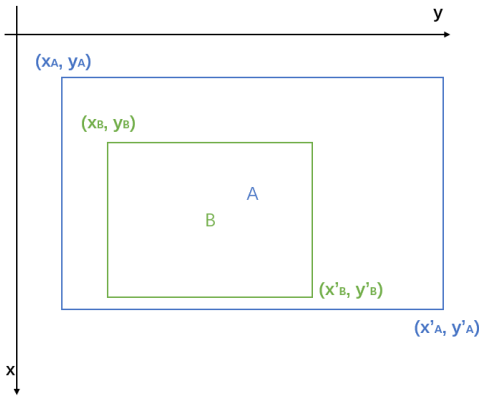

图 3.4　俯视图下非悬空示意图

3.5. 优化目标的建立

本算法每一步的优化目标是选择最小的占用分数$score$值所对应的物品摆放点坐标与摆放姿态，这是一种贪心策略。

对于每种通过合法性测试的摆放方式，都需要对其$score$值进行计算。$score$的计算需要考虑底面积占用和高度占用两个因素，其具体规则如下：

$$min \quad score = \alpha \times \Delta S \times (1 - \alpha) \times h$$

其中，各参数的含义如下：

- ΔS为将当前物体放置在某一可用点$O_1(x_1, y_1, z_1)$后，该物体在当前摆放姿态下的底面积与其支撑物体的底面积之差。若$z_1 = 0$，则将ΔS设置为 1。
- h为将当前物体放置在某一可用点O_1后，该物品顶部距离车厢底部的值。

•α为事先设置的调节因子，用于调节$score$计算中底面积和高度所占的权重，根据大量的实验，α设置为0.2时效果最佳。

本算法在执行过程中维护了一个变量min_score，用于在优化搜索的过程中存储当前搜索到的最小的$score$值。若计算出的$score$小于当前的min_score，则将当前放置点的坐标和摆放姿态设置为当前局部最优解，同时更新min_score，然后继续搜索，直至遍历完$settled_list$中所有的合法的可用点和可用状态。

$score$的计算和选择可以在一定程度上避免不合理放置方式。通过这样的参数设置，要使$score$的值尽可能小，那么箱子将选择底面积占用和高度占用更少的摆放方式，同时更加倾向于摆放在已放置的箱子之上。

3.6.箱子的放置与可用点的更新

我们将每一轮货物放置时min_score值对应的放置坐标与摆放姿势作为最终的放置选择，该放置过程包括设置箱子的坐标属性、将该箱加入已放置箱子集合$settled_list$、计算当前车厢的底面积和高度占用以及修改可用点等过程。假设箱子放置的可用点坐标为$O(x,y,z)$,箱子放置时的长宽高分别为l,w,h，根据图 3.1 可用点的调整方法，坐标$O(x,y,z)$被移除，而新产生的三个可用点坐标分别为$A(x,y,z+h)$、$B(x,y+w,z)$和$C(x+l,y,z)$。若在一轮搜索与遍历结束后，min_score的值仍旧等于初始最大值，则证明没有任何一种摆放方法符合该箱子，该箱子放置失败。

四、实现细节

我们的算法设置了三个主要的类，分别对应点、箱子和车厢。

4.1 点(Point)

4.1.1 属性值

（1）该点的 x、y、z 坐标。

4.1.2 主要方法

（1）Tuple 方法

参数：无

功能：查询该点的坐标

返回值：该方法返回由 x、y、z 坐标值组成的元组。

（2）eq 方法

参数：另一个 Point 类的对象

功能：判断两个 Point 是否相同

返回值：判断结果 True 或 False

4.2 箱子（Cargo）

4.2.1 属性值

（1）箱子的初始长宽高

（2）箱子摆放姿态对应的编号（pose）

（3）改变摆放姿态后的长宽高

（4）坐标点 x、y、z

4.2.2 主要方法

（1）volume 方法

参数：无

功能：根据箱子的长宽高计算体积

返回值：该箱子的体积

（2）set_locate_point 方法

参数：点的 x、y、z 坐标值

功能：设置该箱子放置的坐标点，即修改 Cargo 的 x、y、z 属性

返回值：无

（3）get_x/y/z 方法

参数：无

功能：查询箱子的放置位置

返回值：箱子放置点的 x、y、z 坐标

（4）change_pose 方法

参数：箱子放置姿态的编号 num

功能：根据编号改变箱子的摆放姿态（修改 Cargo 的 pose 属性），并设置改变姿态后箱子的长宽高属性

返回值：无

（5）get_locate_xxx_area 方法

参数：无

功能：计算并返回该箱子 3 个视图的面积，用于车厢占用面积和 score 的计算

返回值：箱子俯/正/侧视图的面积

4.3 车厢（Container）

4.3.1 属性值

（1）车厢的长宽高

（2）可用点坐标集合（available）

（3）已放置的箱子集合（settled_cargo）

（4）已占用底面积和箱子达到的最高高度（用于 score 计算）

（5）调节因子(alpha)

4.3.2 主要方法

（1）calculate_usage 方法：

参数：无

功能：统计已放置箱子的总体积，计算车厢的空间使用率

返回值：车厢的空间使用率

（2）add_cargo_to_container 方法：

参数：Cargo 对象（箱子）

功能：箱子的放置算法，该方法的流程在前文第三节中进行了详细介绍

返回值：一个 bool 值，用于表示箱子是否摆放成功

（3）determine_overflow 方法：

参数：Cargo 对象（箱子）

功能：从箱子是否超出车厢限制的角度进行合法性判断，即溢出性检测。该方法计算箱子摆放后各个角的坐标，并根据各个角的坐标位置判断箱子的摆放是否超出车厢限制

返回值：一个 bool 值，用于表示箱子的摆放是否超出车厢限制

（4）check_crash_and_suspended 方法：

参数：Cargo 对象（箱子）

功能：从该箱子与已放置箱子之间的关系角度进行合法性判断。该方法依次遍历已放置箱子的集合，并分别调用⑤和⑥两类方法，对该箱子与已放置箱子之间进行碰撞检测和悬空检测

返回值：一个 bool 值，用于表示箱子的摆放是否合法

（5）determine_xxx_view_overlaps 方法：

参数：两个 Cargo 对象 A、B，其中 A 为被检测的箱子，B 为某个已放置的箱子

功能：分别从三个视图判断两个箱子之间是否相撞。如果碰撞在两个及以上的视图下发生，则判断 A 和 B 相撞

返回值：一个 bool 值，用于表示两个箱子之间是否相撞

（6）judge_box_suspended 方法：

参数：两个 Cargo 对象 A、B，其中 A 为被检测的箱子，B 为某个已放置的箱子

功能：以 B 为参照对 A 进行悬空检测，根据两个箱子的坐标关系，判断 B 是否悬空

返回值：一个 bool 值，用于表示 B 是否悬空放置

（7）solve_old_point_add_new_point 方法：

参数：一个 Cargo 对象（箱子）和一个 Point 对象（坐标点）

功能：从可用点集合中删除该点，同时计算并添加新的可用点。该步骤在箱子成功放置后执行，具体流程如下：①将该箱子放置的坐标点从可用点集合中删除。②根据删除的可用点坐标和箱子放置后的长宽高，计算出三个新的可用点坐标。③将新坐标加入可用点集合中

返回值：无

五、性能测试

5.1.测试环境

本实验的代码开发与运行环境如下所示：

（1）Windows 版本：Windows 10 专业版

（2）处理器：Intel(R) Core(TM)i5-10400F

（3）CPU 内存：16GB

（4）开发工具：PyCharm Community Edition 2020.2.2 x64

（5）编译器：python3.7

5.2.测试用例

为了证明本实验的有效性，我们设计了如下表 5.1 所示的 5 组测试用例，每组分别包含 3、5、8、10、15 种箱子，且所有的箱子是**打乱顺序后随机到达**的。

表 5.1　测试用例说明表

编号	车厢大小	箱子长宽高	箱子数量
1	(587,233,220)	（108,76,30）	50
		（110,43,25）	60
		（92,81,55）	50
2	(587,233,220)	(108,76,30)	30
		(110,43,25)	40
		(92,81,55)	30
		(81,33,28)	30
		(120,99,73)	30
3	(587,233,220)	(97,81,27)	20
		(102,78,39)	20
		(113,46,36)	20
		(101,30,26)	20
		(66,50,42)	20
		(100,56,35)	20
		(91,50,40)	20
		(106,61,56)	20
4	(587,233,220)	(97,81,27)	10
		(102,78,39)	20
		(113,46,36)	10
		(66,50,42)	10
		(101,43,25)	10
		(100,56,35)	10
		(91,50,40)	10
		(106,61,56)	10

		(103,63,58)	10
		(75,57,41)	10
5	(587,233,220)	(108,76,30)	10
		(110,43,25)	10
		(92,81,55)	10
		(81,33,28)	10
		(120,99,73)	10
		(111,70,48)	10
		(98,72,46)	10
		(95,66,31)	10
		(85,84,30)	10
		(71,32,25)	10
		(36,34,25)	10
		(97,67,62)	10
		(33,25,23)	10
		(95,27,26)	10
		(94,81,44)	10

5.3.用例 1 测试结果

（1）箱子放置过程展示

```
大小为 (110, 43, 25) 成功装入,放在 (0,0,0)
大小为 (108, 76, 30) 成功装入,放在 (110,0,0)
大小为 (92, 81, 55) 成功装入,放在 (0,43,0)
大小为 (110, 43, 25) 成功装入,放在 (0,0,25)
大小为 (92, 81, 55) 成功装入,放在 (0,43,55)
大小为 (92, 81, 55) 成功装入,放在 (0,43,110)
大小为 (108, 76, 30) 成功装入,放在 (110,0,30)
大小为 (108, 76, 30) 成功装入,放在 (110,0,60)
大小为 (92, 81, 55) 成功装入,放在 (0,43,165)
大小为 (92, 81, 55) 成功装入,放在 (218,0,0)
大小为 (110, 43, 25) 成功装入,放在 (0,0,50)
大小为 (110, 43, 25) 成功装入,放在 (0,0,75)
大小为 (92, 81, 55) 成功装入,放在 (218,0,55)
大小为 (110, 43, 25) 成功装入,放在 (0,0,100)
大小为 (110, 43, 25) 成功装入,放在 (0,0,125)
大小为 (108, 76, 30) 成功装入,放在 (110,0,90)
大小为 (108, 76, 30) 成功装入,放在 (110,0,120)
大小为 (108, 76, 30) 成功装入,放在 (110,0,150)
大小为 (92, 81, 55) 成功装入,放在 (218,0,110)
大小为 (92, 81, 55) 成功装入,放在 (218,0,165)
大小为 (92, 81, 55) 成功装入,放在 (110,76,0)
大小为 (110, 43, 25) 成功装入,放在 (0,0,150)
大小为 (92, 81, 55) 成功装入,放在 (110,76,55)
大小为 (110, 43, 25) 成功装入,放在 (0,0,175)
大小为 (110, 43, 25) 成功装入,放在 (0,124,0)
```

```
编号145无法装入
编号146无法装入
编号147无法装入
编号148无法装入
编号149无法装入
编号150无法装入
编号151无法装入
编号152无法装入
编号153无法装入
编号154无法装入
编号155无法装入
编号156无法装入
编号157无法装入
编号158无法装入
编号159无法装入
程序运行时间是: 40.671875s
利用率 0.8195264679314661
```

（2）利用率结果展示

图 5.1 展示了随着箱子的不断到达以及被装入，箱子的累计利用率，以及箱子的最终摆放情况，最终该箱子的累计利用率为 81.95%

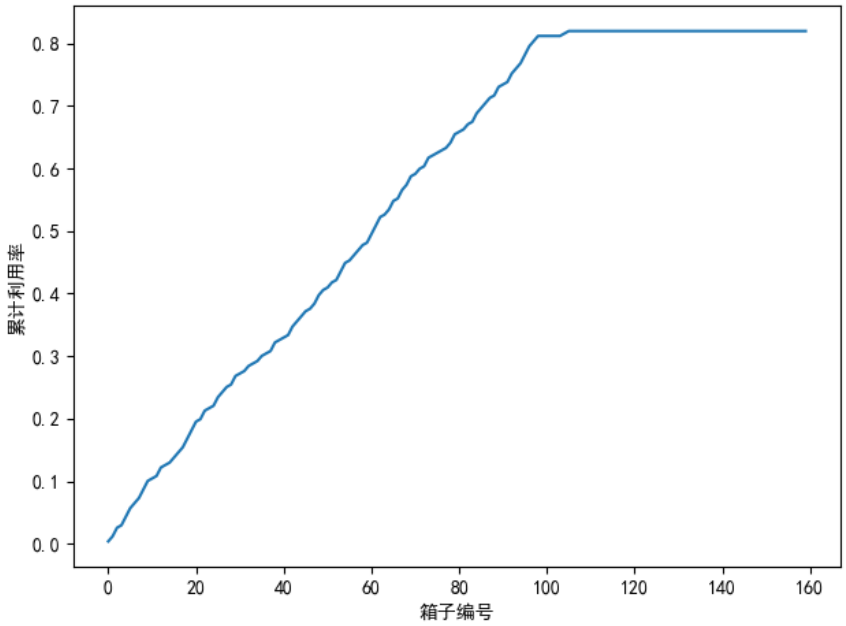

图 5.1a　装箱过程中累计利用率的变化情况

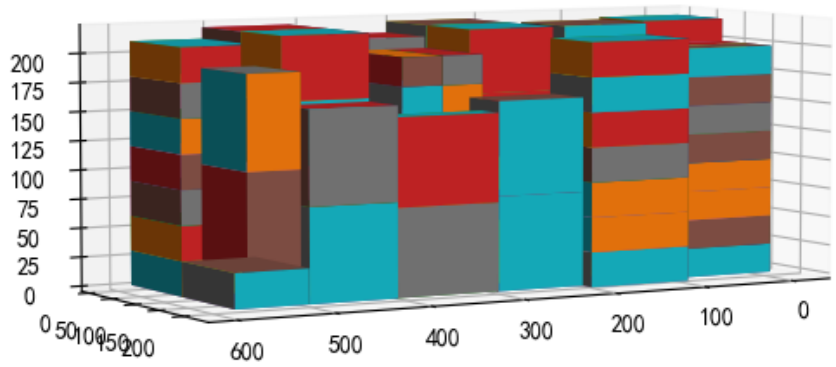

图 5.1b　最终的箱子摆放情况

5.4.用例 2 测试结果

（1）箱子放置过程展示

```
大小为 (110, 43, 25) 成功装入,放在 (0,0,0)
大小为 (108, 76, 30) 成功装入,放在 (110,0,0)
大小为 (110, 43, 25) 成功装入,放在 (0,0,25)
大小为 (108, 76, 30) 成功装入,放在 (110,0,30)
大小为 (110, 43, 25) 成功装入,放在 (0,0,50)
大小为 (92, 81, 55) 成功装入,放在 (0,43,0)
大小为 (110, 43, 25) 成功装入,放在 (0,0,75)
大小为 (108, 76, 30) 成功装入,放在 (110,0,60)
大小为 (110, 43, 25) 成功装入,放在 (0,0,100)
大小为 (110, 43, 25) 成功装入,放在 (0,0,125)
大小为 (92, 81, 55) 成功装入,放在 (0,43,55)
大小为 (92, 81, 55) 成功装入,放在 (0,43,110)
大小为 (110, 43, 25) 成功装入,放在 (0,0,150)
大小为 (108, 76, 30) 成功装入,放在 (110,0,90)
大小为 (110, 43, 25) 成功装入,放在 (0,0,175)
大小为 (108, 76, 30) 成功装入,放在 (110,0,120)
大小为 (110, 43, 25) 成功装入,放在 (218,0,0)
大小为 (92, 81, 55) 成功装入,放在 (0,43,165)
大小为 (108, 76, 30) 成功装入,放在 (110,0,150)
大小为 (25, 43, 110) 成功装入,放在 (191,185,110)
编号144无法装入
编号145无法装入
大小为 (28, 33, 81) 成功装入,放在 (409,200,0)
大小为 (28, 33, 81) 成功装入,放在 (409,200,81)
编号148无法装入
大小为 (28, 81, 33) 成功装入,放在 (548,0,152)
编号150无法装入
编号151无法装入
编号152无法装入
大小为 (28, 81, 33) 成功装入,放在 (548,0,185)
编号154无法装入
编号155无法装入
编号156无法装入
编号157无法装入
大小为 (81, 33, 28) 成功装入,放在 (438,162,162)
编号159无法装入
程序运行时间是:  55.71875s
利用率 0.8261403766481598
```

（2）利用率结果展示

图 5.2 展示了随着箱子的不断到达以及被装入，箱子的累计利用率情况，以及箱子的最终摆放情况，最终该箱子的累计利用率为 82.61%

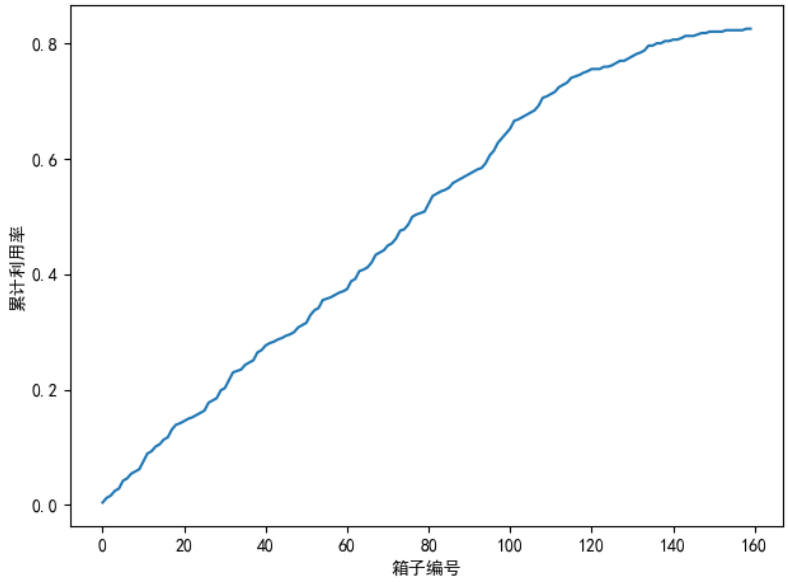

图 5.2a　装箱过程中累计利用率的变化情况

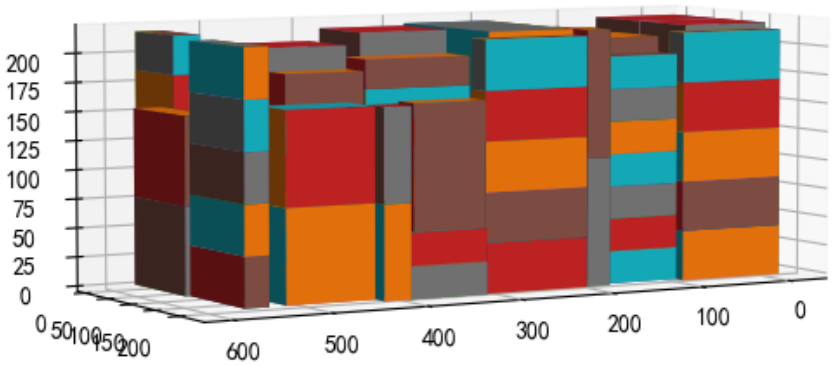

图 5.2b　最终的箱子摆放情况

5.5.用例 3 测试结果

（1）箱子放置过程展示

```
大小为 (101, 30, 26) 成功装入,放在 (0,0,0)
大小为 (97, 81, 27) 成功装入,放在 (101,0,0)
大小为 (61, 106, 56) 成功装入,放在 (0,30,0)
大小为 (101, 30, 26) 成功装入,放在 (0,0,26)
大小为 (97, 81, 27) 成功装入,放在 (101,0,27)
大小为 (100, 56, 35) 成功装入,放在 (198,0,0)
大小为 (91, 50, 40) 成功装入,放在 (101,81,0)
大小为 (66, 50, 42) 成功装入,放在 (0,136,0)
大小为 (113, 46, 36) 成功装入,放在 (298,0,0)
大小为 (66, 50, 42) 成功装入,放在 (0,136,42)
大小为 (100, 56, 35) 成功装入,放在 (198,0,35)
大小为 (91, 50, 40) 成功装入,放在 (101,81,40)
大小为 (100, 56, 35) 成功装入,放在 (198,0,70)
大小为 (100, 56, 35) 成功装入,放在 (198,0,105)
大小为 (66, 50, 42) 成功装入,放在 (0,136,84)
大小为 (113, 46, 36) 成功装入,放在 (298,0,36)
大小为 (101, 30, 26) 成功装入,放在 (0,0,52)
大小为 (113, 46, 36) 成功装入,放在 (298,0,72)
大小为 (61, 106, 56) 成功装入,放在 (0,30,56)
大小为 (100, 56, 35) 成功装入,放在 (198,0,140)
大小为 (61, 106, 56) 成功装入,放在 (0,30,112)
大小为 (66, 50, 42) 成功装入,放在 (0,136,126)
大小为 (97, 81, 27) 成功装入,放在 (101,0,54)
大小为 (97, 81, 27) 成功装入,放在 (101,0,81)
大小为 (91, 50, 40) 成功装入,放在 (101,81,80)
```

```
大小为 (30, 26, 101) 成功装入,放在 (513,197,
编号141无法装入
编号142无法装入
编号143无法装入
编号144无法装入
编号145无法装入
编号146无法装入
编号147无法装入
编号148无法装入
编号149无法装入
编号150无法装入
编号151无法装入
编号152无法装入
编号153无法装入
编号154无法装入
编号155无法装入
编号156无法装入
编号157无法装入
编号158无法装入
编号159无法装入
程序运行时间是: 44.140625s
利用率 0.8007607606875726
```

（2）利用率结果展示

图 5.3 展示了随着箱子的不断到达以及被装入，箱子的累计利用率情况，以

及箱子的最终摆放情况，最终该箱子的累计利用率为 80.08%.

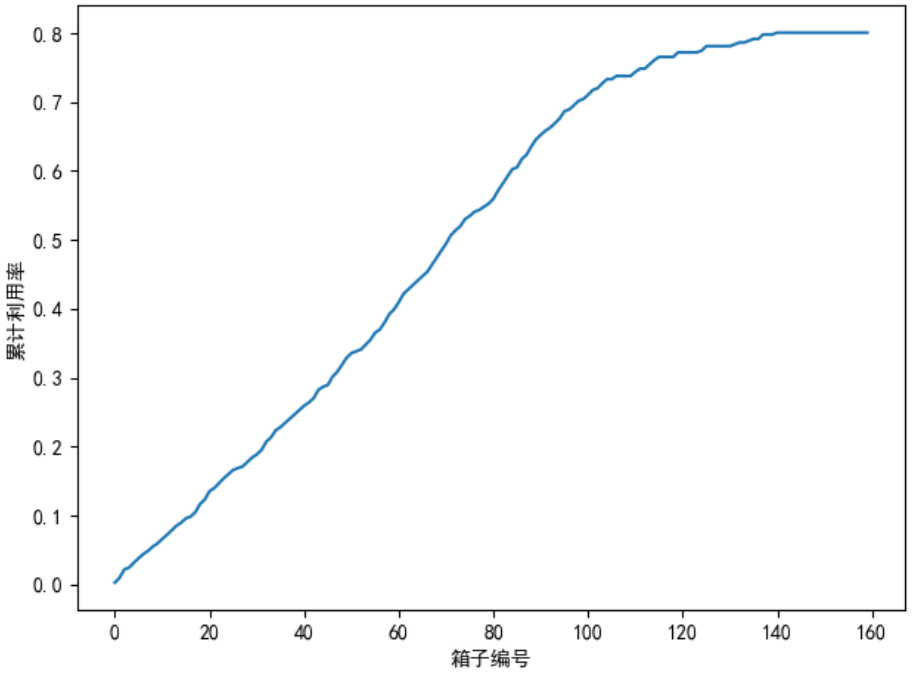

图 5.3a　装箱过程中累计利用率的变化情况

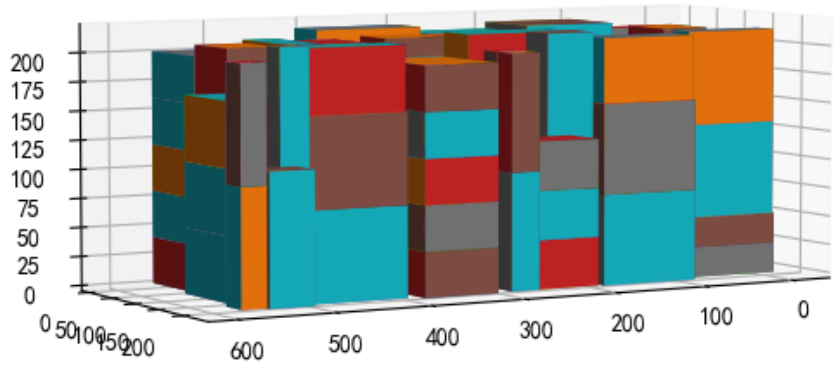

图 5.3b　最终的箱子摆放情况

5.6.用例 4 测试结果

（1）箱子放置过程展示

```
大小为 (102, 78, 39) 成功装入,放在 (0,0,0)
大小为 (110, 43, 25) 成功装入,放在 (102,0,0)
大小为 (113, 46, 36) 成功装入,放在 (0,78,0)
大小为 (103, 63, 58) 成功装入,放在 (212,0,0)
大小为 (91, 50, 40) 成功装入,放在 (113,78,0)
大小为 (103, 63, 58) 成功装入,放在 (212,0,58)
大小为 (97, 81, 27) 成功装入,放在 (0,124,0)
大小为 (103, 63, 58) 成功装入,放在 (212,0,116)
大小为 (66, 50, 42) 成功装入,放在 (315,0,0)
大小为 (113, 46, 36) 成功装入,放在 (0,78,36)
大小为 (106, 61, 56) 成功装入,放在 (212,63,0)
大小为 (100, 56, 35) 成功装入,放在 (113,128,0)
大小为 (113, 46, 36) 成功装入,放在 (0,78,72)
大小为 (100, 56, 35) 成功装入,放在 (113,128,35)
大小为 (106, 61, 56) 成功装入,放在 (212,63,56)
大小为 (103, 63, 58) 成功装入,放在 (381,0,0)
大小为 (103, 63, 58) 成功装入,放在 (381,0,58)
大小为 (106, 61, 56) 成功装入,放在 (212,63,112)
大小为 (106, 61, 56) 成功装入,放在 (318,63,0)
大小为 (100, 56, 35) 成功装入,放在 (113,128,70)
大小为 (91, 50, 40) 成功装入,放在 (113,78,40)
大小为 (100, 56, 35) 成功装入,放在 (113,128,105)
大小为 (100, 56, 35) 成功装入,放在 (113,128,140)
```

```
大小为 (25, 43, 110) 成功装入,放在 (179,184
编号142无法装入
编号143无法装入
编号144无法装入
编号145无法装入
编号146无法装入
编号147无法装入
编号148无法装入
编号149无法装入
编号150无法装入
编号151无法装入
编号152无法装入
编号153无法装入
编号154无法装入
编号155无法装入
编号156无法装入
编号157无法装入
编号158无法装入
编号159无法装入
程序运行时间是:   37.4375s
利用率 0.8240042911808125
```

（2）利用率结果展示

该图展示了随着箱子的不断到达以及被装入，箱子的累计利用率情况，以及

箱子的最终摆放情况，最终该箱子的累计利用率为 82.40%.

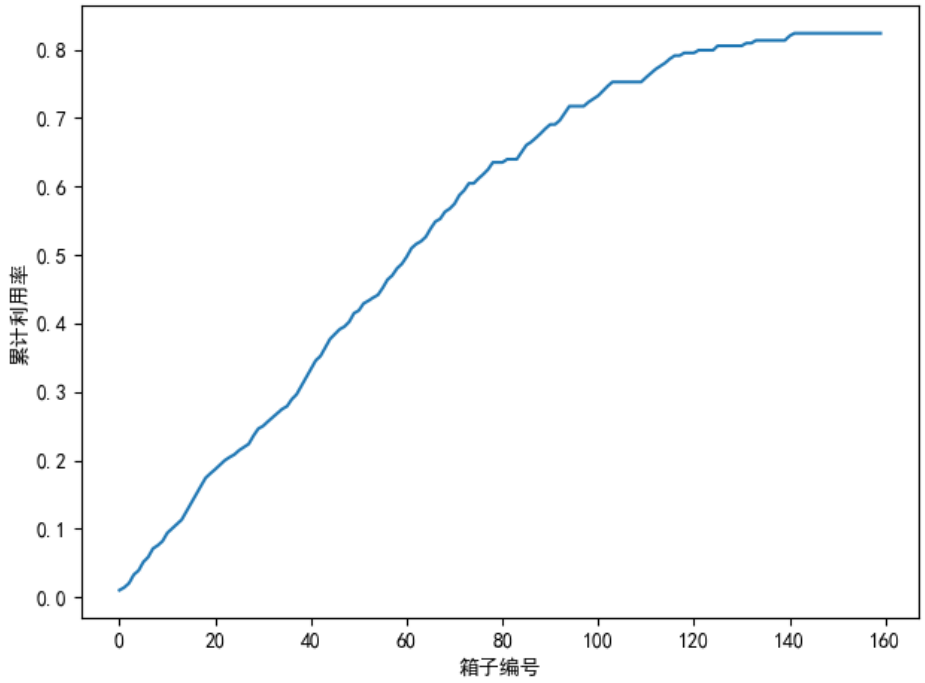

图 5.4a　装箱过程中累计利用率的变化情况

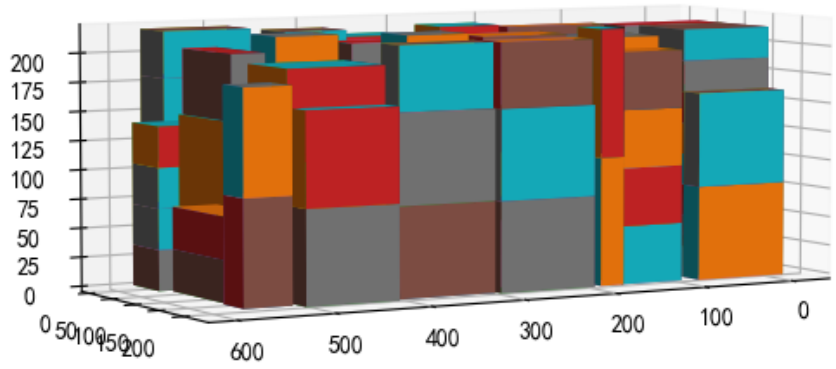

图 5.4b　最终的箱子摆放情况

5.6.用例 5 测试结果

（1）箱子放置过程展示

```
大小为 (98, 72, 46) 成功装入,放在 (0,0,0)
大小为 (92, 81, 55) 成功装入,放在 (98,0,0)
大小为 (85, 84, 30) 成功装入,放在 (0,72,0)
大小为 (94, 81, 44) 成功装入,放在 (190,0,0)
大小为 (95, 66, 31) 成功装入,放在 (98,81,0)
大小为 (108, 76, 30) 成功装入,放在 (0,156,0)
大小为 (92, 81, 55) 成功装入,放在 (98,0,55)
大小为 (110, 43, 25) 成功装入,放在 (284,0,0)
大小为 (98, 72, 46) 成功装入,放在 (0,0,46)
大小为 (110, 43, 25) 成功装入,放在 (284,0,25)
大小为 (71, 32, 25) 成功装入,放在 (193,81,0)
大小为 (36, 34, 25) 成功装入,放在 (108,156,0)
大小为 (108, 76, 30) 成功装入,放在 (0,156,30)
大小为 (94, 81, 44) 成功装入,放在 (190,0,44)
大小为 (71, 32, 25) 成功装入,放在 (193,81,25)
大小为 (94, 81, 44) 成功装入,放在 (190,0,88)
大小为 (95, 66, 31) 成功装入,放在 (98,81,31)
大小为 (95, 27, 26) 成功装入,放在 (394,0,0)
大小为 (120, 99, 73) 成功装入,放在 (284,43,0)
大小为 (85, 84, 30) 成功装入,放在 (0,72,30)
大小为 (94, 81, 44) 成功装入,放在 (190,0,132)
大小为 (85, 84, 30) 成功装入,放在 (0,72,60)
大小为 (94, 81, 44) 成功装入,放在 (190,0,176)
大小为 (98, 72, 46) 成功装入,放在 (0,0,92)
```

```
大小为 (32, 71, 25) 成功装入,放在 (0,120,175)
编号141无法装入
编号142无法装入
大小为 (32, 25, 71) 成功装入,放在 (360,134,0)
编号144无法装入
大小为 (43, 25, 110) 成功装入,放在 (534,195,50)
大小为 (33, 25, 23) 成功装入,放在 (0,0,184)
大小为 (27, 95, 26) 成功装入,放在 (0,25,182)
编号148无法装入
编号149无法装入
编号150无法装入
编号151无法装入
编号152无法装入
编号153无法装入
编号154无法装入
大小为 (32, 25, 71) 成功装入,放在 (360,134,71)
编号156无法装入
编号157无法装入
编号158无法装入
大小为 (23, 33, 25) 成功装入,放在 (391,160,0)
程序运行时间是:   42.9375s
利用率 0.8553305093251427
```

（2）利用率结果展示

图 5.5 展示了随着箱子的不断到达以及被装入，箱子的累计利用率情况，以及箱子的最终摆放情况，最终该箱子的累计利用率为 85.53%.

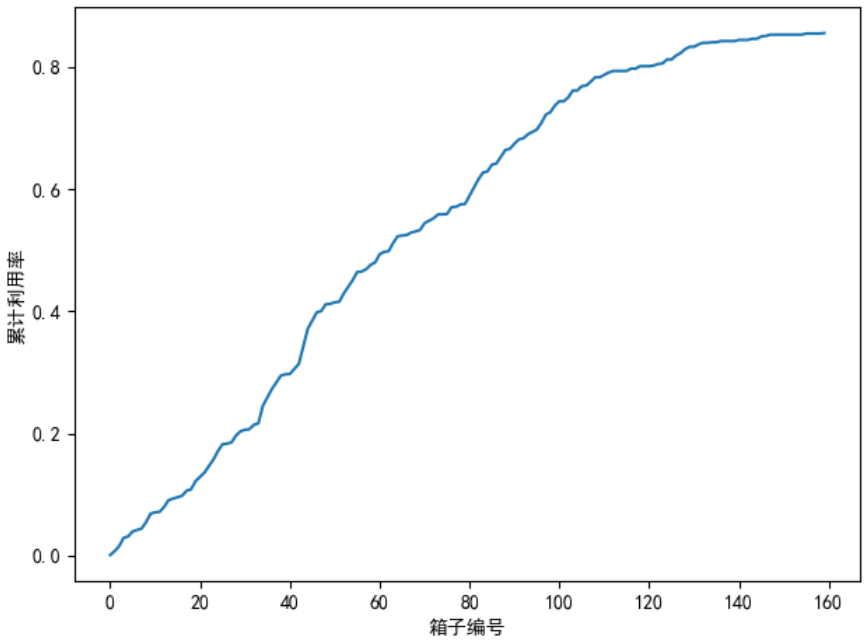

图 5.5a　装箱过程中累计利用率的变化情况

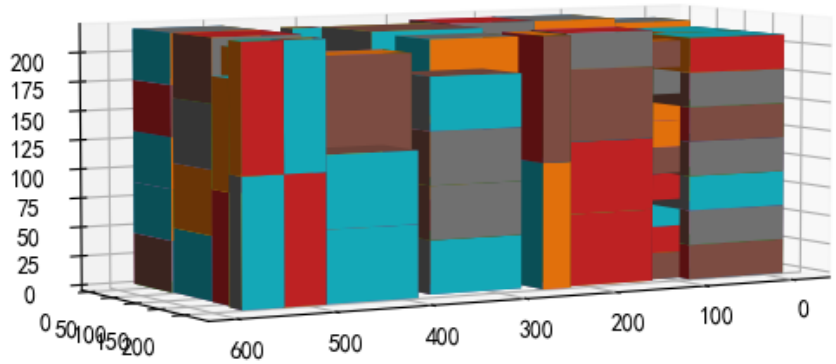

图 5.5b　最终的箱子摆放情况

5.8 时间结果展示

根据用例 1-5 的平均运行时间，在箱子个数为 160 时，每个箱子的放置平均时间小于 0.3s。

表 5.2　时间结果统计

编号	总时间(s)	每个箱子的放置平均时间(s)
用例 1	40.67	0.25
用例 2	55.72	0.35
用例 3	44.14	0.28
用例 4	37.43	0.23
用例 5	42.94	0.27

六、总结

经过优化目标的建立以及基于贪心算法的物品放置后，在上述的五组测试中箱子的利用率均能够达到 80%以上。根据该实验我们可以发现，随着物品箱子个数的增加，车厢利用率并没有出现明显的下降，而且随着已放置货物的增加，放置每个货物的时间不断增加。由于我们采用了贪心算法以及最小“总体积”的优化策略，车厢利用率显著提升，若要实现更高的利用率，还需采取更加细致与科学合理的优化目标。