队伍编号	MC2409179
题号	（C）

基于鲸鱼算法的分拣中心排班问题优化

摘要：当今时代，随着信息技术的高速发展，电商物流行业方兴未艾，根据已有历史数据来预测物流货量未来走向，据此来科学管理人员排班，以减少用人成本尤为重要。

问题一需要根据过去三个月每天的货量数据和过去一个月的每小时货量数据，预测未来 30 天的每天及每小时货量。为此本文首先对所有分拣中心的货量进行可视化，剔除部分异常值，补充部分缺失值。处理之后利用 **LSTM 神经网络**进行预测，取后十天的数据做验证集，平均标准均方误差为 0.081034，最后根据小时比数据，预测每小时货量数据。

问题二要求在问题一结果的基础上，根据调整前线路网络的运输货量及调整后网络情况预测未来 30 天每天与每小时货量。为此，本文首先对调整前后的线路网络图进行可视化，总网络分别分成 5 个连通分量。定义每个分拣中心的平均吞吐货量为分拣中心与周围所有分拣中心运输货量的平均值。接着应用图论的知识，对每个**连通分量**都计算它的**度中心性、中间中心性、特征向量中心性**。然后运用**主成分分析**将 3 个指标降维为一个指标，作为评价分拣中心在网络中的重要程度。编写程序模拟线路运行，根据分拣中心平均货物吞吐量与重要程度计算预测货物吞吐量，得到每日货量预测结果，根据小时比计算每小时货量。

问题三要求在问题二的基础上，给出未来 30 天每个分拣中心每个班次的出勤人数，要求在每天货量处理完成的基础上，安排的人天数尽可能少，且排班尽量均衡。为此，本文采用**双目标优化约束模型**，然后转化为**等价的单目标优化模型**，然后采取用**蒙特卡洛**求初始解，利用**鲸鱼算法**求最优解的方法计算最优方案。

问题四要求在问题三的基础上以 SC60 为例，给出未来 30 天每名正式员工及临时工的班次出勤计划，要求每名正式工出勤率不超过 85%，且连续出勤天数不超过 7 天。在此基础上使每天安排人数尽可能少。对此，采用七天一班排班方式的构建相应符合该问题的目标函数与约束条件的**整数规划模型**，利用**鲸鱼算法和整数线性规划**求最优的排班方案。

关键词：LSTM 神经网络；连通分量；双目标规划；蒙特卡洛方法；鲸鱼算法；整数线性规划

目录

一、问题重述

1.1 问题背景

电商物流网络是订单履约的基石，它由多个环节组成，如图 1-1，其中分拣中心是运输网络的中间环节，它接受从仓库或者其他分拣中心运输进入的物流，同时，商品也从分拣中心向其他分拣中心或营业部输送。

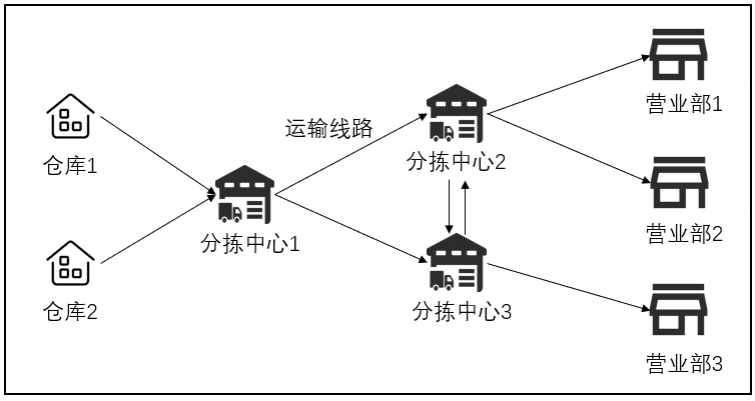


图 1-1 电商物流网络运作图

分拣中心的货量预测是电商物流网络的重要研究课题，对分拣中心的货量的精准预测是后续管理与决策的基础。本文聚焦与对分拣中心的货量预测与分拣中心之间的运输量的预测，帮助管理者建立科学高效的管理方式。帮助管理者给出在完成工作的前提下尽可能降低人员成本的人员分配方案。

1.2 问题要求

围绕电子商务物流分拣中心货量预测与分拣中心人员排班两个问题进行分析，并建立数学模型求解。本文依次解决如下问题：

1.2.1 问题一

根据附件 1 与附件 2，建立货量预测模型，对 57 个分拣中心未来 30 天每天及每小时的货量进行预测，得出预测结果。

1.2.2 问题二

根据附件 3，即过去 90 天各分拣中心之间每个运输线路的平均货量，与附件 4（未来分拣中心的运输线路的变化），预测未来 30 天发生变化后 57 个分拣中心的每天及每小时货量。

1.2.3 问题三

基于问题二建立的模型，给出未来 30 天每个分拣中心每个班次的出勤人数，要求在每天货量处理完成的基础上，安排的人天数尽可能少。

1.2.4 问题四

基于问题二建立的模型，研究特定分拣中心的排班问题，以 SC60 为例，给出未来 30 天每名正式员工及临时工的班次出勤计划，要求每名正式工出勤率不超过 85%，且连续出勤天数不超过 7 天。在此基础上使每天安排人数尽可能少。

二、问题分析

2. 1 问题一分析

2.1.1 数据预处理与数据特征分析

（1）数据预处理

对附件 1 数据，进行初步可视化处理，画出部分分拣中心的货量随时间的折线图，发现在 11 月初，数据有一个较大的波动。查阅相关资料可知，11 月 11 日为电商的重要节日，会带来货量的急剧变化，这里认为其是异常数据。对附件 2 数据，对长数据进行宽数据转化，发现附件 2 存在较多缺失值，发现共 7759 个缺失值，对缺失值进行补零处理。

（2）数据特征分析

对附件 1 数据，取 SC2、SC10、SC15、SC26 四列数据，对其进行 z-score 标准化处理，如（1）

$$Z_{ij} = \frac{x_{ij-}\mu}{\sigma} \tag{1}$$

然后画箱线图进行可视化。根据图 2-1，发现这些数据有一个共同的特征，即上下四分位数与边缘数的跨度较大，有的甚至能达到 3 倍的平均值。说明了附件 1 数据是一个具有较大波动的时间序列数据

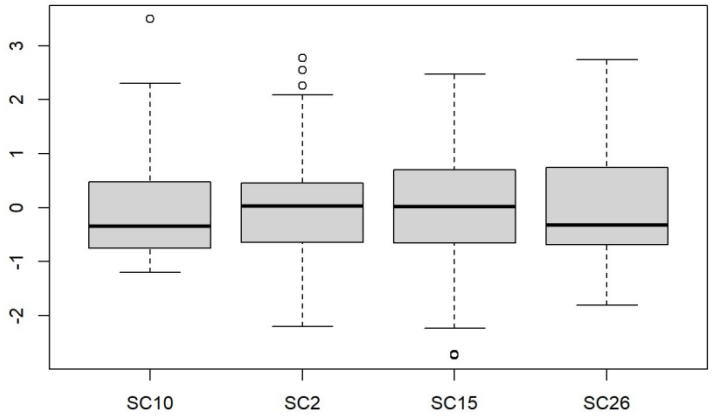


图 2-1 部分分拣中心货量数据箱线图

对于附件二数据，同样地，取部分数据（SC15、SC20、SC31、SC66）观察，如图 2-2，发现附件 2 数据同样是一个高波动的数据，但是考虑到时间以小时计算，并观察数据规律，发现这是一个近似以 1 天为一个周期波动的时间序列数据。

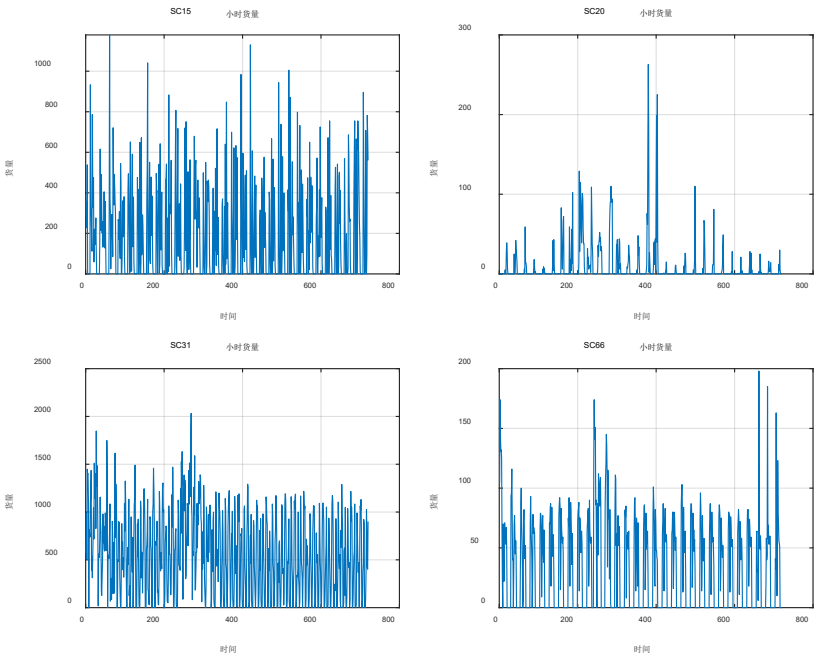


图 2-2 部分分拣中心小时货量折线图

2.1.2 模型的选取

（1）日级数据的预测

考虑到附件 1 为时间序列数据，且由于物流本身的特征，选取时间序列模型（ARIMA 模型）或

LSTM 神经网络进行预测。但根据数据量来判断，删去“双 11”数据集后，剩余数据过少，时间序列模型可能无法根据这么少量的数据进行中长期预测。故采取，先使用时间序列预测部分数据，若拟合效果不佳，采取 LSTM 神经网络进行预测的建模思路。

（2）小时级数据的预测

根据附件 2 数据的周期性，以及日货量与天货量之间的之间联系，可通过天货量来计算日货量。

2.2 问题二分析

2.2.1 数据可视化

对附件 3 与附件 4 的网络图进行可视化，初步得到分拣中心之间的大致关系。

2.2.2 模型的建立

首先，根据附件 3（站点之间的运输货量的大小），建立以货量大小为边的有向网络图。然后根据附件 4，调整网络结构与网络的边，得到对于新的网络的、每个分拣中心的平均可运入或运出的货量预测值。

运用图论的相关知识，根据附件 4，计算出新网络每个分拣中心的度中心性、中间中心性、特征向量中心性。运用主成分分析法或熵权法对三者进行综合评价，得到每个分拣中心在网络中的重要程度。

2.2.3 模型求解

根据每天货量的预测结果，模拟网络运行，每个分拣中心实际运入或运出的值与每个分拣中心货量预测值和分拣中心在网络中的重要程度相关，分拣中心在网络中位置越重要，运入或运出货量越多。

最后，根据日货量与小时货量之间的联系求解小时货量的预测值。

2.3 问题三分析

问题 3 是一个优化问题，本题需要基于问题 2 的预测结果，给出未来三十天每个分拣中心的每个班次的出勤人数，首先要满足每天货量处理完成及正式工总数不超过 60 人的基础上，安排的人员尽可能少，其次让每天实际小时人效尽可能均衡。

因此首先我们需要建立动态调整班次安排人员数的数学模型，在此问题上，我们要考虑每小时货量变化对模型的动态调整。建立每天安排总人员数最小，每天实际小时人效方差最小的双目标作为目标函数，然后建立正式工总人数不超过 60 人与每小时总可处理货量大于等于问题 2 预测的每小

时货量的双约束条件。采用蒙特卡洛求得更好的初始解，然后利用鲸鱼算法求得安排班次人员数的方案。最后对每天安排总人员数、每天实际小时人效方差进行结果分析。

图 2-3 问题三求解流程图

2.4 问题四分析

问题四是一个排班问题，要求我们基于问题 2 的预测结果，对 SC60 的人员进行排班，给出未来 30 天每名正式工和临时工的班次出勤计划。在此之前，我们需要先得到每天每个班次的出勤人数，这里我们采用问题三建立的模型，对正式工人数稍加调整，将 60 名正式工改成 200 名，得到 SC60 未来 30 天每个班次的出勤人数，并且可以满足在每天货量能够处理完成的基础上，安排的人天数尽可能少，每天实际小时人效尽量均衡。

在得知每天出勤人数的基础上，我们对正式工进行排班。对人员先进行分类，分成 7 个班次，每个班次连续工作六天然后休息一天，保证每个人最高出勤率不超过 85%，在此基础上，设立约束条件，建立整数规划模型，得到每个班次的人数。在此基础上，对多处的人次数进行删减，在删减的同时保证每名正式工的出勤率尽量均衡。

三、模型假设

（1）2023 年 12 月 1 日到 2023 年 12 月 30 日电商物流无异常现象发生

（2）假设分拣中心容量无上限

（3）员工严格按照规定完成任务

四、符号说明

符号	解释说明
x_{ij}	小时货量
X_{ij}	每天货量
ω_i	分拣中心i的平均小时比
$C_D(v)$	度中心性
$\mathrm{C_B(v)}$	中间中心性
$\mathrm{x_v}$	特征向量中心性
u_i	所求当天所求站点第i班次的正式工人数
v_i	所求当天所求站点第i班次的临时工人数
E_j	所求当天所求站点第j小时工作的班次集合
c_j	所求当天所求站点第j小时实际人效
$\overline{c}$	所求当天所求站点平均小时人效

五、模型的建立与求解

5.1 问题一

5.1.1 数据预处理

对于附件一：首先对部分数据进行可视化，选取 SC1、SC18、SC24、SC48 做观察其数据特点。如图 2-2，发现在 10 月末和 11 月中旬期间物流数值发生较大的波动。查阅相关资料可知，波动的原因是每年的“双 11”购物节，“双 11”的开始时间是每年的 10 月 20 日 20 点，结束时间是 11 月 11 日 24 点，考虑物流发出存在时间差，故我们进行预测时不考虑 10 月 22 日与 11 月 13 日的数据（共 22 个数据），同时留存 11 月 14 日到 11 月 30 日的数据做验证集，验证预测效果，一共剩余数据 100 个。

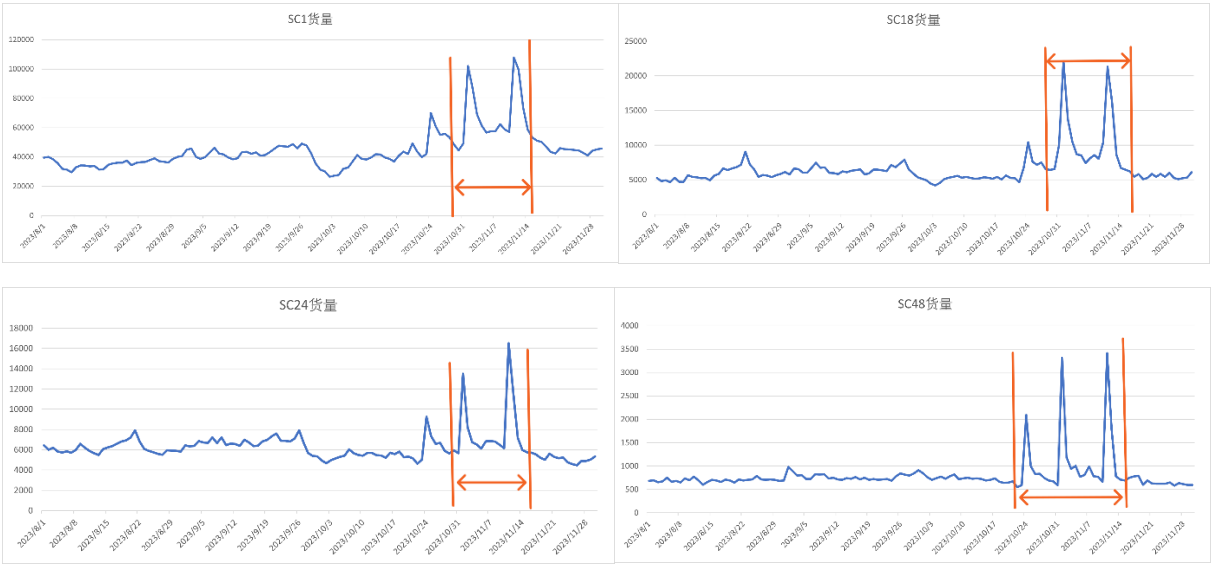


图 5-1 部分分拣中心每天货量折线图

5.1.2 模型选取

首先对于时序数据考虑使用 ARIMA 模型进行预测，但由于给定历史货量数据太少，推断可能预测结果会发生收敛，不符合原数据特征。

对 SC1 数据进行 ARIMA 模型的初步尝试发现预测值在 10 天以后开始收敛，并逐渐收敛到一个值，经分析发现，主要是数据量过少导致预测结果在后期收敛，不符合原数据波动大的特征。ARIMA 模型通常需要足够的数据来捕捉时间序列的长期趋势和周期性变化，数据不足会导致无法准确预测。因此，选取 LSTM 神经网络进行预测。

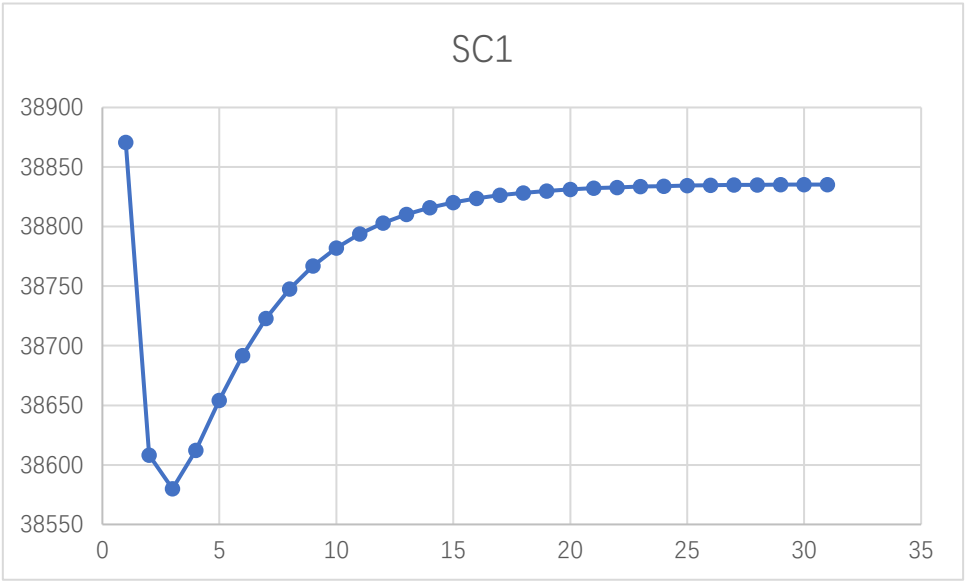


图 5-2 对 SC1 的 ARIMA 预测

5.1.3 模型解释

长短期记忆网络（Long Short-Term Memory，LSTM）是一种特殊的循环神经网络（RNN），在时间序列预测的问题上具有重要作用。LSTM 的核心思想是通过引入“门”机制来控制信息的流动和遗忘。这在一定程度上解决了历史数据不足的问题。LSTM 单元包含三个门：遗忘门、输入门和输出门，以及一个记忆单元（cell state）。以下是 LSTM 单元的详细结构和工作原理：

（1） 遗忘门（Forget Gate）

遗忘门决定了从上一个时刻的记忆单元状态中丢弃哪些信息。它接收上一个时刻的隐藏状态 (h_{t-1}) 和当前时刻的输入 (x_t)，并通过一个 sigmoid 函数输出一个介于 0 和 1 之间的值，表示要遗忘多少信息。

（2） 输入门（Input Gate）

输入门决定了哪些新信息将被添加到记忆单元中。它同样接收 (h_{t-1}) 和 (x_t)，并通过 sigmoid 函数产生一个介于 0 和 1 之间的值，表示要添加多少新信息。同时，一个 tanh 函数会生成一个新的候选记忆单元状态 $(\tilde{C}_t)$。

（3） 记忆单元更新

记忆单元的状态 (C_t) 会根据遗忘门和输入门的结果进行更新。首先，使用遗忘门的结果来遗忘上一个时刻的记忆单元状态 (C_{t-1}) 中的部分信息。然后，使用输入门的结果将新的候选记忆单元状态 $(\tilde{C}_t)$ 添加到记忆单元中。

（4） 输出门（Output Gate）

输出门决定了下一个隐藏状态 (h_t) 的值。它首先通过一个 sigmoid 函数来确定要输出的记忆单元状态的哪些部分。然后，记忆单元状态 (C_t) 通过 tanh 函数进行处理，并与输出门的结果相乘，得到下一个隐藏状态 (h_t)。

5.1.4 模型构建

利用 MATLAB 内置的 Deep Learning Toolbox 工具箱，具体代码详见附件 2。

对 SC14 样本进行初步预测，取前 80 个数据作为训练集，后 10 个为测试集数据，结果图 5-3，残差均值为 10.14447。

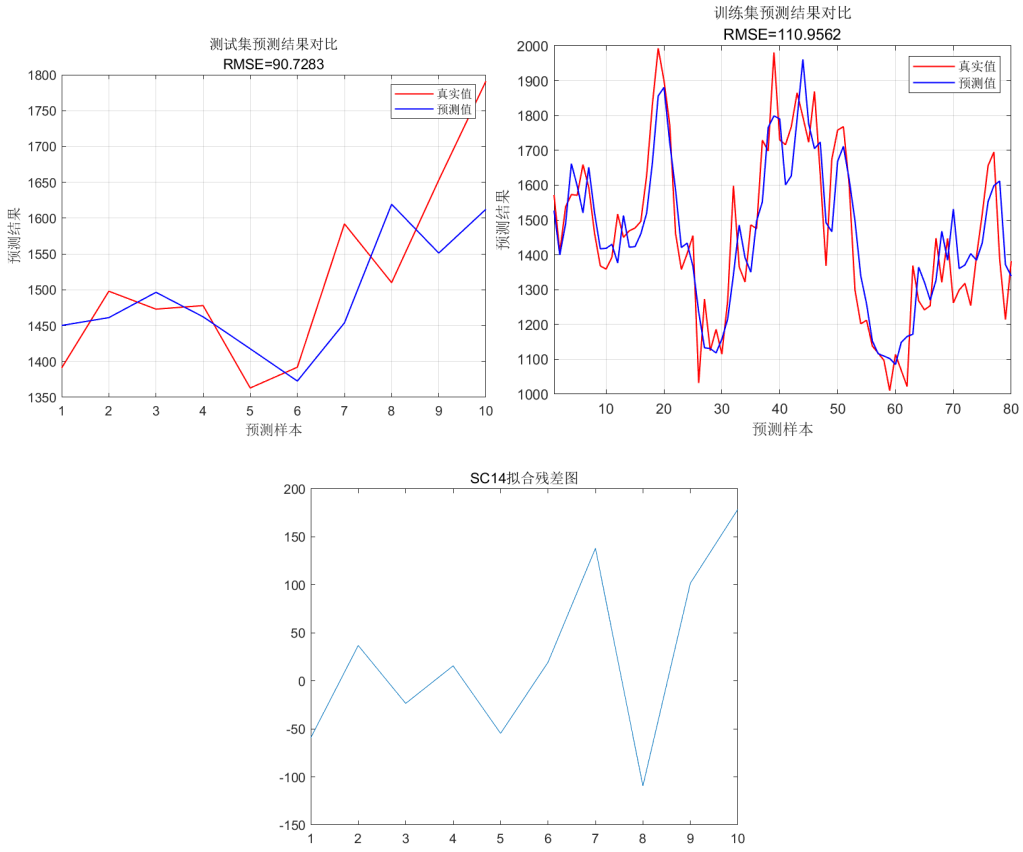


图 5-3 SC14 拟合结果

5.1.5 模型求解与评价

考虑到每个分拣中心具有不同的波动规律与特征，对每个分拣中心都建立神经网络进行训练，

日期	SC1	SC10	SC12	SC14	SC15	SC16	SC17
2023/12/1	49128	47286	9585	1532	4636	1662	22512
2023/12/2	38369	37852	3871	1422	3989	2160	19633
2023/12/3	36140	38509	3837	1462	4065	2108	19485
2023/12/4	39477	38704	4047	1417	4313	2046	18461
2023/12/5	40428	38987	4024	1442	4223	2249	18819
2023/12/6	41933	38847	4010	1482	3986	2265	18251[i]

并测试其训练结果。

代码构建思路如下：

数据数据预处理——划分数据集——数据归一化——构建 LSTM 网络结构——训练神经网络——仿真预测——反归一化——计算均方根误差——进行预测。

预测结果部分如 5-1 所示，完整详见附录 1。

表 5-1 预测结果（部分）

对于与测试集的预测结果我们采取对数据进行标准化后，进行均方误差分析，其标准化后的平

均均方误差为 0.081034，认为拟合效果良好。

表 5-2 测试集的均方误差表（部分）

SC1	SC10	SC12	SC14	SC15	SC16	SC17
0.044516	0.055887	0.019813	0.128007	0.222829	0.060017	0.077629
SC18	SC19	SC2	SC20	SC21	SC22	SC23
0.070577	0.111729	0.095386	0.099047	0.074096	0.067993	0.029886

对于小时货量的预测，利用每天货量预测值来预测小时货量。首先定义小时比ω_i，其中，i为各个分拣中心，j为时间序列。预测值即为预测货量与ω_i的乘积。

$$\omega_i = \frac{\sum_j \frac{x_{ij}}{X_{ij}}}{N} \tag{2}$$

表 5-3 小时货量预测值（部分）

日期	小时	SC1	SC10	SC12	SC14	SC15
2023/12/1	1	3122	3672	136	0	331
2023/12/1	2	3114	1758	227	0	358
2023/12/1	3	3090	2263	188	0	113
2023/12/1	4	3060	1992	268	0	31
2023/12/1	5	2900	2116	175	0	0
2023/12/1	6	2223	2159	140	7	0
2023/12/1	7	391	524	148	13	0
2023/12/1	8	1317	138	277	9	0

5.2 问题二：

5.2.1 数据可视化处理

首先，对附件 3 与附件 4 的数据进行可视化处理。可以发现，分拣中心之间的运输线路相对来说都比较聚集，大致可以分为 5 片网络区。其中 SC35-SC3 这条线路未发生变化，本题不进行预测。由于网络结构的特点：任意节点线路发生变化，都会影响整个网络的变化。因此，不能单纯的通过预测变化的分拣中心来解决第二问，应对分别对剩余 4 个网络进行预测。

图 5-4 附件 3、4 运输路线网络图

5.2.2 模型建立

（1）建立模型

首先，确定基本思路：使用 python 对 30 天中的货量进行模拟求解。

定义分拣中心i的货物平均吞吐量W_i,计算方法为：与i相关的所有分拣中心的运输货量平均值。引入附件 4 数据，更新网络中的站点，更新分拣中心i的W_i。但是，以平均值代替求解会出现重要分拣中心吞吐货量较少，而不重要的分拣中心货物超出处理能力的情况。

对分拣中心的重要性进行定义，在物流网络中，分拣中心的重要性通常与分拣中心的地理位置、运入和运出的分拣中心相关。而分拣中心的重要性对与分拣中心相关的线路的运输量有较大影响，在网络中相对更重要的分拣中心往往会有更大的货物运输量。

主成分分析法是利用降维的思想，在损失很少信息的前提下把多个指标转换为几个综合指标的多元统计方法。转换成的综合指标被称为主成分，其中每个主成分都是原始变量的线性组合，且各个主成分之间互不相关，这就使得主成分比原始变量具有某些更优越的性能。

主成分分析法模型的建立：

Step1：将原始数据标准化，以消除量纲影响，使不同维数据之间具有可比性。假设进行主成分分析的指标变量有m个：x_1，x_2，…，x_m共有 n 个评价对象。第i个评价对象的第j个指标的取值为x_{ij}。将各指标值x_{ij}转换成标准化指标$\widetilde{x_{ij}}$，

$$x'_{ij} = \left(x_{ij} - \bar{x}_j\right)/S_j (i = 1,2,\dots,n; j = 1,2,\dots,m) \tag{3}$$

其中

$$\bar{x_j} = \frac{1}{n}\Sigma_i^n x_{iJ} \tag{4}$$

$$s_j = \frac{1}{n-1}\Sigma_i^n (x_{ij} - \bar{x_j})^2 \text{（}i = 1,2,\dots,m\text{）} \tag{5}$$

即$\bar{x_j}$，s_j为第 j 个指标的样本均值和样本标准差。对应的，称$\widetilde{x_{ij}} = \frac{x_i - \bar{x_i}}{s_i}$,($i$，$j$=1,2,…,$m$)为标准化指标变量。

Step2：建立相关系数矩阵 R

相关系数矩阵$R = \left(r_{ij}\right)_{m\times m}$，其中$r_{ij} = \frac{\Sigma_{k=1}^n \widetilde{x_{k_i}} \cdot \widetilde{x_{k_j}}}{n-1}$ (i，j=1,2,…,m)

式中$r_{ii} = 1$，。$r_{ij} = r_{ji}$是第 i 个指标与第 j 个指标的相关系数。

Step3：计算特征值与特征向量

计算相关系数矩阵 R 的特征值 $\lambda 1 \geqslant \lambda 2 \geqslant \cdots \geqslant \lambda m \geqslant 0$，及对应的特征向量 u_1，u，…，um，其中，$\mu_j = \left(\mu_{1j}, \mu_{2j}, \dots, \mu_{nj}\right)^T$由特征向量组成 m 个新的指标变量。

$$\begin{cases} y_1 = u_{11}\widetilde{X_1} + u_{21}\widetilde{X_2} + u_{n1}\widetilde{X_n} \\ y_2 = u_{12}\widetilde{X_1} + u_{22}\widetilde{X_2} + u_{n2}\widetilde{X_n} \\ \cdots\cdots\cdots \\ y_m = u_{1m}\widetilde{X_1} + u_{2m}\widetilde{X_2} + u_{nm}\widetilde{X_n} \end{cases} \tag{5}$$

式中 y_1 是第一主成分,y_2 是第二主成分，… ，y_m是第 m 主成分。

Step4：计算综合得分

计算特征值 λ_j（j=1,2,…,m）的信息贡献率和累计贡献率。

称$b_j = \frac{\lambda_j}{\Sigma_{k=1}^m \lambda_m}$（$j$=1,2,…,$m$）为主成分$y_j$的信息贡献率：

$$a_p = \frac{\Sigma_{k=1}^P \lambda_k}{\Sigma_{k=1}^m \lambda_k} \tag{6}$$

当a_p接近 1 时（$a_p = 0.85$，0.90，0.95），则选择前 p 个指标变量为 p 个主成分，代替原来 m 个指

标。

计算综合得分

$$z = \Sigma_i^P b_j \cdot y_j \tag{7}$$

其中b_j为第 j 主成分的信息贡献率。

5.2.3 模型求解

（1）分拣中心重要性求解

根据图论的相关知识，描述一个节点争对整个网络的重要程度一般来说可以有 3 个指标，即度中心性、中间中心性、特征向量中心性。

$$C_D(v) = \frac{\deg(v)}{N-1} \tag{8}$$

$$\mathrm{C_B(v)} = \sum_{\mathrm{s\neq v\neq t}} \frac{\sigma_{\mathrm{st}}(\mathrm{v})}{\sigma_{\mathrm{st}}} \tag{9}$$

$$\mathrm{x_v} = \frac{1}{\lambda}\sum_{\mathrm{u}} \mathrm{A_{uv}x_u} \tag{10}$$

对附件 4 的网络进行划分，划分成 4 个主要的子网络，称其为连通分量，对子连通分量进行无向化处理，然后分别求每个连通分量的 3 个特征量，对连通分量的每个计算结果如表 5-4。由于 SC35 与 SC3 之间为简单双向运输，在此不进行展示。

表 5-4 子图的特征量（部分）

	分拣中心	度中心性	中间中心性	特征向量中心性
SC1 连通分量	SC1	0.875	0.428571	0.535973
	SC54	0.125	0	0.139066
SC10 连通分量	SC10	0.5	0.261111	0.426863
	SC12	0.1	0	0.063762
SC36 连通分量	SC36	0.045455	0	0.023397
	SC51	0.227273	0.006494	0.126719
SC53 连通分量	SC53	0.666667	0	0.435162
	SC34	0.666667	0	0.435162

进行主成分分析前要先进行数据相关性分析。跟据公式(8) (9) (10)可以发现三个特征值可能存在某种线性关系，提取主成分时贡献率偏向于 1。

表 5-5 每个连通分量的主成分分析结果

连通分量	第一主成分贡献率	度中心性权重	中间中心性权重	特征向量中心性权重
SC1	0.97941	0.58336	0.57504	0.57361
SC10	1	0.577	0.577	0.577

SC36	0.98524	0.57989	0.57296	0.57916
SC53	1	0.57735	0.57735	0.57735

综合计算权重 C_i，其中：$C_i = k_1 C_D + k_2 C_B + k_3 x_v$

利用综合权重和货物平均吞吐量来建立模拟线路，代码详见附录 3。

根据线路模拟来计算货量，得到每日的货量数据。同样地，和问题一类似，用每日货量数据与小时比相乘计算每小时货量数据。

$$x_{ij} = X_{ij} \cdot \omega_i \tag{11}$$

表 5-6 线路调整后的预测数据（部分）

日期	SC1	SC10	SC12	SC14	SC15	SC16	SC17
2023/12/1	49667	47785	9404	1532	5179	1434	22512
2023/12/2	38908	38351	3690	1422	4532	1932	19633
2023/12/3	36679	39008	3656	1462	4608	1880	19485
2023/12/4	40016	39203	3866	1417	4856	1818	18461
2023/12/5	40967	39486	3843	1442	4766	2021	18819

5.3 问题三：

5.3.1 模型介绍

（1）蒙特卡洛算法简介

蒙特卡洛算法在优化问题中的应用主要是通过随机抽样来探索解空间，从而找到一个相对较好的初始解的算法。

1. 定义解空间：首先，我们需要明确优化问题的解空间，确定决策变量的范围、约束条件等。
2. 随机抽样：在定义好的解空间中，我们通过随机抽样生成一组候选解。这些候选解是随机生成的，因此具有广泛的代表性，有助于探索整个解空间。
3. 评估候选解：对于每个随机生成的候选解，我们使用目标函数进行评估。
4. 选择初始解：根据评估结果，我们从候选解中选择一个相对较好的作为优化算法的初始解。

（2）鲸鱼优化算法简介

鲸鱼优化算法（Whale Optimization Algorithm，简称 WOA）是一种基于仿生学思想的群体智能优化算法。它通过模拟座头鲸群体的狩猎行为来寻找问题的最优解。算法的核心思想在于通过鲸鱼的搜索、包围和捕食行为来模拟优化问题的求解过程。鲸鱼优化算法具有收敛速度快、全局搜索能力强、算法简单易实现等优点，因此在许多优化问题中得到了广泛应用。

1. 初始化鲸鱼群：在算法开始时，通过蒙特卡洛算法生成一定数量的鲸鱼作为初始解，每个鲸鱼的位置和速度用一组向量来表示。这些鲸鱼构成了算法的初始种群，用于在解空间中探索潜在的最

优解。

2. 计算适应度：根据问题的目标函数，计算每条鲸鱼的适应度值。这个值反映了鲸鱼在解空间中的优劣程度，用于指导后续的搜索过程。
3. 搜索猎物：在搜索阶段，鲸鱼通过随机选择的方式来探索问题空间，寻找潜在的解决方案。这个过程模拟了鲸鱼在广阔海洋中寻找食物的行为。
4. 包围猎物：一旦鲸鱼发现了猎物，它们会开始调整自身位置，将猎物包围起来。在算法中，这意味着通过调整候选解的位置，使其更好地逼近最优解。这个过程通过计算参数和更新鲸鱼的位置来实现。
5. 捕食猎物：当鲸鱼群足够接近猎物时，它们会使用气泡网捕食机制来困住猎物。在算法中，这通过合并候选解来产生新的解决方案，进一步逼近最优解。
6. 更新鲸鱼群：在每次迭代中，根据鲸鱼的搜索、包围和捕食行为，更新鲸鱼群的位置和速度。通过不断迭代，鲸鱼群逐渐向最优解靠拢。
7. 终止条件：当达到预设的最大迭代次数，算法停止迭代。此时，算法输出的最优鲸鱼位置即为问题的最优解或近似最优解。

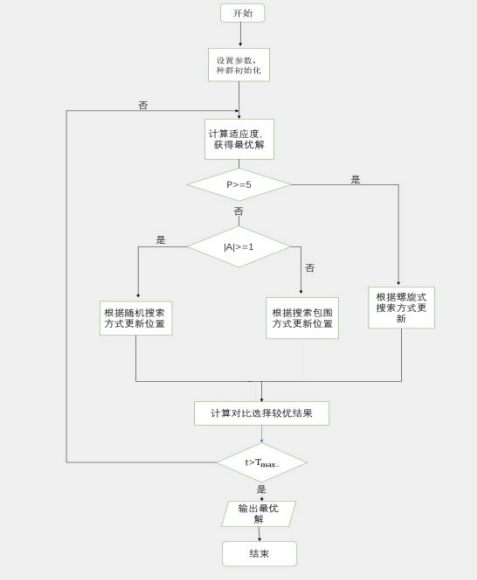


图 5-5 鲸鱼算法流程图

5.3.2 模型建立

（一） 模型简化

每天每个站点的人员安排只与当天各小时货量有关，相互独立，故可将模型简化为对每个站点当天求最优人员分配，从而降低维数且不影响目标函数的准确性。

（二） 定义变量与集合

u_i：所求当天所求站点第i班次的正式工人数

v_i：所求当天所求站点第i班次的临时工人数

E_j：所求当天所求站点第j小时工作的班次集合

c_j：所求当天所求站点第j小时实际人效

$\overline{c}$：所求当天所求站点平均小时人效

（三） 构造约束条件

1. 正式工总人数

每天每站点正式工总人数不能超过分拣中心的正式工人数 60 人

$$\sum_{i=1}^{6} u_i \leq 60 \tag{12}$$

2. 每小时总可处理货量大于等于需处理货量

$$25 \times \sum_{u_i \in E_j} u_i + 20 \times \sum_{v_i \in E_j} v_i > x_{ij} \tag{13}$$

（四） 构造目标函数

1. 每天站点安排总人数尽可能少：

$$K_1 = \sum_{i=1}^{6} u_i + \sum_{i=1}^{6} v_i \tag{14}$$

2. 实际小时人效方差尽可能小：

$$K_2 = \frac{1}{n}\sum_{j=1}^{24}\left(c_j - \overline{c}\right)^2 \tag{15}$$

其中，c_j的计算公式为：

$$c_j = \frac{x_{ij}}{\sum_{u_i \in E_j} u_i + \sum_{v_i \in E_j} v_i} \tag{16}$$

$\overline{c}$的计算公式为：

$$\overline{c} = \frac{\sum_{j=1}^{24} c_j}{24} \tag{17}$$

（五） 带约束条件的双目标函数优化模型

$$minK_1 \tag{18}$$

$$minK_2 \tag{19}$$

$$s.t.\begin{cases}\sum_{i=1}^{6} u_i \le 60 \\ 25\times\sum_{u_i\in E_j} u_i + 20\times\sum_{v_i\in E_j} v_i > x_{ij} \\ K_1 = \sum_{i=1}^{6} u_i + \sum_{i=1}^{6} v_i \\ K_2 = \frac{1}{n}\sum_{j=1}^{24}\left(c_j - \overline{c}\right)^2\end{cases} \tag{20}$$

（六） 双目标优化转化为单目标优化

考虑到二者皆为正数，可以通过线性加权和法找到对模型求解，即

$$minK_1 + \gamma K_2 \tag{21}$$

其中，γ为量目标的权重，用蒙特卡洛算法求得较好的初始解，再用鲸鱼优化算法求解。经过实验我们最终决定取$\gamma = 0.25$。

5.3.3 模型求解

根据建立好的模型，使用 MATLAB 编程解决，先利用蒙特卡洛寻找初始解，然后再利用鲸鱼算法求最优解。

对蒙特卡洛迭代中随机生成初始解的方法优化，在解决问题初期，根据题意，正式工总人数不得超过 60 人，则每个班次正式工的人数在 0~60 之间，故首先采用六个班次的正式工人数都用 0~60 的随机数生成器生成蒙特卡洛迭代内部的初始解。

但在解决后期实验数据量大的情况，发现六个随机生成的变量相加之和极易超出总数小于等于 60 的约束条件，于是我们通过计算随机数产生的期望，六个随机数期望加和等于 60 的方法来大大减少产生符合条件的一组解所需的时间，即一个班次的随机数生成器为 0~20，且因为所需安排人数优先安排正式工的题目条件，所以我们对 60 进行与前五个班次相减，从而将获得的结果赋予第六班次的操作，最后生成初始解的约束条件，便可用第六班次人数是否非负来代替，更优化了临时工的计算方式（代码见附件），使得通过一次计算就能使随机产生的临时工人数增加一定值使得其与正式工的总可处理小时货量大于需处理小时货量，在之后通过蒙特卡洛的筛选后再通过鲸鱼算法对六个班次的正式工人数与临时工人数进行调配，实现在满足题目条件基础上对产生初始解所需时间的优化。

下面给出经过实验效果较好的参数表：

表 5-7 问题三参数表

参数	选择参数数值
蒙特卡洛迭代数	100
鲸鱼数	80
迭代次数	500

表 5-8 规划结果（部分）

分拣中心	日期	班次	正式工	临时工
'SC1'	2023/12/1	'0-8'	1	157
'SC1'	2023/12/1	'5-13'	7	34
'SC1'	2023/12/1	'8-16'	2	58
'SC1'	2023/12/1	'12-20'	4	45
'SC1'	2023/12/1	'14-22'	2	1
'SC1'	2023/12/1	'16-24'	22	130
'SC10'	2023/12/1	'0-8'	8	177

5.3.4 模型评价

（1）鲸鱼算法对优化排班均衡的效果

对于该问题的排班方法，还可以利用整数规划进行优化，但是经过整数规划计算出来的排班方法不能考虑排班人数尽量均衡这个目标函数，所以将整数规划优化所得的排班表与本文利用的算法求解出来的排班表可做等价的比较，用于评价本题模型对排班的优化效果。

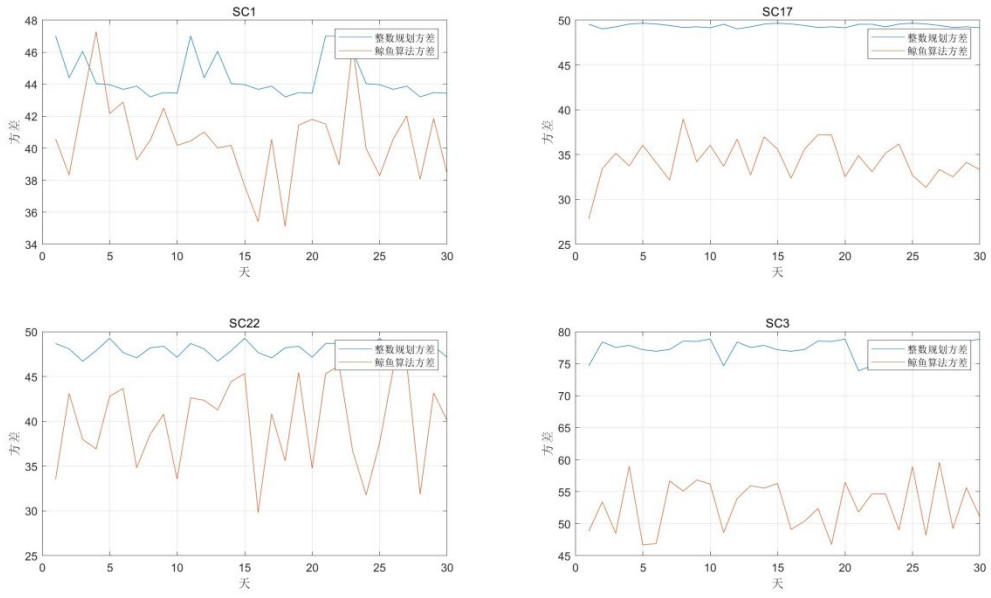
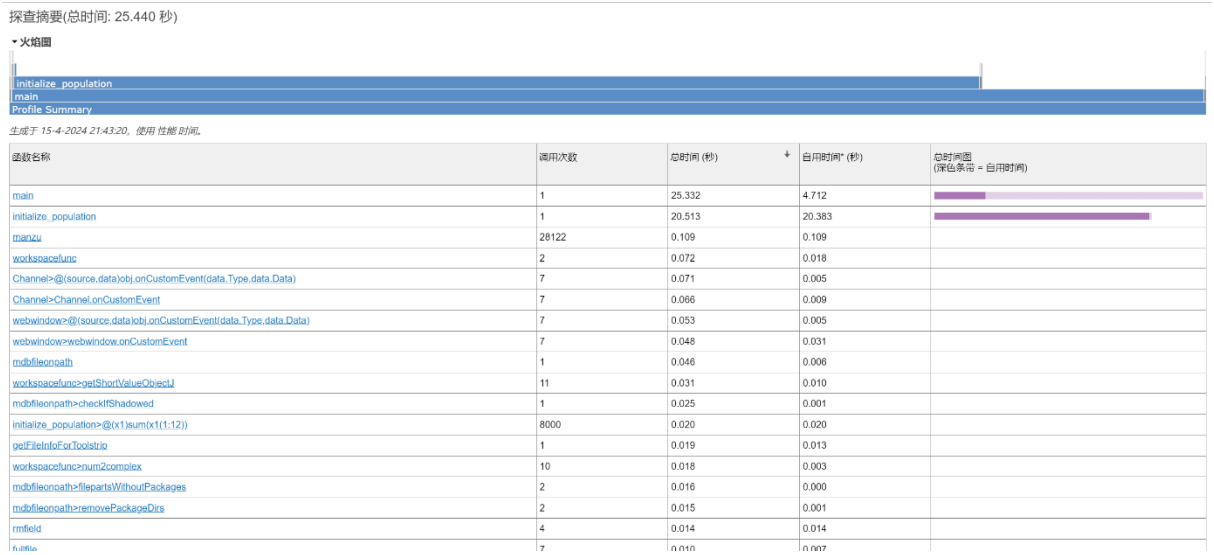


图 5-6 整数规划与鲸鱼算法方差比较

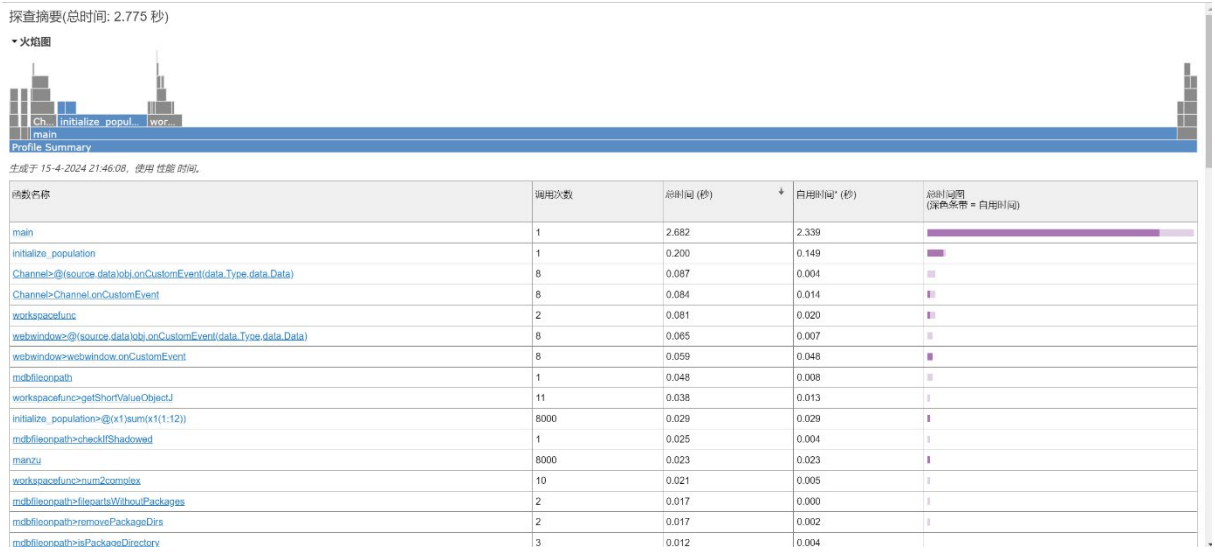
（2）蒙特卡洛算法对时间复杂度的优化

蒙特卡洛随机数对时间复杂度的优化，图 5-7 为不采用随机数产生期望来进行规划的时间复杂度，计算一次样本初始化函数所需时间为 20.383s，图 5-8 为优化后的时间复杂度，计算一次样本初始化函数所需时间为 0.2 秒



函数名称	调用次数	总时间 (秒)	自用时间* (秒)	总时间图 (深色条带 = 自用时间)
main	1	25.332	4.712	
initialize_population	1	20.513	20.383	
manzu	28122	0.109	0.109	
workspacefunc	2	0.072	0.018	
Channel>@(source,data)obj.onCustomEvent(data.Type,data.Data)	7	0.071	0.005	
Channel>Channel.onCustomEvent	7	0.066	0.009	
webwindow>@(source,data)obj.onCustomEvent(data.Type,data.Data)	7	0.053	0.005	
webwindow>webwindow.onCustomEvent	7	0.048	0.031	
mdbfileonpath	1	0.046	0.006	
workspacefunc>getShortValueObjectJ	11	0.031	0.010	
mdbfileonpath>checkIfShadowed	1	0.025	0.001	
initialize_population>@(x1)sum(x1(1:12))	8000	0.020	0.020	
getFileInfoForToolstrip	1	0.019	0.013	
workspacefunc>num2complex	10	0.018	0.003	
mdbfileonpath>filepartsWithoutPackages	2	0.016	0.000	
mdbfileonpath>removePackageDirs	2	0.015	0.001	
rmfield	4	0.014	0.014	
fullfile	7	0.010	0.007	

图 5-7



函数名称	调用次数	总时间 (秒)	自用时间* (秒)	总时间图 (深色条带 = 自用时间)
main	1	2.682	2.339	
initialize_population	1	0.200	0.149	
Channel>@(source,data)obj.onCustomEvent(data.Type,data.Data)	8	0.087	0.004	
Channel>Channel.onCustomEvent	8	0.084	0.014	
workspacefunc	2	0.081	0.020	
webwindow>@(source,data)obj.onCustomEvent(data.Type,data.Data)	8	0.065	0.007	
webwindow>webwindow.onCustomEvent	8	0.059	0.048	
mdbfileonpath	1	0.048	0.008	
workspacefunc>getShortValueObjectJ	11	0.038	0.013	
initialize_population>@(x1)sum(x1(1:12))	8000	0.029	0.029	
mdbfileonpath>checkIfShadowed	1	0.025	0.004	
manzu	8000	0.023	0.023	
workspacefunc>num2complex	10	0.021	0.005	
mdbfileonpath>filepartsWithoutPackages	2	0.017	0.000	
mdbfileonpath>removePackageDirs	2	0.017	0.002	
mdbfileonpath>isPackageDirectory	3	0.012	0.004	

图 5-8

5.4 问题四：

5.4.1 模型简介

整数线性规划（Integer Linear Programming, ILP）是一种特殊的线性规划，其特点在于变量被限制为整数值。这种规划问题在数学上可以表示为在一组线性约束条件下，优化（最大化或最小化）一个线性目标函数，同时所有决策变量都必须是整数。其相关模型表示为：

$$minf^T \tag{22}$$

$$ss.t.\begin{cases} A \cdot X \leq b \\ Aeq \cdot X = beq \\ lb \leq X \leq ub \end{cases} \tag{23}$$

5.4.2 模型建立

SC60 未来 30 天内的出勤人数情况，基于问题二的预测方法，使用问题三的模型求解，仅是将正式工 60 名的条件改为 200 名，这里不在赘述。求解结果如下

表 5-9 部分排班情况

分拣中心	日期	班次	正式工	临时工
'SC60'	2023/12/1	'0-8'	60	285
'SC60'	2023/12/1	'5-13'	28	45
'SC60'	2023/12/1	'8-16'	24	37
'SC60'	2023/12/1	'12-20'	1	27
'SC60'	2023/12/1	'14-22'	8	68
'SC60'	2023/12/1	'16-24'	71	279
'SC60'	2023/12/2	'0-8'	10	214
'SC60'	2023/12/2	'5-13'	2	8

根据这个出勤情况的求解结果，我们来建立 0-1 整数规划模型。

Step1:确定决策变量

为了保证每名正式工的出勤率不高于 85%，我们将正式工分成七个班次，最多每连续工作六天休息一天，轮流休息（例如：第一天是第 1 班的人休息，第二天是第 2 班的人休息……），我们设每个班次的人员数为：

$$\mathrm{n_i}(= 0,1,2,\dots,7) \tag{24}$$

Step2:确定目标函数：

如果班次上班的人数越少，则我们考虑可以将不上班的人数加进来，一方面拉低出勤率，另一方面调整出勤率，让出勤率尽量均衡，所以我们考虑将目标函数设为：

$$f = \sum_{i=1}^{7} n_i \tag{25}$$

Step3:确定约束条件：

题目给出两个两个条件：一是每名正式工的出勤率不能高于 85%；二是每名正式工连续出勤天数不能超过 7 天。

- 对于条件一：在这里，我们如果将班次设成 7 班，则会有 5 班人出勤率高于 85%，在这里我们先不考虑这个条件，在后面删减出勤次数的时候，在达成这个条件。

- 对于条件二：每名正式工不能连续 7 天出勤：我们的排班的建立方式已满足条件。

- 对于排班，我们要保证每天高于所需人数：

$$\sum_{1\le i\le 7,\ i\neq k} n_i \ge s_j \tag{26}$$

其中s_j表示第 j 天出勤的最小正式工人数，即模型三所预测的人数；k 则为第 j 天不上班的班次。

- 所有班次的总人数不能超过正式工人数：

$$\sum_{i=1}^{7} n_i \le 200 \tag{27}$$

5.4.3 整数线性规划模型的求解

根据以上思路利用，我们 MATLAB 编程求解，使用整数线性规划 intlinprog 函数代，码详见附录 5。得出未来 30 天的正式工各班次人数情况如表所示：

表 5-10 正式工各班次人数情况

班次	1	2	3	4	5	6	7
人数	13	29	46	69	19	16	8

5.4.4 数据计算

首先，在我们对每个班次正式工的时间段排班，我们考虑循环使用正式工，（例如：该班次有 9 人，该班次第一天需要上班 7 人，第二天需要上班 8 人，则第一天上班的正式工为正式工 1～正式工 7，第二天上班的正式工为正式工 8、正式工 9、正式工 1～正式工 6）这样循环使用，能保证同一班次的正式工的出勤率相差很小，出勤天数相差不超过 1 天，并且不需要考虑具体的编号排班问题，这里为了方便计算，我们将这个班次的人均出勤率估计为该班次的总出勤人次数/该班次人数×30 天。

接下来我们来解决出勤率高于 85%的问题，由于我们是整数规划，约束条件中，上班所需人数小于等于实际上班人数，所以会每天可能存在一些名额，是多出来的上班人员数，这些多出的数，

我们下面称为休息名额。我们优先将休息名额分配给出勤率高于 85%的班次，让每名正式工的出勤率不高于 85%。

然后，为了让总出勤人次数尽可能少，我们要尽量将休息名额使用完，并且保证出勤率尽量均衡。我们考虑将每一天的休息名额，按照班次的权值（每班次的人数/总人数），分配给每个班次，分配时，为了严格满足条件，我们采用对休息名额向下取整的方式分配。这里为了防止极端班次人员数情况的出现，我们优先分配权值大的班次。

最后，在原始排班的基础上，删减去每个班次每天的休息名额，得到每天每个班次的上班情况。然后采用循环排班的方式，我们得到 200 名正式工的班次出勤计划。

表 5-11 问题四优化结果（部分）

分拣中心	日期	班次	出勤员工
SC60	2023/12/1	0_8	正式工 1
SC60	2023/12/1	0_8	正式工 2
SC60	2023/12/1	0_8	正式工 3
SC60	2023/12/1	0_8	正式工 4
SC60	2023/12/1	0_8	正式工 5
SC60	2023/12/1	0_8	正式工 6

5.4.5 模型分析及评价

由于临时工一般人员对象不固定，所以这里我们不再对临时工具体排班计划作排班，只给出每个班次时段临时工的人数，每天编号依次从 1 开始。

本模型结合鲸鱼优化算法、整数线性规划及一些数学计算，得到一个合理的排班计划，且每名正式工出勤率尽量均衡，正式工的出勤率方差达到了 0.769，（计算详情见附件 5）。由于方便计算处理的关系，休息名额的利用率并没有完全 100%，导致排班总人数比模型三给出的结果多了几十个人数次，这里还可以才考虑采用鲸鱼优化算法、模拟退火或者遗传算法这类启发式算法，对整个月的每个班次或者每人的上班情况进行估计，结合 30 天的约束条件，进一步加强结果。

六、模型总结

6.1 模型的优点

（1）LSTM 神经网络：

LSTM（长短期记忆网络）特别适合于处理和预测时间序列数据，因为它能够学习并记住长期的依赖关系。这对于货量数据这类具有时间序列特性的数据非常有利，能够捕捉到数据随时间变化的趋势和周期性。应用在货量预测上，LSTM 可以考虑到历史数据对未来预测的影响，从而提高预测的准确性。

（2）图论和中心性分析：

通过图论中的连通分量分析和中心性计算（度中心性、中间中心性、特征向量中心性），能够深入理解网络结构和节点间的相互关系。这有助于识别出网络中重要的分拣中心，以及它们在整个货物运输网络中的作用，为后续的货量预测和人员配置提供了重要的决策支持。

（3）主成分分析（PCA）：

PCA 能够将多个相关变量转化为少数几个主成分，这些主成分能够保留原始数据的大部分变异性，从而简化了数据结构。在这里，PCA 被用于将三个中心性指标降维为一个综合指标，便于评估和排序分拣中心的重要性。

（4）双目标优化约束模型和 0-1 整数规划：

双目标优化能够同时考虑多个目标，如最小化人天数和均衡排班，从而得出更加全面和实用的解决方案。0-1 整数规划则确保了排班计划的可行性和实际操作性，特别适用于需要明确决策（如是否出勤）的情况。

（5）蒙特卡洛方法和鲸鱼算法：

蒙特卡洛方法通过随机抽样来估算问题的解，适用于复杂系统的模拟和优化。在这里，它被用于为优化问题提供初始解。鲸鱼算法作为一种启发式优化算法，能够在复杂的解空间中高效地搜索最优解，适用于解决此类排班优化问题。

6.2 模型的缺点

（1）LSTM 神经网络：

LSTM 模型相对复杂，需要更多的计算资源。由于其循环计算的性质和参数的数量，训练时间可能会很长，特别是在大数据集上。此外，尽管 LSTM 设计用来缓解梯度消失问题，但在处理非常长的序列时仍然可能受到影响。同时，它也可能遭受梯度爆炸问题。

（2）图论和中心性分析：

中心性分析可能过于简化网络的复杂性，忽略了节点间更复杂的交互模式。例如，度中心性仅考虑节点的直接连接，可能无法全面反映节点在网络中的实际重要性。此外，对于大型网络，计算中心性指标可能非常耗时。

（3）主成分分析（PCA）：

PCA 在降维过程中可能会丢失一些重要的数据信息,因为它只保留了数据中的主要成分。此外，PCA 对异常值敏感，可能会影响结果的准确性。而且，PCA 基于数据的线性关系，对于处理非线性关系的数据可能效果不理想。

（4）优化模型（如双目标规划）：

多目标优化问题往往复杂且难以求解，特别是当目标之间存在冲突时。此外，加权和方法等传统方法在处理多目标优化时可能引入主观性，并且可能无法充分探索整个解空间。

（5）蒙特卡洛方法：

蒙特卡洛模拟的结果稳定性可能较差，因为它基于随机抽样。此外，为了达到足够的精度，可能需要进行大量的模拟，这会增加计算成本。

（6）鲸鱼算法：

作为一种启发式优化算法，鲸鱼算法可能陷入局部最优解，而无法找到全局最优解。此外，算法的性能和收敛速度可能受到参数设置的影响。

（7）0-1 整数规划：

对于大规模的 0-1 整数规划问题，求解可能会变得非常复杂和耗时。此外，随着问题规模的增加，解空间呈指数级增长，使得全局优化变得困难。

参考文献

[1]叶晓龙,罗瑞,刘金培,等. 基于 X11-WT-LSTM 的物流货运量多尺度组合预测研究[J]. 武汉理工大学学报（信息与管理工程版）,2022,44(2):263-269. DOI:10.3963/j.issn.2095-3852.2022.02.015.

[2]杨丽,吴雨茜,王俊丽,等. 循环神经网络研究综述[J]. 计算机应用,2018,38(z2):1-6,26.

[3]任晓龙,吕琳媛. 网络重要节点排序方法综述[J]. 科学通报,2014,59(13):1175-1197. DOI:10.1360/972013-1280.

[4]林海明,杜子芳. 主成分分析综合评价应该注意的问题[J]. 统计研究,2013,30(8):25-31. DOI:10.3969/j.issn.1002-4565.2013.08.004.

[5]孟红云,张小华,刘三阳. 用于约束多目标优化问题的双群体差分进化算法[J]. 计算机学报,2008,31(2):228-235. DOI:10.3321/j.issn:0254-4164.2008.02.006.

[6]李雅丽,王淑琴,陈倩茹,等. 若干新型群智能优化算法的对比研究[J]. 计算机工程与应用,2020,56(22):1-12. DOI:10.3778/j.issn.1002-8331.2006-0291.

[7]褚鼎立,陈红,王旭光. 基于自适应权重和模拟退火的鲸鱼优化算法[J]. 电子学报,2019,47(5):992-999. DOI:10.3969/j.issn.0372-2112.2019.05.003.

附录

附录 1

介绍支撑材料列表

结果表 1.csv ：问题一中对每天货量数据的预测结果

结果表 2.csv ：问题一中对每小时货量的预测结果

结果表 3.csv ：问题二中转换线路后的每天货量预测结果

结果表 4.csv ：问题二中转换线路后的每小时货量预测结果

结果表 5.csv ：问题三中每个班次的正式工与临时工人数规划结果

结果表 6.csv ：问题四的人员排班优化结果

对问题二总网络中不同连通分量的重要性数据

连通分量 SC1 的中心性.xlsx

连通分量 SC3 的中心性.xlsx

连通分量 SC10 的中心性.xlsx

连通分量 SC36 的中心性.xlsx

连通分量 SC53 的中心性.xlsx

连通分量重要性.xlsx

问题一与问题二使用的平均小时比

问 1 小时比 wi 权值.xlsx

评价问题一预测结果的标准均方误差表

问 1 均方误差.xlsx

附录 2

介绍：MATLAB 编写的 LSTM 神经网络代码 预测问题 1

```
data=readtable('D:\zhuomian\mathercup\附件 1-1.xlsx')
% 创建一个空的数据表
newTable1 = table();
newTable2 = table();
% 获取数据表的列名
columnNames = data.Properties.VariableNames;
for j=1:1:57
```

```
    columnName = columnNames{j}; % 如果你知道列的名称，也可以直接赋值给 columnName

    % 获取第一列的数据
    columnData = data.(columnName);
    result = columnData; % 这里应该是您的模拟数据生成代码
    [error,t_sim2]=lstm11(result);
    newTable1 = [newTable1, table(t_sim2, 'VariableNames', {columnName})];
    newTable2 = [newTable2, table(error, 'VariableNames', {columnName})];

end

%writetable(newTable1, '结果 1-1.csv');
writetable(newTable2, '均方误差表.csv');



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
函数部分
function [error2,a] = lstm11(result)

num_samples=length(result);
kim=10;% 延时步长
zim=1;   %跨越 1 个点

% 划分数据集
for i= 1:num_samples-kim-zim+1
    res(i, :)=[reshape(result(i:i+kim-1),1,kim),result(i+kim+zim-1)];
end

temp=1:1:90;
P_train=res(temp(1:80),1:10)';
T_train=res(temp(1:80),11)';
M=size(P_train,2);

P_test=res(temp(81:end),1:10)';
T_test=res(temp(81:end),11)';
N=size(P_test,2);
 %Index in position 1 exceeds array bounds. Index must not exceed 90.
%%数据归一化
[P_train,ps_input]=mapminmax(P_train,0,1);
P_test=mapminmax('apply',P_test,ps_input);

[t_train,ps_output]=mapminmax(T_train,0,1);
t_test=mapminmax('apply',T_test,ps_output);
%%数据平铺
P_train=double(reshape(P_train,10,1,1,M));
```

```
P_test=double(reshape(P_test,10,1,1,N));


t_train=t_train';
t_test=t_test';

%%数据格式转换
for i = 1:M
    p_train{i,1}=P_train( :, :,1,i);
end
for i = 1:N
    p_test{i,1}=P_test( :, :,1,i);
end
% LSTM 网络结构定义
layers = [
    sequenceInputLayer(10)

    lstmLayer(10, 'OutputMode', 'last') %lstm 层
    reluLayer    %激活层

    fullyConnectedLayer(1)  %全连接层
    regressionLayer
];

% 训练选项（保持不变）
options = trainingOptions('adam', ...    %adam 梯度下降算法
    'MaxEpochs', 500, ...               %最大训练次数
    'InitialLearnRate', 8e-3, ...       %初始学习率
    'LearnRateSchedule', 'piecewise', ...   %学习率下降
    'LearnRateDropPeriod', 480, ...      %经 480 次训练后学习率为 0.008*0.1
    'LearnRateDropFactor', 0.1, ...       %学习率下降因子
    'Verbose', false );

% 训练 LSTM 网络（保持不变）
net = trainNetwork(p_train, t_train, layers, options);

% 仿真预测
t_sim1=predict(net,p_train);

t_sim2=predict(net,p_test);



error2=sqrt(sum((t_sim2'-t_test).^2)./N);
%反归一化
```

```
T_sim1=mapminmax('reverse',t_sim1,ps_output);
T_sim2=mapminmax('reverse',t_sim2,ps_output) ;

%均方根误差
error1=sqrt(sum((T_sim1'-T_train).^2)./M);
a=[];
end
```

附录 3

介绍：运用 python 编写代码，模拟线路，求解问题 2；运用 RStudio 进行连通分量的主成分分析

Python

```
#权重计算
# _*_coding : utf-8_*_
# @Time : 2024/4/14 7:27
# @Auther :jiangzhuo
# @File :  问二权重计算
# @Project :  第二问.py
import pandas as pd
import networkx as nx

future_edges_df = pd.read_csv('D:/zhuomian/2024 年 MathorCup 数学应用挑战赛赛题/C 题/附件/附件 4.csv',encoding='gbk')
G_directed = nx.from_pandas_edgelist(
    future_edges_df,
    source='始发分拣中心',
    target='到达分拣中心',
    create_using=nx.DiGraph()
)
G = G_directed.to_undirected()

centrality_results = {}

# 获取所有的连通分量
connected_components = nx.connected_components(G)

# 遍历所有的连通分量
for component in connected_components:
    subgraph = G.subgraph(component).copy()
```

```
    # 为当前连通分量计算中心性
    degree_centrality = nx.degree_centrality(subgraph)
    betweenness_centrality = nx.betweenness_centrality(subgraph)
    eigenvector_centrality = nx.eigenvector_centrality(subgraph)

    # 创建一个 DataFrame 来存储当前连通分量的中心性结果
    component_df = pd.DataFrame({
        '节点名称': list(degree_centrality.keys()),
        '度中心性': list(degree_centrality.values()),
        '中间中心性': [betweenness_centrality[node] for node in degree_centrality.keys()],
        '特征向量中心性': [eigenvector_centrality[node] for node in degree_centrality.keys()]
    })

    # 将当前连通分量的 DataFrame 存储到字典中，以连通分量中某个节点的标识作为键
    centrality_results[str(list(component)[0])] = component_df

# 输出每个连通分量的中心性结果
for component_key, component_df in centrality_results.items():
    print(f"连通分量 {component_key} 的中心性结果:")
    component_df.to_excel(f'D:/zhuomian/mathercup/第二问/连通分量 {component_key} 的中心性.xlsx', index=False)
    print(component_df)
    print()

#进行线路模拟
import pandas as pd
import networkx as nx
edges_df = pd.read_csv('D:/zhuomian/2024 年 MathorCup 数学应用挑战赛赛题/C 题/附件/附件3.csv',encoding='gbk')

G = nx.from_pandas_edgelist(
    edges_df,
    source='始发分拣中心',
    target='到达分拣中心',
    edge_attr='货量',
```

```
    create_using=nx.DiGraph()
)
future_edges_df = pd.read_csv('D:/zhuomian/2024 年 MathorCup 数学应用挑战赛赛题/C 题/附件/附件 4.csv',encoding='gbk')


# 更新网络图
for _, row in future_edges_df.iterrows():
    src, dst = row['始发分拣中心'], row['到达分拣中心']
    # 检查边是否存在，存在则更新，不存在则添加
    if G.has_edge(src, dst):
        # 使用与该目的地或来源相关的所有边的平均货量
        specific_avg = edges_df[
            (edges_df['始发分拣中心'] == src) | (edges_df['到达分拣中心'] == dst)
        ]['货量'].mean()
        G[src][dst]['货量'] = specific_avg
    else:
        # 对于不存在的边，也计算与目的地或来源相关的边的平均货量
        specific_avg = edges_df[
            (edges_df['始发分拣中心'] == src) | (edges_df['到达分拣中心'] == dst)
        ]['货量'].mean()
        G.add_edge(src, dst, 货量=specific_avg if not pd.isna(specific_avg) else 0)

# 打印图中所有边的信息
for edge in G.edges(data=True):
    print(edge)

daily_prediction = pd.read_csv('D:/zhuomian/2024 年 MathorCup 数学应用挑战赛赛题/C 题/结果/结果表 1.csv',encoding='gbk')

print(daily_prediction)

# 转换日期格式，确保一致性
daily_prediction['日期'] = pd.to_datetime(daily_prediction['日期'])

# 创建空 DataFrame 用于存储更新后的货量
```

```
updated_loads = daily_prediction.copy()

# 将 DataFrame 转换为每天一个字典，便于操作
daily_load_dict = daily_prediction.groupby('日期').apply(lambda x: dict(zip(x['分拣中心'], x['货量
'']))).to_dict()

#加权重
weight= pd.read_excel('D:/zhuomian/mathercup/第二问/连通分量重要性.xlsx')
weight=weight.set_index('节点名称')



# 遍历每天，调整货量
for date, loads in daily_load_dict.items():
    # 遍历图中每条边，调整货量
    for (src, dst, data) in G.edges(data=True):
        # 确保源和目的节点都在当天的预测数据中
        if src in loads and dst in loads:
            # 计算要转移的货量
            transfer_amount = data['货量']

            # 从源分拣中心减去货量
            loads[src] = max(0, loads[src] - transfer_amount*weight.at[src,'权重'])  # 避免负数

            # 向目的地分拣中心添加货量
            loads[dst] += transfer_amount*weight.at[src,'权重']

    # 更新 DataFrame
    for center, new_load in loads.items():
        new_load = int(new_load)
        updated_loads.loc[(updated_loads['日期'] == date) & (updated_loads['分拣中心'] == center), '
货量'] = new_load

updated_loads.to_excel('D:/zhuomian/结果 2-1(new).xlsx',index=False)
```

```
#Rstudio 进行主成分分析
#主成分分析，多变量降维处理
#强相关的指标组合成新的互不相观的新指标
data=read.table()
pca=princomp(data,cor=TRUE,subset=c(1:10))
##cor:TRUE 表示使用相关矩阵求主成分，否则使用协方差矩阵
#scores：表示是否计算每个主成分上的分数
#subset：选取数据部分子集（通常取一般多或）缺省为全取
summary(pca,loadings=TRUE)  #loadings 表示是否进行相关矩阵的特征向量的输出
#自上而下依次是标准差、指标贡献率、累计贡献率  #只要累计贡献率达到 80%或 85%以上就行了
#loadings 是特征向量的输出 即 X1=W 指标一+W 指标二。。。。

screeplot(pca,type='lines') #画出贡献率的折线图

#调用
pca$loadings[,1:3] #前三个主成分的特征向量
pca$scores[,1:3]#三个主成分的综合得分
```

附录 4

介绍：运用 matlab 编写代码，建立蒙特卡洛算法结合鲸鱼算法的优化模型，利用数学期望、线性加权和法等，对 30 天 57 个站点人员进行排班

```
% 鲸鱼优化算法——WOA
clear all
clc
%% 参数列表
load("lastdata.mat");
column_values=hs(2:721,1:57);
matrixOfVectors = cell(30, 57);
for i = 1:30
    for j = 1:57
        vector = column_values(24*(i-1)+1:24*i,j);
        % 将各站点每天每小时货量存储在 cell 数组的相应位置
        matrixOfVectors{i, j} = vector;
    end
end
```

```
whale_num=80;                                                    % 鲸鱼数量
dim=12;                                                          % 维度：六个班次，每个班次两种工种，前六个为第 1-6 班次正式工，后六个为第 1-6 班次临时工
G=500;                                                           % 迭代次数
F=@(x1,input_vector2) 0.8*sum(x1)+0.2*fangcha(x1,input_vector2); % 待求解函数句柄，总人数权重 0.7，方差权重 0.3
min_fitness_sum=zeros(30,57);%存储各站点每天总工人数
matrixOfVectors_workers = cell(30, 57);   %存储各站点每天各细分班次工人数
for i = 1:30
    for j = 1:57
        vector = zeros(1,12);
        matrixOfVectors_workers{i, j} = vector;
    end
end
min_fangcha=zeros(30,57);%存储各站点每天实际小时人效方差
for i1=1:30   %以天数循环
    for j1=1:57    %以站点循环
        %% 种群初始化
        init_whale_group =initialize_population(whale_num, 6, 60,20,300,matrixOfVectors{i1, j1});
        whale_group=init_whale_group;
        whale_min=zeros(G,dim);% 最优鲸鱼所在位置
        min_fitness=inf(G,1);
        %% 迭代主体
        for t=1:G
            %对超出范围的鲸鱼的处理
            for i = 1:whale_num
                whale_group=round(whale_group);
                while sum(whale_group(i, 1:6)) > 60 ||~all(whale_group(i, 1:12) > 0)
                    whale_group(i, 1:12)=reinitialize_individual(6, 60,20,300,matrixOfVectors{i1, j1}) ;
                end
                if ~manzu(whale_group(i, 1:12),matrixOfVectors{i1, j1})
                    whale_group(i, 1:12)=reinitialize_individual2(6,whale_group(i, 1:12),matrixOfVectors{i1, j1}) ;
                end
            end
```

```
whale_fitness=zeros(whale_num,1);
 % 计算每个鲸鱼的函数值
for i=1:whale_num
    whale_fitness(i,1)=F(whale_group(i,1:12),matrixOfVectors{i1, j1});
end
% 记录最好位置
[whale_min_fitness,index_min]=min(whale_fitness);
if(whale_min_fitness<min_fitness)
    min_fitness(t,1)=whale_min_fitness;
    whale_min(t,:)=whale_group(index_min,:);
else
    min_fitness(t,1)=min_fitness(t-1,:);
    whale_min(t,:)=whale_min(t-1,:);
end
% 计算参数
a1=2-2*t/G;
a2=-1-t/G;

for i=1:whale_num
    A=2*a1*rand()-a1;
    C=2*rand();
    b=1;
    l=(a2-1)*rand+1;
    p=rand();
    for j=1:dim
        if(p<0.5)
            if(abs(A)<1)
                % 一、包围猎物
                % 计算移动幅度
                D=abs(C*whale_min(t,j)-whale_group(i,j));
                % 按照移动幅度包围猎物（更新位置，即更新一次 x1,x2）
                whale_group(i,j)=whale_min(t,j)-A*D;
                whale_group(i, j) = round(whale_min(t, j) - A * D); % 取整操作
            else
                % 二、随机搜索猎物
                whale_rand_index=floor(whale_num*rand()+1);
                whale_rand=whale_group(whale_rand_index,:);
```

```
                    D=abs(C*whale_rand(j)-whale_group(i,j));
                    whale_group(i,j)=whale_rand(j)-A*D;
                    whale_group(i, j) = round(whale_rand(j) - A * D); %  取整操作
                end
            else
                % 三、螺旋运动(发泡网攻击)
                D_p=abs(whale_min(t,j)-whale_group(i,j));
                whale_group(i,j)=whale_min(t,j)+D_p*exp(b.*l)*cos(l.*2*pi);
                whale_group(i, j) = round(whale_min(t, j) + D_p * exp(b * l) * cos(l * 2 *
pi)); %  取整操作
            end
        end
        %对超出范围的鲸鱼的处理
        for m = 1:whale_num
            whale_group=round(whale_group);
            while sum(whale_group(m, 1:6)) > 60 ||~all(whale_group(m, 1:12) > 0)
                whale_group(m, 1:12)=reinitialize_individual(6,
60,20,300,matrixOfVectors{i1, j1}) ;
            end
            if ~manzu(whale_group(i, 1:12),matrixOfVectors{i1, j1})
                whale_group(i, 1:12)=reinitialize_individual2(6,whale_group(i,
1:12),matrixOfVectors{i1, j1}) ;
            end
            whale_group;
        end
    end
end
matrixOfVectors_workers{i1,j1}=whale_group(index_min,:);%记录当前站点当天各细分班
次工人数
min_fangcha(i1,j1)=fangcha(matrixOfVectors_workers{i1,j1},matrixOfVectors{i1, j1});%记
录当前站点当天实际小时人效方差
min_fitness_sum(i1,j1)=sum(matrixOfVectors_workers{i1,j1});%记录各站点每天总工人数
  end
end


%%%%%%%%%%%%%%%%
```

```
%函数部分
function population = initialize_population(pop_size, var_num, x_constraint, y_min, y_max,
input_vector)
    %%函数说明
    %初始化鲸鱼种群（各细分班次工人数）
    %%参数说明
    %pop_size  需要的鲸鱼数
    %var_num  每天班次数
    %x_constraint    正式工总数限制
    %y_min, y_max    临时工随机数生成时的上下界
    %input_vector    每小时需处理货量
    %  评估函数
    F = @(x1) sum(x1(1:12));
    %  初始化种群矩阵
    population = zeros(pop_size, 2*var_num);
    for i = 1:pop_size
        %  初始化最优解和最优评估值
        best_solution = zeros(1, 2*var_num);
        best_eval = Inf;

        %  蒙特卡洛迭代次数
        mc_iterations = 100;

        for iter = 1:mc_iterations
            x_solution=zeros(1,var_num);%六个班次正式工初始化为零
            x_solution=randi([0,x_constraint-40],1,var_num-1);
            x_solution(1,6)=x_constraint-sum(x_solution(1:5));
            while x_solution(1,6)<0
                x_solution=randi([0,x_constraint-40],1,var_num-1);
                x_solution(1,6)=x_constraint-sum(x_solution(1:5));
            end
            y_solution = randi([y_min, y_max], 1, var_num); %随机生成各班次临时工人数
            solution = [x_solution, y_solution];
            %  检查解是否满足约束条件
            if ~manzu(solution, input_vector)     %不满足就计算所需增加临时工人数，使解满足
条件
```

```
temp_add_y=zeros(1,24);
colVec = solution';
A=[25 0 0 0 0 0 20 0 0 0 0 0;
    25 0 0 0 0 0 20 0 0 0 0 0;
    25 0 0 0 0 0 20 0 0 0 0 0;
    25 0 0 0 0 0 20 0 0 0 0 0;
    25 0 0 0 0 0 20 0 0 0 0 0;
    25 25 0 0 0 0 20 20 0 0 0 0;
    25 25 0 0 0 0 20 20 0 0 0 0;
    25 25 0 0 0 0 20 20 0 0 0 0;
    0 25 25 0 0 0 0 20 20 0 0 0;
    0 25 25 0 0 0 0 20 20 0 0 0;
    0 25 25 0 0 0 0 20 20 0 0 0;
    0 25 25 0 0 0 0 20 20 0 0 0;
    0 25 25 25 0 0 0 20 20 20 0 0;
    0 0 25 25 0 0 0 0 20 20 0 0;
    0 0 25 25 25 0 0 0 20 20 20 0;
    0 0 25 25 25 0 0 0 20 20 20 0;
    0 0 0 25 25 25 0 0 0 20 20 20;
    0 0 0 25 25 25 0 0 0 20 20 20;
    0 0 0 25 25 25 0 0 0 20 20 20;
    0 0 0 25 25 25 0 0 0 20 20 20;
    0 0 0 0 25 25 0 0 0 0 20 20;
    0 0 0 0 25 25 0 0 0 0 20 20;
    0 0 0 0 0 25 0 0 0 0 0 20;
    0 0 0 0 0 25 0 0 0 0 0 20;];
result=A*colVec;
comparison_result = result > input_vector;
B=[1 0 0 0 0 0 1 0 0 0 0 0;
    1 0 0 0 0 0 1 0 0 0 0 0;
    1 0 0 0 0 0 1 0 0 0 0 0;
    1 0 0 0 0 0 1 0 0 0 0 0;
    1 0 0 0 0 0 1 0 0 0 0 0;
    1 1 0 0 0 0 1 1 0 0 0 0;
    1 1 0 0 0 0 1 1 0 0 0 0;
    1 1 0 0 0 0 1 1 0 0 0 0;
    0 1 1 0 0 0 0 1 1 0 0 0;
    0 1 1 0 0 0 0 1 1 0 0 0;
```

```
                    0 1 1 0 0 0 0 1 1 0 0 0;
                    0 1 1 0 0 0 0 1 1 0 0 0;
                    0 1 1 1 0 0 0 1 1 1 0 0;
                    0 0 1 1 0 0 0 0 1 1 0 0;
                    0 0 1 1 1 0 0 0 1 1 1 0;
                    0 0 1 1 1 0 0 0 1 1 1 0;
                    0 0 0 1 1 1 0 0 0 1 1 1;
                    0 0 0 1 1 1 0 0 0 1 1 1;
                    0 0 0 1 1 1 0 0 0 1 1 1;
                    0 0 0 1 1 1 0 0 0 1 1 1;
                    0 0 0 0 1 1 0 0 0 0 1 1;
                    0 0 0 0 1 1 0 0 0 0 1 1;
                    0 0 0 0 0 1 0 0 0 0 0 1;
                    0 0 0 0 0 1 0 0 0 0 0 1;];

                for j=1:24
                    if ~comparison_result(j,1)
                        temp_add_y(1,j)=ceil(ceil((input_vector(j,1)-
result(j,1))/20)/(sum(B(j,:))/2));
                    end
                end
                solution(1,var_num+1)=solution(1,var_num+1)+max(temp_add_y(1:8))+10;
                solution(1,var_num+2)=solution(1,var_num+2)+max(temp_add_y(6:13))+10;
                solution(1,var_num+3)=solution(1,var_num+3)+max(temp_add_y(9:16))+10;
                solution(1,var_num+4)=solution(1,var_num+4)+max(temp_add_y(13:20))+10;
                solution(1,var_num+5)=solution(1,var_num+5)+max(temp_add_y(15:22))+10;
                solution(1,var_num+6)=solution(1,var_num+6)+max(temp_add_y(17:24))+10;
            end

            % 评估当前解
            current_eval = F(solution);

            % 如果当前解更优，则更新最优解
            if current_eval < best_eval
                best_solution = solution;
                best_eval = current_eval;
            end
        end
```

```
            % 将找到的最优解放入种群中
            population(i, :) = best_solution;
        end
end

function fc = fangcha(input_vector1,input_vector2)
%%函数说明
%计算当天实际小时人效方差
%%参数说明
%input_vector1   一天各班次正式工临时工人数
%input_vector2   各小时需处理货量
A=[1 0 0 0 0 0 1 0 0 0 0 0;
    1 0 0 0 0 0 1 0 0 0 0 0;
    1 0 0 0 0 0 1 0 0 0 0 0;
    1 0 0 0 0 0 1 0 0 0 0 0;
    1 0 0 0 0 0 1 0 0 0 0 0;
    1 1 0 0 0 0 1 1 0 0 0 0;
    1 1 0 0 0 0 1 1 0 0 0 0;
    1 1 0 0 0 0 1 1 0 0 0 0;
    0 1 1 0 0 0 0 1 1 0 0 0;
    0 1 1 0 0 0 0 1 1 0 0 0;
    0 1 1 0 0 0 0 1 1 0 0 0;
    0 1 1 0 0 0 0 1 1 0 0 0;
    0 1 1 1 0 0 0 1 1 1 0 0;
    0 0 1 1 0 0 0 0 1 1 0 0;
    0 0 1 1 1 0 0 0 1 1 1 0;
    0 0 1 1 1 0 0 0 1 1 1 0;
    0 0 0 1 1 1 0 0 0 1 1 1;
    0 0 0 1 1 1 0 0 0 1 1 1;
    0 0 0 1 1 1 0 0 0 1 1 1;
    0 0 0 1 1 1 0 0 0 1 1 1;
    0 0 0 0 1 1 0 0 0 0 1 1;
    0 0 0 0 1 1 0 0 0 0 1 1;
    0 0 0 0 0 1 0 0 0 0 0 1;
    0 0 0 0 0 1 0 0 0 0 0 1;];
B=A*(input_vector1');%各小时工作总人数
```

```
C=input_vector2./B;%小时人效
s=sum(C)/24;%小时人效均值
C=C-s;
C=C.*C;
fc=sum(C)/24;%当天实际小时人效方差
end



function all_manzu = manzu(input_vector1,input_vector2)
%%函数说明
%判断每小时可处理货量数是否都多于每小时需处理货量数
%%参数说明
%input_vector1  各班次细分工人数
%input_vector2    每小时需处理货量
colVec = input_vector1';
A=[25 0 0 0 0 0 20 0 0 0 0 0;
    25 0 0 0 0 0 20 0 0 0 0 0;
    25 0 0 0 0 0 20 0 0 0 0 0;
    25 0 0 0 0 0 20 0 0 0 0 0;
    25 0 0 0 0 0 20 0 0 0 0 0;
    25 25 0 0 0 0 20 20 0 0 0 0;
    25 25 0 0 0 0 20 20 0 0 0 0;
    25 25 0 0 0 0 20 20 0 0 0 0;
    0 25 25 0 0 0 0 20 20 0 0 0;
    0 25 25 0 0 0 0 20 20 0 0 0;
    0 25 25 0 0 0 0 20 20 0 0 0;
    0 25 25 0 0 0 0 20 20 0 0 0;
    0 25 25 25 0 0 0 20 20 20 0 0;
    0 0 25 25 0 0 0 0 20 20 0 0;
    0 0 25 25 25 0 0 0 20 20 20 0;
    0 0 25 25 25 0 0 0 20 20 20 0;
    0 0 0 25 25 25 0 0 0 20 20 20;
    0 0 0 25 25 25 0 0 0 20 20 20;
    0 0 0 25 25 25 0 0 0 20 20 20;
    0 0 0 25 25 25 0 0 0 20 20 20;
    0 0 0 0 25 25 0 0 0 0 20 20;
    0 0 0 0 25 25 0 0 0 0 20 20;
```

```
    0 0 0 0 0 25 0 0 0 0 0 20;
    0 0 0 0 0 25 0 0 0 0 0 20;];
result=A*colVec;   %每小时可处理货量数
comparison_result = result > input_vector2;
all_manzu= all(comparison_result);
end


function best_solution = reinitialize_individual(var_num, x_constraint,y_min,y_max,input_vector)
    %%函数说明
    %重新初始化超出范围的鲸鱼
    %%参数说明
    %var_num  每天班次数
    %x_constraint   正式工总数限制
    %y_min, y_max   临时工随机数生成时的上下界
    %input_vector   每小时需处理货量
    %%重新初始化函数与初始化函数大致相同，一些注释省略
    F=@(x1) sum(x1); %  待求解函数句柄
    best_solution = zeros(1, 2*var_num);
        best_eval = Inf;

        %  蒙特卡洛迭代次数
        mc_iterations = 100;

        for iter = 1:mc_iterations
            x_solution=zeros(1,var_num);
            x_solution=randi([0,x_constraint-40],1,var_num-1);
            x_solution(1,6)=x_constraint-sum(x_solution(1:5));
            while x_solution(1,6)<0
                x_solution=randi([0,x_constraint-40],1,var_num-1);
                x_solution(1,6)=x_constraint-sum(x_solution(1:5));
            end
            y_solution = randi([y_min, y_max], 1, var_num);
            solution = [x_solution, y_solution];
            %  检查解是否满足约束条件
            if ~manzu(solution, input_vector)
                temp_add_y=zeros(1,24);
```

```
colVec = solution';
A=[25 0 0 0 0 0 20 0 0 0 0 0;
    25 0 0 0 0 0 20 0 0 0 0 0;
    25 0 0 0 0 0 20 0 0 0 0 0;
    25 0 0 0 0 0 20 0 0 0 0 0;
    25 0 0 0 0 0 20 0 0 0 0 0;
    25 25 0 0 0 0 20 20 0 0 0 0;
    25 25 0 0 0 0 20 20 0 0 0 0;
    25 25 0 0 0 0 20 20 0 0 0 0;
    0 25 25 0 0 0 0 20 20 0 0 0;
    0 25 25 0 0 0 0 20 20 0 0 0;
    0 25 25 0 0 0 0 20 20 0 0 0;
    0 25 25 0 0 0 0 20 20 0 0 0;
    0 25 25 25 0 0 0 20 20 20 0 0;
    0 0 25 25 0 0 0 0 20 20 0 0;
    0 0 25 25 25 0 0 0 20 20 20 0;
    0 0 25 25 25 0 0 0 20 20 20 0;
    0 0 0 25 25 25 0 0 0 20 20 20;
    0 0 0 25 25 25 0 0 0 20 20 20;
    0 0 0 25 25 25 0 0 0 20 20 20;
    0 0 0 25 25 25 0 0 0 20 20 20;
    0 0 0 0 25 25 0 0 0 0 20 20;
    0 0 0 0 25 25 0 0 0 0 20 20;
    0 0 0 0 0 25 0 0 0 0 0 20;
    0 0 0 0 0 25 0 0 0 0 0 20;];
result=A*colVec;
comparison_result = result > input_vector;
B=[1 0 0 0 0 0 1 0 0 0 0 0;
    1 0 0 0 0 0 1 0 0 0 0 0;
    1 0 0 0 0 0 1 0 0 0 0 0;
    1 0 0 0 0 0 1 0 0 0 0 0;
    1 0 0 0 0 0 1 0 0 0 0 0;
    1 1 0 0 0 0 1 1 0 0 0 0;
    1 1 0 0 0 0 1 1 0 0 0 0;
    1 1 0 0 0 0 1 1 0 0 0 0;
    0 1 1 0 0 0 0 1 1 0 0 0;
    0 1 1 0 0 0 0 1 1 0 0 0;
    0 1 1 0 0 0 0 1 1 0 0 0;
```

```
                    0 1 1 0 0 0 0 1 1 0 0 0;
                    0 1 1 1 0 0 0 1 1 1 0 0;
                    0 0 1 1 0 0 0 0 1 1 0 0;
                    0 0 1 1 1 0 0 0 1 1 1 0;
                    0 0 1 1 1 0 0 0 1 1 1 0;
                    0 0 0 1 1 1 0 0 0 1 1 1;
                    0 0 0 1 1 1 0 0 0 1 1 1;
                    0 0 0 1 1 1 0 0 0 1 1 1;
                    0 0 0 1 1 1 0 0 0 1 1 1;
                    0 0 0 0 1 1 0 0 0 0 1 1;
                    0 0 0 0 1 1 0 0 0 0 1 1;
                    0 0 0 0 0 1 0 0 0 0 0 1;
                    0 0 0 0 0 1 0 0 0 0 0 1;];

                for j=1:24
                    if ~comparison_result(j,1)
                        temp_add_y(1,j)=ceil(ceil((input_vector(j,1)-
result(j,1))/20)/(sum(B(j,:))/2));
                    end
                end
                solution(1,var_num+1)=solution(1,var_num+1)+max(temp_add_y(1:8))+10;
                solution(1,var_num+2)=solution(1,var_num+2)+max(temp_add_y(6:13))+10;
                solution(1,var_num+3)=solution(1,var_num+3)+max(temp_add_y(9:16))+10;
                solution(1,var_num+4)=solution(1,var_num+4)+max(temp_add_y(13:20))+10;
                solution(1,var_num+5)=solution(1,var_num+5)+max(temp_add_y(15:22))+10;
                solution(1,var_num+6)=solution(1,var_num+6)+max(temp_add_y(17:24))+10;
            end

            % 评估当前解
            current_eval = F(solution);

            % 如果当前解更优，则更新最优解
            if current_eval < best_eval
                best_solution = solution;
                best_eval = current_eval;
            end
        end
end
```

```
function best_solution = reinitialize_individual2(var_num,solution,input_vector)
%%函数说明
%对于没有超出正式工总数的鲸鱼，只对其进行临时工增添，使其小时可处理货量均大于需处理货量
%%参数说明
    %var_num  每天班次数
    %solution  小时可处理货量小于需处理货量的鲸鱼
    %input_vector   每小时需处理货量
%与初始化函数类似，不多做说明
    best_solution = zeros(1, 12);

        temp_add_y=zeros(1,24);
                colVec = solution';
                A=[25 0 0 0 0 0 20 0 0 0 0 0;
                    25 0 0 0 0 0 20 0 0 0 0 0;
                    25 0 0 0 0 0 20 0 0 0 0 0;
                    25 0 0 0 0 0 20 0 0 0 0 0;
                    25 0 0 0 0 0 20 0 0 0 0 0;
                    25 25 0 0 0 0 20 20 0 0 0 0;
                    25 25 0 0 0 0 20 20 0 0 0 0;
                    25 25 0 0 0 0 20 20 0 0 0 0;
                    0 25 25 0 0 0 0 20 20 0 0 0;
                    0 25 25 0 0 0 0 20 20 0 0 0;
                    0 25 25 0 0 0 0 20 20 0 0 0;
                    0 25 25 0 0 0 0 20 20 0 0 0;
                    0 25 25 25 0 0 0 20 20 20 0 0;
                    0 0 25 25 0 0 0 0 20 20 0 0;
                    0 0 25 25 25 0 0 0 20 20 20 0;
                    0 0 25 25 25 0 0 0 20 20 20 0;
                    0 0 0 25 25 25 0 0 0 20 20 20;
                    0 0 0 25 25 25 0 0 0 20 20 20;
                    0 0 0 25 25 25 0 0 0 20 20 20;
                    0 0 0 25 25 25 0 0 0 20 20 20;
                    0 0 0 0 25 25 0 0 0 0 20 20;
                    0 0 0 0 25 25 0 0 0 0 20 20;
```

```
                0 0 0 0 0 25 0 0 0 0 0 20;
                0 0 0 0 0 25 0 0 0 0 0 20;];
            result=A*colVec;
            comparison_result = result > input_vector;
            B=[1 0 0 0 0 0 1 0 0 0 0 0;
                1 0 0 0 0 0 1 0 0 0 0 0;
                1 0 0 0 0 0 1 0 0 0 0 0;
                1 0 0 0 0 0 1 0 0 0 0 0;
                1 0 0 0 0 0 1 0 0 0 0 0;
                1 1 0 0 0 0 1 1 0 0 0 0;
                1 1 0 0 0 0 1 1 0 0 0 0;
                1 1 0 0 0 0 1 1 0 0 0 0;
                0 1 1 0 0 0 0 1 1 0 0 0;
                0 1 1 0 0 0 0 1 1 0 0 0;
                0 1 1 0 0 0 0 1 1 0 0 0;
                0 1 1 0 0 0 0 1 1 0 0 0;
                0 1 1 1 0 0 0 1 1 1 0 0;
                0 0 1 1 0 0 0 0 1 1 0 0;
                0 0 1 1 1 0 0 0 1 1 1 0;
                0 0 1 1 1 0 0 0 1 1 1 0;
                0 0 0 1 1 1 0 0 0 1 1 1;
                0 0 0 1 1 1 0 0 0 1 1 1;
                0 0 0 1 1 1 0 0 0 1 1 1;
                0 0 0 1 1 1 0 0 0 1 1 1;
                0 0 0 0 1 1 0 0 0 0 1 1;
                0 0 0 0 1 1 0 0 0 0 1 1;
                0 0 0 0 0 1 0 0 0 0 0 1;
                0 0 0 0 0 1 0 0 0 0 0 1;];

            for j=1:24
                if ~comparison_result(j,1)
                    temp_add_y(1,j)=ceil(ceil((input_vector(j,1)-
result(j,1))/20)/(sum(B(j,:))/2));
                end
            end
            solution(1,var_num+1)=solution(1,var_num+1)+max(temp_add_y(1:8))+10;
            solution(1,var_num+2)=solution(1,var_num+2)+max(temp_add_y(6:13))+10;
            solution(1,var_num+3)=solution(1,var_num+3)+max(temp_add_y(9:16))+10;
```

```
                solution(1,var_num+4)=solution(1,var_num+4)+max(temp_add_y(13:20))+10;
                solution(1,var_num+5)=solution(1,var_num+5)+max(temp_add_y(15:22))+10;
                solution(1,var_num+6)=solution(1,var_num+6)+max(temp_add_y(17:24))+10;
best_solution=solution;
end
```

附录 5

介绍：运用 matlab 编写代码，在模型 3 的对问题四的求解结果上，利用整数规划，以及分权思想，对人员进行分配排班

```
clc,clear;
%num 是每一班的人数
%n 是每天所需的人
%more 是优先休息完之后，每天多于的可休息人数
%arr 是上班图
%aver 是最终平均出勤率
%sleep 是每班可以休息的名额

x0=zeros(6,1);%初始解

b=ones(30,1);
data = readmatrix('41.xlsx');
b=data(:,7);
b=-b;
b=[b;200];

A=ones(31,7);
for i=1:30
A(i,rem(i+5,7)+1)=0;
end
A=-A;
A(31,:)=ones(1,7);

lb=zeros(1,7);
ub=ones(1,7);
for i=1:7
ub(1,i)=200;
```

```
end
intcon = [1 2 3 4 5 6 7];

f = [1 1 1 1 1 1 1];
disp(size(A)); %  应该输出  [31 6]
disp(length(b)); %  应该输出  31
options = optimoptions('intlinprog', 'MaxTime', 10000000, 'Display', 'iter'); %  设置最大求解时间和显示迭代信息
[num, fvalnew, exitflag, output] = intlinprog(f, intcon,A, b,[],[] ,lb, ub,options);



n=zeros(30,1);%计算所有人满班的情况下
A=ones(30,7);
for i=1:30
A(i,rem(i+5,7)+1)=0;
end
n=A*num;
more=n-data(:,7);



n=zeros(30,1);
A=ones(30,7);
for i=1:30
A(i,rem(i+5,7)+1)=0;
end
n=A*num;
more=n-data(:,7);
cnt=num;
aver=sum(n)/200;%这是人均出勤率
arr=zeros(7,30);
for j=1:7
for i=1:30
if rem(i+1,7)==rem(j,7)
continue;
end
arr(j,i)=num(j,1);
end
```

```
end
arr=[arr,num];



%对第一班
for j=1:30
if cnt(1,1)==0
break;
end
if i==7||i==14||i==21||i==28
continue;
end
while(cnt(1,1)>0&&more(j,1)>0)
cnt(1,1)=cnt(1,1)-1;
arr(1,j)=arr(1,j)-1;
more(j,1)=more(j,1)-1;
end
end
%4
for j=1:30
if cnt(4,1)==0
break;
end
if j==10||j==17||j==24||j==3
continue;
end
while(cnt(4,1)>0&&more(j,1)>0)
cnt(4,1)=cnt(4,1)-1;
arr(4,j)=arr(4,j)-1;
more(j,1)=more(j,1)-1;
end
end
%5
for j=1:30
if cnt(5,1)==0
break;
end
if j==11||j==18||j==25||j==4
```

```
continue;
end
while(cnt(5,1)>0&&more(j,1)>0)
cnt(5,1)=cnt(5,1)-1;
arr(5,j)=arr(5,j)-1;
more(j,1)=more(j,1)-1;
end
end
%6
for j=1:30
if cnt(6,1)==0
break;
end
if j==12||j==19||j==26||j==5
continue;
end
while(cnt(6,1)>0&&more(j,1)>0)
cnt(6,1)=cnt(6,1)-1;
arr(6,j)=arr(6,j)-1;
more(j,1)=more(j,1)-1;
end
end
%7
for j=1:30
if cnt(7,1)==0
break;
end
if j==13||j==20||j==27||j==6
continue;
end
while(cnt(7,1)>0&&more(j,1)>0)
cnt(7,1)=cnt(7,1)-1;
arr(7,j)=arr(7,j)-1;
more(j,1)=more(j,1)-1;
end
end
%num 是每一班的人数
%n 是每天所需的人
```

```
%more 是优先休息完之后，每天多于的可休息人数
%arr 是上班图
%aver 是最终平均出勤率
x=D(arr);
wi=zeros(7,1);
for i=1:7
wi(i,1)=num(i,1)/sum(num);
end

%sleep 是每班可以休息的名额
sleep=wi*sum(more);
%4
for j=1:30
if sleep(4,1)<1
break;
end
if j==10||j==17||j==24||j==3
continue;
end
while(arr(4,j)>=1&&sleep(4,1)>=1&&more(j,1)>0)
sleep(4,1)=sleep(4,1)-1;
arr(4,j)=arr(4,j)-1;
more(j,1)=more(j,1)-1;
end
end
%3
for j=1:30
if sleep(3,1)<1
break;
end
if i==2||i==9||i==16||i==23||i==30
continue;
end
while(arr(3,j)>=1&&sleep(3,1)>=1&&more(j,1))
sleep(3,1)=sleep(3,1)-1;
arr(3,j)=arr(3,j)-1;
more(j,1)=more(j,1)-1;
end
```

```
end
%2
for j=1:30
if sleep(2,1)<1
break;
end
if i==1||i==8||i==15||i==22||i==29
continue;
end
while(arr(2,j)>=1&&sleep(2,1)>=1&&more(j,1))
sleep(2,1)=sleep(2,1)-1;
arr(2,j)=arr(2,j)-1;
more(j,1)=more(j,1)-1;
end
end

%5
for j=1:30
if sleep(5,1)<1
break;
end
if j==11||j==18||j==25||j==4
continue;
end
while(arr(5,j)>=1&&sleep(5,1)>=1&&more(j,1)>0)
sleep(5,1)=sleep(5,1)-1;
arr(5,j)=arr(5,j)-1;
more(j,1)=more(j,1)-1;
end
end
%6
for j=1:30
if sleep(6,1)<1
break;
end
if j==12||j==19||j==26||j==5
continue;
end
```

```
while(arr(6,j)>=1&&sleep(6,1)>=1&&more(j,1)>0)
sleep(6,1)=sleep(6,1)-1;
arr(6,j)=arr(6,j)-1;
more(j,1)=more(j,1)-1;
end
end
%对第一班
for j=1:30
if sleep(1,1)<1
break;
end
if i==7||i==14||i==21||i==28
continue;
end
while(arr(1,j)>=1&&sleep(1,1)>=1&&more(j,1)>0)
sleep(1,1)=sleep(1,1)-1;
arr(1,j)=arr(1,j)-1;
more(j,1)=more(j,1)-1;
end
end
%7
for j=1:30
if sleep(7,1)<1
break;
end
if j==13||j==20||j==27||j==6
continue;
end
while(arr(7,j)>=1&&sleep(7,1)>=1&&more(j,1)>0)
sleep(7,1)=sleep(7,1)-1;
arr(7,j)=arr(7,j)-1;
more(j,1)=more(j,1)-1;
end
end

x=D(arr);
```

```
%%对于数组 arr 的方差公式
function fval=D(arr)
averevery=zeros(7,1);
for i=1:7
averevery(i,1)=sum(arr(i,1:30))/arr(i,31);
end
fval=var(averevery);
end
```