运动鞋装箱问题

摘要

关键词 装载问题

1. 问题重述

1.1情况说明

某运动鞋厂家需要把生产的运动鞋统一包装运往销售点，因此需要为不同型号的运动鞋选择合适的鞋盒。选定鞋盒后，需要选择合适的装箱方案，以分别实现使用最少数量和最高利用率的包装箱的目标。

针对问题三：是经典的三维装箱问题，我们简化了模型构造了分层二维装箱问题。并采用C++构建二叉树进行装填。将总体积利用率简化成了总面积利用率最高。

1.2相关信息

附件（sheet1）：不同型号运动鞋的尺寸数据

附件（sheet2）：不同型号鞋盒的尺寸数据

附件（sheet3）：订单1各种型号运动鞋的需求量

附件（sheet4）：订单2各种型号运动鞋的需求量

1.3需要解决的问题

（1）为一批不同型号、尺寸已知的运动鞋选择合适的鞋盒。要求1）鞋与鞋盒之间长宽高间隔空隙分别不少于1cm；2）运动鞋占鞋盒总空间不少于70%。

（2）建立选用最少型号鞋盒的数学模型，并针对问题1数据给出相应的最少型号鞋盒选择方案。

（3）在问题1和问题2基础上，只采用鞋盒正面朝上的放置方法进行装箱，如何进行装箱能使得所用包装箱数量最少，并根据订单1数据给出相应装箱方案。

（4）在包装箱尺寸受一定约束下，采用正放和竖放两种鞋盒放置方式，如何进行装箱能使得包装箱总体积最少，并分别给出订单1、订单2的相应装箱方案和利用率。

1. 问题分析

2.1 问题一分析

问题一是在不同型号的运动鞋尺寸已知，可供选择的鞋盒尺寸已知情况下，满足一定约束条件的匹配问题。因此，我们采用了

2.2 问题二分析

在问题一的基础上，以鞋盒型号数最少建立目标函数，利用C++程序进行优化，得出满足约束条件的最优鞋盒型号选择，

* 1. 问题三分析

在问题1和问题2的基础上，采用分层构建，建立许多鞋盒层，然后遍历所有鞋盒层，如果能放入大箱子就放入大箱子直到所有层都遍历，将剩下放不下的层再次装入剩下的箱子，直到所有鞋盒层都放入大箱子里面，难点就是在一层的有效利用率最高。

2.4 问题四分析

问题为了求解请建立以所用包装箱总体积最少。构建一个类似于问题三的退化版本，将鞋盒订制为 80 \* 55 \* 30 经个人计算不会每个箱子中，装入超过20个箱子

2.5 解题思路

1. 问题假设

（1）附件中给出的数据来源有效。

（2）

1. 符号说明

|  |  |
| --- | --- |
| ***Li*** | 第i个鞋盒的长度，i = 1, 2, …… 23 |
| ***Wi*** | 第i个鞋盒的宽度，i = 1, 2, …… 23 |
| ***Hi*** | 第i个鞋盒的高度，i = 1, 2, …… 23 |
| ***Si*** | 第i个鞋盒的盒底面积，i = 1, 2, …… 23 |
| ***lj*** | 第j双运动鞋的长度，j = 1, 2, …… 15 |
| ***wj*** | 第j双运动鞋的宽度，j = 1, 2, …… 15 |
| ***hj*** | 第j双运动鞋的高度，j = 1, 2, …… 15 |
| ***sj*** | 第j双运动鞋的面积，j = 1, 2, …… 15 |
| ***pj*** | 第j双运动鞋中两只鞋公用空间，j = 1, 2, …… 15 |
| ***aij*** : 0 – 1变量 | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

(3)对于问题三，假设每一层放置可以满足稳定性要求；

(4)对于问题三，假设 鞋盒都按照一个方向放置并不进行旋转

(5)对于问题四，假设可以订制一个鞋盒大小的箱子

1. 模型的建立与求解

5.1 问题一模型建立与求解

5.1.1 模型建立

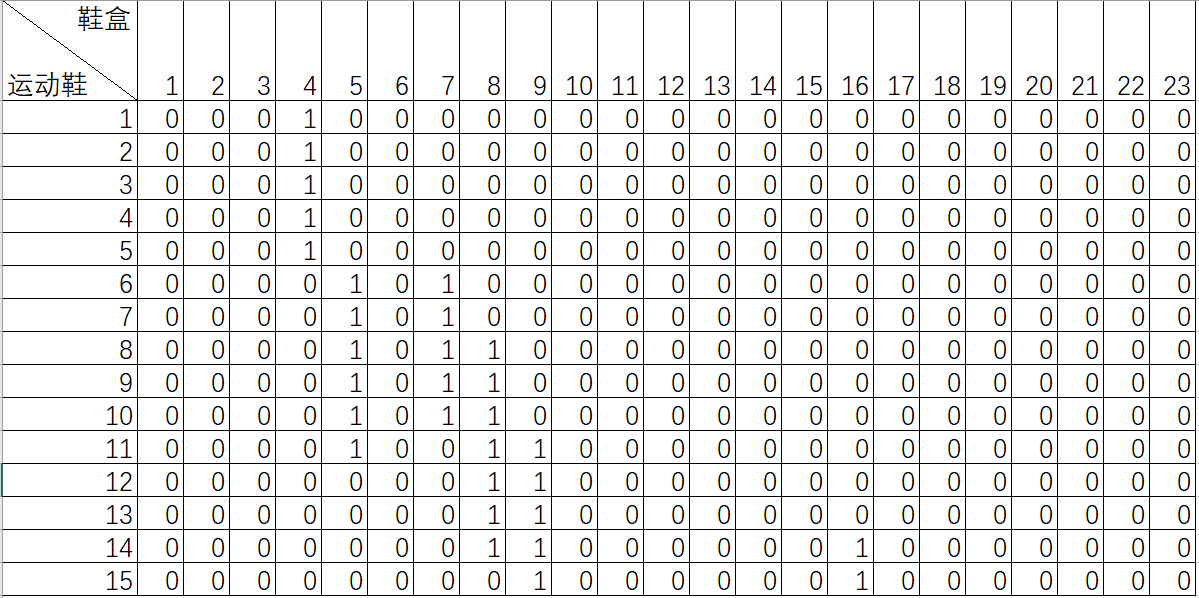
建立如下模型：

目标函数：***find =* 1**

约束条件：

5.1.2 模型求解

利用MATLAB软件编程对上述模型进行计算，并将计算结果输出到Excel软件中，结果如下表所示：



其中，横行表示鞋盒型号，纵行表示运动鞋型号，**aij** = 1 表示第i双鞋子能装入第j个鞋盒。

5.2 问题二模型建立与求解

5.2.1 模型建立

5.2.2 模型求解

5.3 问题三模型建立与求解

5.3.1 模型建立

使用分层构建鞋盒层，然后将鞋盒层放入箱子中

问题转为 求构建的每一层鞋盒层的最大利用率。

5.3.2 模型求解

利用C++求解，根据问题1和问题2可以得到可以使用的两个方案

鞋盒方案[ 4, 5 ,7] && [4, 5, 9]

对于每一种鞋盒方案

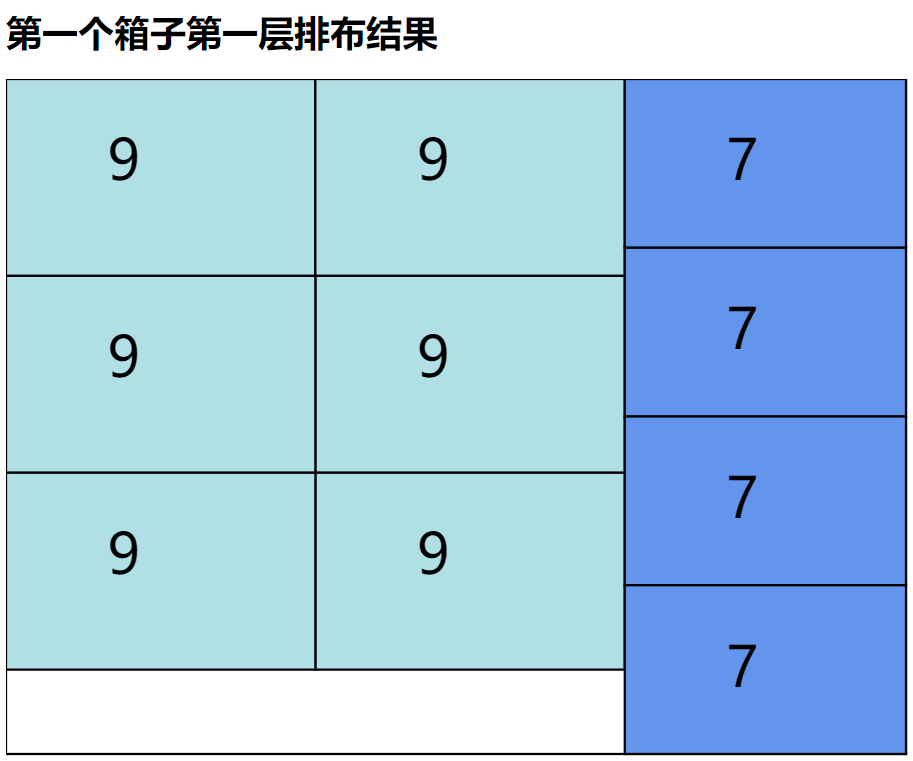
对于鞋子选择鞋盒方案，采用从小号的鞋盒判断到大的鞋盒，先判断4，再判断5，再判断7 例如：如果4号鞋盒满足那么所选的鞋子就放入4号鞋盒。

计算生成每一层使用二维矩形填充算法。采用二叉树分割矩形然后填充小矩形算法。

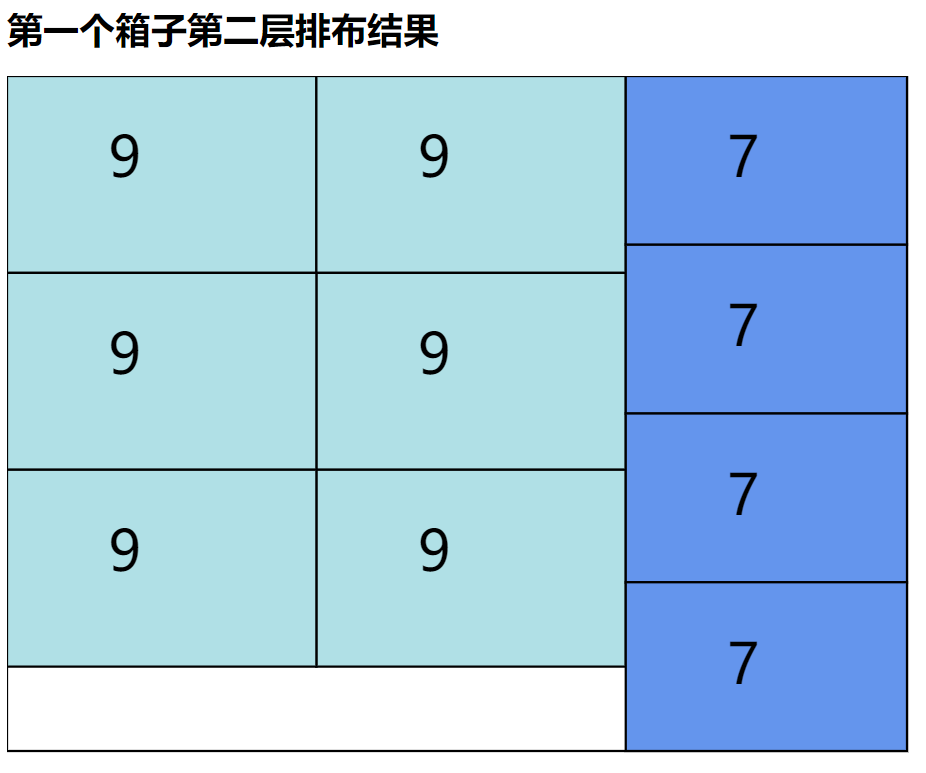
生成的方案 [4 5 9]第一层的矩形分布

矩形在第一层的分布 [起始坐标x, 起始坐标y, 矩形的宽度，矩形的高度]

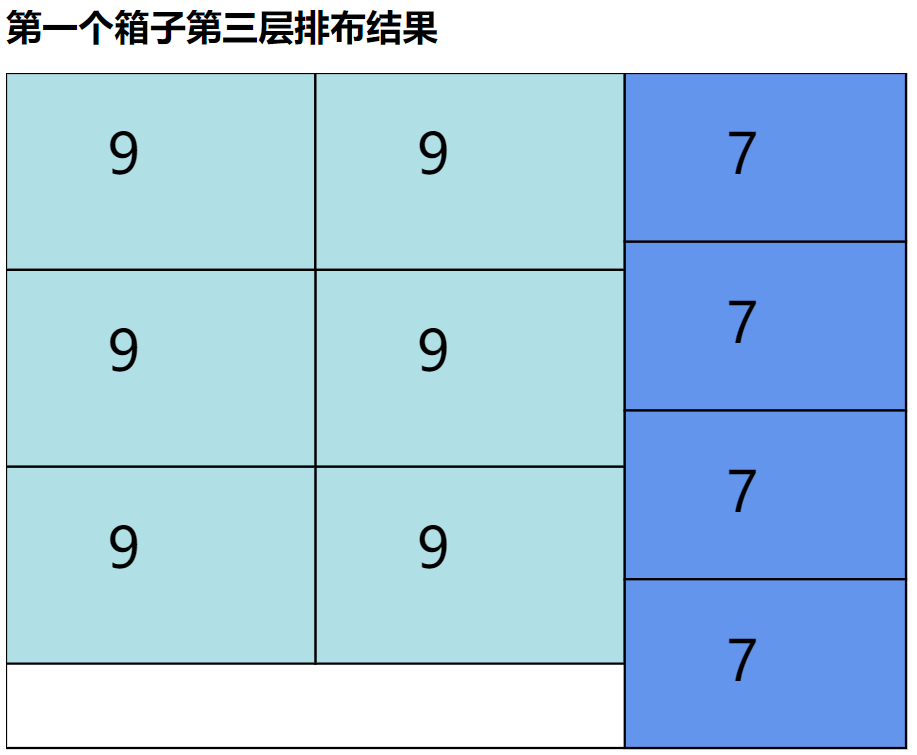
可视化数据后显示结果

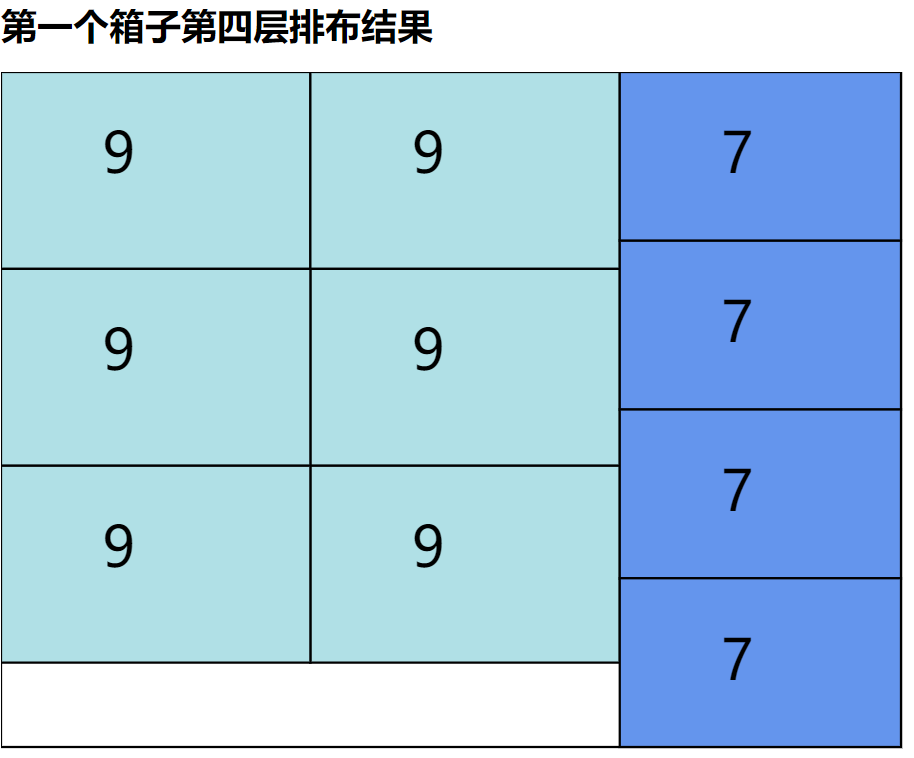


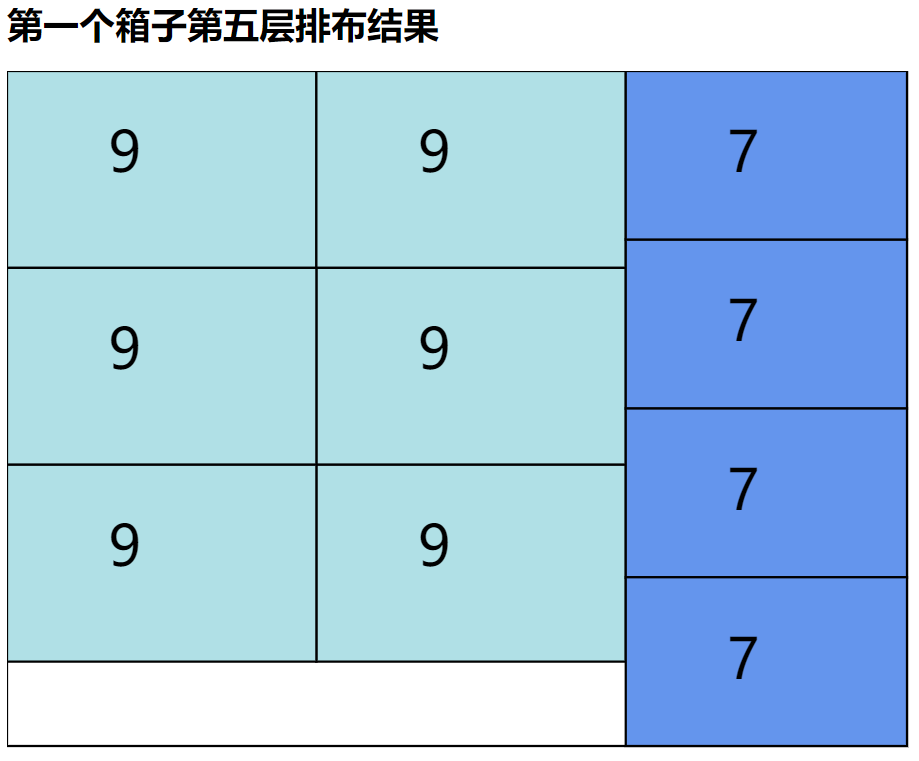
第一个箱子第二层的排布结果：



第一个箱子第三层的排布结果







由于设定了以一层的箱子的最高处作为这一层的箱子的高度可能利用率有损失。

其他箱子的填充依次类推。每个箱子的空间利用率是

箱子1的有效利用率0.773763

箱子2的有效利用率0.773763

箱子3的有效利用率0.773763

箱子4的有效利用率0.884521

箱子5的有效利用率0.3965

// 新增鞋盒的90度旋转方式

箱子 1 的填充率 0.701823

箱子 2 的填充率 0.701823

箱子 3 的填充率 0.695247

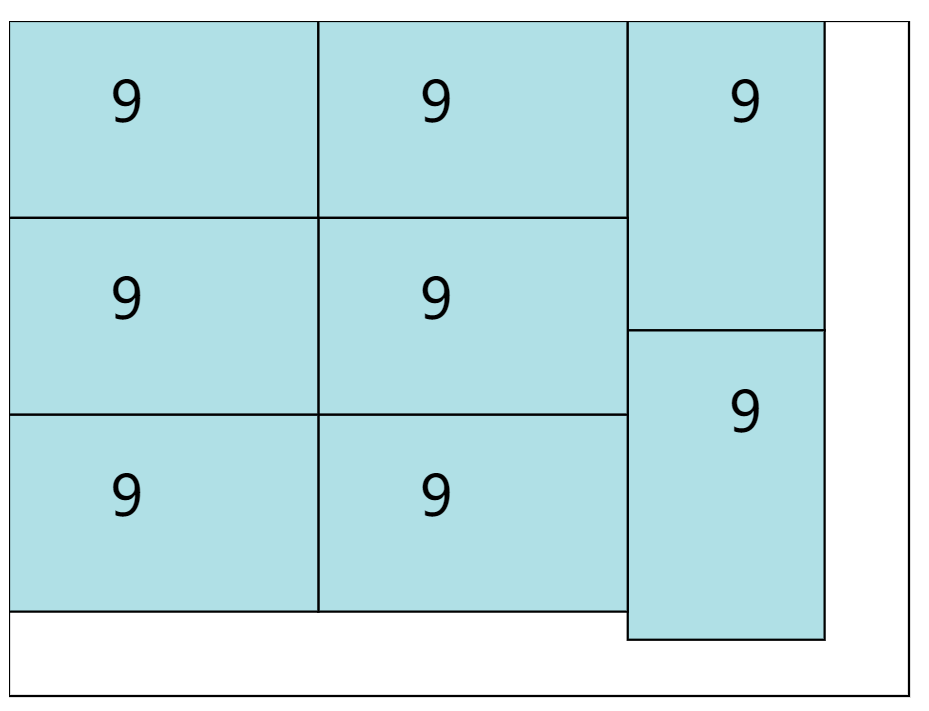
箱子 4 的填充率 0.890625

箱子 5 的填充率 0.612792

并没有提高太多性能

可视化第一个箱子展示图

第一个箱子所有的层都是这样展示



5.4 问题四模型建立与求解

5.4.1 模型建立

将问题4退化为问题3，订制一个统一的箱子，80 \* 55 \* 21（因为最高的两个盒子的高度恰好是这个）经计算每个箱子中不会超过20个盒子。

5.4.2 模型求解

订单1, 包装方案

箱子 0 的填充率 0.875

箱子 1 的填充率 0.875

箱子 2 的填充率 0.875

箱子 3 的填充率 0.875

箱子 4 的填充率 0.875

箱子 5 的填充率 0.875

箱子 6 的填充率 0.752232

箱子 7 的填充率 0.693994

箱子 8 的填充率 0.693994

箱子 9 的填充率 0.693994

箱子 10 的填充率 0.693994

箱子 11 的填充率 0.693994

箱子 12 的填充率 0.684075

箱子 13 的填充率 0.584886

箱子 14 的填充率 0.429545

箱子 15 的填充率 0.0572727

================================利用率============================

订单1的利用率0.701749

第 0 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 0 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 1 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 2 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 3 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 2 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 4 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 5 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 3 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 6 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 7 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 4 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 8 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 9 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 5 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 10 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 11 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 6 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 12 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,25,15],

[0,17.5,25,15],

[0,32.5,25,15],

第 1 层箱子排布方式

选取原层次数中 13 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,25,15],

[52.5,0,25,15],

[52.5,15,25,15],

[52.5,30,25,15],

[27.5,15,25,15],

[27.5,30,25,15],

[0,17.5,25,15],

[0,32.5,25,15],

第 7 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 14 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 1 层箱子排布方式

选取原层次数中 15 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 8 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 16 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 1 层箱子排布方式

选取原层次数中 17 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 9 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 18 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 1 层箱子排布方式

选取原层次数中 19 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 10 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 20 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 1 层箱子排布方式

选取原层次数中 21 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 11 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 22 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 1 层箱子排布方式

选取原层次数中 23 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 12 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 24 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 1 层箱子排布方式

选取原层次数中 25 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,21,14],

第 13 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 26 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,21,14],

[0,15,21,14],

[0,29,21,14],

第 1 层箱子排布方式

选取原层次数中 27 层箱子排布方式

[0,0,25,15],

[25,0,21,14],

[46,0,21,14],

[46,14,21,14],

[46,28,21,14],

[25,14,21,14],

[25,28,21,14],

[0,15,21,14],

[0,29,21,14],

第 14 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 28 层箱子排布方式

[0,0,21,14],

[21,0,21,14],

[42,0,21,14],

[63,0,14,21],

[63,21,14,21],

[42,14,21,14],

[42,28,21,14],

[21,14,21,14],

[21,28,21,14],

第 1 层箱子排布方式

选取原层次数中 29 层箱子排布方式

[0,0,21,14],

[21,0,21,14],

[42,0,21,14],

[63,0,14,21],

[63,21,14,21],

[42,14,21,14],

第 15 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 30 层箱子排布方式

[0,0,21,14],

[21,0,21,14],

订单2，包装方案

箱子 0 的填充率 0.875

箱子 1 的填充率 0.875

箱子 2 的填充率 0.875

箱子 3 的填充率 0.875

箱子 4 的填充率 0.875

箱子 5 的填充率 0.875

箱子 6 的填充率 0.875

箱子 7 的填充率 0.858868

箱子 8 的填充率 0.703835

箱子 9 的填充率 0.693994

箱子 10 的填充率 0.693994

箱子 11 的填充率 0.693994

箱子 12 的填充率 0.693994

箱子 13 的填充率 0.693994

箱子 14 的填充率 0.693994

箱子 15 的填充率 0.594805

箱子 16 的填充率 0.544091

箱子 17 的填充率 0.286364

===============================利用率==============================

订单2的利用率是0.737607

第 0 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 0 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 1 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 2 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 3 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 2 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 4 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 5 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 3 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 6 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 7 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 4 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 8 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 9 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 5 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 10 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 11 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 6 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 12 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 13 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 7 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 14 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

第 1 层箱子排布方式

选取原层次数中 15 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,17.5,27.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,25,15],

第 8 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 16 层箱子排布方式

[0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[55,0,17.5,27.5],

[55,27.5,25,15],

[27.5,17.5,25,15],

[27.5,32.5,25,15],

[0,17.5,25,15],

[0,32.5,25,15],

第 1 层箱子排布方式

选取原层次数中 17 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 9 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 18 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 1 层箱子排布方式

选取原层次数中 19 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 10 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 20 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 1 层箱子排布方式

选取原层次数中 21 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 11 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 22 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 1 层箱子排布方式

选取原层次数中 23 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 12 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 24 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 1 层箱子排布方式

选取原层次数中 25 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 13 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 26 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 1 层箱子排布方式

选取原层次数中 27 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 14 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 28 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 1 层箱子排布方式

选取原层次数中 29 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,25,15],

[0,15,25,15],

[0,30,25,15],

第 15 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 30 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,25,15],

[50,15,25,15],

[50,30,25,15],

[25,15,25,15],

[25,30,21,14],

[0,15,21,14],

[0,29,21,14],

第 1 层箱子排布方式

选取原层次数中 31 层箱子排布方式

[0,0,25,15],

[25,0,25,15],

[50,0,21,14],

[50,14,21,14],

[50,28,21,14],

[25,15,21,14],

[25,29,21,14],

[0,15,21,14],

[0,29,21,14],

第 16 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 32 层箱子排布方式

[0,0,21,14],

[21,0,21,14],

[42,0,21,14],

[63,0,14,21],

[63,21,14,21],

[42,14,21,14],

[42,28,21,14],

[21,14,21,14],

[21,28,21,14],

[0,14,21,14],

第 1 层箱子排布方式

选取原层次数中 33 层箱子排布方式

[0,0,21,14],

[21,0,21,14],

[42,0,21,14],

[63,0,14,21],

[63,21,14,21],

[42,14,21,14],

[42,28,21,14],

[21,14,21,14],

[21,28,21,14],

第 17 个箱子排布方式

第 0 层箱子排布方式

选取原层次数中 34 层箱子排布方式

[0,0,21,14],

[21,0,21,14],

[42,0,21,14],

[63,0,14,21],

[63,21,14,21],

[42,14,21,14],

[42,28,21,14],

第 1 层箱子排布方式

选取原层次数中 35 层箱子排布方式

[0,0,21,14],

[21,0,21,14],

[42,0,21,14],

1. 模型的评价与推广

模型的优点，算法采用纯粹C++编写，运行速度快。

模型的缺点：

在问题3，假设的时候，如果两个层之间有间隔，其实如果充分利用可以进一步加深利用率。

1. 参考文献

[1].数学建模案例分析 白其峥主编 北京 海洋出版社 2000

[2]. 数学建模的原理与方法 蔡锁章主编 北京 海洋出版社 2000

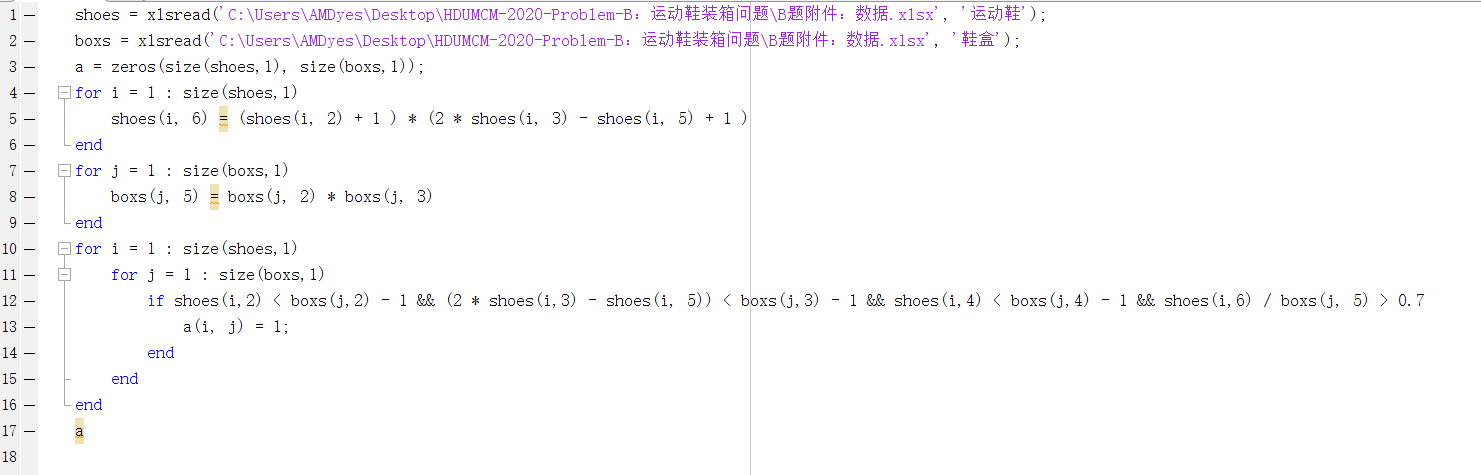
[3].建模、变换、优化--结构综合方法新进展，隋允康著，大连理工大学出版社，(1986)

[4].数学建模案例精选 朱道元等编著 北京：科学出版社，2003

[5].数学建模：原理与方法 蔡锁章主编 北京：海洋出版社，2000

1. 附录

附件一：（问题一 Matlab程序）



附件一：（问题二 Matlab程序）

附件一：（问题三 C++程序）

* 二维矩形填充程序

#ifndef \_\_PACKER\_H\_\_

#define \_\_PACKER\_H\_\_

#include <vector>

#include "packer.h"

struct floors {

double length;

double height;

double width;

double allVolume;//总体积

double validVolume;//有效的体积

int index; // 第几层

floors(double \_length, double \_width, double \_height, double \_allVolume, double \_validVolume, int \_index) {

length = \_length;

width = \_width;

height = \_height;

allVolume = \_allVolume;

validVolume = \_validVolume;

index = \_index;

}

};

struct Node {

double x;

double y;

double width;

double height;

bool isUsed;

Node \* right;

Node \* down;

Node(double \_x, double \_y, double \_width, double \_height) {

x = \_x;

y = \_y;

width = \_width;

height = \_height;

right = NULL;

down = NULL;

isUsed = false;

}

Node(bool \_isUsed, double \_x, double \_y, double \_width, double \_height, Node \* \_down, Node \* \_right) {

isUsed = \_isUsed;

x = \_x;

y = \_y;

width = \_width;

height = \_height;

down = \_down;

right = \_right;

}

};

struct info {

double x;

double y;

double w;

double h;

info(double \_x, double \_y, double \_w, double \_h) {

x = \_x;

y = \_y;

w = \_w;

h = \_h;

}

};

struct block {

double l;// 高度 注意 注意有所不同

double w;// 长度

double h;// 宽度

Node \* fit;

info f;

block(double \_l, double \_w, double \_h) :f(0, 0, 0, 0) {

l = \_l;

w = \_w;

h = \_h;

fit = NULL;

}

public:

double volume() {

return l \* w \* h;

}

};

struct kkey {

double a;

double b;

double width;

double height;

kkey(double \_a, double \_b, double \_c, double \_d) {

a = \_a;

b = \_b;

width = \_c;

height = \_d;

}

};

class Packer { // 二维矩形

public:

double width;

double height;

double x;

double y;

std::vector<kkey> sss;

Packer(double x\_, double y\_, double width\_, double height\_);

Packer() {};

void setPacker(double x\_, double y\_, double width\_, double height\_);

~Packer() {};

void fit(std::vector<block> &blocks);

Node \* findNode(Node \* node, double width, double height);

Node \* splitNode(Node \* node, double w, double h);

Node \* growNode(double w, double h);

Node \* growRight(double w, double h);

Node \* growDown(double w, double h);

bool isOverlap(kkey a, kkey b);

Node \* root;

};

#endif

* 矩形分割Cpp文件

#include "packer.h"

#include <iostream>

const double EPS = 0.0001;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@brief : 构造函数

@author : lee

@input : none

@output : none

@time : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Packer::Packer(double x\_, double y\_, double width\_, double height\_) {

x = x\_;

y = y\_;

width = width\_;

height = height\_;

root = new Node(x, y, width, height);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@brief : 由于没有初始化造成的措施

@author : lee

@input : none

@output : none

@time : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Packer::setPacker(double x\_, double y\_, double width\_, double height\_) {

x = x\_;

y = y\_;

width = width\_;

height = height\_;

root = new Node(x, y, width, height);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@brief : 填充算法

@author : lee

@input : none

@output : none

@time : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Packer::fit(std::vector<block> &blocks) {

int len = blocks.size();

//double width = len > 0 ? blocks[0].w : 0;

//double height = len > 0 ? blocks[0].h : 0;

//root = new Node(0, 0, width, height);

for (int i = 0; i < blocks.size(); i++) {

block &b = blocks[i];

Node \* tmp = findNode(root, blocks[i].w, blocks[i].h);

if (tmp != NULL) {

b.fit = splitNode(tmp, b.w, b.h);

bool canAdd = true;

for (int i = 0; i < sss.size(); i++) {

if (isOverlap({ b.fit->x, b.fit->y, b.w, b.h }, sss[i])) {

canAdd = false;

break;

}

}

if (canAdd) {

sss.push\_back({ b.fit->x , b.fit->y, b.w, b.h });

b.f = { b.fit->x , b.fit->y, b.w, b.h };

//std::cout << b.fit->x << " " << b.fit->y << " " << b.w << " " << b.h << std::endl;

}

}

/\*else {

b.fit = growNode(b.w, b.h);

}\*/

}

std::cout << "一层拟合结束" << std::endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@brief : 寻找节点

@author : lee

@input : none

@output : none

@time : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Node \* Packer::findNode(Node \* root, double width, double height) {

if (root->isUsed) {

Node \* tmpR = findNode(root->right, width, height);

if (tmpR)

return tmpR;

else {

tmpR = findNode(root->down, width, height);

return tmpR;

}

}

else if ((width <= root->width) && (height <= root->height)) {

return root;

}

else

return NULL;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@brief : 节点分离

@author : lee

@input : none

@output : none

@time : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Node \* Packer::splitNode(Node \* node, double w, double h) {

node->isUsed = true;

node->down = new Node(node->x, node->y + h, node->width, node->height - h);

node->right = new Node(node->x + w, node->y, node->width - w, node->height);

return node;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@brief : 矩形重合判断

@author : lee

@input : none

@output : none

@time : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool Packer::isOverlap(kkey rect1, kkey rect2) {

double startX1 = rect1.a;

double startY1 = rect1.b;

double endX1 = startX1 + rect1.width;

double endY1 = startY1 + rect1.height;

double startX2 = rect2.a;

double startY2 = rect2.b;

double endX2 = startX2 + rect2.width;

double endY2 = startY2 + rect2.height;

return !(endY2 < (startY1 + EPS)|| endY1 <( startY2 + EPS) || (startX1 + EPS) > endX2 || (startX2 + EPS) > endX1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@brief : 成长方向

@author : lee

@input : none

@output : none

@time : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Node \* Packer::growNode(double w, double h) {

bool canGrowDown = (w <= root->width);

bool canGrowRight = (h <= root->height);

bool shouldGrowRight = canGrowRight && (root->height >= (root->width + w));

bool shouldGrowDown = canGrowDown && (root->width >= (root->height + h));

if (shouldGrowRight) {

return growRight(w, h);

}

else if (shouldGrowDown) {

return growDown(w, h);

}

else if (canGrowRight) {

return growRight(w, h);

}

else if (canGrowDown) {

return growDown(w, h);

}

else {

return NULL;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@brief : 向右成长

@author : lee

@input : none

@output : none

@time : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Node \* Packer::growRight(double w, double h) {

Node \* tmp = new Node(root->width, 0, w, root->height);

root = new Node(true, 0, 0, root->width + w, root->height, root, tmp);

Node \* node = findNode(root, w, h);

if (node != NULL) {

return splitNode(node, w, h);

}

else {

return NULL;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@brief : 向下成长

@author : lee

@input : none

@output : none

@time : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Node \* Packer::growDown(double w, double h) {

Node \* tmp = new Node(0, root->height, root->width, h);

root = new Node(true, 0, 0, root->width, root->height + h, tmp, root);

Node \* node = findNode(root, w, h);

if (node != NULL) {

return splitNode(node, w, h);

}

else {

return NULL;

}

}

* 解决问题三核心算法流程

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@brief :

1. 思路1 计算问题三 根据订单1得到所用的包装箱数量最少 采用回溯法 可能要采用遗传算法

2. 思路2 计算问题三 可以采用 先构成一层 选这一层中最大的元素 作为这一层的高度

只要尽可能的将一层的面积最大化就好了

然后构成许多层，遍历这么多层选择剩余空间做小的贪心算法 构成这一个箱子

@author : lee

@input ：none

@output ：none

@time : none

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int Solution::solveQuestion3() {

// 1. 选方案 因为 有多个鞋子方案

for (int i = 0; i < answer2.size(); i++) {

std::vector<int> scheme;

std::vector<int> schemeNumber;

std::vector<block> blocks; // 盒子数组

for (int j = 0; j < answer2[i].size(); j++) {

scheme.push\_back(answer2[i][j]); // 先暂时不遍历所有的鞋子可以选择的鞋盒方案 从小号鞋盒开始选择

}

// 先暂时不遍历所有的鞋子可以选择的鞋盒方案 从小号鞋盒开始选择

for (auto it = order.begin(); it != order.end(); it++) {

int number = it->second;

for (auto iit = answer1[it->first].begin(); iit != answer1[it->first].end(); iit++) {

auto findRlt = std::find(scheme.begin(), scheme.end(), \*iit);

if (findRlt == scheme.end()) {

// std::cout << "有部分错误" << it->first << "鞋子所选择的对应的鞋盒" << \*iit << std::endl;

}

else {

std::cout << it->first << "鞋子所选择的对应的鞋盒" << \*findRlt << "个数"<< number<< std::endl;

// 将生成的鞋盒种类和个数加入blocks

for (int i = 0; i < number; i++) {

blocks.push\_back({ boxes[\*findRlt - 1].height, boxes[\*findRlt - 1].length, boxes[\*findRlt - 1].width });

}

break;

}

}

}

// 2. 生成许多“层”

bigBoxes.clear();

std::sort(blocks.begin(), blocks.end(), cmp);

std::vector<floors> v; // 单纯存储一层的长宽高

std::vector<std::vector<block>> floorInfo; // 存储一层的矩形排布

while (blocks.size()) {

Packer packer;

packer.setPacker(0, 0, 80, 60);

packer.fit(blocks);

// 将所有的有效信息存储到 floorInfo

std::vector<block> oneFloorInfo;

for (int i = 0; i < blocks.size(); i++) {

if (blocks[i].f.x != 0 || blocks[i].f.y != 0 || blocks[i].f.w != 0 || blocks[i].f.h != 0) {

oneFloorInfo.push\_back(blocks[i]);

}

}

floorInfo.push\_back(oneFloorInfo);

std::vector<block> tmpBlock;

// 删除所有有有效信息的block

for (int i = 0; i < blocks.size(); i++) {

if (blocks[i].f.x != 0 || blocks[i].f.y != 0 || blocks[i].f.w != 0 || blocks[i].f.h != 0) {

}

else {

tmpBlock.push\_back(blocks[i]);

}

}

blocks.clear();

for (int i = 0; i < tmpBlock.size(); i++) {

blocks.push\_back(tmpBlock[i]);

}

}

// 提取 每一层的核心信息

for (int i = 0; i < floorInfo.size(); i++) {

double maxHight = 0; // 每层的最高距离

double validVolume = 0;

for (int j = 0; j < floorInfo[i].size(); j++) {

// 存储一定的block信息

validVolume += floorInfo[i][j].volume();

if (maxHight < floorInfo[i][j].l) {

maxHight = floorInfo[i][j].l;

}

}

v.push\_back({ 80, 60, maxHight, 80 \* 60 \* maxHight, validVolume, i});

std::cout << "this floor" << maxHight << " allVolume " << 80 \* 60 \* maxHight << " validVolume " << validVolume << " rate " << validVolume / (80 \* 60 \* maxHight) << std::endl;

}

int index = 0;

//

// 先按照顺序填充箱子

while (v.size()) {

BigBox bb(80, 60, 60);

// 遍历一遍填充箱子

std::vector<floors> vv; // 备份

for (int i = 0; i < v.size(); i++) {

if (bb.addFloor(v[i])) {

}

else {

vv.push\_back(v[i]);

}

}

v.clear();

for (int i = 0; i < vv.size(); i++) {

v.push\_back(vv[i]);

}

bb.validPercentageVolume();// 计算有效利用率

index++;

std::cout << "[debug] " << index << std::endl;

std::cout << "箱子的有效利用率[debug]" << bb.validPercentage << std::endl;

bigBoxes.push\_back(bb);

}

// 计算输出每个箱子的填充率

for (int i = 0; i < bigBoxes.size(); i++) {

std::cout << "箱子 " << i << " 的填充率 " << bigBoxes[i].validPercentage << std::endl;

}

for (int i = 0; i < bigBoxes.size(); i++) {

std::cout << "第 " << i << " 个箱子排布方式" << std::endl;

for (int j = 0; j < bigBoxes[i].f.size(); j++) {

std::cout << "第 " << j << " 层箱子排布方式" << std::endl;

std::cout << "选取原层次数中 " << bigBoxes[i].f[j].index << " 层箱子排布方式" << std::endl;

for (int k = 0; k < floorInfo[bigBoxes[i].f[j].index].size(); k++) {

std::cout << "[" << floorInfo[bigBoxes[i].f[j].index][k].f.x << ","

<< floorInfo[bigBoxes[i].f[j].index][k].f.y << ","

<< floorInfo[bigBoxes[i].f[j].index][k].f.w << ","

<< floorInfo[bigBoxes[i].f[j].index][k].f.h << "]," << std::endl;

}

}

}

}

// 3. 填充箱子

// a. 计算并输出每个箱子填充率

// b. 输出每个箱子每一层的填充方案 盒子 - 对应的坐标

return 1;

}

附件一：（问题三 html程序）

可视化效果展示程序

<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8">

    <title>网页标题</title>

    <script src = "https://d3js.org/d3.v5.min.js"></script>

</head>

<body>

    <h1>第二个箱子第4层排布结果</h1>

    <svg width='1000' height='1000'></svg>

    <script>

      var color=['#DA70D6', '#6495ED', '#B0E0E6']

      console.log("dddd",d3.select('svg'))

      var g = d3.select('svg').append('g')

      g.append('rect')

          .attr('x', 0)

          .attr('y', 0)

          .attr('width', 800)

          .attr('height', 600)

          .attr('fill', '#FFF')

          .attr('stroke-width', '2')

          .attr('stroke', '#000')

      var data = [

      [0,0,27.5,17.5],

[27.5,0,27.5,17.5],

[27.5,17.5,27.5,17.5],

[27.5,35,27.5,17.5],

[0,17.5,27.5,17.5],

[0,35,27.5,17.5],

[55,0,25,15],

[55,15,25,15],

[55,30,25,15],

[55,45,25,15],

        ]

      let box = []

      for(var i = 0; i<data.length; i++) {

        if(data[i][2] == 21)

          box.push('4')

        if(data[i][2] == 23.5)

          box.push('5')

        if(data[i][2] == 25)

          box.push('7')

        if(data[i][2] == 27.5)

          box.push('9')

      }

      var colorYinShe ={'9':2,'7':1,'4':0}

      for(let i=0; i<data.length; i++){

        g.append('g')

          .append('rect')

          .attr('x', data[i][0]\* 10)

          .attr('y', data[i][1] \* 10)

          .attr('width', data[i][2]\*10)

          .attr('height', data[i][3]\*10)

          .attr('fill', color[colorYinShe[box[i]]])

          .attr('stroke-width', '2')

          .attr('stroke', '#000')

          .each(function(){

            d3.select(this.parentNode)

              .append('text')

              .text(box[i])

              .attr('x', data[i][0]\* 10 + 90)

              .attr('y', data[i][1] \* 10 + 90)

              .attr('fill', 'black')

              .attr('font-size', 50)

          })

      }

    </script>

</body>

</html>

附件一：（问题四 Matlab程序）