中国研究生创新实践系列大赛

“华为杯”第十八届中国研究生数学建模竞赛

学　　校	杭州电子科技大学
参赛队号	21103360050
队员姓名	1.吴金涛
	2.高天洋
	3.陈奇

中国研究生创新实践系列大赛
“华为杯”第十八届中国研究生数学建模竞赛

题 目　　基于机组排班问题的多目标规划模型

摘　　要:

机组排班问题是运筹学应用的重要领域。本文充分考虑了各种现实约束，对机组排班问题建立了分层的多目标规划模型，并利用贪心算法结合回溯算法求解出最优的机组排班方案。

针对问题一，本文针对题中所给的规划目标建立了一个多约束条件下起飞航班数尽可能大、乘机次数尽可能少、替补资格尽可能少的**多目标规划模型**。由于多个规划目标之间存在重要性排序，因此可以采用**分层序列法**对其进行求解。模型求解时，首先对输入数据进行**预处理**，使其格式满足算法要求，然后使用**贪心算法结合回溯算法**对模型进行求解，目标函数的重要性排序就是贪心算法的贪心策略。最终解得 A 套数据和 B 套数据中，**不满足机组配置航班数**分别为 0、195；**满足机组配置航班数**为 206、13759；**机组人员总体乘机次数**为 8 次、8909 次；**替补资格使用次数**为 0 次、0 次。程序在求解 A 套数据和 B 套数据时的**运行时间**分别为 0.0042035 分钟和 0.6772196 分钟。在最好情况下，算法的时间复杂度为 O（n），最坏情况下**时间复杂度为 O（n^2）**。

针对问题二，添加了执勤相关约束条件，调整了规划目标的顺序，构建了新的多目标规划模型。在求解时，对问题一中所提算法进行了调整，主要是调整贪心算法的贪心策略，使其满足新的规划目标。最终解得 A 套数据和 B 套数据中**不满足机组配置航班数**分别为 0、196；**满足机组配置航班数**分别为 206、13758；**机组总体利用率**分别为 **81.5%**、**51.61%**；**最小/平均/最大一次执勤飞行时长**分别为 135/210.327102803/280 分钟、65/229.2/415 分钟；**最小/平均/最大一次执勤执勤时长**分别为 175/250.420560747/320 分钟、65/366.6/715 分钟；**最小/平均/最大机组人员执勤天数**分别为 3/10.19048/15 天、1/24.8/31 天；**总体执勤成本**分别为 55.6687 万元、5184.252 万元。算法的**时间复杂度**仍然为 **O（n^2）**。**程序运行时间**分别为 **0.0027377482 分钟**和 **0.689067533 分钟**。

针对问题三，添加了任务环相关约束条件，新增了任务环相关的目标函数，并且对所有目标函数的顺序进行了调整，建立了一个新的多目标规划模型。在求解时，对贪心和回溯算法再次进行了改进，使其满足新模型的要求。最终解得针对 A 套数据和 B 套数据的结果中，**不满足机组配置航班数**分别为 0、3257；满足机组配置航班数分别为 206、10687；**机组人员总体乘机次数**分别为 8 次、8442 次；**替补资格使用次数**分别为 0 次、0 次；机组总体利用率分别为 82.98%、57.6%；**最小/平均/最大一次执勤飞行时长**分别为 90/225.6125823/420 分钟、55/270.6/450 分钟；**最小/平均/最大一次执勤执勤时长**分别为 90/271.20320121/560 分钟、55/379.2/705 分钟；**最小/平均/最大机组人员执勤天数**分别为 5/9.5273625/11 天、1/15.41/21 天；**一/二/三/四天任务环数量**分别为 9/4/14/36 个、503/317/239/41 个；**总体执勤成本**分别为 56.8231 万元、4298.88 万元；**总体任务环成本**分别为 7.5373 万元、852.72 万元；**程序运行时间**分别为 0.00364543287 分钟、0.7007563333

分钟。算法的**时间复杂度**仍然为 **O（n^2）。**

关键词：机组排班；多目标规划；分层序列法；贪心算法；回溯算法

目录

1. 问题重述

1.1 问题背景

自 20 世纪 50 年代以来，飞行机组排班问题一直是运筹学领域研究的热土。劳动力成本是航空公司最大的直接运营成本之一，仅次于飞机燃油成本。机组排班是指在满足国家法律法规、国际公约、政府的行政条例的基础上，构造特定时间段的机组日程安排，包括每个机组人员在何时何地以及哪个航班执行什么类型的任务。一个高质量的机组航班任务计划可以降低航空公司的运营成本，合理分配机组人员的工作时间，提高客户满意度，同时提高公司在行业中的竞争力，因此机组排班问题具有巨大的实际应用价值以及非常积极的研究意义。

以飞行员排班问题为例，机组排班问题的示意图如图 1-1 所示：

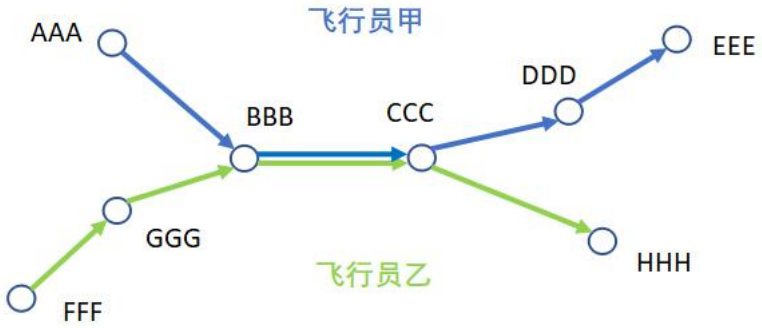

图 1-1 机组排班问题示意图：飞行员甲和飞行员乙在执行航班 BBB->CCC 任务时相遇

近年来，随着航空业的不断发展，业务量不断提高，机组排班问题的规模和复杂性也随之不断扩大。为简化问题，考虑剔除不必要的概念和约束，仅保留核心问题。为此，假设在进行问题的建模和求解之前，航班规划阶段已经完成，机型分配也已经结束，对机组人员数量及资格的需求已经明确，而且可用机组人员也已经确定。另外，本题的描述只针对飞行员，但是对其他机组人员（例如乘务员、乘警）也同样适用。在本题中，每个机组人员都有一个固定的基地，有一个且仅有一个主要资格，但也可以具备其它的替补资格。任务类型分为两种：飞行任务和乘机任务。

每个航班都有给定的最低机组资格配置，用格式 C<数> F<数> 描述，其中 C<数>为正机长数，F<数>为副机长数。一个航班只有满足了最低机组资格配置才能起飞。如图 1-2 所示：

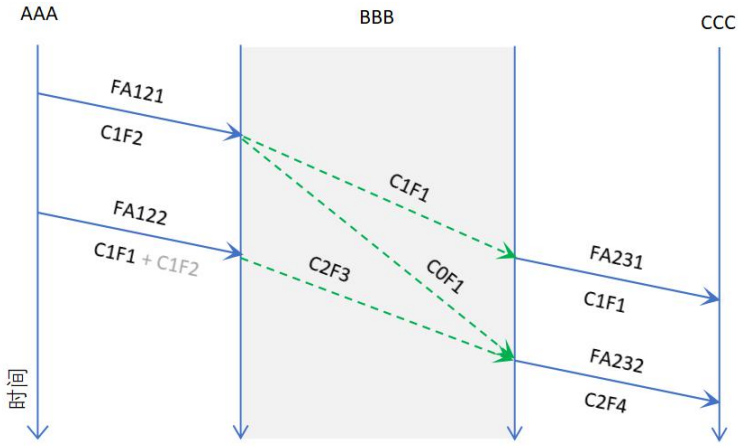

图 1-2 航段及机组分配流动示例图

在图 1-2 中，航班 FA121 上配置了 1 名正机长，2 名副机长，航班 FA122 上配置了 1 名正机长和 2 名副机长，另有 1 名正机长和 2 名副机长搭乘。到达机场 BBB 后，机组人员进行了重新分配，其中航班 FA121 上的 1 名正机长和 1 名副机长继续服务 FA231，剩余 1 名副机长和航班 FA122 上的所有机组人员一起服务航班 FA232。

另外，本文需要明确以下关键概念：

一次执勤由一连串航段（飞行或乘机）和间隔连接时间组成，如图 1-3 所示：

图 1-3 一次执勤示例

任务环由一连串的执勤和休息时间组成，出发和最终到达的机场都为基地，如图 1-4 所示：

图 1-4 一个任务环示例

每个机组人员在每个排班周期都有一个排班计划，这个计划由一系列的任务环和休假组成，任务环之间满足一定的休假天数，如图 1-5 所示：

图 1-5 机组人员排班计划示例

一般情况下，在机组排班问题中需要考虑的优化目标按照重要程度的排序为：

① 尽可能多的航班满足机组配置；
② 机组人员的总体执勤成本最低；
③ 机组人员的总体任务环成本最低；
④ 尽可能少的总体乘机次数；
⑤ 机组人员之间的执勤时长尽可能平衡；
⑥ 机组人员之间的任务环时长尽可能平衡；
⑦ 尽可能少使用替补资格。

1.2 需要解决的问题

针对上述机组排班优化问题，本文需要定义一些参数，如表 1-1 所示：

表 1-1 参数定义

参数名称	取值	含义
MinCT	40 分钟	每个机组人员相邻两个航段之间的最小连接时间
MaxBlk	600 分钟	每次执勤的最大飞行时间
MaxDP	720 分钟	每次执勤的最大时长
MinRest	660 分钟	每个机组人员的相邻两个执勤之间的最短休息时间
MaxTAFB	14400 分钟	每个机组人员每个排班周期的任务环最大总时长
MaxSuccOn	4 天	每个机组人员最大连续执勤天数
MinVacDay	2 天	每个机组人员相邻两个任务环之间最短休息时间

待解决的问题如下：

问题一：依次满足目标①、④、⑦，建立线性规划模型给航班分配机组人员。要求满足如下约束：

1. 每个机组人员初始从基地出发并最终回到基地；
2. 每个机组人员的下一航段的起飞机场必须和上一航段的到达机场一致；
3. 每个机组人员相邻两个航段之间的连接时间不小于 MinCT 分钟。

问题二：引入执勤概念，假定每个机组人员的每单位小时执勤成本给定，依次满足目标①、②、④、⑤、⑦，建立线性规划模型给航班分配机组人员。要求在满足问题一中约束条件的基础上进一步满足如下约束：

1. 每个机组人员每天至多只能执行一个执勤；
2. 每次执勤的飞行时间最多不超过 MaxBlk 分钟；
3. 每次执勤的时长最多不超过 MaxDP 分钟；
4. 每个机组人员下一执勤的起始机场必须和上一执勤的结束机场一致；
5. 每个机组人员的相邻两个执勤之间的休息时间不小于 MinRes 分钟。

问题三：假定每个机组人员的每单位小时任务环成本给定，依次满足目标①、②、③、④、⑤、⑥、⑦，建立线性规划模型编制排班计划。要求在满足问题一和问题二中约束条件的基础上，进一步满足如下约束：

1. 每个机组人员每个排班周期的任务环总时长不超过 MaxTAFB 分钟；
2. 每个机组人员相邻两个任务环之间至少有 MinVacDay 天休息；
3. 每个机组人员连续执勤天数不超过 MaxSuccOn 天。

2. 模型假设

（1）假设每个航班唯一。
（2）假设机型和资格配置只有一种。
（3）假设所有机组人员的初始位置和排班周期结束时的位置都是在其基地。
（4）假设机组人员之间可以任意组合。
（5）假设允许存在因为无法满足最低机组资格配置而不能起飞的航班。
（6）假设不满足最低机组资格配置的航班不能配置任何机组人员。
（7）假设机组人员可以乘机摆渡，即实际机组配置可以超过最低配置要求。
（8）假设航班规划阶段已经完成，机型分配已经结束，对机组人员数量及资格的需求已经明确，而且可用机组人员也已经确定。

3. 符号说明

符号	说明
f_a	航班a是否满足最低机组资格配置
A	所有航班的集合
$f_{a,e}$	机组人员e是否乘坐航班a
E	所有机组人员的集合
C	机组人员中能胜任正机长的人员集合
C_a	航班a的最低机组资格配置中需要的正机长人数
F	机组人员中能胜任副机长的人员集合
F_a	航班a的最低机组资格配置中需要的副机长人数
H_d	从d地出发的航班集合
H'_d	以d地为目的地的航班集合
B_e	机组人员e的基地
T_a	航班a的起飞时间
T'_a	航班a的到达时间
D	所有机场的集合
S_a	起飞时间早于航班a的航班集合
S'_a	起飞时间迟于航班a的航班集合
U_e	机组人员e的每小时执勤成本
$G_{e,z}$	机组人员e在第z天的执勤时长
J_a	航班a使用替补资格的人数
Q_z	起飞日期为z的航班集合
Z	所有航班起飞日期的集合
W_e	机组人员e的每小时任务环成本
L_0	机组人员中执勤总时长的最大值
L_1	机组人员中执勤总时长的最小值
L_2	机组人员中任务环总时长的最大值
L_3	机组人员中任务环总时长的最小值

注：未注明的符号以出现处为准

4. 问题一模型建立与求解

4.1 问题一分析

问题一要求建立一个线性规划模型给航班分配机组人员，在给定的约束条件下依次满

足目标①尽可能多的航班满足机组配置；④尽可能少的总体乘机次数；⑦尽可能少使用替补资格。为此，考虑以题中所给的三个目标为规划目标，建立一个多目标规划模型。由于三个规划目标之间存在主次关系，可以使用分层序列法进行求解。按照目标重要性序列①④⑦依次求得模型的最优解，即首先对目标①求得最优解集，然后在这个最优解集的基础上再对目标④求最优解集，最后在前两个目标的最优解集基础上再去求得目标⑦的最优解，由此可以得到最终答案。

4.2 问题一模型建立

（1）目标函数

首先考虑规划目标①尽可能多的航班满足机组配置，设决策变量 f_a 表示航班 a 是否满足最低机组资格配置，A 表示所有航班的集合。显然有：

$$f_a \in \{0,1\},\ a \in A \tag{4-1}$$

当 f_a 为 0 时表示航班 a 不满足最低机组资格配置，不能起飞，当 f_a 为 1 时表示航班 a 满足最低机组资格配置，可以起飞。为了使得尽可能多的航班满足最低机组资格配置，目标函数 1 为：

$$Max \sum_{a \in A} f_a \tag{4-2}$$

考虑规划目标④尽可能少的总体乘机次数，令决策变量 $f_{a,e}$ 表示机组人员 e 是否乘坐航班 a（无论任务性质是正机长、副机长还是乘机），E 表示所有机组人员的集合。有：

$$f_{a,e} \in \{0,1\},\ \forall a \in A, \forall e \in E \tag{4-3}$$

当 $f_{a,e}$ 为 0 时表示机组人员 e 不乘坐航班 a，当 $f_{a,e}$ 为 0 时表示机组人员 e 乘坐航班 a。

在满足目标函数 1 之后，可以认为满足最低机组资格配置的航班已定，因此总体乘机次数最少可以看做所有可以起飞的航班中正机长、副机长和乘机人次之和最少，设 C 为机组人员中能胜任正机长的人员集合，C_a 为航班 a 的最低机组资格配置中需要的正机长人数。同理，设 F 为机组人员中能胜任副机长的人员集合，F_a 为航班 a 的最低机组资格配置中需要的副机长人数，由此可以得到目标函数 2：

$$Min \sum_{a \in A}((\sum_{e \in E} f_{a,e}) - C_a \cdot f_a + F_a \cdot f_a) \tag{4-4}$$

考虑规划目标⑦尽可能少使用替补资格，有目标函数 3：

$$Min \sum_{a \in A}(f_a \cdot F_a - \sum_{e \in (F-C)} f_{a,e}) \tag{4-5}$$

（2）约束条件

首先，不满足最低机组资格配置的航班不能配置任何机组人员。为了将其表示为线性约束，使用大 M 法[4]，引入一个大于全体机组人员总数的常数 M，因此有：

$$\sum_{e \in E} f_{a,e} \le M \cdot f_a,\ \forall a \in A \tag{4-6}$$

反之，可以起飞的航班一定满足了最低机组资格配置：

$$\begin{cases} \sum_{e\in C\bigcup F} f_{a,e} \geq C_a \cdot f_a + F_a \cdot f_a \\ \sum_{e\in C} f_{a,e} \geq C_a \cdot f_a \\ \sum_{e\in F} f_{a,e} \geq F_a \cdot f_a \end{cases} \quad , \ \forall a \in A \tag{4-7}$$

题目要求每个机组人员初始从基地出发并最终回到基地，即每个机组人员的飞行轨迹都是一个环。设 H_d 表示从 d 地出发的航班集合，H'_d 表示以 d 地为目的地的航班集合，B_e 表示机组人员 e 的所属基地。对于机组人员 e 来说，至少要从自己的基地 B_e 起飞一次：

$$\sum_{b\in H_{B_e}} f_{b,e} \geq 1, \ \forall e \in E \tag{4-8}$$

至少要回到自己的基地 B_e 一次：

$$\sum_{b\in H'_{B_e}} f_{b,e} \geq 1, \ \forall e \in E \tag{4-9}$$

另外，每个机组人员的下一航段的起飞机场必须和上一航段的到达机场一致。结合式（4-8）和式（4-9），可以理解为对于某个机场来说，若有一个航班出发，则必有一个航班可以到达，否则机组人员的飞行轨迹无法成环。从图算法的角度来说，每个节点的出度必须等于入度。

除此之外，飞行轨迹能否成环还与时间有关。每个机组人员的下一航段的起飞时间必须早于上一航段的到达时间。设 D 为所有机场的集合，S_a 为起飞时间早于航班 a 的航班集合，S'_a 为起飞时间迟于航班 a 的航班集合，因此有：

$$f_{a,e} \leq \sum_{b\in H_d \bigcap S'_a} f_{b,e}, \ \forall a \in \left(H'_d - B_e\right), \ \forall d \in D, \forall e \in E \tag{4-10}$$

考虑每个机组人员相邻两个航段之间的连接时间不小于 MinCT 分钟。设 T_a 为航班 a 的起飞时间，T'_a 为航班 a 的到达时间，则有：

$$T_a - T_b \geq MinCT \cdot (f_{a,e} + f_{b,e} - 1), \quad \forall b \in A, \forall e \in E, \ \forall a \in S'_b \tag{4-11}$$

$$f_{a,e} + f_{b,e} \leq 1, \quad \forall b \in A, \ \forall a \notin S'_b \bigcup S_b \tag{4-12}$$

最后，为了让使用替补资格的次数最少，需要确定每个航班使用替补资格的人数，设 J_a 为航班 a 使用替补资格的人数，有：

$$J_a \geq 0, \ \forall a \in A \tag{4-13}$$

$$J_a \geq f_a \cdot F_a - \sum_{e\in(F-C)} f_{a,e}, \ \forall a \in A \tag{4-14}$$

综上所述，得到最终的多目标规划模型为：

$$Max \sum_{a\in A} f_a$$

$$Min \sum_{a\in A}((\sum_{e\in E} f_{a,e}) - C_a \cdot f_a + F_a \cdot f_a)$$

$$Min \sum_{a\in A}(f_a \cdot F_a - \sum_{e\in(F-C)} f_{a,e})$$

$$s.t. \begin{cases} f_a \in \{0,1\}, \ \forall a \in A \\ f_{a,e} \in \{0,1\}, \ \forall a \in A, \forall e \in E \\ \sum_{e\in C\bigcup F} f_{a,e} \geq C_a \cdot f_a + F_a \cdot f_a, \ \forall a \in A \\ \sum_{e\in C} f_{a,e} \geq C_a \cdot f_a, \ \forall a \in A \\ \sum_{e\in F} f_{a,e} \geq F_a \cdot f_a, \ \forall a \in A \\ \sum_{e\in E} f_{a,e} \leq M \cdot f_a, \ \forall a \in A \\ \sum_{b\in H_{B_e}} f_{b,e} \geq 1, \ \forall e \in E \\ \sum_{b\in H'_{B_e}} f_{b,e} \geq 1, \ \forall e \in E \\ f_{a,e} \leq \sum_{b\in H_d \cap S'_a} f_{b,e}, \ \forall a \in (H'_d - B_e), \ \forall d \in D, \forall e \in E \\ T_a - T_b \geq MinCT \cdot (f_{a,e} + f_{b,e} - 1), \ \forall b \in A, \forall e \in E, \ \forall a \in S'_b \\ f_{a,e} + f_{b,e} \leq 1, \ \forall b \in A, \ \forall a \notin S'_b \bigcup S_b \\ J_a \geq 0, \ \forall a \in A \\ J_a \geq f_a \cdot F_a - \sum_{e\in(F-C)} f_{a,e}, \ \forall a \in A \end{cases}$$

4.3 模型求解及结果分析

4.3.1 算法设计

（1）数据预处理

本文提出的算法对输入数据有一定的要求，需要先对所给数据进行预处理。具体来说，需要先将题中所给的航班计划数据进行二次排序。所谓二次排序就是首先按照第一字段排序，然后再对第一字段相同的行按照第二字段排序，并且不能破坏第一次排序的结果。

在本题中，我们需要对题中所给附件“机组排班 Data A-Flight.csv”和附件“机组排班

Data B-Flight.csv”中的数据进行预处理。我们选择起飞日期 DptrDate 字段作为第一字段，起飞时间 DptrTime 字段为第二字段，两次排序全部采用升序排列。由于题目所给原数据中 DptrDate 字段值不是标准日期类型，因此为避免排序结果错误，在排序之前需要对 DptrDate 字段值进行统一的格式化处理。

图 4-1 数据预处理流程

经过对数据的预处理，可以保证输入数据中的航班计划严格按照起飞时间从前往后的顺序排列。

（2）算法流程

为了对 4.2 节中所述的数学模型进行求解，本文提出了一种改进的贪心算法，同时借鉴了回溯算法的思想。在描述算法流程之前，需要先定义一个关键概念：若航班 1 的出发机场等于航班 2 的到达机场，航班 1 的到达机场等于航班 2 的出发机场，我们称这两个航班互为返程航班。由此，具体算法流程如下：

输入：航班计划数据集 DataD={d_1，d_2，...，d_m}；机组人员信息数据集 DataC={c_1，c_2，...，c_n}.
过程：
Step1：遍历 DataC，对每个基地 b 维护一个机组人员集合 SetC 和一个出发航班集合 SetF；
Step2：遍历 DataD，对其中的每一条数据 d_i 执行：
IF【从某个基地 b 出发】：
　　若当前机组人员集合 SetC 可以满足其最低配置，将当前航班加入出发航班集合 SetF，更新 SetC，当前航班可以起飞；
ELSE：
　　IF【目的地为某个基地 b】：
　　　　IF【集合中有返程航班】：
　　　　　　以此为起点向下遍历 DataD，找到下一个与当前航班出发地和目的地都相同的航班，若有，将 1 个返程航班从 SetF 中删除，更新 SetC，当前航班可以起飞。否则当前航班将所有历史返程航班配置的机组人员带回，将所有返程航班从 SetF 中删除，更新 SetC；
　　　　ELSE：
　　　　　　以此为起点，反向遍历 DataD（回溯），找到满足（到达时间-当前航班起飞时间）> MinCT 的航班 a（对应数据 d_i），判断当时 SetC 是否可以满足当前航班最低配置，若满足，为航班 a 增加当前航班的最低配置，更新 SetC，从 d_{i+1} 开始继续执行 Step2，否则当前航班无法起飞，继续遍历下一条数据；
　　ELSE：
　　　　IF【出发地是 SetF 中某航班 a 的目的地且航班 a 与当前航班满足最小连接时间约束】：
　　　　　　将航班 a 的目的地改为当前航班的目的地，当前航班可以起飞；
　　　　ELSE：
　　　　　　当前航班无法起飞；

算法流程图如图 4-2 所示：

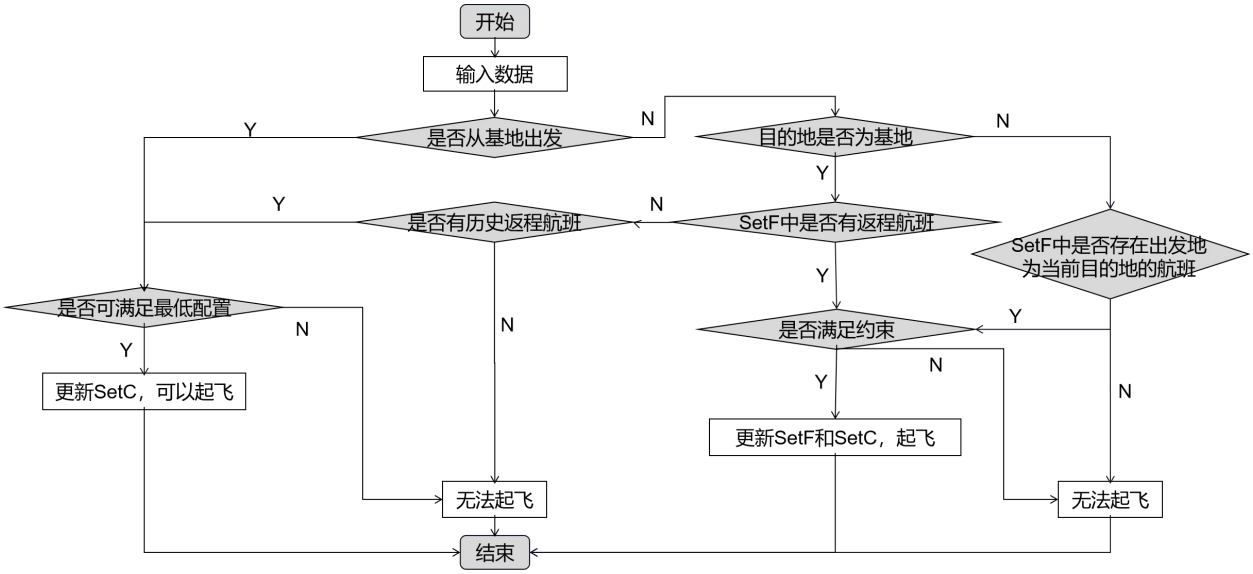

图 4-2 贪心算法流程

4.3.2 求解结果

根据 4.3.1 节中所述的算法，对 4.2 节中建立的多目标规划模型进行编程求解，得到了最终的排班计划。由于数据量较大，不宜在论文中占用过多篇幅进行全部展示，所以本文选取部分数据进行展示，全部详细求解结果见附件 CrewRosters1a.csv、附件 CrewRosters1b.csv 和附件 UncoveredFlights1b.csv。

（1）A 套数据

在针对 A 套数据的求解结果中，以员工 A0001 为例，该员工的排班计划如表 4-1 所示，由于篇幅所限，文中出发和到达日期字段中省略了年份，结果附件 CrewRosters1a.csv 中展示了完整的日期，其中年份都为 2021 年：

表 4-1 A 套数据中员工 A0001 的排班计划

员工号	航段序号	航班号	出发日期	出发时间	出发机场	到达日期	到达时间	到达机场	任务性质
A0001	1	FA680	8/11	8:00	NKX	8/11	9:30	PGX	C
A0001	2	FA681	8/11	10:10	PGX	8/11	11:40	NKX	C
A0001	12	FA680	8/12	8:00	NKX	8/12	9:30	PGX	C
A0001	16	FA3	8/12	10:25	PGX	8/12	11:40	NKX	C
A0001	36	FA864	8/13	17:30	NKX	8/13	19:15	PXB	C
A0001	37	FA865	8/13	20:00	PXB	8/13	21:45	NKX	C
A0001	57	FA884	8/15	11:30	NKX	8/15	13:50	XGS	C
A0001	60	FA885	8/15	14:30	XGS	8/15	16:50	NKX	C
A0001	79	FA890	8/17	7:30	NKX	8/17	9:50	XGS	C
A0001	84	FA891	8/17	10:30	XGS	8/17	12:50	NKX	C
A0001	105	FA864	8/18	17:30	NKX	8/18	19:15	PXB	C
A0001	107	FA865	8/18	20:00	PXB	8/18	21:45	NKX	C
A0001	125	FA680	8/20	8:00	NKX	8/20	9:30	PGX	C
A0001	127	FA681	8/20	10:10	PGX	8/20	11:40	NKX	C
A0001	145	FA854	8/21	13:50	NKX	8/21	15:30	CTH	C

A0001	148	FA855	8/21	16:10	CTH	8/21	17:55	NKX	C
A0001	167	FA680	8/23	8:00	NKX	8/23	9:30	PGX	C
A0001	169	FA681	8/23	10:10	PGX	8/23	11:40	NKX	C
A0001	187	FA854	8/24	13:50	NKX	8/24	15:30	CTH	C
A0001	190	FA855	8/24	16:10	CTH	8/24	17:55	NKX	C

注：任务性质中 C 表示正机长，F 表示副机长，D 表示乘机

在给定数据的时间范围内，员工 A0001 的上班安排如图 4-3 所示：

图 4-3 A 套数据求解结果中员工 A00001 上班安排

可以看出，上班时间与休息时间的分配均匀合理，比例协调，更贴合实际。

所有航班的起飞情况如图 4-4 所示：

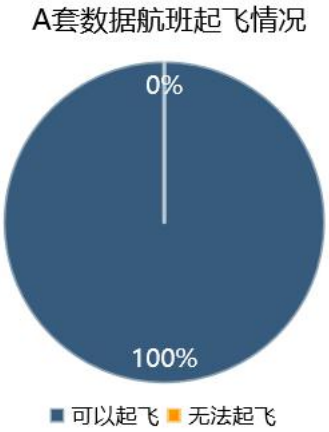

图 4-4 A 套数据求解结果中航班起飞情况

从图中可以看出，在我们的求解结果中，**所有航班全部满足机组配置，可以起飞。**

共计 420 人次成功起飞，其中任务性质为正机长的人次为 206，副机长 206 人次，**乘机 8 人次，替补 0 人次**，如图 4-5 所示：

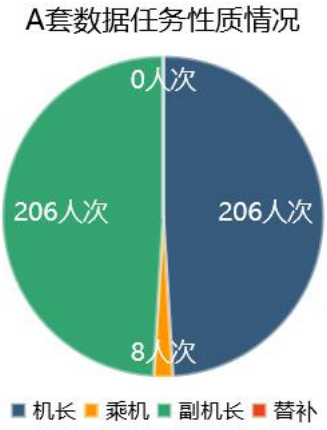

图 4-5 A 套数据求解结果中任务性质情况

以 2021 年 8 月 12 日的排班计划为例，当天正常起飞的航班及其机组配置情况如图 4-6 所示：

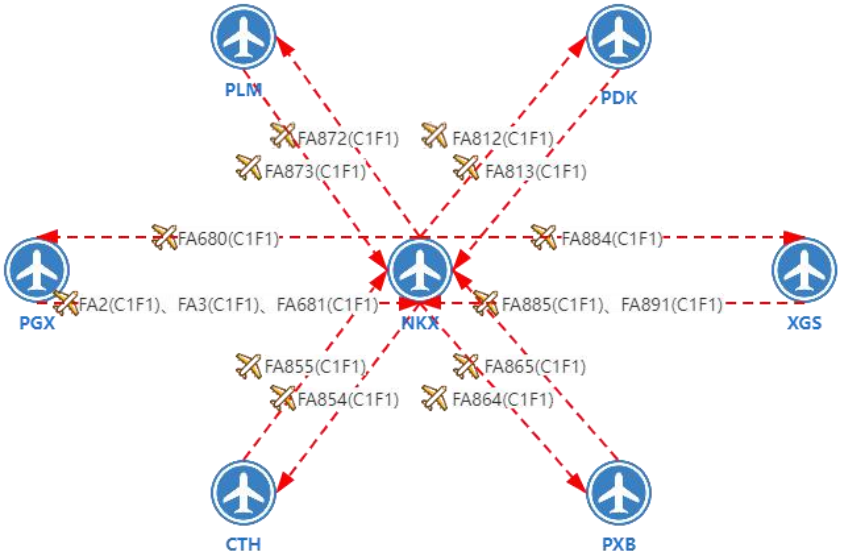

图 4-6 A 套数据求解结果中 2021 年 8 月 12 日航段示意图

利用本文所提的算法针对 A 套数据求解排班计划，在求解结果中不存在因不满足最低机组资格配置而无法起飞的航班，即**所有航班全部满足机组配置。总体乘机次数为 8 次，没有使用替补资格**。

（2）B 套数据

在针对 B 套数据的求解结果中，以员工 B0001 为例，该员工的排班计划如表 4-2 所示，同样地，在此省略了年份，实际应为 2019 年。结果附件 CrewRosters1b.csv 中展示了全部机组人员排班计划，UncoveredFlights1b.csv 中罗列了无法起飞的航班详情。

表 4-2 B 套数据中员工 B0001 的排班计划（部分）

员工号	航段序号	航班号	出发日期	出发时间	出发机场	到达日期	到达时间	到达机场	任务性质
B0001	1	FB8559	8/01	0:05	HOM	8/01	2:05	SXA	C
B0001	3	FB1000	8/01	3:35	SXA	8/01	5:55	TGD	C
B0001	92	FB17	8/01	9:00	TGD	8/01	10:10	HOM	C
B0001	2167	FB779	8/05	18:40	HOM	8/05	20:25	NOU	C
B0001	2420	FB1142	8/06	11:50	NOU	8/06	13:40	HOM	C
B0001	3140	FB8559	8/08	0:05	HOM	8/08	2:05	SXA	C
B0001	3283	FB1734	8/08	10:40	SXA	8/08	12:45	HOM	C
B0001	3823	FB1169	8/09	13:10	HOM	8/09	15:15	SXA	C
B0001	2531	FB1170	8/06	15:55	SXA	8/06	17:55	HOM	C
B0001	3140	FB8559	8/08	0:05	HOM	8/08	2:05	SXA	C
B0001	3283	FB1734	8/08	10:40	SXA	8/08	12:45	HOM	C

注：任务性质中 C 表示正机长，F 表示副机长，D 表示乘机

所有航班的起飞情况如图 4-7 所示：

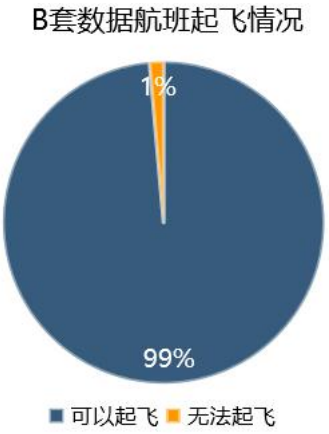

图 4-7 B 套数据求解结果中航班起飞情况

在我们的求解结果中，仅有 195 个航班因无法满足最低机组配置而不能起飞。起飞航班中任务性质的情况如图 4-8 所示：

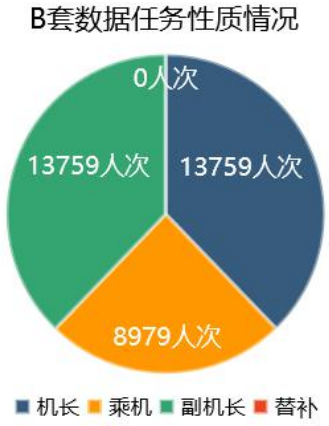

图 4-8 B 套数据求解结果中任务性质情况

问题一中 AB 两套数据的求解结果指标如表 4-3 所示：

表 4-3 问题一结果指标

结果指标	A 套数据	B 套数据
不满足机组配置航班数	0	195
满足机组配置航班数	206	13759
机组人员总体乘机次数	8	8909
替补资格使用次数	0	0
程序运行分钟数	0.0042035	0.6772196

4.3.3 算法有效性及复杂度分析

本文提出的算法整体思想采用了贪心算法，但在局部处理特殊情况时采用了回溯算法。这种算法的好处是思路比较容易想到，但缺点是比较难以实现，尤其是面对大型数据集时需要考虑许多可能发生的情况。

用本文所提算法对模型进行求解时目标的次序非常重要，直接影响了结果指标。本算法对首要目标最为友好，若以首要目标为标准来衡量求解结果，那么结果可以堪称完美。但对次要目标，以及排序更为靠后的规划目标不太友好，往往只能得到近优解。

整体贪心和局部回溯的方法，在最坏情况下，算法的**时间复杂度达到 $O(n^2)$，最好情**

况下时间复杂度为 O(n)。在 A 套数据上程序的运行时间为 0.0042035 分钟，在 B 套数据上程序的运行时间为 0.6772196 分钟。

5. 问题二模型建立与求解

5.1 问题二分析

问题二引入了执勤概念，在问题一的基础上增加了两个规划目标，并且附加了 5 个约束条件。问题二完全建立在问题一的基础上，因此只要对问题一中所建立的多目标规划模型进行扩展即可。首先要增加规划目标②机组人员的总体执勤成本最低和⑤机组人员之间的执勤时长尽可能平衡，然后调整规划目标的顺序①、②、④、⑤、⑦，然后增加新的约束以符合题目要求。在求解时，仍然可以采用分层序列法对目标函数依次进行求解，算法仍然可以使用改进贪心算法，不过要在问题一所提算法的基础上做出进一步的调整。

5.2 问题二模型建立

问题二的多目标规划模型建立在问题一模型的基础上，因此问题一模型中解释过的目标函数和约束条件详见 4.2 节，此处不再赘述。

（1）目标函数

对于规划目标②机组人员的总体执勤成本最低，设机组人员 e 的每小时执勤成本为 U_e，其在第 z 天的执勤时长为 $G_{e,z}$，则该目标函数可以表示为：

$$Min \sum_{e\in E} U_e \cdot \sum_{z\in Z} G_{e,z} \tag{5-1}$$

为了表示规划目标⑤机组人员之间的执勤时长尽可能平衡，需要先对平均执勤时长进行表示：

$$\bar{U} = \frac{1}{|E|}\sum_{e\in E} U_e \cdot \sum_{a\in A}\left(T'_a - T_a\right)\cdot f_{a,e} \tag{5-2}$$

规划目标⑤可以用机组人员的最长执勤时长与最短执勤时长的差值来度量，差值越小，说明执勤时长越平均。为此，我们引入变量 L_0 和 L_1 分别表示机组人员中每个人的执勤总时长的最大值和最小值，有：

$$\sum_{z\in Z} G_{e,z} \le L_1,\ \ \forall e \in E \tag{5-3}$$

$$L_0 \le \sum_{z\in Z} G_{e,z},\ \ \forall e \in E \tag{5-4}$$

因此该目标函数可以写为：

$$Min\ L_1 - L_0 \tag{5-5}$$

（2）约束条件

首先，根据题中给出的执勤定义，同一次执勤里的每个航段必须在同一天起飞。附加约束指出，每个机组人员每天至多只能执行一个执勤。也就是说，最终求得的每个机组人员的排班计划中，所有起飞时间为同一天的航段必定属于同一次执勤。有了这个前提，我们可以去表示其他约束。

对于任意一个机组人员来说，每次执勤的总飞行时间即为起飞时间在同一天的所有航班飞行时间之和。设Q_z表示起飞日期为z的航班集合，Z表示所有航班起飞日期的集合，每次执勤的飞行时间最多不超过 MaxBlk 分钟可以表示为：

$$\sum_{a\in Q_z}\left(T'_a-T_a\right)\cdot f_{a,e}\le MaxBlk,\quad \forall z\in Z,\forall e\in E \tag{5-6}$$

其中T'_a-T_a表示航班a的飞行时间。

考虑每次的执勤时长不超过 MaxDP 分钟，若每次执勤中任意两次航班中，前一航班的起飞时间到后一航班的到达时间之间的间隔不超过 MaxDP 分钟，那么一定可以满足这条约束。因此有：

$$(T'_b-T_a)\cdot(f_{a,e}+f_{b,e}\text{-}1)\le G_{e,z},\quad \forall a\in Q_z,\forall b\in Q_z\bigcap S'_a,\quad \forall z\in Z,\forall e\in E \tag{5-7}$$

$$G_{e,z}\le MaxDP,\quad \forall e\in E,\forall z\in Z \tag{5-8}$$

为了相邻执勤的时间间隔不小于 MinRest 分钟，在任意一个机组人员的排班计划中，只要任意两个不在同一天起飞的航班之间的时间间隔不小于 MinRest 分钟，即可满足这条约束，因此有：

$$(T_b-T'_a)\ge(f_{a,e}+f_{b,e}-1)\cdot MinRest,\quad \forall a\in Q_z,\forall b\in Q_i,\forall i>z,\forall z\in Z,\forall e\in E \tag{5-9}$$

综上所述，得到针对问题二的多目标规划模型为：

$$Max\ \sum_{a\in A}f_a$$

$$Min\ \sum_{e\in E}U_e\cdot\sum_{z\in Z}G_{e,z}$$

$$Min\ \sum_{a\in A}((\sum_{e\in E}f_{a,e})-C_a\cdot f_a+F_a\cdot f_a)$$

$$Min\ L_1-L_0$$

$$Min\ \sum_{a\in A}(f_a\cdot F_a-\sum_{e\in(F-C)}f_{a,e})$$

$$
s.t.\ \begin{cases}
f_a \in \{0,1\},\ \ \forall a \in A \\
f_{a,e} \in \{0,1\},\ \ \forall a \in A, \forall e \in E \\
\sum_{e \in C \cup F} f_{a,e} \geq C_a \cdot f_a + F_a \cdot f,\ \ \forall a \in A \\
\sum_{e \in C} f_{a,e} \geq C_a \cdot f_a,\ \ \forall a \in A \\
\sum_{e \in F} f_{a,e} \geq F_a \cdot f_a,\ \ \forall a \in A \\
\sum_{e \in E} f_{a,e} \leq M \cdot f_a,\ \ \forall a \in A \\
\sum_{b \in H_{B_e}} f_{b,e} \geq 1,\ \ \forall e \in E \\
\sum_{b \in H'_{B_e}} f_{b,e} \geq 1,\ \ \forall e \in E \\
f_{a,e} \leq \sum_{b \in H_d \cap S'_a} f_{b,e},\ \ \forall a \in (H'_d - B_e),\ \forall d \in D, \forall e \in E \\
T_a - T_b \geq MinCT \cdot (f_{a,e} + f_{b,e} - 1),\ \ \forall b \in A, \forall e \in E,\ \forall a \in S'_b \\
f_{a,e} + f_{b,e} \leq 1,\ \ \forall b \in A,\ \forall a \notin S'_b \cup S_b \\
J_a \geq 0,\ \ \forall a \in A \\
J_a \geq f_a \cdot F_a - \sum_{e \in (F-C)} f_{a,e},\ \ \forall a \in A \\
\sum_{a \in Q_z} (T'_a - T_a) \cdot f_{a,e} \leq MaxBlk,\ \ \forall z \in Z, \forall e \in E \\
(T'_b - T_a) \cdot (f_{a,e} + f_{b,e} - 1) \leq G_{e,z},\ \ \forall a \in Q_z, \forall b \in Q_z \cap S'_a,\ \ \forall z \in Z, \forall e \in E \\
G_{e,z} \leq MaxDP,\ \ \forall e \in E, \forall z \in Z \\
(T_b - T'_a) \geq (f_{a,e} + f_{b,e} - 1) \cdot MinRest,\ \ \forall a \in Q_z, \forall b \in Q_i, \forall i > z, \forall z \in Z, \forall e \in E \\
\sum_{z \in Z} G_{e,z} \leq L_1,\ \ \forall e \in E \\
L_0 \leq \sum_{z \in Z} G_{e,z},\ \ \forall e \in E
\end{cases}
$$

5.3 模型求解及结果分析

5.3.1 算法设计

本文针对问题二提出的算法是对问题一所提贪心算法的改进，仍然是基于贪心和回溯的思想，在问题一所提算法的基础上我们做出了一些调整，使其符合问题二所给的条件。问题一所提算法详见 4.3.1 节。

（1）数据预处理

在本文提出的针对问题二的算法中，仍然对输入数据有一定的要求，因此首先需要对题目所给数据进行预处理。此处数据预处理流程与问题一中数据预处理流程完全相同，详见 4.3.1 节，此处不再赘述。

（2）算法流程

在算法流程上，我们做出了较大的调整。由于引入执勤的概念，我们需要在保证目标①的基础上优先考虑目标②，即降低总执勤成本，这直接体现在选择机组人员这一环。具体来说，我们为每个基地维护一个机组人员的集合，假设某一时刻有航班需要起飞，基地中的机组人员足以满足此航班的最低机组配置，此时某基地中的机组人员集合如图 5-1 所示：

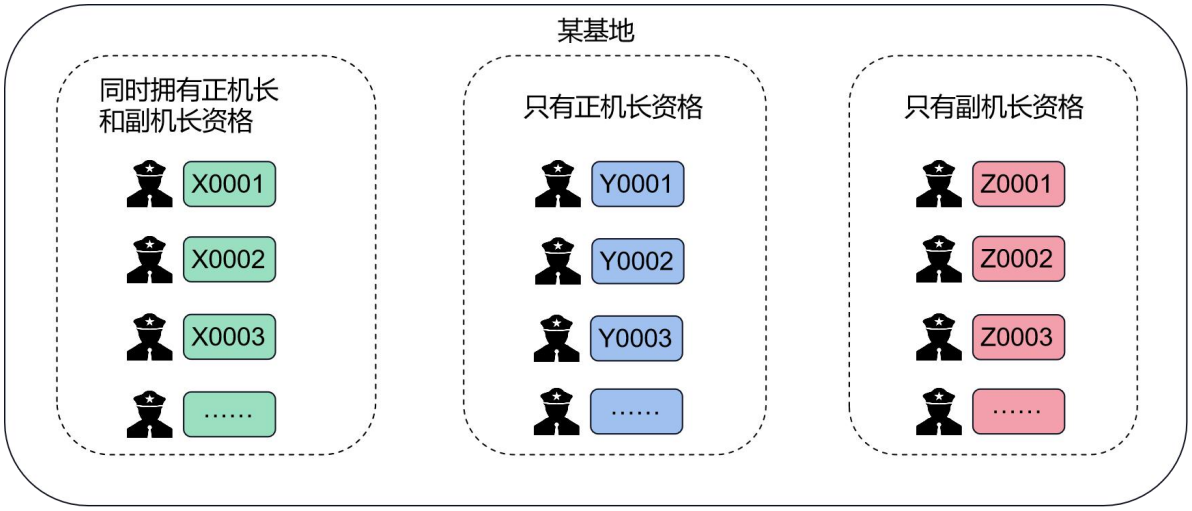

图 5-1 某基地中机组人员集合示意图

按照贪心算法的思想，在满足尽可能多的航班起飞的条件的基础上，我们优先选择同时拥有正机长和副机长资格的机组人员担任此航班的正机长，因为他们的每小时执勤成本比只有正机长资格的机组人员更低。

除此之外，我们还需考虑目标⑤，即机组人员之间的执勤时长尽可能平衡。因此在满足目标①、目标②和目标④尽可能少的总体乘机次数之后，在选择机组人员时可以考虑让拥有相同资格的人轮流执勤。具体来说，集合内部使用三个队列来维护具有相同资格的机组人员，每次执勤选择队列头部的机组人员，执勤结束后进入队列尾部，利用队列先进先出的特性保证了执勤时长尽量平衡。以拥有正机长和副机长双重资格的机组人员队列为例，如图 5-2 所示：

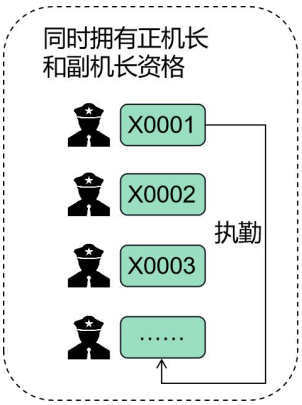

图 5-2 一次执勤示意图

5.3.2 求解结果

根据 5.3.1 节中所述的算法，对 5.2 节中建立的多目标规划模型进行编程求解，得到了最终的排班计划。由于数据量较大，不宜在论文中占用过多篇幅进行全部展示，所以本文选取部分数据进行展示，全部详细求解结果见附件 CrewRosters2a.csv、附件 CrewRosters2b.csv 和附件 UncoveredFlights2b.csv。

（1）A 套数据

在针对 A 套数据的求解结果中，以员工 A0001 为例，该员工的排班计划如表 5-1 所示，由于篇幅所限，文中出发和到达日期字段中省略了年份，结果附件 CrewRosters2a.csv 中展示了完整的日期，其中年份都为 2021 年：

表 5-1 A 套数据中员工 A0001 的排班计划

员工号	航段序号	航班号	出发日期	出发时间	出发机场	到达日期	到达时间	到达机场	任务性质
A0001	19	FA812	8/12	12:20	NKX	8/12	14:05	PDK	C
A0001	22	FA813	8/12	14:50	PDK	8/12	16:40	NKX	C
A0001	63	FA864	8/15	17:30	NKX	8/15	19:15	PXB	C
A0001	64	FA865	8/15	20:00	PXB	8/15	21:45	NKX	C
A0001	121	FA864	8/19	17:30	NKX	8/19	19:15	PXB	C
A0001	122	FA865	8/19	20:00	PXB	8/19	21:45	NKX	C
A0001	191	FA864	8/24	17:30	NKX	8/24	19:15	PXB	C
A0001	192	FA865	8/24	20:00	PXB	8/24	21:45	NKX	C

注：任务性质中 C 表示正机长，F 表示副机长，D 表示乘机

可以看出，相对问题一求解结果中员工 A0001 的排班计划来说，减少了许多排班，这是因为加入了执勤成本约束，在同等条件下我们的贪心算法会优先选择具有正机长和副机长两种资格的机组人员，而员工 A0001 是一名仅拥有正机长资格的机组人员，这也意味着他的每小时执勤成本是最高的。因此为了节约成本，他的排班数量必然会减少，这是由算法的特性决定的。

所有航班的起飞情况如图 5-3 所示：

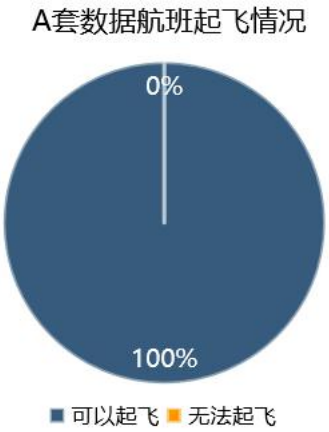

图 5-3 A 套数据求解结果中航班起飞情况

从图中可以看出，在我们的求解结果中，**所有航班全部满足机组配置，可以起飞。**共计 420 人次成功起飞，其中任务性质为正机长的人次为 206，副机长 206 人次，**乘**

机 8 人次，替补 0 人次，如图 5-4 所示：

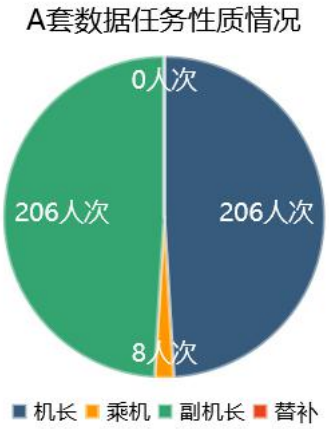

图 5-4 A 套数据求解结果中任务性质情况

为了衡量执勤的效果，我们计算了关于一次执勤飞行时长、一次执勤执勤时长和机组人员执勤天数的最小值、最大值和平均值，如图 5-5 所示：

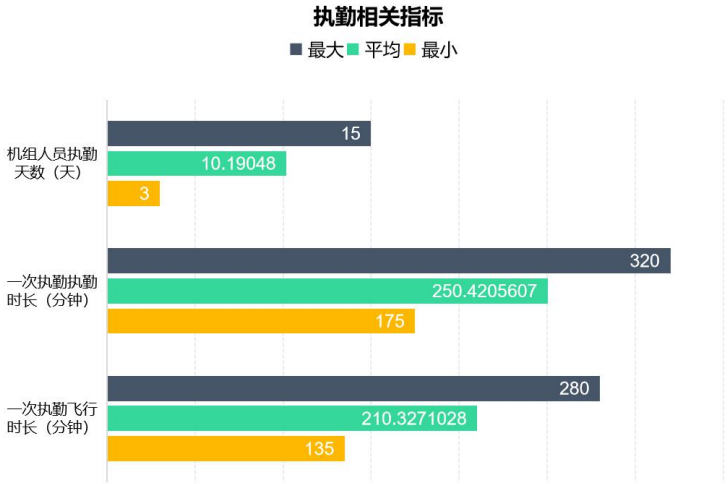

图 5-5 A 套数据求解结果中执勤相关指标

以 2021 年 8 月 12 日的排班计划为例，当天正常起飞的航班及其机组配置情况如图 5-6 所示：

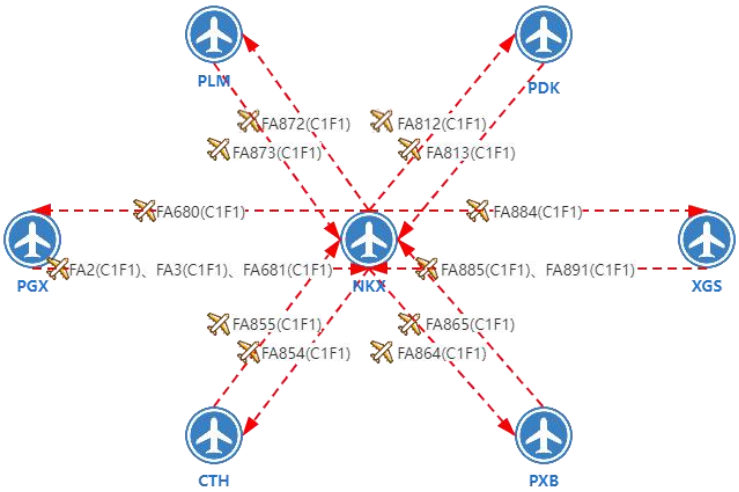

图 5-6 A 套数据求解结果中 2021 年 8 月 12 日航段示意图

利用本文所提的算法针对 A 套数据求解排班计划，在求解结果中不存在因不满足最低机组资格配置而无法起飞的航班，即**所有航班全部满足机组配置。总体乘机次数为 8 次，没有使用替补资格**。

（2）B 套数据

在针对 B 套数据的求解结果中，以员工 B0001 为例，该员工的排班计划如表 5-2 所示，同样地，在此省略了年份，实际应为 2019 年。结果附件 CrewRosters2b.csv 中展示了全部机组人员排班计划，UncoveredFlights2b.csv 中罗列了无法起飞的航班详情。

表 5-2 B 套数据中员工 B0001 的排班计划（部分）

员工号	航段序号	航班号	出发日期	出发时间	出发机场	到达日期	到达时间	到达机场	任务性质
B0001	1	FB8559	8/01	0:05	HOM	8/01	2:05	SXA	C
B0001	3	FB1000	8/01	3:35	SXA	8/01	5:55	TGD	C
B0001	92	FB17	8/01	9:00	TGD	8/01	10:10	HOM	C
B0001	2124	FB427	8/05	17:20	HOM	8/05	18:35	FBX	D
B0001	2296	FB1702	8/06	8:00	FBX	8/06	9:05	HOM	C
B0001	2789	FB523	8/07	9:15	HOM	8/07	10:25	XMJ	C
B0001	3179	FB522	8/08	7:35	XMJ	8/08	8:40	HOM	C
B0001	4072	FB94	8/10	7:00	HOM	8/10	8:10	TGD	D
B0001	4148	FB17	8/10	9:00	TGD	8/10	10:10	HOM	D
B0001	4480	FB511	8/10	19:55	HOM	8/10	21:00	XMH	C

注：任务性质中 C 表示正机长，F 表示副机长，D 表示乘机

所有航班的起飞情况如图 5-7 所示：

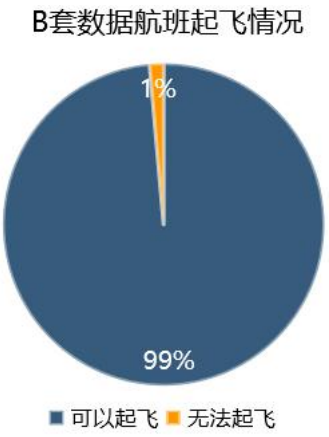

图 5-7 B 套数据求解结果中航班起飞情况

在我们的求解结果中，仅有 196 个航班因无法满足最低机组配置而不能起飞。起飞航班中任务性质的情况如图 5-8 所示：

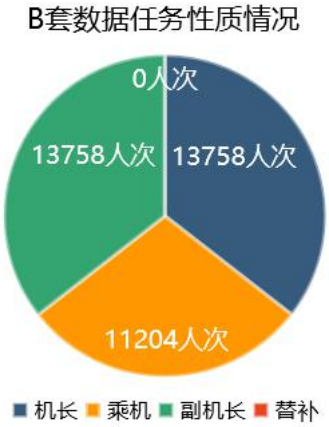

图 5-8 B 套数据求解结果中任务性质情况

由于引入了执勤概念，因此需要对执勤相关指标进行可视化展示，如图 5-9 所示：

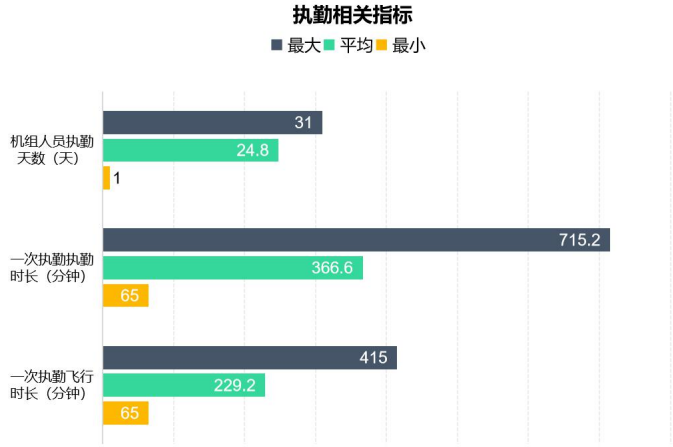

图 5-9 B 套数据求解结果中执勤相关指标

问题二中 AB 两套数据的求解结果指标如表 5-3 所示：

表 5-3 问题二结果指标

结果指标	A 套数据	B 套数据
不满足机组配置航班数	0	196
满足机组配置航班数	206	13758
机组人员总体乘机次数	8	11204
替补资格使用次数	0	0
机组总体利用率	81.5%	51.61%
最小一次执勤飞行时长（分钟）	135	65
平均一次执勤飞行时长（分钟）	210.327102803	229.2
最大一次执勤飞行时长（分钟）	280	415
最小一次执勤执勤时长（分钟）	175	65
平均一次执勤执勤时长（分钟）	250.420560747	366.6
最大一次执勤执勤时长（分钟）	320	715
最小机组人员执勤天数	3	1
平均机组人员执勤天数	10.19048	24.8

最大机组人员执勤天数	15	31
总体执勤成本（万元）	55.6687	5184.252
程序运行分钟数	0.0027377482	0.68906753333

5.3.3 算法有效性及复杂度分析

本题引入了执勤的概念，因此我们对问题一中提出的算法进行了进一步的优化。主要是调整了贪心的策略，使其依次满足新的目标函数。虽然我们对算法流程进行了调整，但是并没有改变算法的时间复杂度。在最坏情况下，算法的**时间复杂度仍然是 $O(n^2)$，最好情况下时间复杂度为 O(n)。程序在 A 套数据上运行时间为 0.0027377482 分钟，在 B 套数据上运行时间为 0.68906753 分钟。**

6. 问题三模型建立与求解

6.1 问题三分析

问题三引入了任务环的概念，在问题二的基础上再次增加了 2 个规划目标和 3 个约束条件。只需在问题二建立的多目标规划模型的基础上，将新增目标函数③机组人员的总体任务环成本最低和⑥机组人员之间的任务环时长尽可能平衡加入进去，并且添加一些约束条件以满足题目要求即可。在求解时，根据分层序列法按照目标顺序①、②、③、④、⑤、⑥、⑦依次进行求解。算法可以对问题二所提算法进行进一步的优化，使其满足问题三的新要求。

6.2 问题三模型建立

问题三的多目标规划模型建立在问题二模型的基础上，因此问题一和问题二模型中解释过的目标函数和约束条件详见 4.2 节和 5.2 节，此处不再赘述。

（1）目标函数

设 W_e 为机组人员 e 的每小时任务环成本，则规划目标③机组人员的总体任务环成本最低可以表示为：

$$Min \sum_{e\in E}\Big(\sum_{b\in H'_{B_e}} T'_b\cdot f_{b,e}-\sum_{b\in H_{B_e}} T_b\cdot f_{b,e}\Big)\cdot W_e \tag{6-1}$$

参照式（5-3），引入变量 L_2 和 L_3 分别表示机组人员中任务环总时长的最大值和最小值，有：

$$\sum_{b\in H'_{B_e}} T'_b\cdot f_{b,e}-\sum_{b\in H_{B_e}} T_b\cdot f_{b,e}\le L_3,\ \ \forall e\in E \tag{6-2}$$

$$L_2\le \sum_{b\in H'_{B_e}} T'_b\cdot f_{b,e}-\sum_{b\in H_{B_e}} T_b\cdot f_{b,e},\ \ \forall e\in E \tag{6-3}$$

因此可以将规划目标⑥机组人员之间的任务环时长尽可能平衡表示为：

$$Min\ L_3-L_2 \tag{6-4}$$

（2）约束条件

题中要求每个机组人员每个排班周期的任务环总时长不超过 MaxTAFB 分钟，可以表示为：

$$\sum_{b\in H'_{B_e}} T'_b \cdot f_{b,e} - \sum_{b\in H_{B_e}} T_b \cdot f_{b,e} \geq MaxTAFB, \quad \forall e \in E \tag{6-5}$$

每个机组人员相邻两个任务环之间至少有 MinVacDay 天休息，因此有：

$$(T_b - T'_a - MinVacDay) \cdot f_{b,e} \geq 0, \quad \forall e \in E, \forall a \in H'_{B_e}, \forall b \in S'_a \tag{6-6}$$

每个机组人员连续执勤天数不超过 MaxSuccOn 天：

$$\sum_{z=i}^{i+MaxSuccOn} f_{a,e} \leq MaxSuccOn, \quad \forall a \in Q_z, \forall i \in Z \tag{6-7}$$

最终，我们得到的多目标规划模型为：

$$Max \sum_{a\in A} f_a$$

$$Min \sum_{e\in E} U_e \cdot \sum_{z\in Z} G_{e,z}$$

$$Min \sum_{e\in E} (\sum_{b\in H'_{B_e}} T'_b \cdot f_{b,e} - \sum_{b\in H_{B_e}} T_b \cdot f_{b,e}) \cdot W_e$$

$$Min \sum_{a\in A} ((\sum_{e\in E} f_{a,e}) - C_a \cdot f_a + F_a \cdot f_a)$$

$$Min\ L_1 - L_0$$

$$Min\ L_3 - L_2$$

$$Min \sum_{a\in A} (f_a \cdot F_a - \sum_{e\in (F-C)} f_{a,e})$$

$$
s.t.\ \begin{cases}
f_a \in \{0,1\}, \ \ \forall a \in A \\
f_{a,e} \in \{0,1\}, \ \ \forall a \in A, \forall e \in E \\
\sum_{e \in C \bigcup F} f_{a,e} \geq C_a \cdot f_a + F_a \cdot f, \ \ \forall a \in A \\
\sum_{e \in C} f_{a,e} \geq C_a \cdot f_a, \ \ \forall a \in A \\
\sum_{e \in F} f_{a,e} \geq F_a \cdot f_a, \ \ \forall a \in A \\
\sum_{e \in E} f_{a,e} \leq M \cdot f_a, \ \forall a \in A \\
\sum_{b \in H_{B_e}} f_{b,e} \geq 1, \ \ \forall e \in E \\
\sum_{b \in H'_{B_e}} f_{b,e} \geq 1, \ \ \forall e \in E \\
f_{a,e} \leq \sum_{b \in H_d \bigcap S'_a} f_{b,e}, \ \ \forall a \in \left(H'_d - B_e\right), \ \forall d \in D, \forall e \in E \\
T_a - T_b \geq MinCT \cdot (f_{a,e} + f_{b,e} - 1), \quad \forall b \in A, \forall e \in E, \ \forall a \in S'_b \\
f_{a,e} + f_{b,e} \leq 1, \quad \forall b \in A, \ \ \forall a \notin S'_b \bigcup S_b \\
J_a \geq 0, \ \ \forall a \in A \\
J_a \geq f_a \cdot F_a - \sum_{e \in (F-C)} f_{a,e}, \ \ \forall a \in A \\
\sum_{a \in Q_z} \left(T'_a - T_a\right) \cdot f_{a,e} \leq MaxBlk, \ \ \forall z \in Z, \forall e \in E \\
(T'_b - T_a) \cdot (f_{a,e} + f_{b,e} - 1) \leq G_{e,z}, \ \ \forall a \in Q_z, \forall b \in Q_z \bigcap S'_a, \ \ \forall z \in Z, \forall e \in E \\
G_{e,z} \leq MaxDP, \ \ \forall e \in E, \forall z \in Z \\
(T_b - T'_a) \geq (f_{a,e} + f_{b,e} - 1) \cdot MinRest, \ \ \forall a \in Q_z, \forall b \in Q_i, \forall i > z, \forall z \in Z, \forall e \in E \\
\sum_{z \in Z} G_{e,z} \leq L_1, \ \ \forall e \in E \\
L_0 \leq \sum_{z \in Z} G_{e,z}, \ \ \forall e \in E \\
\sum_{b \in H'_{B_e}} T'_b \cdot f_{b,e} - \sum_{b \in H_{B_e}} T_b \cdot f_{b,e} \geq MaxTAFB, \ \ \forall e \in E \\
(T_b - T'_a - MinVacDay) \cdot f_{b,e} \geq 0, \ \ \forall e \in E, \forall a \in H'_{B_e}, \forall b \in S'_a \\
\sum_{z=i}^{i+MaxSuccOn} f_{a,e} \leq MaxSuccOn, \ \ \forall a \in Q_z, \forall i \in Z \\
\sum_{b \in H'_{B_e}} T'_b \cdot f_{b,e} - \sum_{b \in H_{B_e}} T_b \cdot f_{b,e} \leq L_3, \ \ \forall e \in E \\
L_2 \leq \sum_{b \in H'_{B_e}} T'_b \cdot f_{b,e} - \sum_{b \in H_{B_e}} T_b \cdot f_{b,e}, \ \ \forall e \in E
\end{cases}
$$

6.3 模型求解及结果分析

6.3.1 算法设计

本题所用算法是对问题二所提贪心算法的进一步优化，仍然是基于贪心和回溯的思想，在问题二所提算法的基础上我们做出了一些调整，使其符合问题三所给的条件。问题二所提算法详见 5.3.1 节。

（1）数据预处理

在本文提出的针对问题三的算法中，仍然对输入数据有一定的要求，因此首先需要对题目所给数据进行预处理。此处数据预处理流程与问题一中数据预处理流程完全相同，详见 4.3.1 节，此处不再赘述。

（2）算法流程

在算法流程上，我们做出了较大的调整。由于引入了任务环的概念，这对我们的约束条件有了更高的要求。因此我们需要在保证基本约束的前提下，尽可能实现目标①，使得航班能完全满足最低配置。具体来说，我们分别为拥有正机长资格的机组人员和只拥有副机长资格的机组人员创建了集合。假设某一时刻有航班需要起飞，基地中的机组人员足以满足此航班的最低机组配置。此时某基地中的机组人员集合如图 6-1 所示：

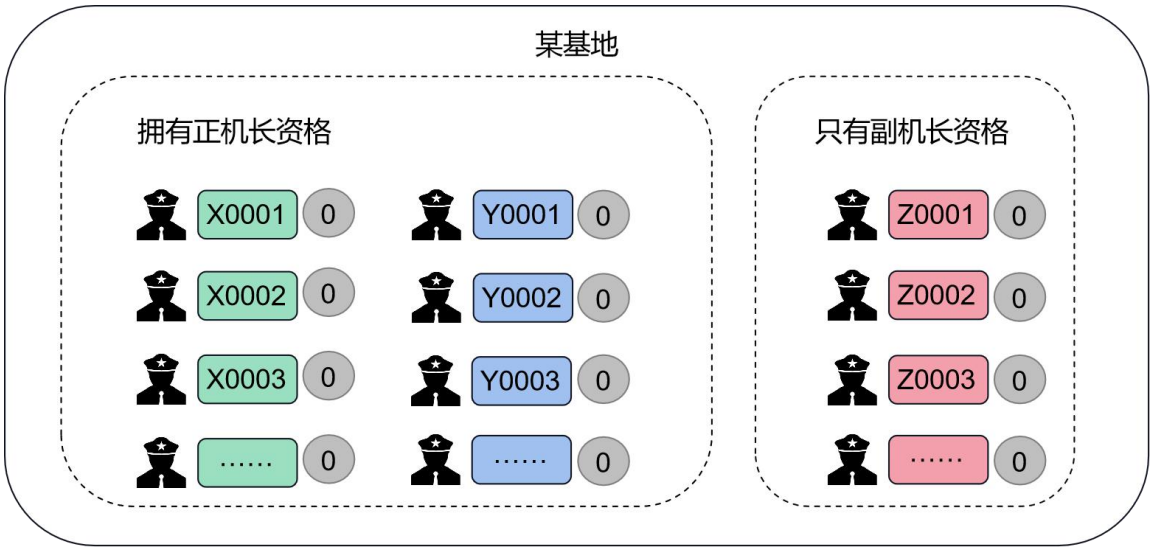

图 6-1 某基地中机组人员集合示意图

我们采用栈的数据结构来维护这两个集合，尽可能让先返回的机组人员继续出发，因为这样能最大程度地降低机组人员的执勤成本。我们考虑的分配方案具体为：将各个栈中的机组人员，按照其工号的奇偶性分为两类，分别对应每三天的工作。例如前三天的工作由工号末尾为奇数的机组人员负责，后三天的工作由工号末尾为偶数的机组人员负责。这样做的好处在于，可以保证所有机组人员的任务环始终控制在四天以内，哪怕遇到特殊情况，如第三天某机组成员接受任务前往另一机场，需等待第二天的飞机才能返回基地，他仍然能保证任务环的时间维持在 4 天以内，并在休息 2 天后可以赶上进度，执行下一个任务环。

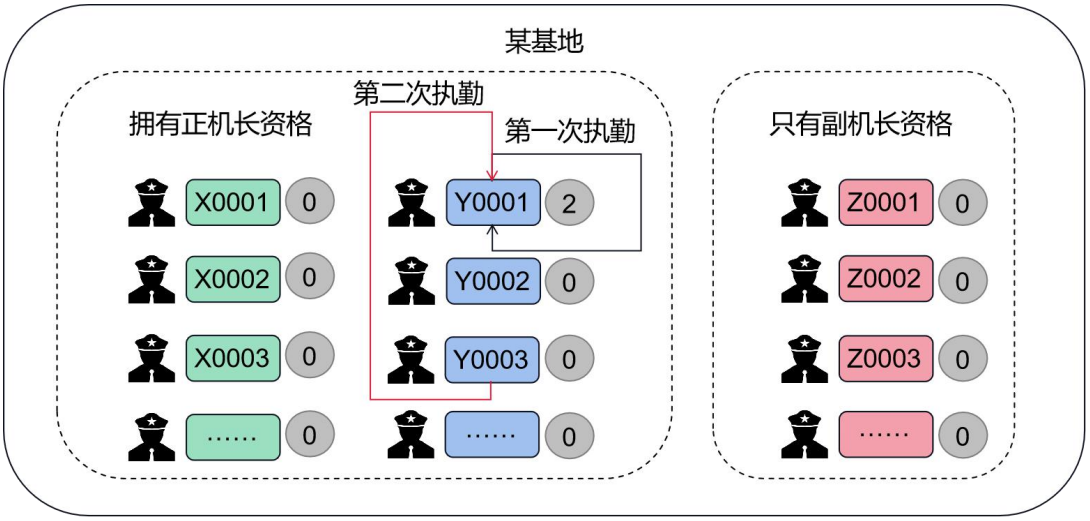

图 6-2 任务分配示意图

对于工号尾数奇偶性相同的机组人员，我们利用栈的后进先出特性对数据集合进行维护，但需要满足约束条件。例如，机组人员 Y0001 接收了第一个任务出发，并于当日返回，那么在下一个任务到来时，立刻对其进行判断，他今日的剩余飞行时间以及执勤时间能否允许他完成下一个任务。若能，我们将下一个任务分配给他；若不能，则我们让他今日休息，并向后查询下一个机组人员执行当前任务。

这一处理方法借用了贪心算法的理念，综合考虑了机组人员的利用率以及成本控制，同时还能兼顾任务环以及机组成员每日飞行时长和执勤时长的限制。

6.3.2 求解结果

根据 6.3.1 节中所述的算法，对 6.2 节中建立的多目标规划模型进行编程求解，得到了最终的排班计划。由于数据量较大，不宜在论文中占用过多篇幅进行全部展示，所以本文选取部分数据进行展示，全部详细求解结果见附件 CrewRosters3a.csv、附件 CrewRosters3b.csv 和附件 UncoveredFlights3b.csv。

（1）A 套数据

在针对 A 套数据的求解结果中，以员工 A0001 为例，该员工的排班计划如表 6-1 所示，由于篇幅所限，文中出发和到达日期字段中省略了年份，结果附件 CrewRosters3a.csv 中展示了完整的日期，其中年份都为 2021 年：

表 6-1 A 套数据中员工 A0001 的排班计划

员工号	航班号	出发日期	出发时间	出发机场	到达日期	到达时间	到达机场	任务性质
A0001	FA872	8/12	7:55	NKX	8/12	9:00	PLM	C
A0001	FA873	8/12	9:40	PLM	8/12	10:50	NKX	C
A0001	FA884	8/12	11:30	NKX	8/12	13:50	XGS	C
A0001	FA885	8/12	14:30	XGS	8/12	16:50	NKX	C
A0001	FA890	8/14	7:30	NKX	8/14	9:50	XGS	C
A0001	FA891	8/14	10:30	XGS	8/14	12:50	NKX	C
A0001	FA890	8/18	7:30	NKX	8/18	9:50	XGS	C
A0001	FA891	8/18	10:30	XGS	8/18	12:50	NKX	C
A0001	FA890	8/20	7:30	NKX	8/20	9:50	XGS	C

A0001	FA891	8/20	10:30	XGS	8/20	12:50	NKX	C
A0001	FA890	8/24	7:30	NKX	8/24	9:50	XGS	C
A0001	FA891	8/24	10:30	XGS	8/24	12:50	NKX	C

注：任务性质中 C 表示正机长，F 表示副机长，D 表示乘机

所有航班的起飞情况如图 6-3 所示：

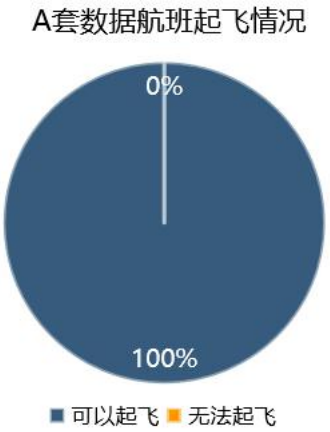

图 6-3 A 套数据求解结果中航班起飞情况

从图中可以看出，在我们的求解结果中，**所有航班全部满足机组配置，可以起飞。**

共计 420 人次成功起飞，其中任务性质为正机长的人次为 206，副机长 206 人次，**乘机 8 人次，替补 0 人次**，如图 6-4 所示：

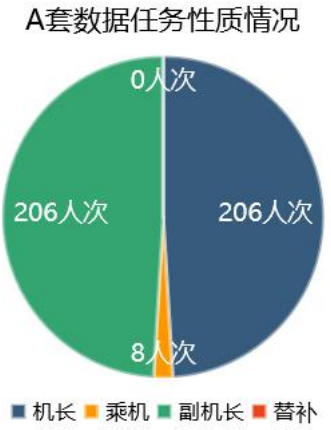

图 6-4 A 套数据求解结果中任务性质情况

为了衡量执勤的效果，我们计算了关于一次执勤飞行时长、一次执勤执勤时长和机组人员执勤天数的最小值、最大值和平均值，如图 6-5 所示：

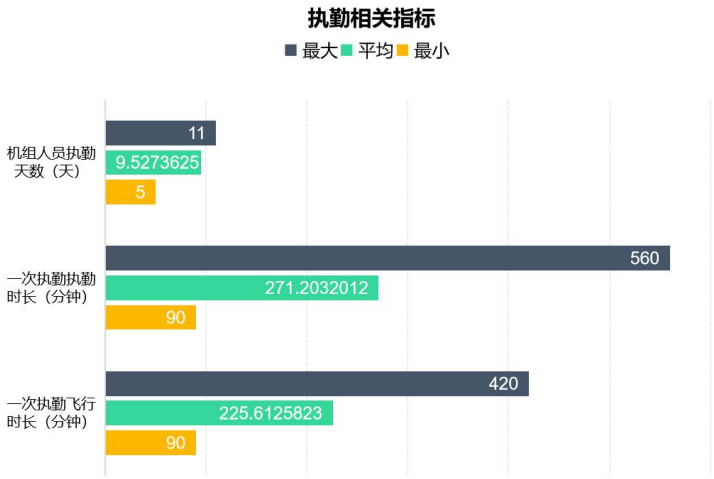

图 6-5 A 套数据求解结果中执勤相关指标

除此之外，由于本题引入了任务环概念，因此需要对任务环分配情况进行统计，结果如图 6-6 所示：

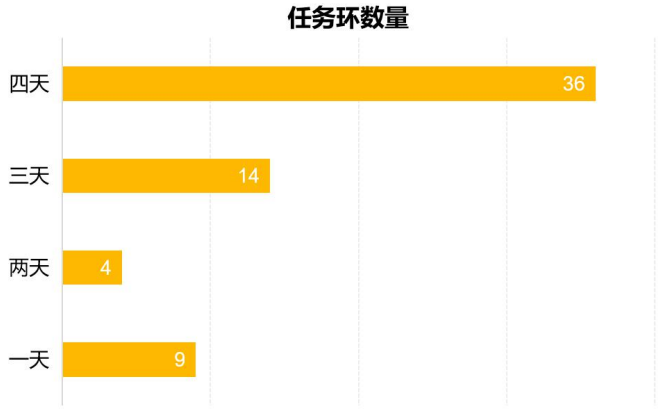

图 6-6 A 套数据求解结果中任务环数量情况

利用本文所提的算法针对 A 套数据求解排班计划，在求解结果中不存在因不满足最低机组资格配置而无法起飞的航班，即**所有航班全部满足机组配置。总体乘机次数为 8 次，没有使用替补资格**。

（2）B 套数据

在针对 B 套数据的求解结果中，以员工 B0001 为例，该员工的排班计划如表 6-2 所示，同样地，在此省略了年份，实际应为 2019 年。由于结果集较大，此处仅展示前 10 条数据。结果附件 CrewRosters3b.csv 中展示了全部机组人员排班计划，UncoveredFlights3b.csv 中罗列了无法起飞的航班详情。

表 6-2 B 套数据中员工 B0001 的排班计划（部分）

员工号	航段序号	航班号	出发日期	出发时间	出发机场	到达日期	到达时间	到达机场	任务性质
B0001	1	FB8559	8/01	0:05	HOM	8/01	2:05	SXA	C
B0001	461	FB8560	8/02	6:35	SXA	8/02	8:30	HOM	C
B0001	2746	FB1731	8/07	8:00	HOM	8/07	10:00	SXA	D
B0001	2830	FB1734	8/07	10:40	SXA	8/07	12:45	HOM	D
B0001	3994	FB38	8/09	19:00	HOM	8/09	20:15	TGD	D

B0001	4045	FB99	8/09	21:00	TGD	8/09	22:20	HOM	D
B0001	4638	FB20	8/11	10:00	HOM	8/11	11:15	TGD	D
B0001	4778	FB31	8/11	15:00	TGD	8/11	16:05	HOM	C
B0001	5357	FB38	8/12	19:00	HOM	8/12	20:15	TGD	C
B0001	5403	FB97	8/12	20:30	TGD	8/12	21:50	HOM	D

注：任务性质中 C 表示正机长，F 表示副机长，D 表示乘机

所有航班的起飞情况如图 6-7 所示：

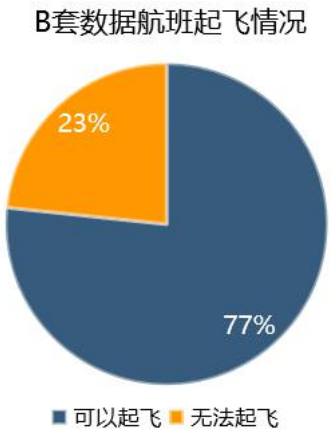

图 6-7 B 套数据求解结果中航班起飞情况

在我们的求解结果中，有 3257 个航班因无法满足最低机组配置而不能起飞。起飞航班中任务性质的情况如图 6-8 所示：

图 6-8 B 套数据求解结果中任务性质情况

执勤相关指标如图 6-9 所示：

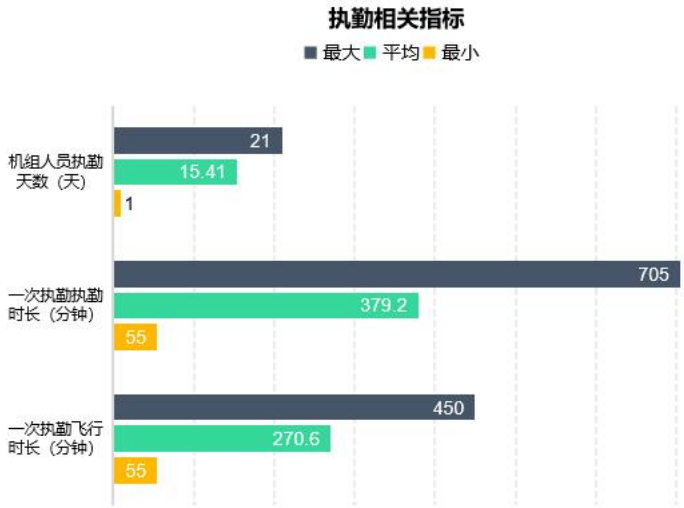

图 6-9 B 套数据求解结果中执勤相关指标

除此之外，由于本题引入了任务环概念，因此需要对任务环分配情况进行统计，结果如图 6-10 所示：

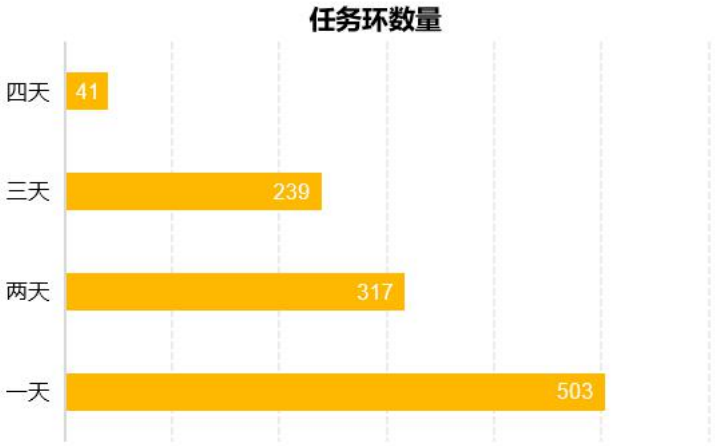

图 6-10 A 套数据求解结果中任务环数量情况

问题二中 AB 两套数据的求解结果指标如表 6-3 所示：

表 6-3 问题三结果指标

结果指标	A 套数据	B 套数据
不满足机组配置航班数	0	3257
满足机组配置航班数	206	10697
机组人员总体乘机次数	8	8442
替补资格使用次数	0	0
机组总体利用率	82.98%	57.6%
最小一次执勤飞行时长（分钟）	90	55
平均一次执勤飞行时长（分钟）	225.6125823	270.6
最大一次执勤飞行时长（分钟）	420	450
最小一次执勤执勤时长（分钟）	90	55
平均一次执勤执勤时长（分钟）	271.20320121	379.2
最大一次执勤执勤时长（分钟）	560	705
最小机组人员执勤天数	5	1

平均机组人员执勤天数	9.5273625	15.41
最大机组人员执勤天数	11	21
一天任务环数量	9	503
两天任务环数量	4	317
三天任务环数量	14	239
四天任务环数量	36	41
总体执勤成本（万元）	56.8231	4298.88
总体任务环成本（万元）	7.5373	852.72
程序运行分钟数	0.00364543287	0.7007563333

6.3.3 算法有效性及复杂度分析

本题引入了任务环的概念，因此我们对问题二中提出的算法再次进行了优化。虽然我们对算法流程进行了调整，但是并没有改变算法的时间复杂度。在最坏情况下，算法的**时间复杂度仍然是 $O(n^2)$，最好情况下时间复杂度为 O(n)。程序在 A 套数据上运行时间为 0.00364543287 分钟，在 B 套数据上运行时间为 0.68906753 分钟。**

7. 模型的评价

7.1 模型的优点

本文在仔细分析题意的基础上，针对航班和机组人员的对应关系和航班能否起飞设定了决策变量，并根据题目中给出的约束条件，对模型中的决策变量进行了线性约束。

问题一中，我们根据题目要求对机组人员所能搭乘的航班进行了约束，并给出了符合题意的三层目标函数。若存在多个满足首要目标的解，我们则进一步将该首要目标的最优解作为新的约束条件，并求解次要目标，以此类推。

问题二中，我们引入了机组人员执勤的概念，在问题一的约束条件基础上，针对机组人员的执勤排班情况添加了相应的约束条件，使其满足执勤排班的要求以及问题二中新添加的限制条件。另外，我们还对新增的目标函数进行了表达，以此控制机组人员的总执勤成本并使得机组人员的总执勤时间的最大值和最小值之差尽可能小。

问题三中，引入了任务环的概念。因此我们添加了成员排班计划的约束条件，使得成员的排班更贴近实际情况。首先，根据题意添加了与机组人员排班相关的新条件约束，同时我们也给出了新的目标函数，用于控制机组人员的总任务环成本，并使得机组人员的总任务环时间最大值和最小值的差尽可能小。

三个问题的模型层层递进，通过添加新的约束和目标使得整个模型更加完整，更具实用意义。

本文提出的多目标规划模型具有以下优点：

（1）模型结构简单明了，使用二维决策变量，使得决策空间较小。

（2）模型的约束条理清晰，通过多重线性进行约束，求解变得高效，且易于实现。

（3）模型中保留了大量的已知常量，使其易被拓展，当约束条件发生改变，该模型仍可用于求解，具有良好的拓展性。

7.2 模型的缺点

由于比赛时间限制，未能对模型进行进一步的优化，本文所提出的模型还存在着一些缺点：

（1）模型的求解结果比较固定，分层的目标使得求得的结果专注于首要目标，在次要目标的表现不一定达到最优。

（2）模型求解的结果对数据精确度要求高，遇到航班延误等特殊情况难以进行有效调整。

参考文献

[1] Xiaodong Luo, Yogesh Dashora, Tina Shaw (2015) Airline Crew Augmentation: Decades of Improvements from Sabre. Interfaces 45(5): 409-424.

[2] Saeed Saemi, Alireza Rashidi Komijan, Reza Tavakkoli-Moghaddam and Mohammad Fallah, A new mathematical model to cover crew pairing and rostering problems simultaneously, Journal of Engg. Research Vol. 9 No. (2) June 2021 pp. 218-233

[3] Mohamed Haouari, Farah Zeghal Mansour, Hanif D. Sherali (2019) A New Compact Frmulation Fr the Daily Crew Pairing Problem. Transportation Science 53(3): 811-828

[4] 龚强. 人工变量法—大 M 法:测绘运筹学线性规划问题的进一步讨论[J]. 地理信息世界, 1996(3).

[5] 向杜兵. 航空机组排班计划一体化优化研究[D].北京交通大学,2020.

[6] 肖真真. 基于任务均衡的航空公司机组人员指派问题研究[D].中国民用航空飞行学院,2012.

[7] A hybrid meta-heuristic algorithm for optimization of crew scheduling[J] . A. Azadeh,M. Hosseinabadi Farahani,H. Eivazy,S. Nazari-Shirkouhi,G. Asadipour. Applied Soft Computing Journal . 2013 (1)

[8] Modeling and solving a Crew Assignment Problem in air transportation[J] . F.M. Zeghal,M. Minoux. European Journal of Operational Research . 2005 (1)

附录

注：附录中只含有部分代码，详细完整代码请见附件。

```
import csv
import datetime
import json
import numpy as np
import pandas as pd
#from gurobipy import *
import time

def read_data():
    crew =
pd.read_csv(r'C:\Users\84704\Desktop\TP_crew_shift-master\params\A-Crew.csv',
usecols=['EmpNo','Captain', 'FirstOfficer', 'Deadhead', 'Base', 'DutyCostPerHour',
'ParingCostPerHour'])
    print(crew)

    plane =
pd.read_csv(r'C:\Users\84704\Desktop\TP_crew_shift-master\params\A-Flight.csv',
usecols=['FltNum','DptrDate', 'DptrTime', 'DptrStn', 'ArrvDate', 'ArrvTime',
'ArrvStn','Comp'])
    print(plane)

    return crew,plane


class Pcrew():
    def __init__(self,EmpNo,Captain, FirstOfficer, Deadhead, Base, DutyCostPerHour,
ParingCostPerHour):
        self.EmpNo = EmpNo
        self.Captain = Captain
        self.FirstOfficer = FirstOfficer
        self.Deadhead = Deadhead
        self.Base = Base
        self.DutyCostPerHour = DutyCostPerHour
        self.ParingCostPerHour = ParingCostPerHour
        self.TaskNum=[]
        self.TaskProp=[]

class Ncrew():
    def __init__(self,EmpNo,Captain, FirstOfficer, Deadhead, Base, DutyCostPerHour,
ParingCostPerHour):
        self.EmpNo = EmpNo
```

```
        self.Captain = Captain
        self.FirstOfficer = FirstOfficer
        self.Deadhead = Deadhead
        self.Base = Base
        self.DutyCostPerHour = DutyCostPerHour
        self.ParingCostPerHour = ParingCostPerHour
        self.TaskNum=[]
        self.TaskProp=[]

class CrewScheduling(object):
    def __init__(self,crew, plane):
        self.crew, self.plane = crew,plane
        self.crew_size = len(self.crew['Captain'])
        self.line_sum = len(self.plane['DptrDate'])

        print(self.crew_size)

        self.Pcrew = 0
        self.Ncrew = 0
        for i in range(0, self.crew_size):
            if(self.crew['Captain'][i]=='Y'):
                self.Pcrew += 1
            else:
                self.Ncrew += 1

        print(self.Pcrew)
        print(self.Ncrew)
        # print(self.crew['DutyCostPerHour'][0])
        # print(self.plane['DptrDate'][0])

        self.init_P1=init_station('NKX',self.crew)
        self.station_vec=[]
        self.station_vec.append('NKX')
        self.P=[]
        self.P.append(self.init_P1)
        for i in range(0,self.line_sum):
            if(self.plane['ArrvStn'][i] not in self.station_vec ):
                self.station_vec.append(self.plane['ArrvStn'][i])
                self.P.append(station(self.plane['ArrvStn'][i]))
        print(self.station_vec)
        print(self.P[0].name)
        print(self.P[0].Pcrew_num)

    def fly(self, a, b ,add_num, airline_num):
```

```
        pcrew_list = []
        ncrew_list = []
        for i in range(0,add_num + 1):
            pcrew_list.append(i)
            ncrew_list.append(i)


        pcrew,ncrew=a.leave(pcrew_list,ncrew_list,airline_num)
        b.arrive(pcrew,ncrew)

        print(a.name," to ",b.name," use ",len(pcrew_list)," P ",len(ncrew_list),"
N.")

        return 0


    def find_unpair(self):

        pair_list=[]
        pair_time=[]
        pair_day = []
        re_unpair_index=[]
        add_index=[]
        add_num=[]

    #寻找不对称点
        for i in range(0,self.line_sum):
            start_sta = self.P[self.station_vec.index(self.plane['DptrStn'][i])]
            arrive_sta = self.P[self.station_vec.index(self.plane['ArrvStn'][i])]
            start_time = self.plane['DptrTime'][i]
            start_day = self.plane['DptrDate'][i]
            arrive_time = self.plane['ArrvTime'][i]
            arrive_day = self.plane['ArrvDate'][i]
            if(start_sta.name == self.init_P1.name):     #加入队列，如从初始点出发
                pair_list.append(arrive_sta.name)
                pair_time.append(arrive_time)
                pair_day.append(arrive_day)
            print(pair_list)
            if(arrive_sta.name == self.init_P1.name):    #弹出队列，如到达初始点

                if(len(pair_list)==0):
                    print(i,'line',start_sta.name,'to',arrive_sta.name,'error')
                    re_unpair_index.append(i)
```

```
                else:
                    # d1=datetime.datetime.strptime(start_time, '%Y-%m-%d %H:%M:%S')
                    d1 = datetime.datetime.strptime(start_time, '%H:%M')
                    d2 =
datetime.datetime.strptime(pair_time[pair_list.index(start_sta.name)], '%H:%M')
                    day1 = datetime.datetime.strptime(start_day, '%m/%d/%Y')
                    day2 =
datetime.datetime.strptime(pair_day[pair_list.index(start_sta.name)], '%m/%d/%Y')

                    delta_d = day1.day - day2.day
                    delta_h = d1.hour - d2.hour
                    delta_m = d1.minute - d2.minute
                    delta = delta_d*1440 + delta_h * 60 + delta_m

                    if(delta>=40 or (start_sta.Pcrew_num>=1 and
start_sta.Ncrew_num>=1)):
                        pair_day.pop(pair_list.index(start_sta.name))
                        pair_time.pop(pair_list.index(start_sta.name))
                        pair_list.pop(pair_list.index(start_sta.name))

            print('剩余航班数',len(pair_list))

    # 寻找不对称点的乘机捎带点
        for i in range(0, len(re_unpair_index)):
            staname=self.plane['DptrStn'][re_unpair_index[i]]
            start_time = self.plane['DptrTime'][re_unpair_index[i]]
            start_day = self.plane['DptrDate'][re_unpair_index[i]]

            print(staname)
            for j in range(re_unpair_index[i],-1,-1):
                start = self.plane['DptrStn'][j]
                end = self.plane['ArrvStn'][j]
                end_time=self.plane['ArrvTime'][j]
                end_day = self.plane['ArrvDate'][j]


                d1 = datetime.datetime.strptime(start_time, '%H:%M')
                d2 = datetime.datetime.strptime(end_time, '%H:%M')
                day1 = datetime.datetime.strptime(start_day, '%m/%d/%Y')
                day2 = datetime.datetime.strptime(end_day, '%m/%d/%Y')

                delta_d = day1.day - day2.day
                delta_h = d1.hour - d2.hour
                delta_m = d1.minute - d2.minute
```

```
            delta = delta_d*1440 + delta_h * 60 + delta_m

            if(start==self.init_P1.name and end == staname and delta>=40 ):
                add_index.append(j)
                break
    print(re_unpair_index)
    print(add_index)
    return re_unpair_index,add_index

def Schedu(self):
    #for i in range(0, self.line_sum):

    starttime = datetime.datetime.now()
    re_unpair_index,add_index = self.find_unpair()
    endtime = datetime.datetime.now()
    for i in range(0,len(self.plane['DptrStn'])):
        add_num=0
        for j in range(0,len(add_index)):
            if(i==add_index[j]):
                add_num+=1

        a=self.P[self.station_vec.index(self.plane['DptrStn'][i])]
        b=self.P[self.station_vec.index(self.plane['ArrvStn'][i])]
        print(i+1,'次航班')
        self.fly(a,b,add_num,i+1)
        print(a.name,'have',a.Pcrew_num,'Pcrew')
        print(b.name,'have',b.Pcrew_num,'Pcrew')

    print('总乘机次数为',len(add_index)*2)

    empno=[]
    air_num=[]
    Prop=[]
    line_num=[]
    stdate=[]
    sttime=[]
    ststa=[]
    ardate=[]
    artime=[]
    arsta=[]
    for i in range(0,len(self.init_P1.Pcrew)):
        for j in range(0,len(self.init_P1.Pcrew[i].TaskNum)):
             empno.append(self.init_P1.Pcrew[i].EmpNo)
            air_num.append(self.init_P1.Pcrew[i].TaskNum[j])
```

```
                Prop.append(self.init_P1.Pcrew[i].TaskProp[j])

line_num.append(self.plane['FltNum'][self.init_P1.Pcrew[i].TaskNum[j] - 1])

stdate.append(self.plane['DptrDate'][self.init_P1.Pcrew[i].TaskNum[j] - 1])

sttime.append(self.plane['DptrTime'][self.init_P1.Pcrew[i].TaskNum[j] - 1])
                ststa.append(self.plane['DptrStn'][self.init_P1.Pcrew[i].TaskNum[j]
- 1])

ardate.append(self.plane['ArrvDate'][self.init_P1.Pcrew[i].TaskNum[j] - 1])

artime.append(self.plane['ArrvTime'][self.init_P1.Pcrew[i].TaskNum[j] - 1])
                arsta.append(self.plane['ArrvStn'][self.init_P1.Pcrew[i].TaskNum[j]
- 1])

        for i in range(0,len(self.init_P1.Ncrew)):
            for j in range(0,len(self.init_P1.Ncrew[i].TaskNum)):
                # empno=self.init_P1.Ncrew[i].EmpNo
                                empno.append(self.init_P1.Ncrew[i].EmpNo)
                air_num.append(self.init_P1.Ncrew[i].TaskNum[j])
                Prop.append(self.init_P1.Ncrew[i].TaskProp[j])

line_num.append(self.plane['FltNum'][self.init_P1.Ncrew[i].TaskNum[j] - 1])

stdate.append(self.plane['DptrDate'][self.init_P1.Ncrew[i].TaskNum[j] - 1])

sttime.append(self.plane['DptrTime'][self.init_P1.Ncrew[i].TaskNum[j] - 1])
                ststa.append(self.plane['DptrStn'][self.init_P1.Ncrew[i].TaskNum[j]
- 1])

ardate.append(self.plane['ArrvDate'][self.init_P1.Ncrew[i].TaskNum[j] - 1])

artime.append(self.plane['ArrvTime'][self.init_P1.Ncrew[i].TaskNum[j] - 1])
                arsta.append(self.plane['ArrvStn'][self.init_P1.Ncrew[i].TaskNum[j]
- 1])

        dataframe = pd.DataFrame({'empno': empno, 'air_num':
air_num,'line_num':line_num,'stdate':stdate,'sttime':sttime,

'ststa':ststa,'ardate':ardate,'artime':artime,'arsta':arsta,'Prop':Prop})
        dataframe.to_csv("1A.csv", index=False, sep=',')
        print((endtime - starttime))
        return 0
```

```
class init_station(object):
    def __init__(self,name,crew):
        self.crew = crew
        self.crew_size = len(crew['Captain'])
        self.Pcrew_num=0
        self.Ncrew_num=0
        self.Pcrew = []
        self.Ncrew = []

        self.name=name

        for i in range(0, self.crew_size):
            if(crew['Captain'][i]=='Y'):
                self.Pcrew_num += 1

self.Pcrew.append(Pcrew(crew['EmpNo'][i],crew['Captain'][i],crew['FirstOfficer'][
i],crew['Deadhead'][i],crew['Base'][i],crew['DutyCostPerHour'][i],crew['ParingCos
tPerHour'][i]))
            else:
                self.Ncrew_num += 1
                self.Ncrew.append(
                    Ncrew(crew['EmpNo'][i], crew['Captain'][i],
crew['FirstOfficer'][i], crew['Deadhead'][i],
                          crew['Base'][i], crew['DutyCostPerHour'][i],
crew['ParingCostPerHour'][i]))


    def leave(self,a,b,airline_num):
        if(len(self.Pcrew)==0):
            print("error")
        if(len(self.Ncrew)==0):
            flag=0
            if(len(self.Pcrew)==1):
                print("error")
            else:
                for i in range(0, len(self.Pcrew)):
                    temp=self.Pcrew[i]
                    if(temp.FirstOfficer=='Y'):
                        flag = 1
                if(flag==0):
                    print("error")
```

```
        REP=[]
        REN=[]
        for i in range(0,len(a)):
            self.Pcrew[a[i]].TaskNum.append(airline_num)

            self.Pcrew[a[i]].TaskProp.append('C')
            REP.append(self.Pcrew.pop(a[i]))
            self.Pcrew_num -= 1
        for i in range(0, len(b)):
            self.Ncrew[b[i]].TaskNum.append(airline_num)
            self.Ncrew[b[i]].TaskProp.append('F')
            REN.append(self.Ncrew.pop(b[i]))
            self.Ncrew_num -= 1

        return REP,REN

    def arrive(self,a,b):
        for i in range(0,len(a)):
            self.Pcrew.append(a[i])
            self.Pcrew_num += 1
        for i in range(0, len(b)):
            self.Ncrew.append(b[i])
            self.Ncrew_num += 1
        return 0


class station(object):
    def __init__(self,name):
        self.Pcrew=0
        self.Ncrew=0
        self.name=name
        self.Pcrew_num=0
        self.Ncrew_num=0
        self.Pcrew = []
        self.Ncrew = []

    def leave(self,a,b,airline_num):
        if(len(self.Pcrew)==0):
            print("error")
        if(len(self.Ncrew)==0):
            flag=0
            if(len(self.Pcrew)==1):
                print("error")
```

```
            else:
                for i in range(0, len(self.Pcrew)):
                    temp=self.Pcrew[i]
                    if(temp.FirstOfficer=='Y'):
                        flag = 1
                if(flag==0):
                    print("error")


        REP=[]
        REN=[]
        for i in range(0,len(a)):
            self.Pcrew[a[i]].TaskNum.append(airline_num)

            self.Pcrew[a[i]].TaskProp.append('C')
            REP.append(self.Pcrew.pop(a[i]))
            self.Pcrew_num -= 1
        for i in range(0, len(b)):
            self.Ncrew[b[i]].TaskNum.append(airline_num)
            self.Ncrew[b[i]].TaskProp.append('F')
            REN.append(self.Ncrew.pop(b[i]))
            self.Ncrew_num -= 1

        return REP,REN

    def arrive(self,a,b):
        for i in range(0,len(a)):
            self.Pcrew.append(a[i])
            self.Pcrew_num += 1
        for i in range(0, len(b)):
            self.Ncrew.append(b[i])
            self.Ncrew_num += 1
        return 0


if __name__ == '__main__':
    crew, plane = read_data()
    q=CrewScheduling(crew, plane)
    q.Schedu()
```