五一数学建模竞赛

承　诺　书

我们仔细阅读了五一数学建模竞赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与本队以外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其它公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们愿意承担由此引起的一切后果。

我们授权五一数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

参赛题号（从 A/B/C 中选择一项填写）：A

参赛队号：T20681765246464

参赛组别（研究生、本科、专科、高中）：本科

所属学校（学校全称）：四川大学锦城学院

参赛队员：　队员 1 姓名：杨然

队员 2 姓名：唐国清

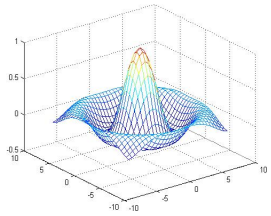
队员 3 姓名：李睿熙

联系方式：　Email：1910895806@qq.com　联系电话：15583206832

日期：　2021　年　5　月　1　日

（除本页外不允许出现学校及个人信息）

五 一 数 学 建 模 竞 赛



疫苗生产问题

关键词：排队论；贪心算法；高斯分布；累加；

摘　要：本文对疫苗生产中遇到一系列问题进行研究，建立相关数学模型推广到现实生活中其他类型加工厂进行生产规划。

针对问题一，将附表一数据按照疫苗类型和工位进行数据分组，计算出各疫苗在工位上加工数据均值、方差、最值、概率分布等数学指标。将其做成图 2、图 3、图 4，并对这些图表等特征进行简要分析，为下问做好铺垫。

针对问题二，根据 10 种类型疫苗在 CJ1-CJ4 工位上加工所需时间的长短不一，疫苗加工顺序控制加工最短时间。在本文中建立数学模型利用贪心算法，每次选择最优解将 10 种类型的疫苗进行排序，结合排队论中的排队思想得到疫苗加工最短时间顺序，具体各疫苗进入和离开工位时间表在文中表 2 体现。

针对问题三，运用高斯分布函数生成每个工位生产每种疫苗的所需时间，这样得到的数据将具有随机性。将这些数据带入问题二中建立的模型求得多条疫苗加工顺序，再在 java 中按照交货时间比问题 2 中的总时间缩短 5%的条件将多条加工顺序数据进行筛选，得到最优加工顺序 YM4-YM1-YM2-YM5-YM7-YM8-YM6-YM10-YM3-YM9，并且在后文中用图像表示出缩短时间比例于与完成概率的关系图，以此得出缩短时间越大则完成概率越小的结论。

针对问题四，同样利用高斯分布函数生成相关数据，并且根据附件 2 中要求的各类型疫苗生产数量进行生产。由于题目要求，可将四个工位加工同一类型疫苗看作一个整体，通过不断累加计算出最后一次空闲时间总值，再套用第二题模型求解。

针对问题五，通过数据分析，发现各工位某时刻加工时间总和 T1、T2、T3、T4，在所有工位运转时如果满足 $T1 \le T2 \le T3 \le T4$ 和 $k = \sqrt[2]{(T2-T1)^2+(T3-T2)^2+(T4-T3)^2}$ 中 k 越小时，销售额更大。

1. 问题重述

1.1 问题背景

自从 2020 年新冠肺炎传染病爆发以来，全世界各国都在为控制疫情努力着。截至今日，印度、美国等国力强厚的国家依旧无法完全控制新冠疫情，还有欧洲等国亦是如此。甚至在近日，印度以单日新增确诊病例报告 31.4 万例创下世界各国单日新增确诊病例最高记录[1]。并且出现突变病株，使其国内疫情变的更加严峻。于是中国自主研发的 2.23 亿剂疫苗分发至世界各国，并且未来还会有更大的疫苗需求量，因此对于疫苗生产工厂生产效率提高成为了当前要务。

1.2 问题重述

1.2.1 问题一

疫苗在生产时将依次按照固定顺序经过 4 个工位加工，且目前有 10 种类型的疫苗待加工。每种类型疫苗以 100 剂为单位，装入加工箱进行加工（每次每个工位处理同类型的 1 箱试剂）。本题将对每个工位上每箱疫苗所需要的生产时间进行均值、方差、最值、概率分布等统计方式来反映各工位生产每种疫苗的能力强弱。

1.2.2 问题二

在问题一的条件下，且取每种试剂在各工位的平均值作为定量。现需要对 10 种类型的疫苗各 100 剂（一加工箱）进行快速检测。由于每种类型的疫苗在每个加工位所需的加工时间都不同，且要求同一个工位不能同时进行不同类型疫苗加工，为了响应检测部门的紧急情况，我们要如何安排疫苗生产的顺序才能在最短时间内全部生产完毕呢？

1.2.3 问题三

在日常生活中很难达到理想状态，对于每个工位生产疫苗的时间其实是具有不稳定性的。如果要求实际交货时间比问题二中的时间缩短 5%，并且以最大概率完成疫苗生产。那么疫苗加工顺序又该如何排序？而缩短时间比例与最大完成概率之间存在着怎样的关系呢？

1.2.4 问题四

每天各工位工作时间在 16 小时内，且每种类型疫苗加工需一次性完成。现在决定将 10 种类型不同的疫苗按照不同的产量目标生产，在 90%的可靠性下，至少需要多少天加工完毕呢？

1.2.5 问题五

在以最大销售额为前提且工位每天生产时间不能超过 16 小时的条件下，公司决定在 100 天内对疫苗进行部分拆分生产，即不一定全部经过 4 个工位加工。在这些条件下，我们应该怎样安排生产计划呢？

2. 问题分析

2.1 问题一分析

本题将对每个工位上每箱疫苗所需要的生产时间进行均值、方差、最值、概率分布等统计方式来反映各工位生产各类型疫苗的能力强弱。

2.2 问题二分析

根据附件 1 提供的数据分析，发现疫苗放入工位加工顺序不同时，每个工位的空闲时间会根据顺序不同而变化。总耗时是服务时间与等待时间的总和，但服务时间是恒定值，因此尽量缩短等待时间即可使总耗时变短（使下一加工类型疫苗的前一工位服务时间与上一加工类型疫苗的后一工位服务时间尽量相等）。根据贪心算法原理

（每次选择最优）[2]，要想得到最短时间内疫苗加工顺序，则要在每次两种疫苗工位加工时间、空闲时间等数据指标比较下，计算出前一种类型疫苗各相邻工位空闲时间的差值与后一种类型疫苗各相邻工位空闲时间的差值，再计算已求两种疫苗对应差值的差值，最后求和，永远选择和最小的那种疫苗先进行加工。

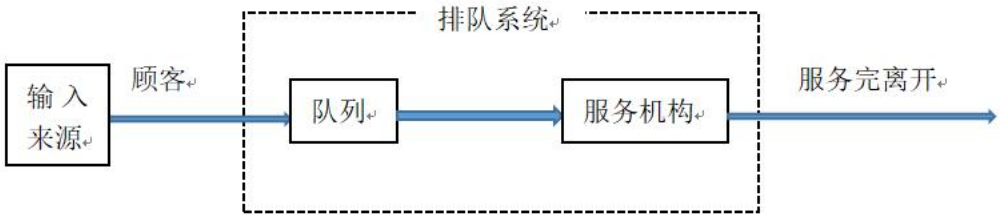


图 1 排队系统结构图

图 1 为排队论[3]的结构图[4]，与本题结合起来看，图中顾客是疫苗（YM1-YM10），服务机构是四个工位(CJ1-CJ4)。在决定疫苗加工顺序时，可以从输入过程、排队规则、服务机构三个部分来排序[5]。

2.3 问题三分析

为得到随机数据，在 matlab 中利用 normrnd　正态（高斯）分布的随机数生成器，normrnd(a,b,c,d)：产生均值为 a、方差为 b 大小为 c*d 的随机矩阵[6]，随机生成符合条件的数据。根据随机数据分析套用问题二中的模型，得出 10 万条疫苗加工顺序并记录此顺序最终所用时间。再利用 java 建立如下数据结构统计并分析[7]，matlab 中输出的随机数据的执行结果。根据缩短时间 5%条件来筛选，最终得到符合本题意的疫苗生产顺序。

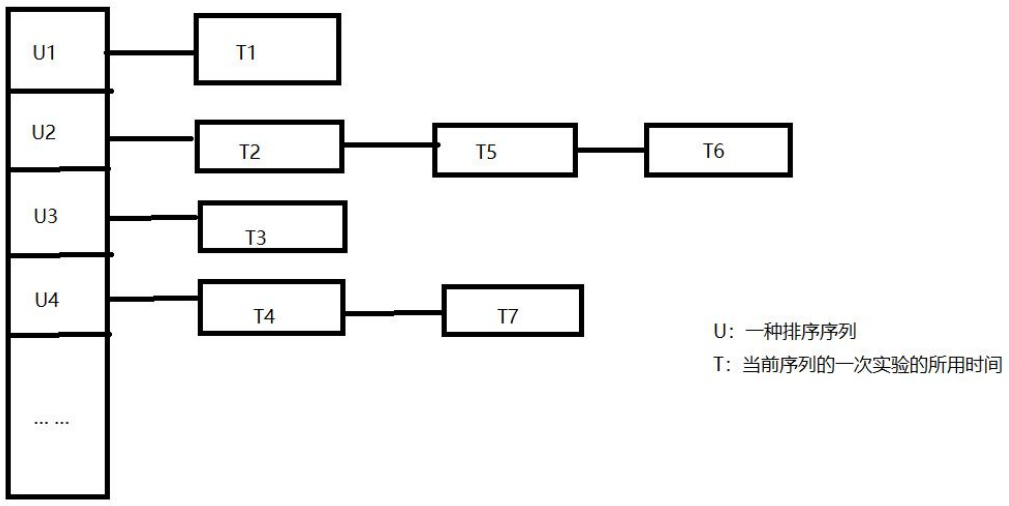


图 2java 数据结构图

在本题第二个小问中，同样将大量模拟疫苗数据进行统计分析，从而画出缩短时间比例与完成概率之间的关系图。

2.4 问题四分析

求出均值、方差等数据后，利用高斯函数生成大量随机数据。由于题目要求每种类型疫苗加工需一次性完成,可以将一种类型的多剂疫苗在 4 个工位加工时间看作一个整体（即通过带权累加出每种疫苗单独在各工位的空闲时间）[8]，将本题简化为十箱不同类型疫苗的排序求解问题，即将各类型疫苗在各工位空闲时间进行带权累加后，再套用问题二中所建立的模型进行求解，多次实验得出完成概率与完成天数

之间的关系。

2.5 问题五分析

在限制生产时间为100天且各类型疫苗可拆分生产的条件下，最大销售额跟每种疫苗的生产数量与销售价格相关。在加工的过程，每个工位处都有等待区，等待区内都有已完成前一工位加工的疫苗。根据每个疫苗各自的加工时间，得出规律：当疫苗进入某个工位时其所需加工时间必须小于此工位等待区的加工时间总和。

3. 模型假设

1. 假设第二、三问中各工位加工机器连续工作无故障。
2. 假设第二、三问中除等待时间与生产时间外其余时间不计。
3. 假设第三问中产生的随机数据范围足够广数量足够大，具有参考性。
4. 假设第四问中，机器在每天16个小时的工作时间内不出现故障，理想工位空闲时间与实际工位空闲时间无误差。

5. 符号说明

表1 符号说明

符号	符号说明
S	剩余9种疫苗标号
U	任意一种疫苗标号
AVG	平均值汇总
i	下标（疫苗类型）
st	定义序列
$Temp_{(sumi)}$	st序列中两个差值的差值的各项累加和
ST	累加后的序列
j	st序列中的下角标

（下文中未提及的符号将及时做出说明）

5. 模型建立

5.1 第二问：最短时间建模

算法分析：

第一步：AVG为十种类型疫苗在四个工位上所需时间平均值的汇总。

第二步：用函数U来存放一组1种疫苗标号，S存放剩余9种疫苗标号。

第三步：定义一个st序列，用来存储四个工位的当前的空闲时间。

第四步：运用贪心算法原理，算出各个工位的时间差，求出最小工位空闲时间差。

$temp_{sumi}=abs[(st_{(i)}(2)-st_{(i)}(1))-(ST_{(i-1)}(3)-ST_{(i-1)}(4))]+abs[(st_{(i)}(3)-st_{(i)}(2))-(ST_{(i-1)}(4)-ST_{(i-1)}(3))]$

第五步：在得到的$temp_{sumi}$中找到最大的那个值，并记录下其i值，即下一个进入工位加工的疫苗类型。

第六步：更新st序列：将st(j)与st(j-1)进行比较，分两种情况讨论：

st(j)=st(j)+AVG(j,i)　　条件：st(j)>st(j-1)

st(j)=st(j-1)+AVG(j,i)　条件：st(j)≤st(j-1)

第七步：将i在S中删除，并放入U中。

第八步：重复以上步骤四到步骤七，直到 S 为空。最终即可得到疫苗加工的最短时间，以及最优加工顺序。

6. 模型求解

6.1 问题 1 均值、方差等求解

对于每箱疫苗在每个工位上所需时间采用求均值、方差、最值等数学手段进行统计，以此反映的各工位的工作水平。

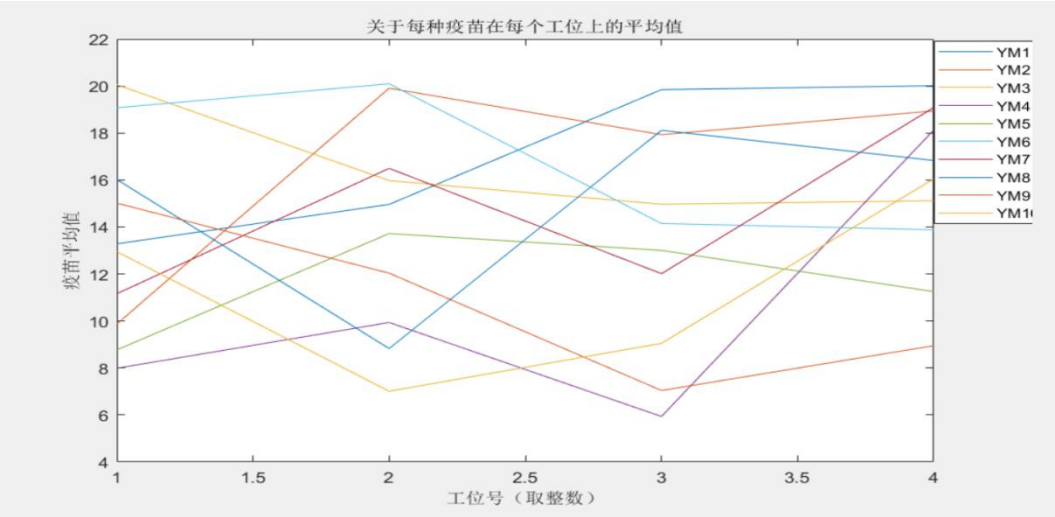


图 3 每种疫苗在每个工位上的平均值

其中据图 3 所知，每种类型的疫苗其加工所需的时间都是不一样的。相对于在 CJ1 时所需的时间来看，每个类型疫苗在 CJ2、CJ3、CJ4 三个工位所需加工时间有所减少或者增加，并且其变化幅度各不相同。从整体来看在不同工位所需时间上下变化较大的是 YM4、YM9 这两种类型疫苗，这提示我们在提高效率的情况下要合理安排各种类型疫苗进行加工的顺序，避免时间浪费。

表 2 每种疫苗在每个工位上的方差

	YM1_var	YM2_var	YM3_var	YM4_var	YM5_var	YM6_var	YM7_var	YM8_var	YM9_var	YM10_var
CJ1	1.5940	0.8333	0.8448	0.9322	0.7451	1.3350	1.0263	1.1317	0.9738	0.2075
CJ2	1.0831	1.2529	0.6934	0.9534	1.1396	0.9735	0.9295	0.2614	1.1581	0.2822
CJ3	1.1875	0.9022	1.0400	0.0380	0.8313	0.8755	0.7862	1.0700	0.1337	0.2074
CJ4	1.8897	0.8905	0.8122	1.1277	1.2308	1.1517	0.6564	0.9128	0.1752	0.2634

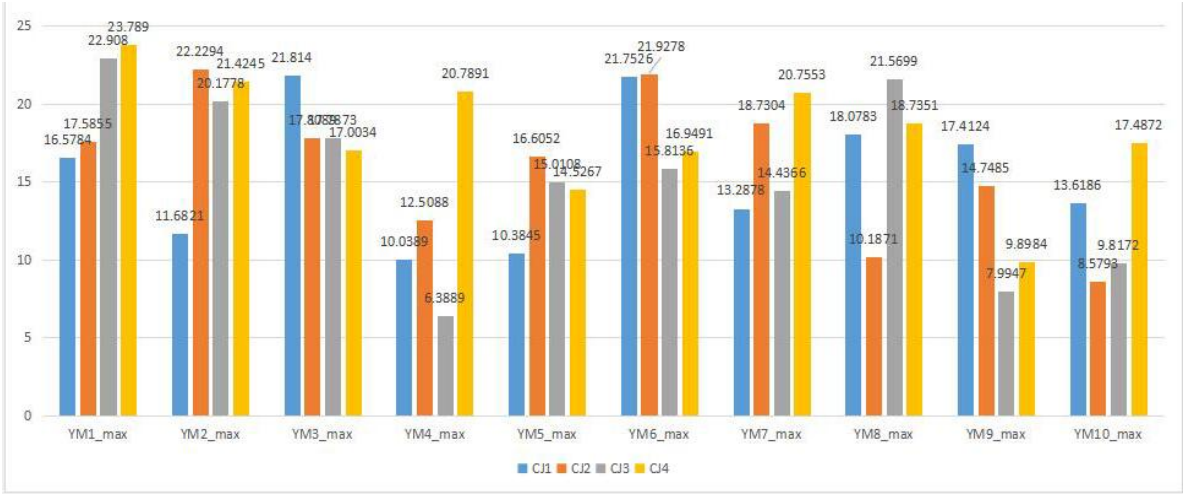


图 4 关于疫苗生产时间最大值比较

图 5 关于疫苗生产时间最小值比较

由图 3 与图 4 所知，在 CJ1、CJ2、CJ3、CJ4 上加工所需时间最值最大的分别是 YM3、YM2、YM1、YM1 这 4 种疫苗，而加工所需时间最值最小的分别是 YM4、YM10、YM4、YM9 疫苗。

6.2 问题 2：最短时间求解

将本题 10 种疫苗类型具体在每个工位加工所需的平均值带入模型中，求得其最短加工时间排序。

表 3 各类型疫苗所需时间均值

	YM1_avg	YM2_avg	YM3_avg	YM4_avg	YM5_avg	YM6_avg	YM7_avg	YM8_avg	YM9_avg	YM10_avg
CJ1	13.2840	9.8709	20.0584	7.9887	8.7701	19.0741	11.1601	16.0201	15.0146	12.9524
CJ2	14.9621	19.9075	15.9726	9.9366	13.7220	20.0944	16.4961	8.8275	12.0351	7.0110
CJ3	19.8460	17.9282	14.9704	5.9359	13.0052	14.1485	12.0137	18.1144	7.0419	9.0492
CJ4	20.0129	18.9424	15.1164	18.1284	11.2495	13.8839	19.0876	16.8314	8.9497	16.0524
总时间	68.1050	66.6490	66.1178	41.9896	46.7468	67.2009	58.7575	59.7934	43.0413	45.0650

表 4 问题 2 的结果

加工顺序（填疫苗编号）	进入 CJ1 时刻	离开 CJ4 时刻
YM4	00:00	00:42
YM1	00:08	01:16
YM2	00:21	01:35
YM5	00:31	01:46
YM8	00:40	02:03
YM7	00:56	02:22
YM10	01:07	02:38
YM6	01:20	02:52
YM3	01:39	03:07
YM9	01:59	03:16

6.3 问题 3 求解

第一个小问求解：

时间缩短 5%左右时，各种类型疫苗的加工顺序为：YM4-YM1-YM2-YM5-YM7-YM8-YM6-YM10-YM3-YM9（4 1 2 5 7 8 6 10 3 9)

第二个小问求解：

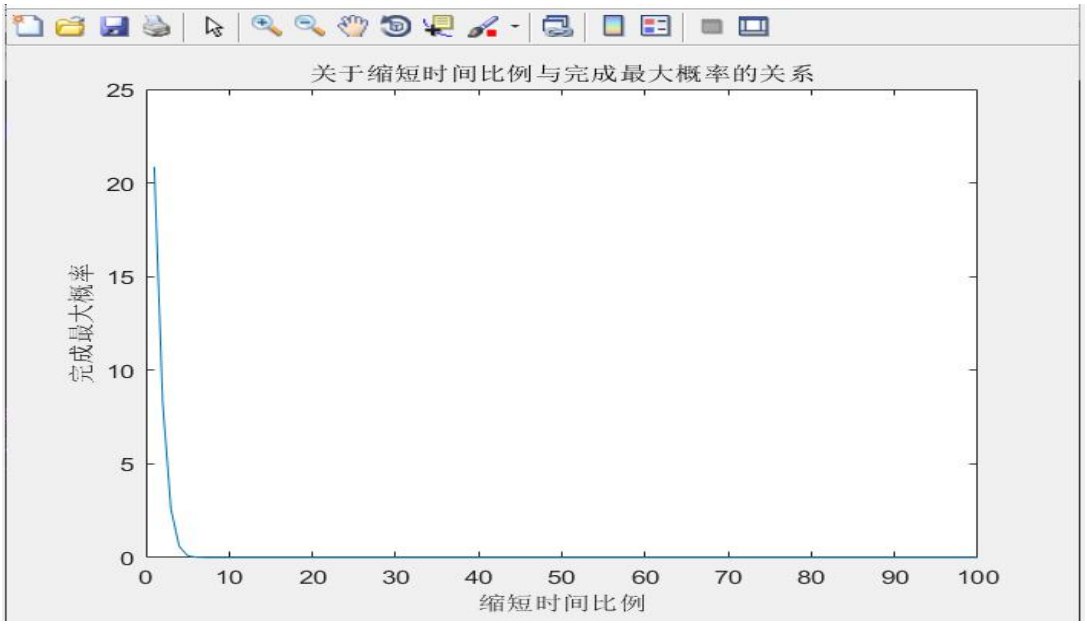


图 6 缩短时间与完成概率关系

根据图像显示可得出结论：当缩短时间越大则完成概率越小

6.4 问题 4 求解

结论:在可靠性为 90%且每个工位生产时间不超过 16 个小时的条件下,至少需要 226 天才可以完成任务。

6.5 问题 5 求解

设各工位某时刻加工时间总和为：T1、T2、T3、T4，当所有工位运转后满足条件：T1≤T2≤T3≤T4

各工位间存在的运算关系为 $k = \sqrt[2]{(T2-T1)^2+(T3-T2)^2+(T4-T3)^2}$，当 k 的值越小时，各工位任务间等待时间越短。

7.模型评价及改进

7.1 模型的优点

在第二问中建立的模型对贪心算法进行改良，结和排队论的相关知识，是模型具备运转更加高效的优点，时间复杂度为 0（$nLog^n$）[9]。本文提出的模型，提高了贪心算法的可用性，扩大其使用范围[7]。

7.2 模型的缺点

利用本模型求解有一定的误差，在实际监测中，每个工位的数据观测有误差，即本题使用的每种类型疫苗在各工位加工时间与实际加工时间有一定的误差。并且在模型建立过程中，有模型误差。模型使用的各工位加工时间数据均使用的 50 次模拟数据均值，跟实际值间存在误差。在开始进行数据清理时，数据存在一定的偏差，无法进行有效规避。

7.3 模型改进

在第二个问中利用贪心算法构建的模型，仅能求出当下情况的局部最优解[10]。所以还需寻找更加强大的算法来得出全局的最优解，是本文所建模型改进的方向。

如果能够通过每种疫苗在各工位的运行时间推算出一定的规律，那么即可使模型变得更加有效率。

8.模型推广及应用

本模型可以适用于普遍加工计划安排，例如口罩厂加工，机械厂加工，食品加工等各类工厂。并且根据各个工厂的机器运转、环境等实际情况，更改相应参数进行目标计划产品加工顺序规划。这样有利于提高工厂加工效率和产品质量，并且创造最大利润值。

9.参考文献

[1]形势紧迫:印度疫情全球第二严重[J].课堂内外(作文独唱团),2020(11):10.

[2]阮玉生,屈慧洁.基于贪心算法的通信网络能耗均衡分簇路由优化方法[J].科学技术创新,2021(10):118-119.

[3][ISBN 978-7-302-44576-0]李工农.运筹学基础及其 MATLAB 应用.北京：清华大学出版社，2016.

[4]吴希.基于边际分析法的银行窗口数量最优化设计[J].中国集体经济,2016(33):83-84.

[5]石芳玉. 基于排队论的银行网点顾客排队优化管理研究[D].成都理工大学,2016.

[6]张志军,刘行兵,段新涛.高斯随机数生成算法对比研究[J].河南科技学院学报(自然科学版),2014,42(03):56-59.

[7]陈衡,刘玉文.基于二维数组和十字链表的 Apriori 算法[J].德州学院学报,2017,33(02):63-67

[8] 李俊杰,吴根秀,焦黎洋.一种基于证据焦元距离的权重计算方法[J].江西师范大学学报(自然科学版),2020,44(03):275-281.

[9]傅晓航，郑欢欢.k 分搜索的时间复杂度分析[J].计算机技术与发展,2021,31(02):175-179.

[10]杨卓林,吕宜光,曹灿.贪心算法在科研仪器共享系统中的应用[J].自动化技术与应用,2020,39(11):169-173.

10. 附录

附录 1：第一题代码

```
clear;
clc;
D = xlsread('C:\Users\86187\Desktop\建模提交\分隔后表.xls',1,'C2:AZ41');
% 数据导入
YM1 = D(1:4,:);
YM2 = D(5:8,:);
YM3 = D(9:12,:);
YM4 = D(13:16,:);
YM5 = D(17:20,:);
YM6 = D(21:24,:);
YM7 = D(25:28,:);
YM8 = D(29:32,:);
YM9 = D(33:36,:);
YM10 = D(37:40,:);
%求均值
YM1_avg = mean(YM1,2)
YM2_avg = mean(YM2,2)
YM3_avg = mean(YM3,2)
YM4_avg = mean(YM4,2)
YM5_avg = mean(YM5,2)
YM6_avg = mean(YM6,2)
YM7_avg = mean(YM7,2)
YM8_avg = mean(YM8,2)
YM9_avg = mean(YM9,2)
YM10_avg = mean(YM10,2)
%求方差(标准差平方)
YM1_var = std(YM1,0,2).^2
YM2_var = std(YM2,0,2).^2
YM3_var = std(YM3,0,2).^2
YM4_var = std(YM4,0,2).^2
YM5_var = std(YM5,0,2).^2
YM6_var = std(YM6,0,2).^2
YM7_var = std(YM7,0,2).^2
YM8_var = std(YM8,0,2).^2
YM9_var = std(YM9,0,2).^2
YM10_var = std(YM10,0,2).^2
%求最大值
YM1_max = max(YM1,[],2)
YM2_max = max(YM2,[],2)
YM3_max = max(YM3,[],2)
YM4_max = max(YM4,[],2)
```

```
YM5_max = max(YM5,[],2)
YM6_max = max(YM6,[],2)
YM7_max = max(YM7,[],2)
YM8_max = max(YM8,[],2)
YM9_max = max(YM9,[],2)
YM10_max = max(YM10,[],2)
%求最小值
YM1_min = min(YM1,[],2)
YM2_min = min(YM2,[],2)
YM3_min = min(YM3,[],2)
YM4_min = min(YM4,[],2)
YM5_min = min(YM5,[],2)
YM6_min = min(YM6,[],2)
YM7_min = min(YM7,[],2)
YM8_min = min(YM8,[],2)
YM9_min = min(YM9,[],2)
YM10_min = min(YM10,[],2)
%概率分布(均值总图)
t = 1:1:4;
plot(t,YM1_avg);
hold on;
plot(t,YM2_avg);
hold on;
plot(t,YM3_avg);
hold on;
plot(t,YM4_avg);
hold on;
plot(t,YM5_avg);
hold on;
plot(t,YM6_avg);
hold on;
plot(t,YM7_avg);
hold on;
plot(t,YM8_avg);
hold on;
plot(t,YM9_avg);
hold on;
plot(t,YM10_avg);
xlabel('工位号（取整数）');
ylabel('疫苗平均值');
title('关于每种疫苗在每个工位上的平均值');
legend('YM1','YM2','YM3','YM4','YM5','YM6','YM7','YM8','YM9','YM10');
%概率分布（每一箱疫苗在每一个工位的概率分布图）
figure;
```

```
hist(YM1(1,:),30);
figure;
hist(YM1(2,:),30);
figure;
hist(YM1(3,:),30);
figure;
hist(YM1(4,:),30);
figure;
hist(YM2(1,:),30);
figure;
hist(YM2(2,:),30);
figure;
hist(YM2(3,:),30);
figure;
hist(YM2(4,:),30);
figure;
hist(YM3(1,:),30);
figure;
hist(YM3(2,:),30);
figure;
hist(YM3(3,:),30);
figure;
hist(YM3(4,:),30);
figure;
hist(YM4(1,:),30);
figure;
hist(YM4(2,:),30);
figure;
hist(YM4(3,:),30);
figure;
hist(YM4(4,:),30);
figure;
hist(YM5(1,:),30);
figure;
hist(YM5(2,:),30);
figure;
hist(YM5(3,:),30);
figure;
hist(YM5(4,:),30);
figure;
hist(YM6(1,:),30);
figure;
hist(YM6(2,:),30);
figure;
```

```
hist(YM6(3,:),30);
figure;
hist(YM6(4,:),30);
figure;
hist(YM7(1,:),30);
figure;
hist(YM7(2,:),30);
figure;
hist(YM7(3,:),30);
figure;
hist(YM7(4,:),30);
figure;
hist(YM8(1,:),30);
figure;
hist(YM8(2,:),30);
figure;
hist(YM8(3,:),30);
figure;
hist(YM8(4,:),30);
figure;
hist(YM9(1,:),30);
figure;
hist(YM9(2,:),30);
figure;
hist(YM9(3,:),30);
figure;
hist(YM9(4,:),30);
figure;
hist(YM10(1,:),30);
figure;
hist(YM10(2,:),30);
figure;
hist(YM10(3,:),30);
figure;
hist(YM10(4,:),30);
```

第二题模型建立代码：

```
clear;
clc;
D = xlsread('.\分隔后表.xlsx',1,'C2:AZ41');
% 数据导入
YM1 = D(1:4,:);
YM2 = D(5:8,:);
YM3 = D(9:12,:);
YM4 = D(13:16,:);
```

```
YM5 = D(17:20,:);
YM6 = D(21:24,:);
YM7 = D(25:28,:);
YM8 = D(29:32,:);
YM9 = D(33:36,:);
YM10 = D(37:40,:);
%求均值
YM1_avg = mean(YM1,2);
YM2_avg = mean(YM2,2);
YM3_avg = mean(YM3,2);
YM4_avg = mean(YM4,2);
YM5_avg = mean(YM5,2);
YM6_avg = mean(YM6,2);
YM7_avg = mean(YM7,2);
YM8_avg = mean(YM8,2);
YM9_avg = mean(YM9,2);
YM10_avg = mean(YM10,2);
%汇总平均值 AVG
AVG = ones(4,10);
for i = 1:4
   AVG(i,1) = YM1_avg(i);
end
for i = 1:4
   AVG(i,2) = YM2_avg(i);
end
for i = 1:4
   AVG(i,3) = YM3_avg(i);
end
for i = 1:4
   AVG(i,4) = YM4_avg(i);
end
for i = 1:4
   AVG(i,5) = YM5_avg(i);
end
for i = 1:4
   AVG(i,6) = YM6_avg(i);
end
for i = 1:4
   AVG(i,7) = YM7_avg(i);
end
for i = 1:4
   AVG(i,8) = YM8_avg(i);
end
for i = 1:4
```

```
   AVG(i,9) = YM9_avg(i);
end
for i = 1:4
   AVG(i,10) = YM10_avg(i);
end
AVG
Time_true = 999999999;
U_true = ones(1,10);
for first = 1:10
    % 定义已入队列U，未入队列S。
    U = ones(1,1);
    S = ones(1,9);
    U(1) = first;
    for i = 1:10
        S(i) = i;
    end
    S(first) = [];
    %定义一个st序列并初始化，存储四个工位的当前的空闲时间。
    st = ones(1,4);
    st(1) = AVG(1,first);
    for i = 2:4
        st(i) = st(i-1) + AVG(i,first);
    end
    st
    min = ones(1,2); %[S中下标，差值和]
    while ~isempty(S)
       min(1) = 1;
       if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2))
           min(2) = 0;
       end
       if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) <= 0
           min(2) = abs((AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)));
       end
       if (AVG(3,S(1)) - AVG(2,S(1)))-(st(4)-st(3)) <= 0
           min(2)      =      min(2)      +      abs((AVG(3,S(1))      -
AVG(2,S(1)))-(st(4)-st(3)));
       end
       for i = 2:length(S)
           temp_sum = zeros(1,1);
           if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) > 0
               temp_sum = 0;
           end
           if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) <= 0
               temp_sum = abs((AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)));
```

```
            end
            if (AVG(3,S(1)) - AVG(2,S(1)))-(st(4)-st(3)) <= 0
                temp_sum    =    temp_sum    +    abs((AVG(3,S(1))    -
AVG(2,S(1)))-(st(4)-st(3)));
            end
            if temp_sum < min(2)
                min(1) = i;
                min(2) = temp_sum;
            end
        end
        st(1) = st(1) + AVG(1,S(min(1)));
        for i = 2:4
            if st(i-1) < st(i)
                st(i) = st(i) + AVG(i,S(min(1)));
            end
            if st(i-1) >= st(i)
                st(i) = st(i-1) + AVG(i,S(min(1)));
            end
                   end

        U(length(U) + 1) = S(min(1));
        S(min(1)) = [];
        st
    end
    U
    if st(4) < Time_true
        Time_true = st(4);
        for i = 1:10
                        U_true(i) = U(i);

        end
    end
end
U_true,Time_true
```

第三题第一个小问：

```
clear;
clc;
D = xlsread('.\分隔后表.xlsx',1,'C2:AZ41');
% 数据导入
YM1 = D(1:4,:);
YM2 = D(5:8,:);
YM3 = D(9:12,:);
YM4 = D(13:16,:);
YM5 = D(17:20,:);
```

```
YM6 = D(21:24,:);
YM7 = D(25:28,:);
YM8 = D(29:32,:);
YM9 = D(33:36,:);
YM10 = D(37:40,:);
%求均值
YM1_avg_t = mean(YM1,2);
YM2_avg_t = mean(YM2,2);
YM3_avg_t = mean(YM3,2);
YM4_avg_t = mean(YM4,2);
YM5_avg_t = mean(YM5,2);
YM6_avg_t = mean(YM6,2);
YM7_avg_t = mean(YM7,2);
YM8_avg_t = mean(YM8,2);
YM9_avg_t = mean(YM9,2);
YM10_avg_t = mean(YM10,2);
%求方差(标准差平方)
YM1_var = std(YM1,0,2).^2
YM2_var = std(YM2,0,2).^2;
YM3_var = std(YM3,0,2).^2;
YM4_var = std(YM4,0,2).^2;
YM5_var = std(YM5,0,2).^2;
YM6_var = std(YM6,0,2).^2;
YM7_var = std(YM7,0,2).^2;
YM8_var = std(YM8,0,2).^2;
YM9_var = std(YM9,0,2).^2;
YM10_var = std(YM10,0,2).^2;
for cnt = 1:10000
    % 模拟数据，随机均值数
    YM1_avg = normrnd(YM1_avg_t,YM1_var,4,1);
    YM2_avg = normrnd(YM2_avg_t,YM2_var,4,1);
    YM3_avg = normrnd(YM3_avg_t,YM3_var,4,1);
    YM4_avg = normrnd(YM4_avg_t,YM4_var,4,1);
    YM5_avg = normrnd(YM5_avg_t,YM5_var,4,1)
    YM6_avg = normrnd(YM6_avg_t,YM6_var,4,1);
    YM7_avg = normrnd(YM7_avg_t,YM7_var,4,1);
    YM8_avg = normrnd(YM8_avg_t,YM8_var,4,1);
    YM9_avg = normrnd(YM9_avg_t,YM9_var,4,1);
    YM10_avg = normrnd(YM10_avg_t,YM10_var,4,1);
    %汇总平均值 AVG
    AVG = ones(4,10);
    for i = 1:4
       AVG(i,1) = YM1_avg(i);
    end
```

```
for i = 1:4
   AVG(i,2) = YM2_avg(i);
end
for i = 1:4
   AVG(i,3) = YM3_avg(i);
end
for i = 1:4
   AVG(i,4) = YM4_avg(i);
end
for i = 1:4
   AVG(i,5) = YM5_avg(i);
end
for i = 1:4
   AVG(i,6) = YM6_avg(i);
end
for i = 1:4
   AVG(i,7) = YM7_avg(i);
end
for i = 1:4
   AVG(i,8) = YM8_avg(i);
end
for i = 1:4
   AVG(i,9) = YM9_avg(i);
end
for i = 1:4
   AVG(i,10) = YM10_avg(i);
end
%循环计算下一个进入的元素，直到 S 集合为空。
Time_true = 999999999;
U_true = ones(1,10);
for first = 1:10
    % 定义已入队列 U，未入队列 S。
    U = ones(1,1);
    S = ones(1,9);
    U(1) = first;
    for i = 1:10
        S(i) = i;
    end
    S(first) = [];
    %定义一个 st 序列并初始化，存储四个工位的当前的空闲时间。
    st = ones(1,4);
    st(1) = AVG(1,first);
    for i = 2:4
        st(i) = st(i-1) + AVG(i,first);
```

```
        end
        min = ones(1,2); %[S 中下标，差值和]
        while ~isempty(S)
           min(1) = 1;
           if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) > 0
               min(2) = 0;
           end
           if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) <= 0
               min(2) = abs((AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)));
           end
           if (AVG(3,S(1)) - AVG(2,S(1)))-(st(4)-st(3)) <= 0
               min(2)     =     min(2)     +     abs((AVG(3,S(1))     -
AVG(2,S(1)))-(st(4)-st(3)));
           end
           for i = 2:length(S)
               temp_sum = zeros(1,1);
               if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) > 0
                   temp_sum = 0;
               end
               if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) <= 0
                   temp_sum         =         abs((AVG(2,S(1))         -
AVG(1,S(1)))-(st(3)-st(2)));
               end
               if (AVG(3,S(1)) - AVG(2,S(1)))-(st(4)-st(3)) <= 0
                   temp_sum    =    temp_sum    +    abs((AVG(3,S(1))    -
AVG(2,S(1)))-(st(4)-st(3)));
               end
               if temp_sum < min(2)
                   min(1) = i;
                   min(2) = temp_sum;
               end
           end
           st(1) = st(1) + AVG(1,S(min(1)));
           for i = 2:4
               if st(i-1) < st(i)
                   st(i) = st(i) + AVG(i,S(min(1)));
               end
               if st(i-1) >= st(i)
                   st(i) = st(i-1) + AVG(i,S(min(1)));
               end
           end
           U(length(U) + 1) = S(min(1));
           S(min(1)) = [];
        end
```

```
        if st(4) < Time_true
            Time_true = st(4);
            for i = 1:10
                U_true(i) = U(i);
            end
        end
    end
    Time_true
    U_true
end
```

第三问第二个小问：

```
clear;
clc;
D = xlsread('.\分隔后表.xlsx',1,'C2:AZ41');
% 数据导入
YM1 = D(1:4,:);
YM2 = D(5:8,:);
YM3 = D(9:12,:);
YM4 = D(13:16,:);
YM5 = D(17:20,:);
YM6 = D(21:24,:);
YM7 = D(25:28,:);
YM8 = D(29:32,:);
YM9 = D(33:36,:);
YM10 = D(37:40,:);
%求均值
YM1_avg_t = mean(YM1,2);
YM2_avg_t = mean(YM2,2);
YM3_avg_t = mean(YM3,2);
YM4_avg_t = mean(YM4,2);
YM5_avg_t = mean(YM5,2);
YM6_avg_t = mean(YM6,2);
YM7_avg_t = mean(YM7,2);
YM8_avg_t = mean(YM8,2);
YM9_avg_t = mean(YM9,2);
YM10_avg_t = mean(YM10,2);
%求方差(标准差平方)
YM1_var = std(YM1,0,2).^2;
YM2_var = std(YM2,0,2).^2;
YM3_var = std(YM3,0,2).^2;
YM4_var = std(YM4,0,2).^2;
YM5_var = std(YM5,0,2).^2;
YM6_var = std(YM6,0,2).^2;
YM7_var = std(YM7,0,2).^2;
```

```
YM8_var = std(YM8,0,2).^2;
YM9_var = std(YM9,0,2).^2;
YM10_var = std(YM10,0,2).^2;
% 使用 Time 数组存储符合条件的数据个数，如下标为 i 的数据为缩短 i%能完成的数据个数
Time = zeros(1,100);
for cnt = 1:100000
    % 模拟数据，随机均值数
    YM1_avg = normrnd(YM1_avg_t,YM1_var,4,1);
    YM2_avg = normrnd(YM2_avg_t,YM2_var,4,1);
    YM3_avg = normrnd(YM3_avg_t,YM3_var,4,1);
    YM4_avg = normrnd(YM4_avg_t,YM4_var,4,1);
    YM5_avg = normrnd(YM5_avg_t,YM5_var,4,1);
    YM6_avg = normrnd(YM6_avg_t,YM6_var,4,1);
    YM7_avg = normrnd(YM7_avg_t,YM7_var,4,1);
    YM8_avg = normrnd(YM8_avg_t,YM8_var,4,1);
    YM9_avg = normrnd(YM9_avg_t,YM9_var,4,1);
    YM10_avg = normrnd(YM10_avg_t,YM10_var,4,1);
    %汇总平均值 AVG
    AVG = ones(4,10);
    for i = 1:4
       AVG(i,1) = YM1_avg(i);
    end
    for i = 1:4
       AVG(i,2) = YM2_avg(i);
    end
    for i = 1:4
       AVG(i,3) = YM3_avg(i);
    end
    for i = 1:4
       AVG(i,4) = YM4_avg(i);
    end
    for i = 1:4
       AVG(i,5) = YM5_avg(i);
    end
    for i = 1:4
       AVG(i,6) = YM6_avg(i);
    end
    for i = 1:4
       AVG(i,7) = YM7_avg(i);
    end
    for i = 1:4
       AVG(i,8) = YM8_avg(i);
    end
```

```
    for i = 1:4
       AVG(i,9) = YM9_avg(i);
    end
    for i = 1:4
       AVG(i,10) = YM10_avg(i);
    end
    Time_true = 999999999;
    U_true = ones(1,10);
    for first = 1:10
        % 定义已入队列U，未入队列S。
        U = ones(1,1);
        S = ones(1,9);
        U(1) = first;
        for i = 1:10
            S(i) = i;
        end
        S(first) = [];
        %定义一个st序列并初始化，存储四个工位的当前的空闲时间。
        st = ones(1,4);
        st(1) = AVG(1,first);
        for i = 2:4
            st(i) = st(i-1) + AVG(i,first);
        end
        min = ones(1,2); %[S中下标，差值和]
        while ~isempty(S)
           min(1) = 1;
           if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) > 0
               min(2) = 0;
           end
           if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) <= 0
               min(2) = abs((AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)));
           end
           if (AVG(3,S(1)) - AVG(2,S(1)))-(st(4)-st(3)) <= 0
               min(2)     =     min(2)     +     abs((AVG(3,S(1))     -
AVG(2,S(1)))-(st(4)-st(3)));
           end
           for i = 2:length(S)
               temp_sum = zeros(1,1);
               if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) > 0
                   temp_sum = 0;
               end
               if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) <= 0
                   temp_sum         =         abs((AVG(2,S(1))         -
AVG(1,S(1)))-(st(3)-st(2)));
```

```
                end
                if (AVG(3,S(1)) - AVG(2,S(1)))-(st(4)-st(3)) <= 0
                    temp_sum    =    temp_sum    +    abs((AVG(3,S(1))    -
AVG(2,S(1)))-(st(4)-st(3)));
                end
                if temp_sum < min(2)
                    min(1) = i;
                    min(2) = temp_sum;
                end
            end
            st(1) = st(1) + AVG(1,S(min(1)));
            for i = 2:4
                if st(i-1) < st(i)
                    st(i) = st(i) + AVG(i,S(min(1)));
                end
                if st(i-1) >= st(i)
                    st(i) = st(i-1) + AVG(i,S(min(1)));
                end
            end
            U(length(U) + 1) = S(min(1));
            S(min(1)) = [];
        end
        if st(4) < Time_true
            Time_true = st(4);
            for i = 1:10
                U_true(i) = U(i);
            end
        end
    end
    % 因为我们需要的数据为缩短时间，所以将所用时间大于参考时间的数据暂不
计入计数（但记入总数据量）。
    if Time_true <= 196.2069
        %求出当前时间与参考时间的比值（百分比）
        g1 = (Time_true/196.2069)*100;
        %将再次之前的 Time 数据都加 1。
        for g_i = 1:(100-g1)
            Time(g_i) = Time(g_i) + 1;
        end
    end
end
plot(1:100,(Time./1000));
xlabel('缩短时间比例');
ylabel('完成最大概率');
title('关于缩短时间比例与完成最大概率的关系');
```

第四题：

```
clear;
clc;
D = xlsread('C:\Users\86187\Desktop\建模提交\分隔后表.xls',1,'C2:AZ41');
% 数据导入
YM1 = D(1:4,:);
YM2 = D(5:8,:);
YM3 = D(9:12,:);
YM4 = D(13:16,:);
YM5 = D(17:20,:);
YM6 = D(21:24,:);
YM7 = D(25:28,:);
YM8 = D(29:32,:);
YM9 = D(33:36,:);
YM10 = D(37:40,:);
%求均值
YM1_avg_t = mean(YM1,2);
YM2_avg_t = mean(YM2,2);
YM3_avg_t = mean(YM3,2);
YM4_avg_t = mean(YM4,2);
YM5_avg_t = mean(YM5,2);
YM6_avg_t = mean(YM6,2);
YM7_avg_t = mean(YM7,2);
YM8_avg_t = mean(YM8,2);
YM9_avg_t = mean(YM9,2);
YM10_avg_t = mean(YM10,2);
%求方差(标准差平方)
YM1_var = std(YM1,0,2).^2;
YM2_var = std(YM2,0,2).^2;
YM3_var = std(YM3,0,2).^2;
YM4_var = std(YM4,0,2).^2;
YM5_var = std(YM5,0,2).^2;
YM6_var = std(YM6,0,2).^2;
YM7_var = std(YM7,0,2).^2;
YM8_var = std(YM8,0,2).^2;
YM9_var = std(YM9,0,2).^2;
YM10_var = std(YM10,0,2).^2;
%统计分布
FB = zeros(1,400);
for cnt = 1:1000
    % 1000箱YM1疫苗的各工位用时随机数。
    YM1_AVG = ones(4,1000);
    for i = 1:1000
        YM1_avg = normrnd(YM1_avg_t,YM1_var,4,1);
```

```
    for j = 1:4
        YM1_AVG(j,i) = YM1_avg(j,1);
    end
end
% 定义 YM_st 存储当前各机器空闲时间。
YM1_st = zeros(1,4);
% YM1_st 不断累加，计算出最后一次的空闲时间结果。
for i = 1:4
   YM1_st(1,i) = YM1_AVG(i,1);
end
YM1_st(1,1) = YM1_st(1,1) + YM1_AVG(1,i);
for i = 2:1000
   for j = 2:4
       if YM1_st(1,j) >= YM1_st(1,j-1)
           YM1_st(1,j) = YM1_st(1,j) + YM1_AVG(j,i);
       end
       if YM1_st(1,j) < YM1_st(1,j-1)
           YM1_st(1,j) = YM1_st(1,j-1) + YM1_AVG(j,i);
       end
   end
end
%对 YM2~10 的耗时进行计算。
% YM2 的计算
YM2_AVG = ones(4,500);
for i = 1:500
    YM2_avg = normrnd(YM2_avg_t,YM2_var,4,1);
    for j = 1:4
        YM2_AVG(j,i) = YM2_avg(j,1);
    end
end
% 定义 YM_st 存储当前各机器空闲时间。
YM2_st = zeros(1,4);
% YM1_st 不断累加，计算出最后一次的空闲时间结果。
for i = 1:4
   YM2_st(1,i) = YM2_AVG(i,1);
end
YM2_st(1,1) = YM2_st(1,1) + YM2_AVG(1,i);
for i = 2:500
   for j = 2:4
       if YM2_st(1,j) >= YM2_st(1,j-1)
           YM2_st(1,j) = YM2_st(1,j) + YM2_AVG(j,i);
       end
       if YM2_st(1,j) < YM2_st(1,j-1)
           YM2_st(1,j) = YM2_st(1,j-1) + YM2_AVG(j,i);
```

```
        end
    end
end
% YM3 的计算
YM3_AVG = ones(4,600);
for i = 1:600
    YM3_avg = normrnd(YM3_avg_t,YM3_var,4,1);
    for j = 1:4
        YM3_AVG(j,i) = YM3_avg(j,1);
    end
end
% 定义 YM_st 存储当前各机器空闲时间。
YM3_st = zeros(1,4);
% YM1_st 不断累加，计算出最后一次的空闲时间结果。
for i = 1:4
   YM3_st(1,i) = YM3_AVG(i,1);
end
YM3_st(1,1) = YM3_st(1,1) + YM3_AVG(1,i);
for i = 2:600
   for j = 2:4
       if YM3_st(1,j) >= YM3_st(1,j-1)
           YM3_st(1,j) = YM3_st(1,j) + YM3_AVG(j,i);
       end
       if YM3_st(1,j) < YM3_st(1,j-1)
           YM3_st(1,j) = YM3_st(1,j-1) + YM3_AVG(j,i);
       end
   end
end
% YM4 的计算
YM4_AVG = ones(4,1000);
for i = 1:1000
    YM4_avg = normrnd(YM4_avg_t,YM4_var,4,1);
    for j = 1:4
        YM4_AVG(j,i) = YM4_avg(j,1);
    end
end
% 定义 YM_st 存储当前各机器空闲时间。
YM4_st = zeros(1,4);
% YM1_st 不断累加，计算出最后一次的空闲时间结果。
for i = 1:4
   YM4_st(1,i) = YM4_AVG(i,1);
end
YM4_st(1,1) = YM4_st(1,1) + YM4_AVG(1,i);
for i = 2:1000
```

```
   for j = 2:4
       if YM4_st(1,j) >= YM4_st(1,j-1)
           YM4_st(1,j) = YM4_st(1,j) + YM4_AVG(j,i);
       end
       if YM4_st(1,j) < YM4_st(1,j-1)
           YM4_st(1,j) = YM4_st(1,j-1) + YM4_AVG(j,i);
       end
   end
end
% YM5 的计算
YM5_AVG = ones(4,1200);
for i = 1:1200
    YM5_avg = normrnd(YM5_avg_t,YM5_var,4,1);
    for j = 1:4
        YM5_AVG(j,i) = YM5_avg(j,1);
    end
end
% 定义 YM_st 存储当前各机器空闲时间。
YM5_st = zeros(1,4);
% YM1_st 不断累加，计算出最后一次的空闲时间结果。
for i = 1:4
   YM5_st(1,i) = YM5_AVG(i,1);
end
YM5_st(1,1) = YM5_st(1,1) + YM5_AVG(1,i);
for i = 2:1200
   for j = 2:4
       if YM5_st(1,j) >= YM5_st(1,j-1)
           YM5_st(1,j) = YM5_st(1,j) + YM5_AVG(j,i);
       end
       if YM5_st(1,j) < YM5_st(1,j-1)
           YM5_st(1,j) = YM5_st(1,j-1) + YM5_AVG(j,i);
       end
   end
end
% YM6 的计算
YM6_AVG = ones(4,1600);
for i = 1:1600
    YM6_avg = normrnd(YM6_avg_t,YM6_var,4,1);
    for j = 1:4
        YM6_AVG(j,i) = YM6_avg(j,1);
    end
end
% 定义 YM_st 存储当前各机器空闲时间。
YM6_st = zeros(1,4);
```

```
% YM1_st 不断累加，计算出最后一次的空闲时间结果。
for i = 1:4
   YM6_st(1,i) = YM6_AVG(i,1);
end
YM6_st(1,1) = YM6_st(1,1) + YM6_AVG(1,i);
for i = 2:1600
   for j = 2:4
       if YM6_st(1,j) >= YM6_st(1,j-1)
           YM6_st(1,j) = YM6_st(1,j) + YM6_AVG(j,i);
       end
       if YM6_st(1,j) < YM6_st(1,j-1)
           YM6_st(1,j) = YM6_st(1,j-1) + YM6_AVG(j,i);
       end
   end
end
% YM7 的计算
YM7_AVG = ones(4,1800);
for i = 1:1800
    YM7_avg = normrnd(YM7_avg_t,YM7_var,4,1);
    for j = 1:4
        YM7_AVG(j,i) = YM7_avg(j,1);
    end
end
% 定义 YM_st 存储当前各机器空闲时间。
YM7_st = zeros(1,4);
% YM1_st 不断累加，计算出最后一次的空闲时间结果。
for i = 1:4
   YM7_st(1,i) = YM7_AVG(i,1);
end
YM7_st(1,1) = YM7_st(1,1) + YM7_AVG(1,i);
for i = 2:1800
   for j = 2:4
       if YM7_st(1,j) >= YM7_st(1,j-1)
           YM7_st(1,j) = YM7_st(1,j) + YM7_AVG(j,i);
       end
       if YM7_st(1,j) < YM7_st(1,j-1)
           YM7_st(1,j) = YM7_st(1,j-1) + YM7_AVG(j,i);
       end
   end
end
% YM8 的计算
YM8_AVG = ones(4,800);
for i = 1:800
    YM8_avg = normrnd(YM8_avg_t,YM8_var,4,1);
```

```
    for j = 1:4
        YM8_AVG(j,i) = YM8_avg(j,1);
    end
end
% 定义 YM_st 存储当前各机器空闲时间。
YM8_st = zeros(1,4);
% YM1_st 不断累加，计算出最后一次的空闲时间结果。
for i = 1:4
   YM8_st(1,i) = YM8_AVG(i,1);
end
YM8_st(1,1) = YM8_st(1,1) + YM8_AVG(1,i);
for i = 2:800
   for j = 2:4
       if YM8_st(1,j) >= YM8_st(1,j-1)
           YM8_st(1,j) = YM8_st(1,j) + YM8_AVG(j,i);
       end
       if YM8_st(1,j) < YM8_st(1,j-1)
           YM8_st(1,j) = YM8_st(1,j-1) + YM8_AVG(j,i);
       end
   end
end
% YM9 的计算
YM9_AVG = ones(4,600);
for i = 1:600
    YM9_avg = normrnd(YM9_avg_t,YM9_var,4,1);
    for j = 1:4
        YM9_AVG(j,i) = YM9_avg(j,1);
    end
end
% 定义 YM_st 存储当前各机器空闲时间。
YM9_st = zeros(1,4);
% YM1_st 不断累加，计算出最后一次的空闲时间结果。
for i = 1:4
   YM9_st(1,i) = YM9_AVG(i,1);
end
YM9_st(1,1) = YM9_st(1,1) + YM9_AVG(1,i);
for i = 2:600
   for j = 2:4
       if YM9_st(1,j) >= YM9_st(1,j-1)
           YM9_st(1,j) = YM9_st(1,j) + YM9_AVG(j,i);
       end
       if YM9_st(1,j) < YM9_st(1,j-1)
           YM9_st(1,j) = YM9_st(1,j-1) + YM9_AVG(j,i);
       end
```

```
   end
end
% YM10 的计算
YM10_AVG = ones(4,900);
for i = 1:900
    YM10_avg = normrnd(YM10_avg_t,YM10_var,4,1);
    for j = 1:4
        YM10_AVG(j,i) = YM10_avg(j,1);
    end
end
% 定义 YM_st 存储当前各机器空闲时间。
YM10_st = zeros(1,4);
% YM1_st 不断累加，计算出最后一次的空闲时间结果。
for i = 1:4
   YM10_st(1,i) = YM10_AVG(i,1);
end
YM10_st(1,1) = YM10_st(1,1) + YM10_AVG(1,i);
for i = 2:900
   for j = 2:4
       if YM10_st(1,j) >= YM10_st(1,j-1)
           YM10_st(1,j) = YM10_st(1,j) + YM10_AVG(j,i);
       end
       if YM10_st(1,j) < YM10_st(1,j-1)
           YM10_st(1,j) = YM10_st(1,j-1) + YM10_AVG(j,i);
       end
   end
end
%套用第二题模型。
AVG = zeros(4,10);
for i = 1:4
    AVG(i,1) = YM1_st(i);
end
for i = 1:4
    AVG(i,2) = YM2_st(i);
end
for i = 1:4
    AVG(i,3) = YM3_st(i);
end
for i = 1:4
    AVG(i,4) = YM4_st(i);
end
for i = 1:4
    AVG(i,5) = YM5_st(i);
end
```

```
    for i = 1:4
        AVG(i,6) = YM6_st(i);
    end
    for i = 1:4
        AVG(i,7) = YM7_st(i);
    end
    for i = 1:4
        AVG(i,8) = YM8_st(i);
    end
    for i = 1:4
        AVG(i,9) = YM9_st(i);
    end
    for i = 1:4
        AVG(i,10) = YM10_st(i);
    end
    Time_true = 999999999;
    U_true = ones(1,10);
    for first = 1:10
        % 定义已入队列U，未入队列S。
        U = ones(1,1);
        S = ones(1,9);
        U(1) = first;
        for i = 1:10
            S(i) = i;
        end
        S(first) = [];
        %定义一个st序列并初始化，存储四个工位的当前的空闲时间。
        st = ones(1,4);
        st(1) = AVG(1,first);
        for i = 2:4
            st(i) = st(i-1) + AVG(i,first);
        end
        min = ones(1,2); %[S中下标，差值和]
        while ~isempty(S)
           min(1) = 1;
           if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) > 0
               min(2) = 0;
           end
           if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) <= 0
               min(2) = abs((AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)));
           end
           if (AVG(3,S(1)) - AVG(2,S(1)))-(st(4)-st(3)) <= 0
               min(2)     =     min(2)     +     abs((AVG(3,S(1))     -
AVG(2,S(1)))-(st(4)-st(3)));
```

```
            end
            for i = 2:length(S)
                temp_sum = zeros(1,1);
                if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) > 0
                    temp_sum = 0;
                end
                if (AVG(2,S(1)) - AVG(1,S(1)))-(st(3)-st(2)) <= 0
                    temp_sum         =         abs((AVG(2,S(1))         -
AVG(1,S(1)))-(st(3)-st(2)));
                end
                if (AVG(3,S(1)) - AVG(2,S(1)))-(st(4)-st(3)) <= 0
                    temp_sum    =    temp_sum    +    abs((AVG(3,S(1))    -
AVG(2,S(1)))-(st(4)-st(3)));
                end
                if temp_sum < min(2)
                    min(1) = i;
                    min(2) = temp_sum;
                end
            end
            st(1) = st(1) + AVG(1,S(min(1)));
            for i = 2:4
                if st(i-1) < st(i)
                    st(i) = st(i) + AVG(i,S(min(1)));
                end
                if st(i-1) >= st(i)
                    st(i) = st(i-1) + AVG(i,S(min(1)));
                end
            end
            U(length(U) + 1) = S(min(1));
            S(min(1)) = [];
        end
        if st(4) < Time_true
            Time_true = st(4);
            for i = 1:10
                U_true(i) = U(i);
            end
        end
    end
    % 定义耗时 H（小时） D（天数）
    H = st(4)/60;
    D = ceil(H/16);
    for t = D:400
        FB(t) = FB(t) + 1;
    end
```

end
FB