参赛密码 _____

（由组委会填写）

“华为杯”第十五届中国研究生数学建模竞赛

学　　校	上海大学
参赛队号	18102800248
队员姓名	1. 应祺超
	2. 顾银娟
	3. 焦琪涵

参赛密码 _____

（由组委会填写）

“华为杯”第十五届中国研究生数学建模竞赛

题 目 基于启发式混合模型的航班中转规划问题

摘 要：

为停靠机场的航班无冲突地分配停机位，是机场调度工作的核心业务，关系到整个机场的正常运作。调度算法的性能，不仅直接关系到机场以及与之有密切业务来往的各航空公司的核心利益，也关系到到站旅客的乘坐体验，其重要性不言而喻。不恰当的分配方式会导致航班延误和冲突、降低旅客满意度，影响旅客的日常生活，更不排除因此发生严重拥挤，造成事故的可能。在求解难度上，机场停机位分配问题(ASA)属于 NP-hard 问题，直接通过遍历解集空间得到该问题全局最优解的方法一般需要非常高的时间复杂度，难以满足该问题的实时性需求。因此，本论文旨在通过建立混合启发式模型，将求解问题离散化，利用合理的限制条件尽可能使得启发式算法提供的解空间在规模上不断缩

小，并能够不断逼近理论存在的全局最优解。此外，针对不同场景的设计，我们根据需要变化了我们的模型，使得本文中的三个待求解难题都可以基于特定的模型得到尽可能优化的结果。

对于问题一,我们的首要目标是尽可能将最多的飞机安排到固定停机位中，因此我们建立了禁忌算法与贪婪算法的启发式混合模型，总体以贪婪算法作为模型初始化的原则，并在求解过程中使用禁忌算法，防止求解中陷入局部最优解的情况。对于求取近似全局最优解的过程中出现的同解情况，我们用两种不同的策略设计了两个对生成结果筛选精细程度不同的启发式模型，第一个模型全面利用机场停机位分配的特点，并着重考虑机位均衡利用，第二个模型在每次迭代时可以提供更多的解，因此进一步提升性能。两种方法平均可以达到83.17%分配成功率。

作为问题一的合理延申，我们也为同时优化使用最少数量的停机坪与使用最少数量的固定机位设计了带平衡项的启发式求导模型，可以为解决这对矛盾提供很好的动态方案。

对于问题二，我们更新了问题一中模型的约束条件，使得问题一中的模型在遇到优化临时登机口同解的情况下,进一步考虑并优化中转乘客的换乘时间，在此过程中，我们为航班分配导致的乘客换乘失败的情况加上了高额的惩罚，让模型也具备了优先保证最多数量乘客可以顺利换乘的能力。我们的模型最终可以让所有的乘客都顺利换乘，且将平均换乘时间控制在 33 分钟。

对于第三问，我们考虑了前两问模型的应用局限，设计了基于蚁群算法的启发式模型，将问题转化为对图的顶点着色问题，通过设置邻接矩阵，使蚂蚁在可行域内一步步构建有化解。这个模型可以给出与之相比更好的优化结果。

最后，我们通过实验仿真数据，对算法的有效性进行验证和分析。

关键词：启发式模型；禁忌算法；蚁群算法；NP-hard 问题；航班安排

目 录

一、问题的背景与重述

1.1 问题背景

随着社会经济的发展和人民生活水平的提高，中国旅游业正处于高速发展期，航空出行作为远距离出行的一种交通工具也成为越来越多旅客的首选方式。但众所周知，随着飞机航班的增多，机场航站楼旅客流量呈现饱和状态。为应对未来机场的发展，现增设卫星厅用于实现旅客的分流，缓解机场航站楼登机口不足的压力。然而，虽然增设卫星厅可以有效地扩容登机口数量，但是对于中转旅客的航班衔接也提出了很高的要求。飞机在机场登机口的一次停靠通常由到达航班和出发航班(转场)来标识，如下图一所示。航班-登机口分配就是把这样的航班对分配到合适的登机口。中转旅客就是从到达航班换乘到由同一架或不同架飞机执行的出发航班的旅客。单纯的航班-登机口优化分配问题已有很多优秀的算法对此进行优化改善，但现增设了卫星厅，在分配登机口这个问题上必须考虑中转旅客在转场过程中所消耗的时间。

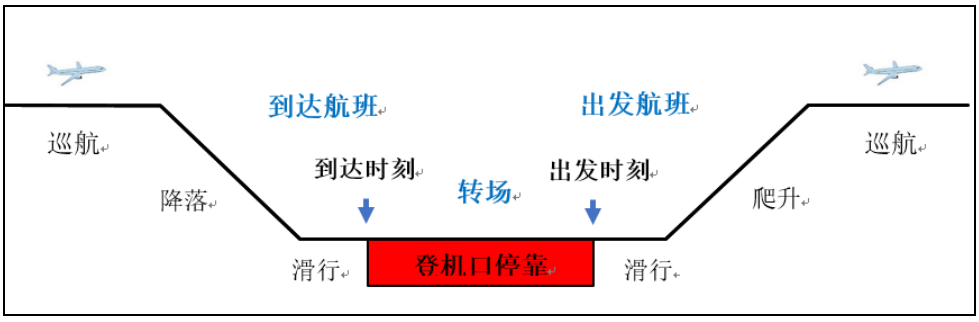


图 1 航班及其停靠示意图

1.2 问题重述

本文需要在优化登机口分配问题的基础上最小化旅客行走时间。真实的航班-登机口分配调度方案会受到各种制约因素的影响，但作为评估，本文根据三种情况对其进行约束。首先是只考虑航班-登机口分配，建立航班-登机口分配问题数学优化模型，尽可能多地分配航班到合适的登机口，在此基础上，最小化倍数用登机口的数量。其次，加入旅客换乘因素，最小化中转旅客的最短流程时间。最后，如上文所述，新增卫星厅对航班的最大影响是中转旅客换乘时间可能被延长，因此需要在前面的基础上考虑中转旅客的换乘时间，要求最小化换乘旅客总体的紧张度。

1.2.1 问题一

问题一分析新增卫星厅对航班影响，首先只考虑航班-登机口分配问题，对于中转旅客的换乘情况不做考虑。航站楼有 28 个登机口、卫星厅有 41 个登机口，共 69 个登机口。本题主要根据 2018 年 1 月 30 日到达或 20 日出发的航班和旅客进行分析，尽可能多的分配航班到合适的登机口，并在此基础上最小化被使用登机口的数量。问题一需要优先求出使用临时停机位最少的结果、给出成功分配到登机口的航班数量和比例、航站楼和卫星厅的各个登机口的使用次数以及登机口的平均使用率。

1.2.2 问题二

问题二要求在最小化临时停机位前提下，考虑最小化中转旅客最短流程时间，并在此基础上最小话登机口的数量，对于旅客乘坐捷运时间和行走时间不作考虑。问题二要求计算出问题一包含的各项问题，同时给出换成失败旅客的数量和比率。此处换乘失败是指未能赶上中转航班。另外还需根据中转流程时间算出总体旅客的换乘时间分布图(分布图不需考虑捷运时间和步行时间)。

1.2.3 问题三

问题三要求考虑换乘旅客总体紧张度，在保证尽可能少的使用临时停机位的条件下，考虑最小化换乘旅客的总体紧张度，最后是考虑最小化登机口使用数量。问题三在求解出问题二包含的问题基础上，要求得到总体旅客的换乘紧张分布图。

二、符号说明与模型假设

2.1 部分符号说明

符号	符号说明
N	表示航班的数量
M	表示登机口的数量
$M+1$	表示停机坪
D	表示国内航班
I	表示国际航班
γ	表示分配在同一登机口的两架飞机最小空档间隔时间
a_i	表示航班 i 使用某一登机口的开始时间
d_i	表示航班 i 使用某一登机口的结束时间
F_i	表示航班 i 的机型，当且仅当 i 为宽体机时 F_i =1，反之为 F_i =0
S_k	表示登机口 k 的大小，当且仅当 k 为大登机口时 S_k =1，否则 S_k =0
y_{ik}	0-1 变量，若航班 i 被分配到登机口 k 中，y_{ik} =1，否则 y_{ik} =0
z_{ijk}	0-1 变量，当且仅当航班 i 和航班 j 均被分配到登机口 k，并且航班 i 恰好先于航班 j 时，z_{ijk} =1，否则 z_{ijk} =0
$X_1(k)$	表示第 k 个登机口的到达类型，$X_1(k)\in\{D,I\}$
$X_2(k)$	表示第 k 个登机口的出发类型，$X_2(k)\in\{D,I\}$
$X_1(i)$	表示第 i 个航班的到达类型，$X_1(i)\in\{D,I\}$
$X_2(i)$	表示第 i 个航班的出发类型，$X_1(i)\in\{D,I\}$
$i, j,$	下标 i, j 表示第 i 或是第 j 个航班；
m	下标 m 表示某一航班中转旅客的组号
k	下标 k 表示第 k 个登机口；
$R_{\mathrm{i,m}}$	表示第 i 个航班中第 m 组中转旅客的人数

$t_{\mathrm{i,m}}$	表示第 i 个航班中第 m 组中转旅客最短流程时间
$\bar{T}$	表示中转旅客的总的最短流程时间
m_i	表示第 i 个航班中中转旅客的总组数
$TE(k)$	表示第 k 个登机口的终端厅类型,$TE(k)\in\{T,S\}$
A	A=T 表示可以办理出入境， A=S 不能办理出入境
t	表示换乘总时长
F_{ail}	0-1 变量， F_{ail}=1，表示换乘失败，反之 F_{ail}=0，换乘成功
e_k	0-1 变量， e_k=1，第 k 个登机口未被使用，反之 e_k=0，被使用
$X_Z Y_W$	$X,Y\in\{D,I\}$， $Z,W\in\{T,S\}$，表示某一航班到达类型到达终端厅以及出发类型出发终端厅，例如 $\mathrm{D}_T\mathrm{I}_S$ 表示该航班到达类型为国内到达，终端厅为航站楼，出发类型为国际出发，终端厅为卫星厅。

【注】：部分建立在模型中的符号，在模型中进行具体说明。

2.2 模型基本假设

假设一：每架飞机转场的到达和出发两个航班分配在同一登机口进行，期间不能挪移。

假设二：分配在同一登机口的两飞机之间的空档间隔时间必须大于等于 45 分钟。

假设三：临时停机位数量无限制。

假设四：每个登机口的功能属性不能改变。

假设五：不考虑卫星厅对始发旅客和终到旅客的影响。

假设六：建模和数据分析都是针对航站楼 T 和卫星厅 S 同时使用的情形。

假设七：问题一不需要考虑中转旅客的换乘情况。

假设八：问题二中捷运时间和旅客行走时间不计入最短流程时间

三、模型选择

本章将介绍我们的团队在解答本论文中三个问题时，对于离散数学模型选择时的主要考虑与分析。3.1 节简单介绍经典的全局最优化算法，3.2 节将介绍一系列经典的启发式算法，3.3 节对 3.2 节介绍的算法进行应用时分析，并给出各方法给出的待解决组合优化问题的可行解的一般特性。值得注意的是，由于启发式算法可行解与最优解的偏离程度往往是不能被准确估计的。因此，现阶段的启发式算法仍然以仿照自然体系的算法为主.

3.1 全局最优化算法

本节介绍经典的全局最优化算法。全局最优化算法通过对解集空间的遍历，可以提供实际问题的全局最优解。不过，全局最优化算法的不足之处也是非常明显的。由于全局最优化算法一般只能应用于离散数学模型，又由于需要对几乎所有可能出现的解进行运算，因此使得全局最优解的时间复杂度往往非常高，往往无法在规定时间内得到有效的解。尽管可以通过人为干预手段减小时间开销，例如通过提早跳出循环、通过“备忘录”数据结构记录重叠求解结果等，但从根本上，这样的手段无法有效将全局最优化算法的时间复杂度降低一个或几个数量级，因此，全局最优算法更多只能应用于有中小规模全局解集的实际问题中。

3.1.1 回溯算法

回溯算法实际上一个类似枚举的搜索尝试过程，主要是在搜索尝试过程中寻找问题的解，当发现已不满足求解条件时，就“回溯”返回，尝试别的路径。回溯法是一种选优搜索法，按选优条件向前搜索，以达到目标。但当探索到某一步时，发现原先选择并不优或达不到目标，就退回一步重新选择，这种走不通就退回再走的技术为回溯法，而满足回溯条件的某个状态的点称为“回溯点”。比较经典的回溯算法应用实例是著名的“八皇后”问题。

回溯算法从原理上与“递归”的思维紧密联系，如果一个问题选择使用回溯算法来求解，则该问题的解空间通常是在搜索问题的解的过程中动态产生的，这是回溯算法的一个重要特性。回溯法是有组织的进行穷举，在试探过程中不断通过题设要求减少搜索空间，而这种减少不是一个一个解的减少，而是对搜索空间进行大规模剪枝，从而使得实际搜索空间远远小于问题的解空间。使用

回溯算法解决问题的一般步骤可以总结为以下几步：

(1) 针对所给问题，定义问题的解空间，它至少包含问题的一个（最优）解。

(2) 确定易于搜索的解空间结构,使得能用回溯法方便地搜索整个解空间 。

(3) 以深度优先的方式搜索解空间，并且在搜索过程中用剪枝函数避免无效搜索。

3.1.2 穷举法

穷举法的基本思想是根据题目的部分条件确定答案的大致范围，并在此范围内对所有可能的情况逐一验证，直到全部情况验证完毕。若某个情况验证符合题目的全部条件，则为本问题的一个解；若全部情况验证后都不符合题目的全部条件，则本题无解。穷举法也称为枚举法。穷举法也是常见的暴力破解法之一。

使用穷举法求解时，就是按照某种方式列举问题答案的过程。针对问题的数据类型而言，常用的列举方法一有如下三种：

(1) 顺序列举：是指答案范围内的各种情况很容易与自然数对应甚至就是自然数，可以按自然数的变化顺序去列举。

(2) 排列列举：有时答案的数据形式是一组数的排列，列举出所有答案所在范围内的排列，为排列列举。

(3) 组合列举：当答案的数据形式为一些元素的组合时，往往需要用组合列举。组合是无序的。

3.2 启发式算法

计算机科学的基础目标除了发现最优解外，同样重要的任务是提供可证明其执行效率良好且可得次佳解的算法。而启发式算法则试图一次提供一组目标（候选解集），因此，启发式算法常能发现很不错的解，但也没办法证明它不会得到较坏的解；它通常可在合理时间解出答案，但也没办法知道它是否每次都可以这样的速度求解。有时候人们会发现在某些特殊情况下，启发式算法会得到很坏的答案或效率极差，然而造成那些特殊情况的数据组合，也许永远不会在现实世界出现。因此现实世界中启发式算法常用来解决问题。启发式算法处理许多实际问题时通常可以在合理时间内得到不错的答案。

3.2.1 贪婪算法

贪婪算法又称贪心算法。在对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，它所做出的仅是在某种意义上的局部最优解。

贪心算法没有固定的算法框架，算法设计的关键是贪心策略的选择，贪心策略使用的前提是局部最优能导致全局最优。必须注意的是，贪心算法不是对所有问题都能得到整体最优解，选择的贪心策略必须具备无后效性，即某个状态以后的过程不会影响以前的状态，只与当前状态有关。所以对所采用的贪心策略一定要仔细分析其是否满足无后效性。

利用贪婪算法求解的基本思路可以归纳为以下四个步骤：

(1) 建立数学模型来描述问题。

(2) 把求解的问题分成若干个子问题。

(3) 对每一子问题求解，得到子问题的局部最优解。

(4) 把子问题的解局部最优解合成原来解问题的一个解。

3.2.2 遗传算法

遗传算法是一种通过模拟自然进化过程搜索最优解的方法。其主要特点是直接对结构对象进行操作，不存在求导和函数连续性的限定；具有内在的隐并行性和更好的全局寻优能力；采用概率化的寻优方法，不需要确定的规则就能自动获取和指导优化的搜索空间，自适应地调整搜索方向。

遗传算法直接对结构对象操作，无需函数求导和连续性的限定，全局搜索能力强，自适应的调整搜索方向和搜索空间且可并行。

遗传算法有以下几个重要的步骤

(1) 初始化：设置最大迭代进化次数 T，随机生成 M 个个体作为初始种群 P(0);

(2) 个体评价：计算当前种群 P(t)中的个体适应度；

(3) 选择：在个体评价之后，对群体进行选择操作目的是将优秀个体的基因通过组合配对交叉遗传到下一代种群中；

(4) 交叉：遗传算法中的核心部分；

(5) 变异：在个体基因的基础上进行变动，模拟自然界的基因突变，其变异

结果的好坏不定；

(6) 终止条件：若迭代次数达到预先设定的 T，将迭代过程中具有最优适应度的个体作为问题的解输出。

3.2.3 禁忌搜索算法

禁忌搜索算法是组合优化算法的一种，是局部搜索算法的扩展。禁忌搜索算法是人工智能在组合优化算法中的一个成功应用。禁忌搜索算法的特点是采用了禁忌技术。所谓禁忌就是禁止重复前面的工作。禁忌搜索算法用一个禁忌表记录下已经到达过的局部最优点，在下一次搜索中，利用禁忌表中的信息即可不再或有选择地搜索这些点。

禁忌算法中一些主要的概念包括：

(1) 初始解的获取：可以随机给出初始解，也可以事先使用其他启发式等算法给出一个较好的初始解；

(2) 移动邻域：移动是从当前解产生新解的途径，从当前解可以进行的所有移动构成邻域，也可以理解为从当前解经过“一步”可以到达的区域；

(3) 禁忌表：禁忌表的作用是为了防止搜索出现循环，记录前若干步走过的点、方向或目标值，禁止重复。并且随着优化搜索的进行，禁忌表的记录是动态更新的。禁忌的步数称为禁忌长度，表示禁忌表记忆的长度，禁忌长度的选择是很重要的， 禁忌长度过短，一旦陷入局部最优点，出现循环无法跳出；反之，禁忌长度过长，候选解全部被禁忌，造成计算时间较大，也可能造成计算无法继续下去；

(4) 候选集：候选集的选择一般由邻域中的邻居组成，可以选择所有邻居，也可以选择表现较好的邻居，还可以随机选择几个邻居；

(5) 评价函数和终止规则。

使用禁忌算法的基本思路可以概括为以下两步内容：

(1) 给定一个禁忌表 H,并选定一个初始解 X_now；

(2) 如果满足终止规则，则停止计算并输出结果；否则，在 X_now 的移动领域中选出满足不受禁忌的候选集 N(X_now)，并在 N(X_now)中选择一个评价值最好的解 X_next，用这个解更新历史记录 H, 并重复步骤（2）直到满足终止条件。

禁忌算法来解决本次的问题是因为与传统优化算法方法相比，禁忌算法的主要优势在于：

(1) 该算法从一个初始可行解出发，选择一系列的特定搜索方向（移动）作为试探，选择实现让特定的目标函数值变化最多的移动，在搜索过程中可以接受次优解，转向解空间的其他区域，以避免陷入局部最优解，增大获得更优的全局最优解的概率；

(2) 通过禁忌表的建立，禁忌搜索中采用了一种灵活的“记忆”技术，对已经进行的优化过程进行记录和选择，指导下一步的搜索方向；

(3) 搜素的新解不是在当前解的邻域空间中随机产生的，而是优于当前最好的解，或是非禁忌的最佳解，这样选取的优良解的概率远远增大。

3.3 选择合适的模型

选择禁忌算法来解决本次的问题是因为与传统优化算法方法相比，禁忌算法的主要优势在于：

(1) 该算法从一个初始可行解出发，选择一系列的特定搜索方向（移动）作为试探，选择实现让特定的目标函数值变化最多的移动，在搜索过程中可以接受次优解，转向解空间的其他区域，以避免陷入局部最优解，增大获得更优的全局最优解的概率；

(2) 通过禁忌表的建立，禁忌搜索中采用了一种灵活的“记忆”技术，对已经进行的优化过程进行记录和选择，指导下一步的搜索方向；搜素的新解不是在当前解的邻域空间中随机产生的，而是优于当前最好的解，或是非禁忌的最佳解，这样选取的优良解的概率远远增大。

本文中将用到两种不同的初始化方法。第一种初始化方法是 FIFO（先进先出）调度算法，在本文中所需解决的问题的背景下，初始化对需要分配的航班采取先到先服务的原则，为每个航班分配合理的停靠位置，按照登机口的业务类型等约束条件优先为每个航班安排最匹配的登机口，以供航班停靠。

对比贪婪算法在对问题求解时，总是做出在当前看来是最好的选择，我们应用的这种初始化方法得到的解也是最适合当前情况的最好选择，可以看出我们所采取的 FIFO 算法在一定意义上包含了贪婪算法的思想，但是也同贪婪算法一样，得到的仅是某种意义上的局部最优解。

此外，我们发现，对于不同的启发式算法，在每一次迭代中给出的解空间一般都是不同的，其中，一些算法给出的解空间中的解普遍与迭代前的原始解较为接近，不过也有其他算法会给出基于上一步迭代的新计算的解空间。例如，相比贪婪算法、禁忌算法，蚁群算法、遗传算法得到的新的解空间往往与原始解体现更高的相似程度。

启发式算法给出的解集对于模型的选择来说十分重要。例如，在本论文中的第一个问题，由于飞机交换后会可以得到一个基于当前机位给定的解集，因此，“先进先出”的初始化方案在第一个问题中是一个很好的选择，然而对于本论文中第三个问题，如果模型的初始化，以及以当前解作为启发求解时，如果仍然使用“先进先出”的原则，会在一定程度上删除先前迭代的求解结果。因此，在第一、第二问的模型无法给出合适的解空间的情况下，我们需要寻找其他的建模方法，更好地让每次递归都可以尽可能保留自身的计算结果，并不断迫近全局最优解。

3.4 本章小结

本章主要是对近些年来各类可用于航班分配问题的算法进行一个概括与总结，尤其是在第四小节对本次需要使用的禁忌搜索算法进行了详细的分析，包括禁忌算法的基本概念、实现特点、结构特点等，说明禁忌搜素算法在航班分配问题上的具有很多优点。接着，简单分析了不同的数学模型在不同问题中的选择问题，为问题一二三提供了理论算法基础。

四、问题一

4.1 问题分析

由于现今旅行业的迅速发展，某家航空公司在某机场的现有航站楼的旅客流量已经达到饱和状态，为了应对未来更加长远的发展，现正在增设卫星厅。但是引入卫星厅之后，虽然可以在一定程度上缓解原有的航站楼登机口供应不足的现状，却有可能对航班的规划分配带来影响，尤其可能会对中转旅客造成航班衔接问题，为航空公司带来负面影响和较大的经济损失。与新增设的卫星厅相比，航站楼具有完整的国际机场航站功能，包括出发、到达、出入境和候机，而卫星厅可以候机，但没有出入境的功能。同时，受机型（宽体机/窄体机）与等属性与登机口是否匹配以及分配在同一登机口的两飞机之间的空档间隔时间不得少于 45 分钟的制约，为了分析新建卫星厅对航班的影响，问题一只考虑航班 - 登机口的分配问题。该问题中，航站楼有 28 个登机口，卫星厅有 41 个登机口。在不考虑中转旅客的换乘信息的情况下，尽可能多地分配航班到适合的登机口，并且在此基础上使得所使用的登机口数量最小化。我们需要对航班登机口进行合理的安排和调整，得到航班登机口重新分配的最优结果。

问题一要求得到航班登机口重新分配的最佳结果，能够尽可能多的分配航班到合适的登机口且在此基础上最小化登机口数量。该问题属于较大规模的 NPhard 问题，该问题的约束条件包括：航站楼与卫星厅一共可用的登机口 69 个，以及各个登机口是否有出入境功能和宽机体/窄机体是否匹配等属性问题；每架飞机转场的到达和出发两个航班必须安排在同一个登机口；同一登机口的两飞机之间的空档间隔时间要大于等于 45 分钟，同时，分配不到固定登机口的飞机可以停靠在临时停机位，这里假设临时停机位数量无限制。为了尽可能多的分配航班到合适的登机口，应该使得停靠到临时停机位的飞机数量尽可能少，则应该使得使用的临时停靠机位数量尽量小，即目标函数可以转化为在上述各个约束条件下使临时停靠机位数最小化。

4.2 模型建立

登机口重新分配问题属于较大规模的 NP-Hard 问题，在本题目的背景下，有多达 1400 余个航班需要在考虑范围内，在这种规模下，该模型通过精确算法很难在可以接受的计算时间和存储容量的代价下求出满意的解，所以此类问题

需要采用禁忌算法、遗传算法等现代优化算法进行近似求解。所以，一个高效可行的求解算法是解决本模型求解问题的关键。本文中选择利用禁忌算法来实现登机口重新分配的优化方案。

在求解航班-登机口分配问题之前，首先回顾一下分配问题的描述：N 个航班被分配到 M 个登机口上。一个登机口在任何时候最多只能为一个航班提供服务。一个航班不能同时在两个登机口接受服务。同时，在同一个登机口的前后两个航班空档间隔时间必须要大于 45 分钟。如何在上述条件的前提下尽可能多的分配航班到合适的登机口并在此基础上最小化被使用登机口的数量？

本次比赛提供的飞机转场计划和中转旅客信息表包含 2018 年 1 月 19、20、21 三天，根据要求，只考虑 20 日到达或 20 日出发的航班和旅客。因此，将表格中不合要求的数据先删除。此外，表格中存在一些不合逻辑的数据，此类数据不作考虑。

4.2.1 目标函数

本题的目标主要是尽可能多的将航班分配到合适的登机口，在此基础上最小化被使用登机口的数量。因此，模型的目标是最小化停机坪的数量。

机场登机口的数量共有 M 个，临时停机位视为第 M+1 个登机口，若航班 i 被分配到临时停机位上，则 $y_{i,M+1}$=1。

建立如下目标函数：

$$\min Z_1 = \sum_{i=1}^{N} y_{i,M+1}$$

4.2.2 约束条件

由题设，将已知条件作为目标函数的约束条件引入优化模型中：

(1) 飞机最小间隔约束：每架飞机迁移航班到达时间与后一航班起飞时间的最小间隔时间为 45 分钟，故有如下约束：

$$a_{jk} \geq d_{ik} + \gamma, i, j = 1, 2, ..., \mathrm{N}; \mathrm{k} = 1, 2, ..., \mathrm{M}$$

(2) 唯一性约束：每个航班必须且仅能被分配到一个登机口。

$$\sum_{k=1}^{M} y_{ik} = 1, i = 1, 2, ..., \mathrm{N}$$

(3) 独占约束：同一登机口在任何时刻最多只能停一架飞机，不能分配给一个以上的航班，即对于登机口 k，此时使用该登机口的航班其前后最多只能有一个航班：

$$y_{ik} \geq \sum_{j=1}^{N} z_{ijk}, i = 1,2,...,\mathrm{N}; \mathrm{k} = 1,2,...,\mathrm{M}$$

$$y_{jk} \geq \sum_{i=1}^{N} z_{ijk}, j = 1,2,...,\mathrm{N}; \mathrm{k} = 1,2,...,\mathrm{M}$$

(4) 匹配约束：宽体机只能停放在大型登机口，窄体机只能停放在小型登机口：

$$\mathrm{S}_k = F_i, i = 1,2,...,N; k = 1,2,...,M$$

(5) 登机口航班类型约束：每个航班的到达类型和出发类型不同，因此在分配登机口时需要考虑是否与登机口类型业务相匹配：

$$\left(X_1(i) \subset X_1(k)\right) \cap \left(X_2(i) \subset X_2(k)\right),$$

$$i = 1,2,...,\mathrm{N}; \mathrm{k} = 1,2,...,\mathrm{M}$$

综合上述公式，对问题一我们可以建立如下的航班-登机口分配优化模型：

$$\min Z_1 = \sum_{i=1}^{N} y_{i,M+1},$$

$$\text{s.t.} \begin{cases} a_{jk} \geq d_{ik} + \gamma, \quad i, j = 1,2,...,\mathrm{N}; \mathrm{k} = 1,2,...,\mathrm{M} \\ \sum_{k=1}^{M} y_{ik} = 1, i = 1,2,...,\mathrm{N} \\ y_{ik} \geq \sum_{j=1}^{N} z_{ijk}, i = 1,2,...,\mathrm{N}; \mathrm{k} = 1,2,...,\mathrm{M} \\ y_{jk} \geq \sum_{i=1}^{N} z_{ijk}, j = 1,2,...,\mathrm{N}; \mathrm{k} = 1,2,...,\mathrm{M} \\ \mathrm{S}_k = F_i, i = 1,2,...,N; k = 1,2,...,M \\ \left(X_1(i) \subset X_1(k)\right) \cap \left(X_2(i) \subset X_2(k)\right) \end{cases}$$

4.3 速度最优原则

在实现每次迭代解集的运算时，为了能够尽可能多地将不容易成为最优解的备选解排除出每次迭代生成的解集，我们可以规定可以成为最优解的备选解

出自停机坪前几位的航班与登机口航班互换所生成的解集。这样的假设是合理的，一方面因为位于停机坪靠前位置的航班的到达时间相较停机坪靠后的航班靠前，因此很大程度上可以保证求出的解更优，另一方面，由于航班调整对航班安排的影响只体现在比换进登机口的航班到达时间更靠后的航班上，因此将停机坪的第一个航班优化后，对后续停机坪航班的优化对这个航班无影响。

本题算法步骤如下：

步骤 1：初始解的产生方法

本模型采用 FIFO 先进先出调度算法得到初始分配方案。由于题目中登机口由三种情况组成：*DD*(国内到达、国内出发)，*DI* (国内到达、国际出发)、*ID*(国际到达、国内出发)。通用的登机口以上三种情况都可以进行，但是有一些特定登机口只能办理某一项或某两项情况(例如：某个登机口只能办理国内到达国内出发而无法办理国内到达国际出发)。因此，我们在初始分配登机口航班时,优先将这些特定登机口进行航班分配，从而保证通用登机口利用率更大。

步骤 2：根据禁忌搜索算法更新当前航班-登机口分配情况

为了保证整个算法速度，规定用临时停机位上的第一个没被分配过登机口的航班依次替换已分配好的登机口的航班，再按照 FIFO 原则产生新的分配方案，从所有方案中选取停放在临时停机位上的航班数量最少的方案，接着，将此时未分配到登机口的第二个航班依此方法代入，产生优化方案，依此类推，直至临时停机位上没有未分配过的航班。

4.4 业务最优原则

由于上述算法在寻求解空间的过程中采用了较为“贪婪”的策略，也即仅将位于停机坪上第一位的航班依次与固定登机口上的航班交换位置，用以加速运算，因此若对于每一步迭代，最能够接近全局最优解的解没有出现在替换第一个航班所生成的解集中，就有一定概率接近不了全局最优解。

假定分配到临时停机位的航班为 P，按照时间先后顺序排序，选取前 P/2 个航班依次替换已分配好的登机口的航班，从所有方案中选取停放在临时停机位上的航班数量最少的方案。假设此时方案临时停机位的航班数量为 Q(P>Q)，继续选取 Q/2 个航班进行代入，产生优化方案，依次类推，直到临时停机位航班数量不再变小。

4.5 平衡原则

在上述的两个原则下，我们的优化目标都只考虑尽可能多的分配航班到合适的登机口，在此小节中，我们加入登机口使用数量平衡原则，在尽可能多的分配航班到合适登机口的基础上使得所使用的登机口数量最少。此时，我们的目标函数形式调整为:

$$\min Z_1 = \sum_{i=1}^{N} y_{i,M+1} + \alpha(M - \sum_{k} e_k)$$

此时加入了登机口数量平衡项，需要在尽可能多的分配航班到合适登机口的基础上使得使用的登机口数量尽可能少，之前的初始化方法就不再适用，因为它要优先选择最匹配航班业务的登机口，一定程度上增大了要占用未使用登机口的几率。所以我们使用了一种更加严谨的初始化方法，即在为每个航班分配登机口时，仍然遵循先到先服务的原则，只是搜索合适的登机口时遵循的优先级条件发生了变化：在搜素登机口时只要登机口的提供的业务包含航班所需的业务，并且也满足其他的约束条件时，就可以将该航班分配到搜所到的登机口，不需要遵循第一种初始化方式中的分配登机口的优先级顺序。例如，某个航班所需业务类型为（D,D）,那么为其安排登机口时搜索提供业务类型为（D,D），（D/I,D）或（D,D/I）以及（D/I,D/I）的满足各项约束条件的登机口，只要可以搜索到这样的合适的登机口，就可以将该航班安排到合适登机口；如果经过上述搜索过程之后找不到合适的登机口停放该航班，则将该航班分配到临时停机位。

这样的做法可以增大航班被安排到合适登机口的几率，也可以增大已经使用的登机口的使用率，减少对未使用登机口的占用的几率，达到在尽量多安排航班到指定登机口的基础上使得使用登机口数量尽可能小。

因此，按照上述的更加严谨的初始化条件进行初始化，得到能使航班能安排到合适登机口的几率增大并且可以使占用未使用登机口几率减小的解作为初始解。也即此时我们考虑将所有可能的空闲登机口都用来安置新到来的航班，因此在每一步迭代中，解集的大小被大大增加了。

平衡原则没有一个确定的最好的结果，其具体取值根据 alpha 加权项决定。由下图(2)可以看出，随着 alpha 加权项变大，“被使用登机口的数量需要最小”这个条件的影响因素变大，登机口数量越来越少，停机坪数量越来越多。

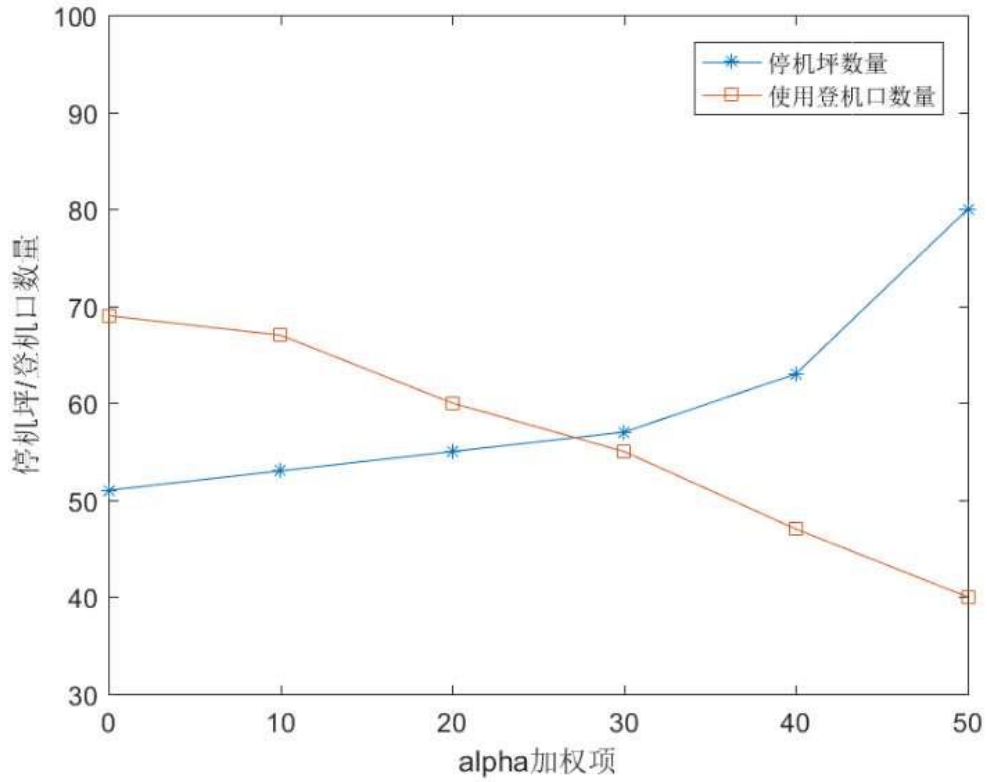


图 2 平衡原则结果示意图

4.6 结果与分析

本章主要对问题一的相关内容进行了建模和求解，并在求解模型的过程中提出了两个不同的原则：速度最优原则和业务最优原则，分别侧重于保证算法运行速度和保证算法运行取得的最终效果。还加入了使用登机口数量平衡的原则，在尽量多安排航班到合适登机口的前提下，使得使用的登机口数量尽量少。

本题的背景只考虑 20 日达到和出发的航班共 304 次航班，且要尽可能多的分配航班到合适的登机口。速度最优原则下，得到的实验结果为使用临时停机位 53 个，成功分配航班数 251 个，使用登机口数量 69 个；业务最优原则下，得到的实验结果为使用临时停机位 51 个，成功分配航班数 253 个，使用登机口数量 69 个。

根据速度最优原则进行算法求解，结果如下表(1)所示：

临时停机位	被使用登机口数量	成功分配航班数量
53	69	251

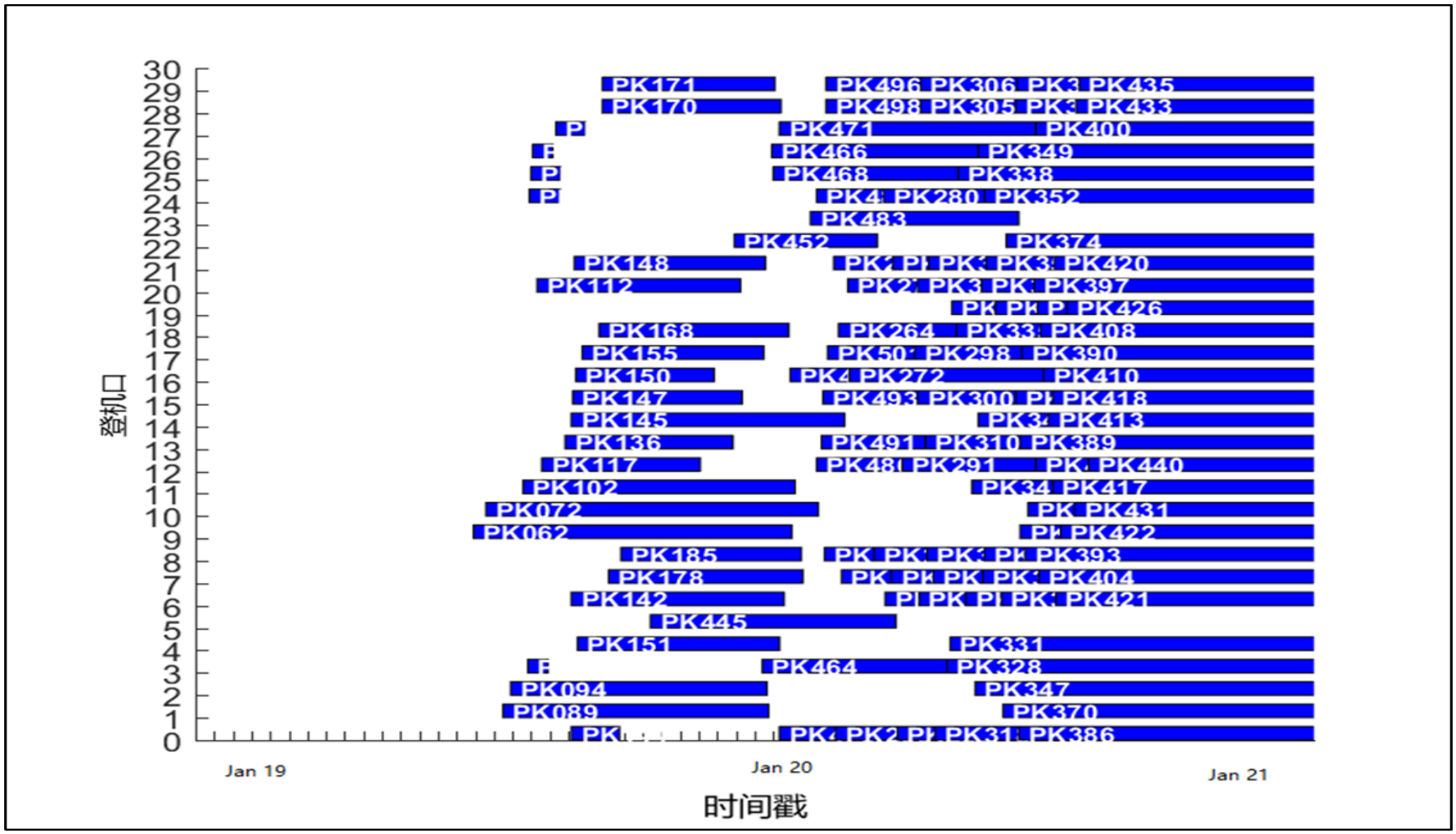
表(1)：速度最优原则结果表

根据业务最优原则进行算法求解，结果如下表(2)所示：

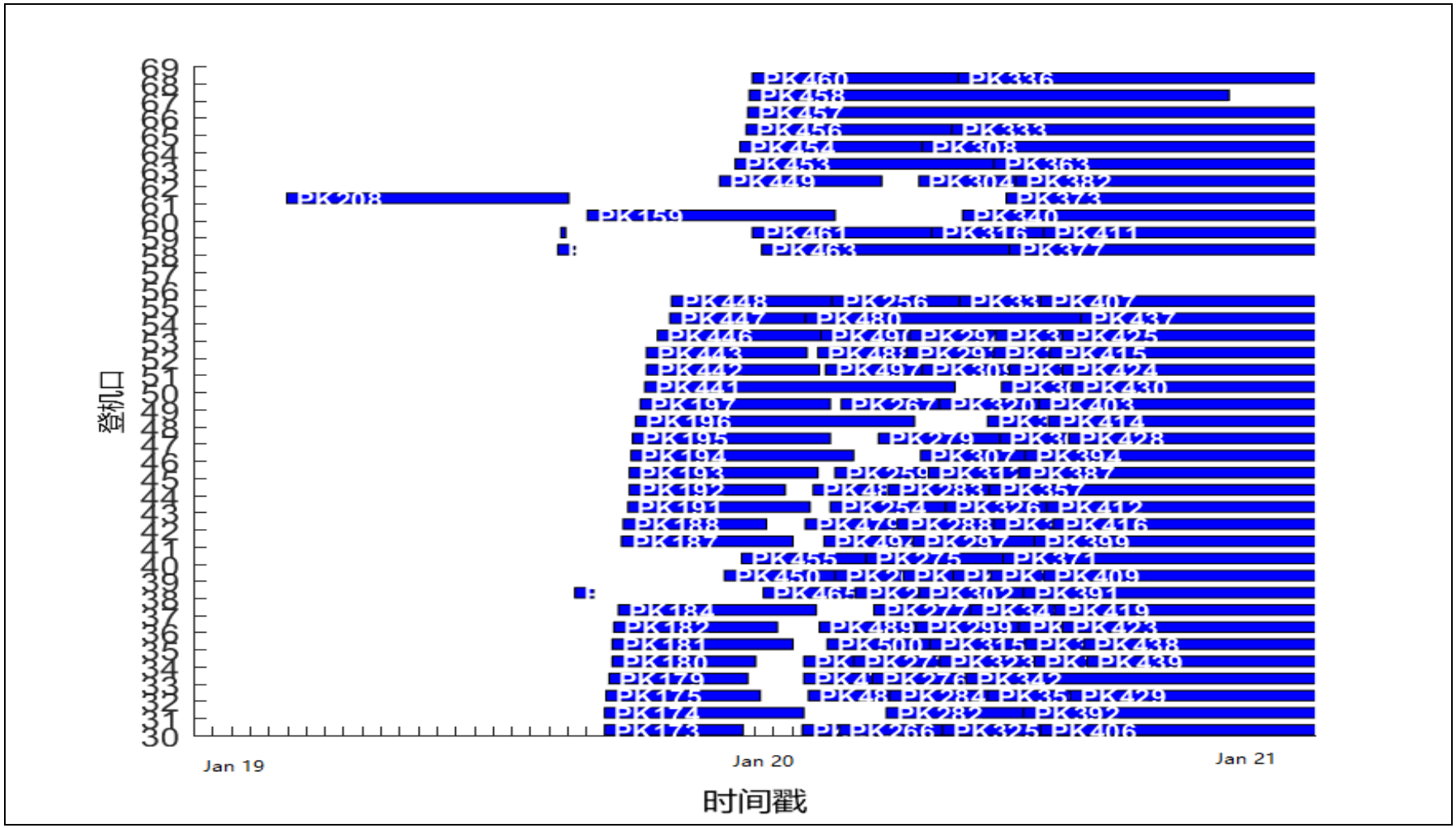
临时停机位	被使用登机口数量	成功分配航班数量
51	69	253

表(2)：业务最优原则结果表

根据业务最优原则，我们得到了各个登机口航班分配情况的甘特图，如图(a)(b)所示：



图(a)


登机口
Jan 19
Jan 20
Jan 21
时间戳

a) 给出成功分配到登机口的航班数量和比例，按宽、窄体机分别画线状图

根据速度最优原则，我们按照宽窄机型号的区别画出了成功分配到登机口的航班数量与比例，如下图(3)所示：横坐标代表飞机机型，左边为窄体机，右边为宽体机，纵坐标表示分配数量。蓝色代表分配成功的数量，红色代表分配失败的数量。由于题目要求分析 2018 年 1 月 20 日到达或出发的航班，因此根据给出的数据得知，1 月 20 日共有 304 个航班，其中窄体机共有 246 个，宽体机共有 58 个。根据算法，得出结果：成功分配的窄体机共有 195 个，成功率约为 79.27%，成功分配的宽体机共有 56 个，成功率约为 96.55%。

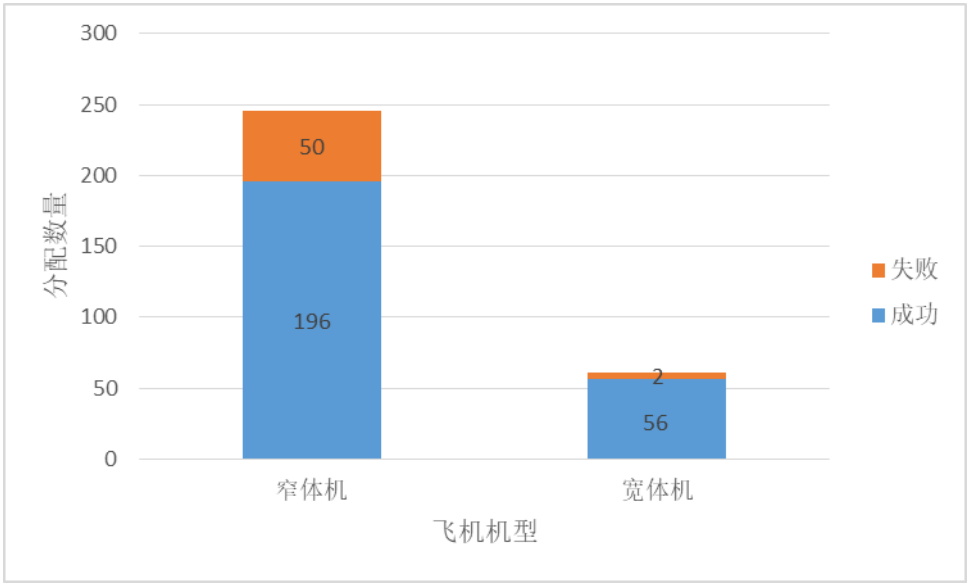


图 3

根据业务最优原则，按照宽窄机型号的区别画出了成功分配到登机口的航班数量与比例，如下图(4)所示：成功分配的窄体机共有 196 个，成功率约为 79.67%，成功分配的宽体机共有 57 个，成功率约为 98.27%。

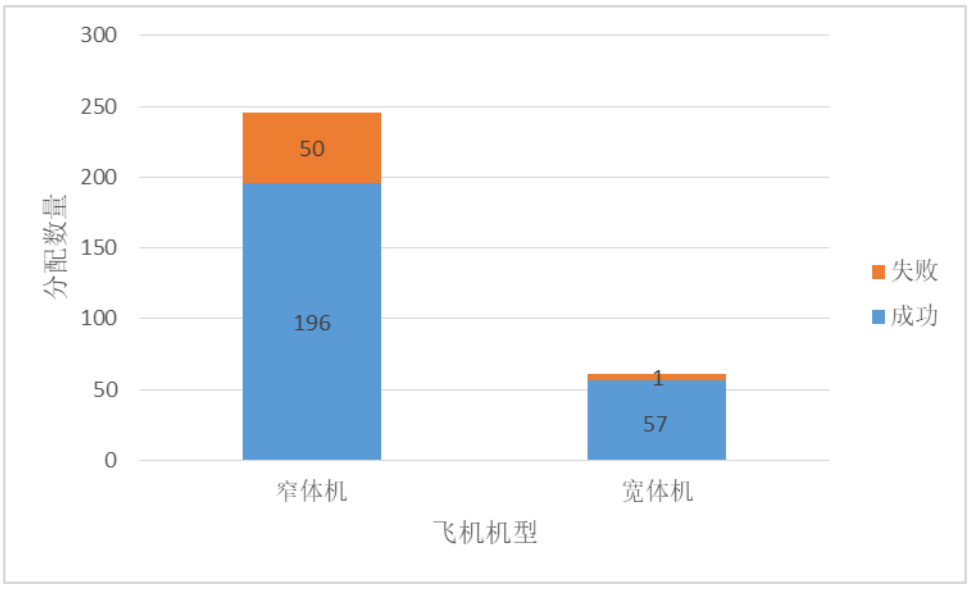


图 4

b) 给出 T 和 S 登机口的使用数目和被使用登机口的平均使用率（登机口占用时间比率），要求画线状图。本题是根据业务最优原则绘图，下图(5)为登机口使用次数统计，下图(6)为登机口占用时间比率。

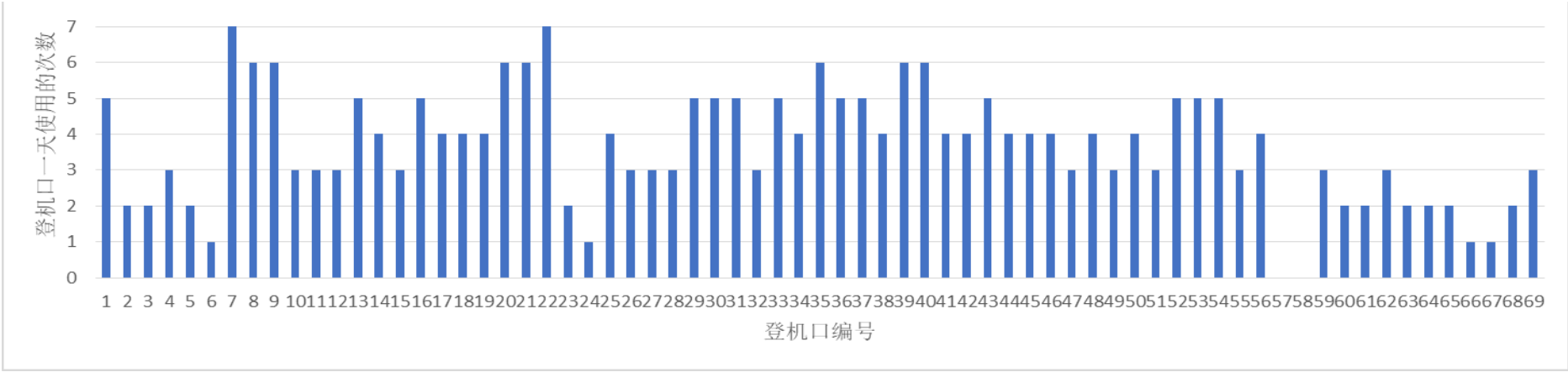


图 5 登机口使用次数统计

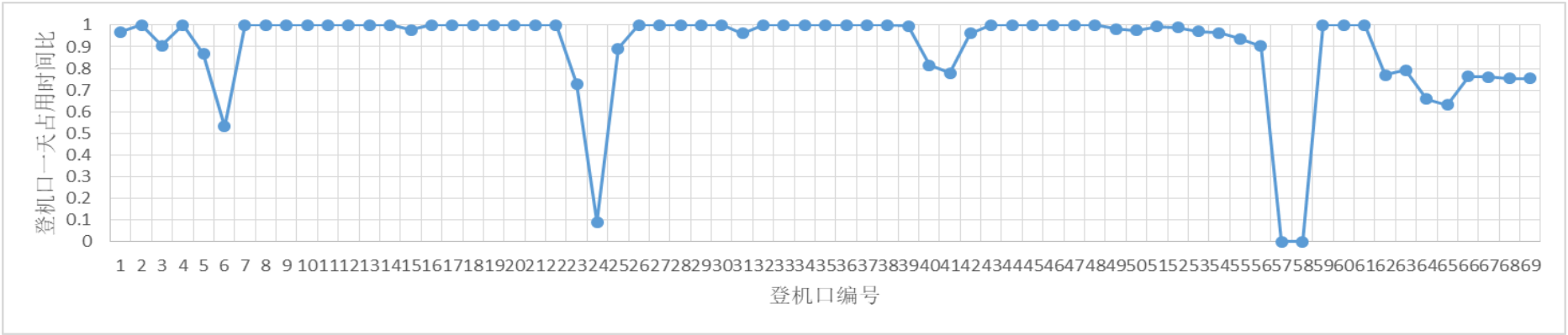


图 6 登机口占用时间

五、问题二

5.1 问题分析

问题二在问题一的基础上，在重新分配登机口时考虑到中转旅客换乘的因素，为了使旅客因为中转所花费的时间总代价尽可能最小，来确保中转旅客后续的行程被错过的风险降低，我们需要使中转旅客的总体流程时间最小化，并且在此基础上最小化被使用的登机口数量。由于新建卫星厅没有办理出入境业务的功能，所以需要中转的旅客从前一航班的达到到后一航班的出发之间的流程，根据行程的起止点是国内（D）还是国际（I），以及所在登机口位于航站楼(T)还是卫星厅（S），可以组合成 16 种不同的情况，不同情况的最短流程时间和捷运乘坐次数由下表(3)给出：

出发 / 到达		国内出发（D）		国际出发（I）	
		航站楼 T	卫星厅 S	航站楼 T	卫星厅 S
国内到达（D）	航站楼 T	15/0	20/1	35/0	40/1
	卫星厅 S	20/1	15/0	40/1	35/0
国际到达（I）	航站楼 T	35/0	40/1	20/0	30/1
	卫星厅 S	40/1	45/2	30/1	20/0

表(3) 最短流程时间及捷运乘坐次数

根据上表中给出的数据以及处于不同情况下的乘客的数量，我们可以求和得到中转旅客的总体流程时间。

本题目的目标是要找到使得中转旅客的总体流程时间最小的登机口分配方案，并且在此基础上尽可能将使用的登机口数量最小化。我们在重新分配登机口时仍然可以延续第一题中使用的禁忌算法的思想来优化搜索最优解，不同的

是不仅要考虑第一题中包含的各个约束条件，还要加上使中转旅客总体流程时间最小的约束，并且即使得到一种分配方案使得使用登机口数量比较少，但是如果该种方案使得中转旅客总体流程时间变大，我们也会舍弃这种方案，也就是说，中转旅客总体流程时间最小化有更大的权重。我们在进行最优解的搜索时在约束条件项中可以将中转旅客总体流程时间最小化的约束项也整合进去，在新的约束条件下再用禁忌算法进行最优解搜索，得到符合条件的最佳分配方案。

5.2 模型建立

5.2.1 目标函数

目标函数建立：

本题的目标主要是在问题一的基础上加入旅客换乘因素，考虑中转旅客的最短流程时间，并在此基础上最小化被使用登机口的数量。因此，本问题的模型目标还是最小化临时停机位的数量，但是需在约束条件中增加中转旅客总体最短流程时间最小化这一项。可建立如下目标函数：

$$\min Z_1 = \sum_{i=1}^{N} y_{i,M+1}$$

5.2.2 约束条件

问题二的约束条件包括飞机最小间隔约束、唯一性约束、独占约束、匹配约束、登机口航班类型约束，上述五项约束在问题一中已表明，在此不再赘述。其他新增约束条件如下：

(1) 出入境业务约束：航站楼可以办出入境业务，卫星厅无法办理出入境业务:

$$\mathrm{A}=TE\left(k\right)，\ \mathrm{k}=1,2,...,\mathrm{M}$$

(2) 最短流程时间约束$t_{\mathrm{i,m}}$

$$t_{i,m}\begin{cases} 0, \ if \quad y_{i,M+1}=1 \\ 15, \ if \quad D_T D_T \ \ or \ \ D_S D_S \\ 20, \ if \quad D_T D_S \ \ or \ \ D_S D_T \ \ or \ \ I_T I_T \ \ or \ \ I_S I_S \\ 30, \ if \quad I_S I_T \ \ or \ \ I_T I_S \\ 35, \ if \quad D_T I_T \ \ or \ \ D_S I_S \ \ or \ \ I_T D_T \\ 40, \ if \quad D_S I_T \ \ or \ \ D_T I_S \ \ or \ \ I_S D_T \ \ or \ \ I_T D_S \\ 45, \ if \quad I_S D_S \end{cases}$$

$$if \quad F_{ail}=1, \ t_{im}=360$$

(3) 中转旅客总的最短流程时间约束

$$\bar{T}=\min\sum_{i=1}^{N}\sum_{m=1}^{m_i}R_{\mathrm{i,m}}t_{\mathrm{i,m}},$$

综合上述公式，对问题一我们可以建立如下的航班-登机口分配优化模型：

$$\min Z_1=\sum_{i=1}^{N}y_{i,M+1}$$

$$\text{s.t.}\begin{cases} a_{jk}\geq d_{ik}+\gamma, \ \ i,j=1,2,\ldots,\mathrm{N};\mathrm{k}=1,2,\ldots,\mathrm{M} \\ \sum_{k=1}^{M}y_{ik}=1, i=1,2,\ldots,\mathrm{N} \\ y_{ik}\geq\sum_{j=1}^{N}z_{ijk}, i=1,2,\ldots,\mathrm{N};\mathrm{k}=1,2,\ldots,\mathrm{M} \\ y_{jk}\geq\sum_{i=1}^{N}z_{ijk}, j=1,2,\ldots,\mathrm{N};\mathrm{k}=1,2,\ldots,\mathrm{M} \\ \mathrm{S}_k=F_j, i=1,2,\ldots,N;k=1,2,\ldots,M \\ \left(X_1(i)\subset X_1(k)\right)\cap\left(X_2(i)\subset X_2(k)\right) \\ \bar{T}=\sum_{i=1}^{N}\sum_{m=1}^{m_i}R_{\mathrm{i,m}}t_{\mathrm{i,m}}, \\ t_{i,m}\begin{cases} 0, \ if \quad y_{i,M+1}=1 \\ 15, \ if \quad D_T D_T \ \ or \ \ D_S D_S \\ 20, \ if \quad D_T D_S \ \ or \ \ D_S D_T \ \ or \ \ I_T I_T \ \ or \ \ I_S I_S \\ 30, \ if \quad I_S I_T \ \ or \ \ I_T I_S \\ 35, \ if \quad D_T I_T \ \ or \ \ D_S I_S \ \ or \ \ I_T D_T \\ 40, \ if \quad D_S I_T \ \ or \ \ D_T I_S \ \ or \ \ I_S D_T \ \ or \ \ I_T D_S \\ 45, \ if \quad I_S D_S \end{cases} \\ if \quad F_{ail}=1, \ t_{im}=360 \end{cases}$$

5.3 模型求解

基于第一问的求解方法，我们仍然使用禁忌算法来优化搜索最佳的登机口分配方案，不同的是，在优化目标函数时增加了要使得中转旅客总体流程时间最小化这一约束条件，也就是说，当通过算法得到多个相同的优于当前结果的候选解时，还要计算出每个不同的解对应的情况下中转旅客的总体流程时间进行比较，，每次迭代最终选取的解对应的中转旅客总体流程时间要是最小的。具体的求解步骤如下：

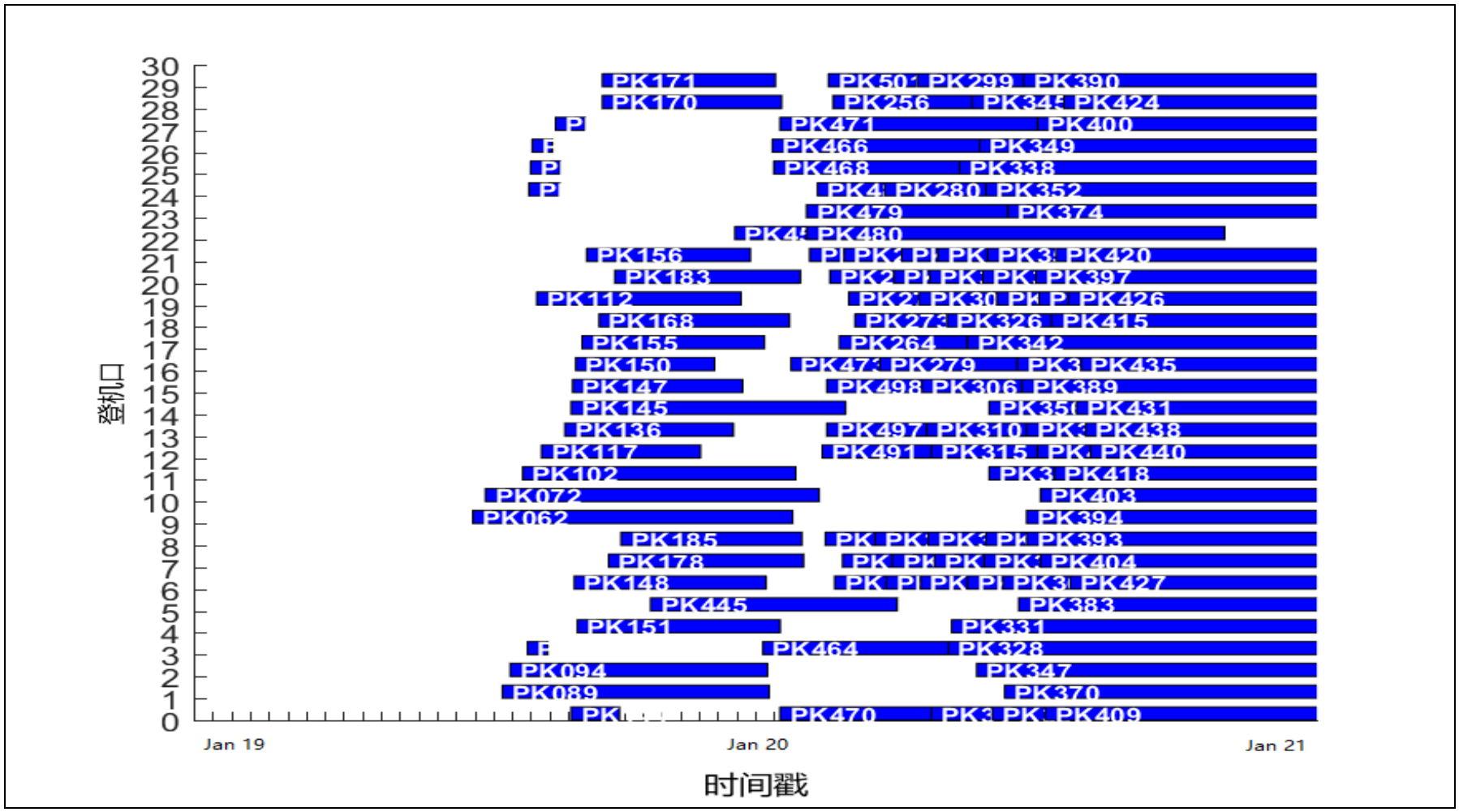
步骤 1：初始解的产生。仍然采用 FIFO 先进先出调度算法得到初始分配方案，在初始分配登机口航班时,优先将有特定登机口匹配条件的航班分配到最匹配的登机口，以此来保证通用功能更多的登机口利用率更大。

步骤 2：根据禁忌算法搜索优解，来更新当前的登机口分配情况。本小题采取的是问题一中的速度最优原则，用临时停机位上的第一个没被分配过登机口的航班依次替换已分配好的登机口的航班，再按照 FIFO 原则产生新的分配方案，从所有方案中选取停放在临时停机位上的航班数量最少的方案；

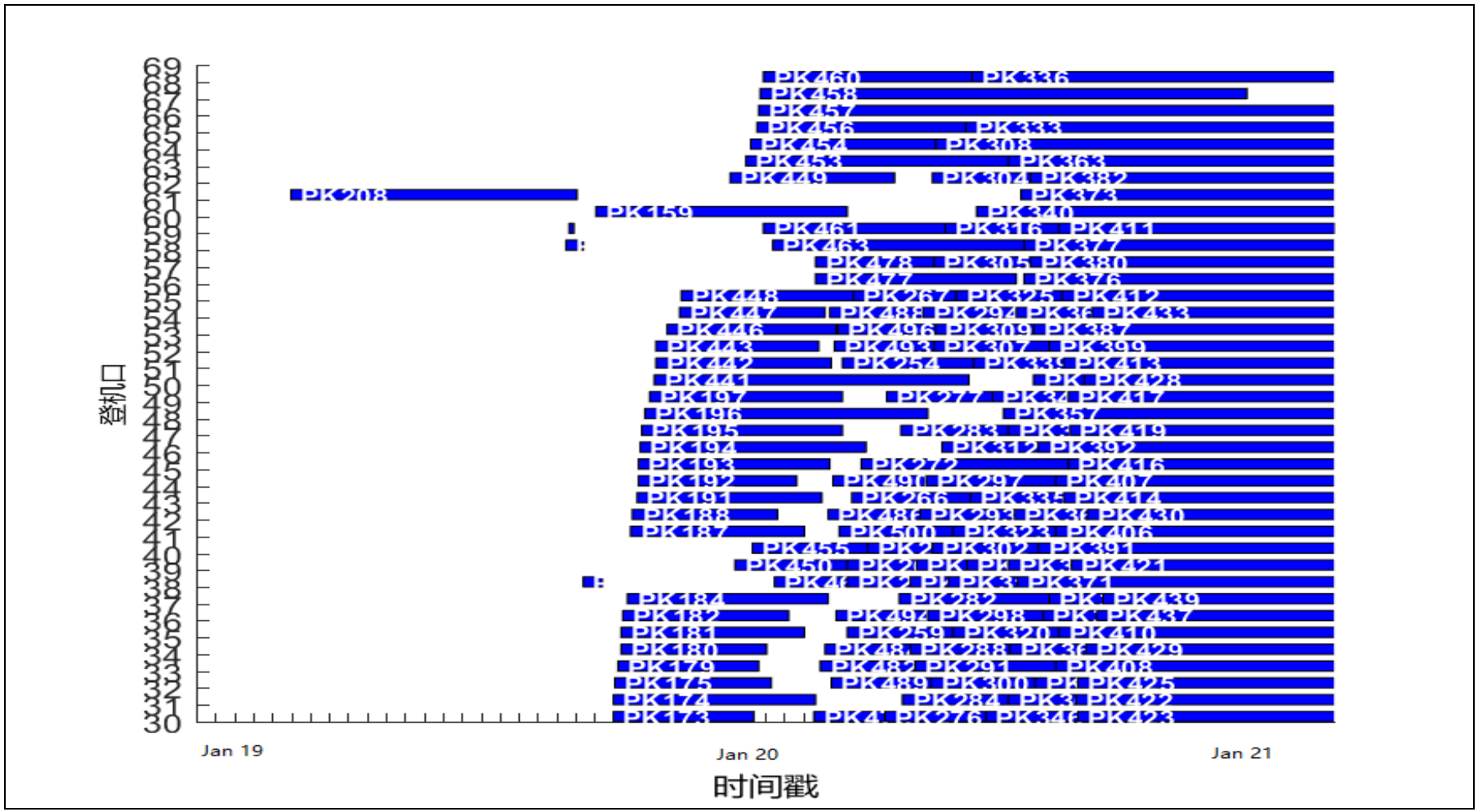
步骤 3：对产生的候选方案分别进行中转旅客总体流程时间的计算，并且比较选出使得中转旅客总体流程时间最小的方案；

步骤 4：重复步骤 2 和步骤 3 的过程，直到目标函数收敛，即临时停机位的数量不再减少。

根据求解情况，我们得到了各个登机口航班分配情况的甘特图，如图(c)(d)所示：


登机口
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Jan 19
Jan 20
Jan 21
时间戳

图(c)



图(d)

5.4 结果与分析

本章主要对问题二的相关内容进行了建模和求解，在第一问的基础上又加入了中转旅客总体流程时间最小化的约束条件，依然使用禁忌算法进行优化求解，最终得到的实验结果为使用临时停机位 51 个，成功分配航班数 253 个，使用登机口数量为 69 个，中转旅客总体流程时间最小为 145435 分钟，换乘失败旅客数量为 0 个，其它详细实验结果见第八章实验结果分析。求解结果如下表(2)所示：

临时停机位	**51**
被使用登机口数量	69
成功分配航班数量	253
中转旅客总的最短流程时间	145435

表(4) 问题二结果表

a) 给出成功分配到登机口的航班数量和比例，按宽、窄体机分别画线状图

按照宽窄机型号的区别画出了成功分配到登机口的航班数量与比例，如下图(7)所示：成功分配的窄体机共有 196 个，成功率约为 79.67%，成功分配的宽体机共有 57 个，成功率约为 98.27%。

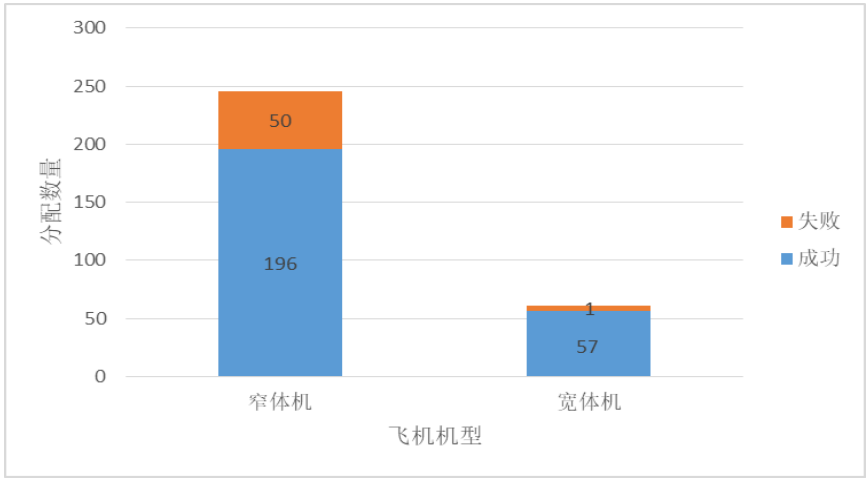


图 7 分配航班图

b) 给出 T 和 S 登机口的使用数目和被使用登机口的平均使用率（登机口占用时间比率），下图(8)为登机口使用次数统计，下图(9)为登机口占用时间比率。

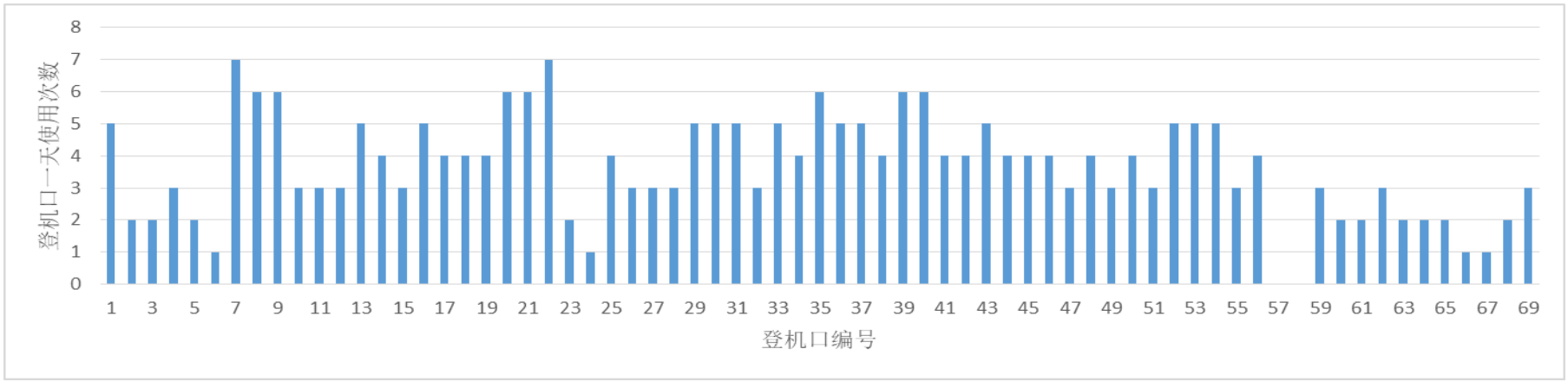


图 8 登机口使用次数统计

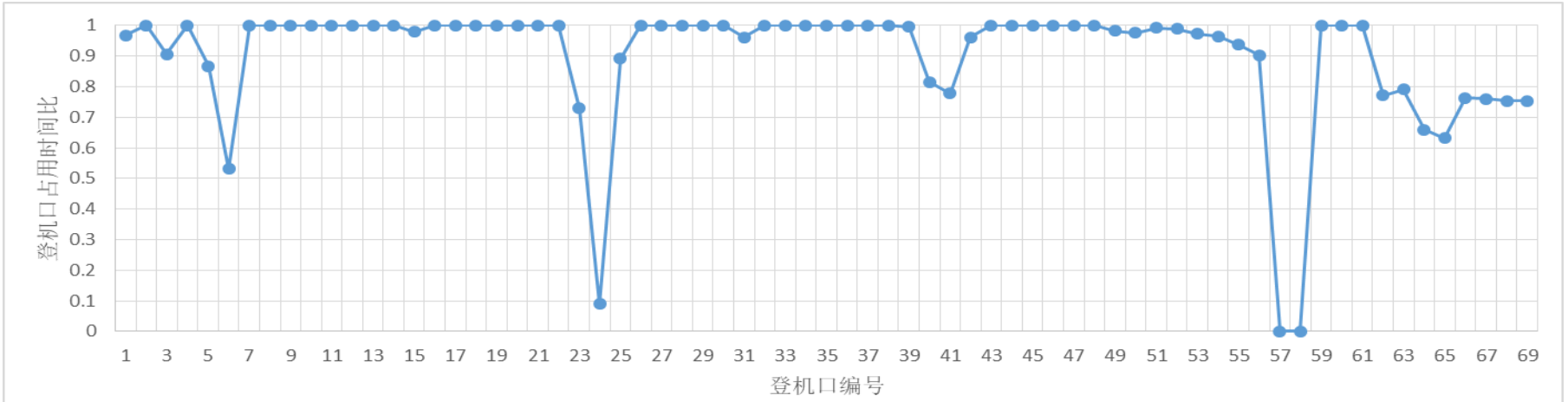


图 9 登机口占用时间比率

c) 换乘失败旅客数量为 0，换乘失败率为 0。

d) 旅客换乘时间分布图，本题平均旅客换乘时间为 33 分钟。

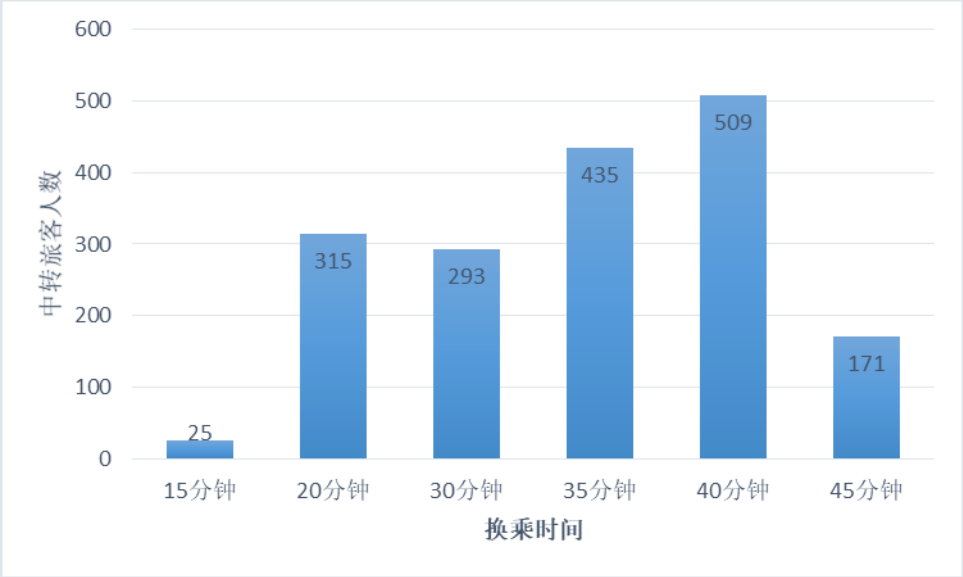


图 10 旅客换乘时间分布

六、问题三

6.1 问题分析

本问题在问题二的基础上进行细化，考虑中转旅客的换乘时间的因素。因为新建卫星厅对航班的最大影响就是可能造成中转旅客换乘时间的延长。这里引入了一个换乘紧张度的概念，定义如下：

$$
\begin{aligned}
\text{换乘紧张度} &= \frac{\text{旅客换乘时间}}{\text{航班连接时间}} \\
\text{旅客换乘时间} &= \text{最短流程时间} + \text{捷运时间} + \text{行走时间} \\
\text{航班连接时间} &= \text{后一航班出发时间} - \text{前一航班到达时间}
\end{aligned}
$$

图 11 概念定义

因此，最终需要考虑换乘旅客总体紧张度的最小化，并且在此基础上最小化被使用登机口的数量。其中，各种情况下行走时间可由下列表格查找（单位：分钟）

登机口区域	T-North	T-Center	T-South	S-North	S-Center	S-South	S-East
T-North	10	15	20	25	20	25	25
T-Center		10	15	20	15	20	20
T-South			10	25	20	25	25
S-North				10	15	20	20
S-Center					10	15	15
S-South						10	20
S-East							10

表(5)

捷运单程一次需要 8 分钟，可以根据表(6)来获得各种情况下的捷运搭乘次数进而求得捷运时间。为了后续计算方便，我们将捷运时间整理为下表的形式（单位：分钟）：其中登机口位置表示到达/出发航班所在的登机口位于航站楼 T 还是卫星厅 S，航班类型表示到达/出发的航班是国内航班（D）还是国际航班（I）。例如 S/T 表示到达航班在卫星厅 S，出发航班登机口在航站楼 T，D/I 表示到达航班是国内航班 D，出发航班是国际航班 I。

航班类型 登机口位置	D/D	D/I	I/D	I/I
S/S	0	16	16	0
S/T	8	8	8	8
T/S	8	8	8	8
T/T	0	8	8	0

表(6)

再将最短流程时间、捷运时间以及行走时间相加得到旅客换乘时间；再通过换乘旅客后一航班出发时间减去前一航班到达时间得到航班连接时间，然后根据换乘紧张度定义得到换乘紧张度的值，对每个旅客都进行上述的过程求出对应的换乘紧张度再求和，得到换乘旅客总体紧张度。

本题目的目标是找到登机口的一个最佳分配方案，在该方案下换乘旅客的总体紧张度最小化，并且将其作为目标函数中的首要因素。同时，在此基础上最小化所使用的登机口的数量。我们依然延用问题一、二的模型的思想来解决这个问题，在约束条件中要加入使得换乘旅客总体紧张度最小化的约束条件，即使得到更优的分配方案，但是如果该种方案使得换乘旅客总体紧张度变大，我们也会舍弃这种方案。在加入了换乘旅客总体紧张度最小化的约束条件下再用禁忌算法进行最优解搜索，得到符合条件的最佳分配方案。

6.2 模型建立

本题是第二问的细化，只是对目标函数的约束条件有一些改动，目标函数为：

$$\min Z_1 = \sum_{i=1}^{N} y_{i,M+1}$$

这里与第二问的其他约束条件相同，只是将第二问中中转旅客总体最短流程时间最小化这一约束项换成本题中要求的换乘旅客总体紧张度最小化，换乘旅客总体紧张度最小化约束项表示为：

$$T_t = \min \frac{\sum_{i=1}^{N}(\sum_{m=1}^{m_i} R_{i,m}(t_{i,m} + t_{i,m}^{jy}) + \sum_{n=1}^{n_i} rs_{i,n} t_{i,n}^{bx})}{\sum_{i=1}^{N}\sum_{l=1}^{l_i}(t_l^{cf} - t_l^{dd})}$$

$$t_{i,m}^{jy} = \begin{cases} 0, \text{ if } D_S D_S \text{ or } D_T D_T \text{ or } I_S I_S \text{ or } I_T I_T \\ 16, \text{ if } D_S I_S \text{ or } I_S D_S \\ 8, \text{ Others} \end{cases}$$

$t_{i,n}^{bx}$ 可以根据表 1 中的数据查询，这里由于取值情况复杂，不详细列出。

6.3 模型求解

基于前面两问的求解方法，我们依然使用禁忌算法来优化搜索最佳的登机口分配方案，不同的是，在优化目标函数时增加了要使得换乘旅客总体紧张度最小化这一约束条件，也就是说，当通过算法得到多个相同的优于当前结果的候选解时，还要计算出每个不同的解对应的情况下换乘旅客的总体紧张度进行比较，每次迭代最终选取的解对应的换乘旅客总体紧张度要是最小的。具体的求解步骤如下：

步骤 1：初始解的产生。仍然采用 FIFO 先进先出调度算法得到初始分配方案,在初始分配登机口航班时,优先将有特定登机口匹配条件的航班分配到最匹配的登机口，以此来保证通用功能更多的登机口利用率更大。

步骤 2：根据禁忌算法搜索优解，来更新当前的登机口分配情况。本小题采取的是问题一中的速度最优原则，用临时停机位上的第一个没被分配过登机口的航班依次替换已分配好的登机口的航班,再按照 FIFO 原则产生新的分配方案，从所有方案中选取停放在临时停机位上的航班数量最少的方案；

步骤 3：对产生的候选方案分别进行换乘旅客总体紧张度的计算，并且比较选出使得换乘旅客总体紧张度最小的方案；

步骤 4：重复步骤 2 和步骤 3 的过程，直到目标函数收敛，即临时停机位的数量不再减少。

6.4 改进方案

在本题中，我们延续用问题二的模型得到的性能并不是很好，主要因为禁忌算法的初始化会丢弃一部分前一轮迭代的结果，在这一问中，我们要尽可能找到与前一轮迭代结果相近的结果,但是基于 FIFO 的调度算法在迭代优化的过程中会引起解的较大的变化，与前一轮结果相似程度可能比较小。

通过查找阅读相关的文献，我们发现蚁群算法更适合这个问题的解答，可

能会获得更好的结果。

蚁群算法（ant colony algorithm，ACA）是一种比较新的模拟进化算法，该方法的提出模拟蚂蚁觅食时的最短路径原理：蚁群可以在不同的环境下，寻找最短到达食物源的路径。这是因为蚁群内的蚂蚁可以通过某种信息机制实现信息的传递。蚂蚁会在其经过的路径上释放一种可以称之为“信息素”的物质，蚁群内的蚂蚁对“信息素”具有感知能力，它们会沿着“信息素”浓度较高路径行走，而每只路过的蚂蚁都会在路上留下“信息素”，这就形成一种类似正反馈的机制，这样经过一段时间后，整个蚁群就会沿着最短路径到达食物源了。该方法用以求解 TSP 问题，取得了较好的实验效果。这种算法具有分布计算、信息正反馈和启发式搜索的特征，本质上是进化算法中的一种启发式全局优化算法。

用蚁群算法来求解登机口分配问题时，航班机位分配二元图 G(V,E)，航班编号为$V=\{v_1,v_2,...,v_n\}$，对于到达离开时间上不相冲突的航班，建立使得任意$(v_i,v_j)\in E$，以 $C=\{c_1,c_2,...,c_p\}$表示每个航班的 p 种不同可分配停机位的一个集合，$C(v_i)\neq C(v_j)$，D(n×n) 表示图 G(V,E) 的邻接矩阵，满足：

$$D(i,j)=\begin{cases}1，v_i 与 v_j 相关联\\ 0，\ v_i 与 v_j 不关联\end{cases}$$

S(n×p)表示分配矩阵，满足：

$$S(i,j)=\begin{cases}1,给\ v_i 着\ c_j 色\\ 0，不给\ v_i 着\ c_j 色\end{cases}$$

若着色蚂蚁 k 经过 S(i,j)，则表示给航班 v_i 分配停机位 c_j 。

算法的具体实施流程设计如下：

Step1：参数初始化。设置算法最大迭代次数 max_n ，以及当前迭代次数 n←0；设置每条弧的信息素 τ_{ij} ←c（c 为较小的正数）；设置 v_{i-1} 着色后使用的停机位数 Num ，所有航班进入停机位的滑行总距离为 D；启发式信息

$\eta_{ij} \leftarrow 1/(Num \cdot D)$ 将待着色图以关联矩阵形式存储，随机生成顶点序列 $V=\{v_1,v_2,...,v_n\}$；

Step2：蚂蚁位点初始化。对每只蚂蚁 k(k=1,2,...,q)，全部置于航班对应的 v1 点，在分配集 $C=\{c_1,c_2,...,c_p\}$ 中选择第 1 个停机位 c_1 分配给航班 v_1，并将 v_1 移到已分配航班集中；

Step3：蚂蚁自主构造路。对每只蚂蚁 k，当给航班 v_i 分配时，若其着色集不为空，则按概率 P_{ij}^k 给 v_i 分配 c_j，并将 v_i 移到已分配点集中，这里以“轮盘赌”的方式分配概率 P_{ij}^k；

Step4：记录本次迭代最优解。如果每只蚂蚁 k(k=1,2,...,q)都完成航班分配，比较并记录当次迭代的最优化解 $f_{best} \leftarrow \min(Z_1,Z_2,Z_3)$，否则，返回 Step3；

Step5：信息素的更新。对于每只蚂蚁构造路径上的边，更新信息素 $\tau_{ij} \leftarrow \rho \cdot \tau_{ij} + Q$（其中 1-ρ 为信息素挥发度，Q 为单位蚂蚁所留轨迹数量的常数）；更新当前迭代次数 n←n+1；

Step6：输出最优解。如果当前迭代次数 n≤max_n，转 Step2；否则，输出最后一次迭代所得的停机位分配方案，即为算法所得优化解。

6.5 本章小结

本章对问题三的相关问题进行阐述，在问题三的求解过程中延用第二问中用到的模型，只是将第二问中中转旅客总体最短流程时间最小化这一约束项换成本题中要求的换乘旅客总体紧张度最小化，在此过程中我们发现第二问的模型并不适用于解决第三问的问题，得到的效果并不是很好，所以额外介绍了一种可能更适合解决本问题的算法——蚁群算法。

七、模型评估

本章我们通过一系列迭代、优化数据的折线图来对问题一、问题二、问题三的模型进行评估。

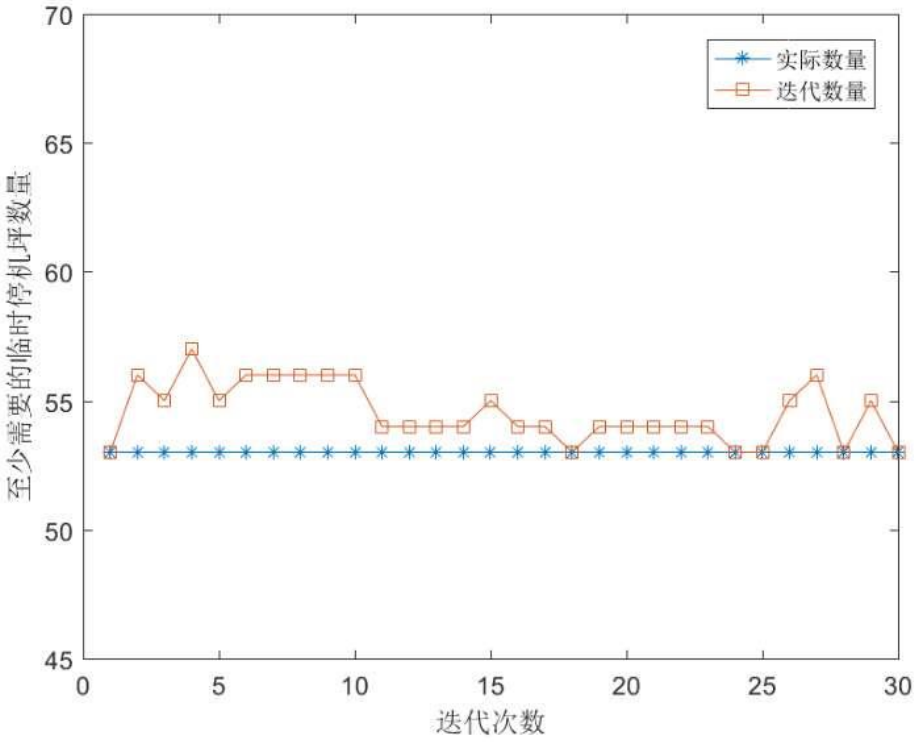


图 12

上图(12)为第一个问题中快速求解算法的优化曲线，可以发现，优化只发生于对停机坪第一个航班的替换上，其后以所有其他停机坪上的航班作为优化对象进行交换都没有能够刷新全局最优解。尽管这种快速求解的算法存在一定缺陷，但它的时间开销非常低，由于限制条件的设定让算法可以直接忽略无法完成航班交换的解集(时间冲突等)，因此该算法在每次迭代中对目标函数的计算次数较少。平均来说，对于一个拥有 300 个航班，70 个登机口的分配问题，这种算法每秒就可以完成一轮新的迭代。

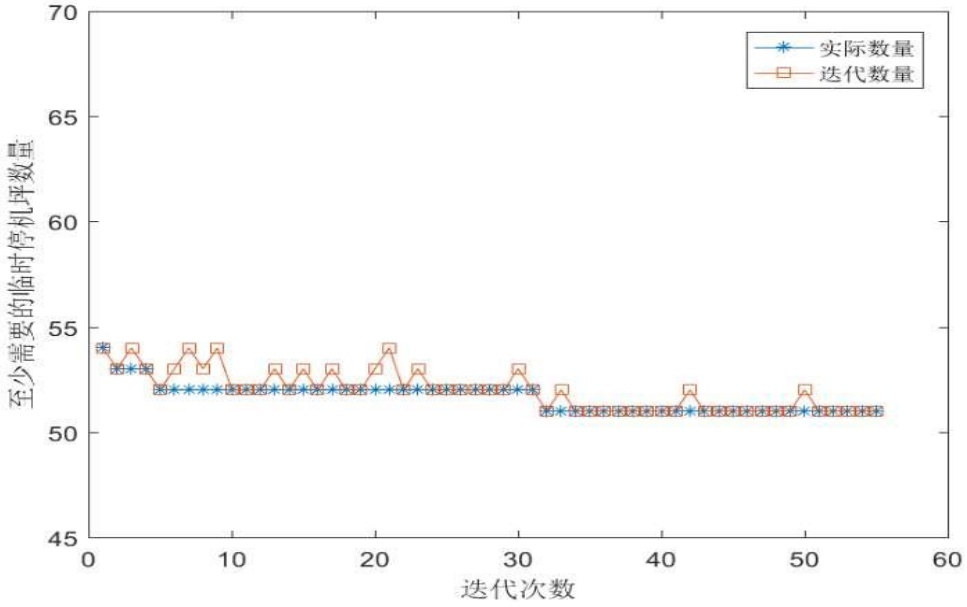


图 13

上图为第一个问题中业务优先求解算法的优化曲线，相比快速求解的算法，这种引入了一个更大解空间的算法可以得到一个更好的优化结果，随着迭代的深入，算法仍然可以不断地找到解集中优于当前最优解的新解。但这也伴随着运算时间的提升，该算法每一轮迭代的计算时间随着迭代的深入会越来越长。平均来说，完成一轮相同数据规模的迭代，这种算法所需要的时间为上述快速求解算法的三倍左右。

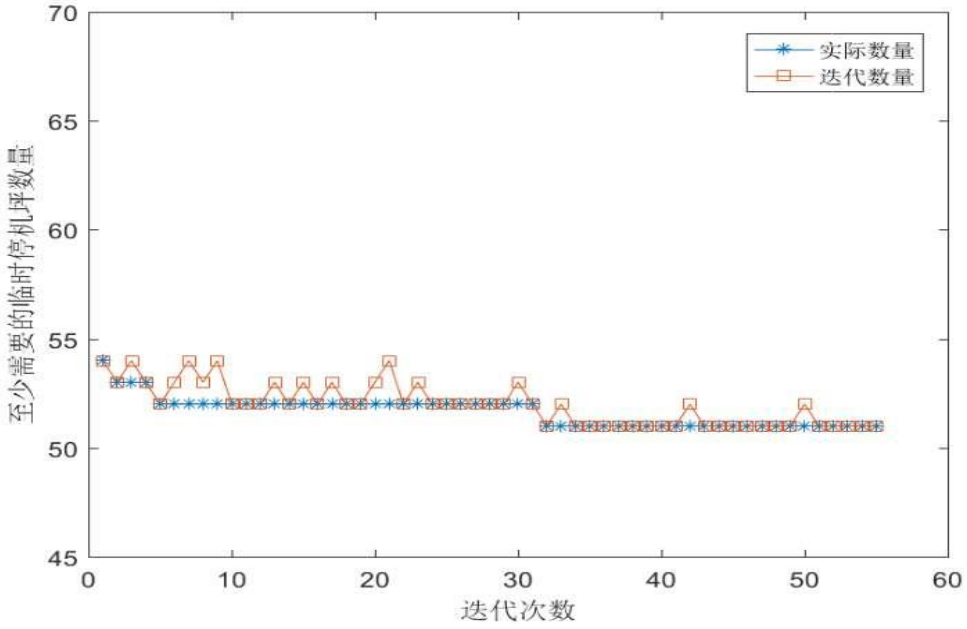


图 14

上图为第二个问题的优化曲线，由于第二问的模型总体建立于第一问的业务优先算法模型之上，并且在出现同解时还需要通过计算总体中转时间来决定解集的取舍，因此此题的模型也需要较长的运行时间。

八：总结与展望

由于旅游业的蓬勃发展，航空出行成为越来越多人的首选出行方式，这也导致机场航站楼的旅客容量达到饱和状态。本课题为解决航站楼旅客容量过大，登机口数量偏少的情况，在航站楼附近新增卫星厅以扩增登机口。但新增卫星厅会导致中转旅客航班衔接问题。

针对这个问题，本文进行了建模求解。问题主要背景是根据一定的约束条件分析机场 2018 年 1 月 20 日到达或出发航班的分配优化情况，包括中转旅客的最短流程时间、中转旅客的换成紧张度等。

本文创造性地采用了一种混合启发式模型来解决登机口分配问题。对于问题一，只考虑航班-登机口分配，优先级是尽可能先保证更多的航班分配到合适的登机口，即尽可能少的使用临时停机位，在此基础上再考虑最小化登机口使用数量。我们采用贪婪算法与禁忌算法相结合的启发式混合模型对问题进行建模和求解。同时，在求解模型的过程中提出了两个不同的原则：速度最优原则和业务最优原则，分别侧重于保证算法运行速度和保证算法运行取得的最终效果。最终，仅需使用临时停机位 51 个，成功分配航班数 253 个，使用登机口数量 69 个。

在问题一的基础上，问题二需要考虑中转旅客的最短流程时间。此时的优先级依旧是优先保证尽可能少的使用临时停机位，在此基础上最小化中转旅客的最短流程时间，最终才考虑最小化登机口使用数量。本题不需要考虑中转旅客的行走时间和坐捷运的时间。本题的模型目标还是最小化临时停机位的数量，但是需在约束条件中增加中转旅客总体最短流程时间最小化这一项，当遇到优化临时登机口同解的情况下，进一步考虑并优化中转乘客的最短流程时间，最后可以得到中转乘客总体流程时间最小为 145435 分钟，换乘失败乘客为 0 个。

在问题三中，加入了换成旅客总体紧张度，此时的优先级是保证尽可能少的使用临时停机位，在此基础上考虑最小化换乘旅客的总体紧张度，最后是考虑最小化登机口使用数量。由于前两题的模型对于求解更新后的目标函数效率很低，因此我们引入了蚁群算法进行求解。

参考文献

【1】Hemchand Kochukuttan and Sergey Shebalov, New Gate Planning Optimizer Perfects Gate Assignment Process, Ascend Magazine, https://www.sabreairlinesolutions.com/pdfs/Plan_Ahead.pdf

【2】Abdelghani Bouras, Mageed A. Ghaleb, Umar S. Suryahatmaja, and Ahmed M. Salem, The Airport Gate Assignment Problem: A Survey, The Scientific World Journal, Volume 2014, http://dx.doi.org/10.1155/2014/923859

【3】Dong Zhang, Diego Klabjan, Optimization for gate re-assignment, Transportation Research Part B: Methodological, Volume 95, January 2017

【4】Shuo Liu, Wenhua Chen, Jiyin Liu, Optimizing airport gate assignment with operational safety constraints, 2014 20th International Conference on Automation and Computing

【5】张彦峰. 机场停机位分配优化研究[D]. 中国民航大学, 2007.

【6】李大卫, 王莉, 王梦光. 遗传算法与禁忌搜索算法的混合策略[J]. 系统工程学报, 1998(3):28-34.

【7】陈华群. 基于图论和蚁群算法的机场停机位分配优化研究[J]. 科技通报, 2015, 31(10):235-238.

【8】文军, 李冰, 王清蓉,等. 机场停机位分配问题的图着色模型及其算法[J]. 系统管理学报, 2005, 14(2):136-140.

【9】Altiparmak F, Karaoglan I. An Adaptive Tabu-Simulated Annealing for Concave Cost Transportation Problems[J]. Journal of the Operational Research Society, 2008, 59(3):331-341.

【10】邢文训. 现代优化计算方法[M]. 清华大学出版社, 2005.

【11】高海昌, 冯博琴, 朱利 b. 智能优化算法求解 TSP 问题[J]. 控制与决策, 2006, 21(3):241-247.

附录

A. 本文算法的 Matlab 实现

A.1 程序运行结果

《第一问 快速求解法（基于贪婪算法）》 运行 flightAssignment_Simplified.m

《第一问 综合求解法（使用更严格的禁忌约束》 运行 flightAssignment.m，对应两种禁忌策略，得到的结果近似

《第一问 平衡求解法（同时考虑最小化停机坪与登机口数量）》 运行 flightAssignment_withAlpha.m

《第二问》运行 flightAssignment2.m

《第三问 禁忌模型（沿用一、二问）》 运行 flightAssignment_backup.m 注意：由于需要考虑同解较多，运行非常缓慢

A.2函数介绍

注：由于篇幅原因，这里仅列出第一问求解建模函数以及工具函数，与它们代码耦合度较高的第二、第三问代码在这里不做罗列。

A.2.1 arrange_ite.m

功能：用位于停机坪的航班 indJ 替换位于固定机位的航班 indI，并根据“先进先出”原则更新 indI 航班后续所有航班的安排方案

输入变量	输出变量
优化前机位安排情况 Arrangement	最新机位安排情况 Arrangement
所有航班信息 flightData	每个登机口何时可用 LastDepart
停机坪的航班 indJ	每个登机口已安排航班数量

	SizeOfPort
固定机位的航班 indI	位于停机坪的航班队列 Stack
登机口信息 GATE	

```
    function [Arrangement,LastDepart,SizeOfPort,Stack] =
arrange_ite(Arrangement,flightData,indI,indJ,GATE)
       [K,~] = size(GATE);[N,~] = size(flightData);
       %Remove indI to END flights
       for i=indI:N
          [r,c] = find(Arrangement==i);
          if(length(r)>1)
             debug =1;
          end
          Arrangement(r,c) = 0;
       end
       SizeOfPort = zeros(K,1);LastDepart = zeros(K,1);
       for j=1:K
           temp = find(Arrangement(j,:)>0,1,'last');
           c = length(temp);
           if(c==0)
              SizeOfPort(j) = 1;LastDepart(j) = 0;
           else
              SizeOfPort(j) = temp+1;LastDepart(j) =
flightData{Arrangement(j,SizeOfPort(j)-1),6};
           end

       end
       %Add indJ
       [Arrangement,LastDepart,SizeOfPort,~] =
FIFO(Arrangement,LastDepart,SizeOfPort,flightData,indJ,in
dJ,GATE,0);
```

```
        %Regular FIFO from indI+1 to END(skip indJ)
        skip = indJ;
        [Arrangement,LastDepart,SizeOfPort,Stack2] =
FIFO(Arrangement,LastDepart,SizeOfPort,flightData,indI+1,
N,GATE,skip);
        %final check
        Stack1 = [];
        for k=1:indI
            [r,~] = find(Arrangement==k);
            if(isempty(r))
                Stack1 = [Stack1 k];
            end
        end
        Stack = [Stack1 Stack2];
    End
```

A2.2 calcMovement.m

功能：根据最新的航班安排，计算所有旅客换乘所需时间

输入变量	输出变量
到达航班映射数据结构 arriveMap	所有旅客转机时间总和 move
出发航班映射数据结构 departMap	换乘失败旅客人数 failure
所有票务信息 TICKET	
机位安排情况 Arrangement	
登机口信息 GATE	

```
    function [move,failure] =
calcMovement(arriveMap,departMap,TICKET,Arrangement,Stack
,flightData)
```

```
        [r,~] = size(TICKET);invalid = 0;move = 0;T_num =
28;failure = 0;
        for i=1:r
            arrFlight =
[TICKET{i,3},'_',num2str(TICKET{i,4})];depFlight =
[TICKET{i,5},'_',num2str(TICKET{i,6})];
            isValidA = isKey(arriveMap,arrFlight);
            isValidB = isKey(departMap,depFlight);
            %排除无效航班记录：包含 19 号就飞来的一部分飞机以及错误信
息
            if(isValidA+isValidB==2)
                flightA = arriveMap(arrFlight);flightB =
departMap(depFlight);
                %排除在停机坪的情况
                indA = find(Stack==flightA);indB =
find(Stack==flightB);
                [arrA,~] = find(Arrangement==flightA);[arrB,~]
= find(Arrangement==flightB);
                if(isempty(indA) && isempty(indB))
                    persons = TICKET{i,2};
                    %D:0,I:1,T:0,S:1
                    s1 = flightData{flightA,4}=='I';s2 =
flightData{flightB,8}=='I';
                    s3 = arrA>T_num;s4 = arrB>T_num;
                    transfer = s1*1000+s2*100+s3*10+s4;
                    switch transfer
                        case {0,11}
                            temp = 15;
                        case {10,1,1100,1111}
                            temp = 20;
                        case {1101,1110}
```

```
                    temp = 30;
                case {1000,100,111}
                    temp = 35;
                case {1010,110,101}
                    temp = 40;
                 case {1011}
                    temp = 45;
            end
            land = flightData{flightA,2};depart =
flightData{flightB,6};
            %检查是否错过航班
            if(land+temp/1440>depart-0.0312)
                temp = 360;
                failure = failure+persons;
            end
            move = move+(persons*temp);
        end
    else
        invalid = invalid+1;
    end
end
disp(num2str(invalid));
end
```

A2.3 dataPreprocessing.m

功能：格式化数据集

输入变量	输出变量
航班数据 PUCK	航班数据 PUCK
所有票务信息 TICKET	所有票务信息 TICKET
登机口信息 GATE	登机口信息 GATE

```
    %dataset preprocessing
    function [PUCK,TICKET,GATE] =
datasetPreprocessing(PUCK,TICKET,GATE)
        %PUCK
        PUCK(1,:)=[];
        PUCK(:,11:12)=[];
    %     xlswrite('processedData.xlsx',PUCK);
        i=1;[r,~] = size(PUCK);
        while(i<=r)

PUCK{i,3}=str2double(PUCK{i,2}(end-1:end))-19+PUCK{i,3};%
到达时间：0 表示 19 号的 0 点，20 号就是 1.0 开始

PUCK{i,8}=str2double(PUCK{i,7}(end-1:end))-19+PUCK{i,8};%
出发时间：同上 ,0.0312 是 45 分钟
            %删除时间为 19 和 21 号的航班信息
            if((PUCK{i,3}<1 && PUCK{i,8}<=1) || (PUCK{i,3}>=2
&& PUCK{i,8}>2) || (PUCK{i,3}<1 && PUCK{i,8}>2) )
                PUCK(i,:)=[];[r,~] = size(PUCK);
                continue;
            end
            PUCK{i,8} = PUCK{i,8} + 0.0312;
            if(isnumeric(PUCK{i,6}))
                if(PUCK{i,6}==332 || PUCK{i,6}==333 ||
PUCK{i,6}==773)
                    PUCK{i,6} = 'W';
                else
                    PUCK{i,6} = 'N';
                end
            else
```

```
            if(strcmp(PUCK{i,6},'332')==1 ||
strcmp(PUCK{i,6},'333')==1 || strcmp(PUCK{i,6},'773')==1 ||
strcmp(PUCK{i,6},'33E')==1 || strcmp(PUCK{i,6},'33H')==1 ||
strcmp(PUCK{i,6},'33L')==1)
                PUCK{i,6} = 'W';
            else
                PUCK{i,6} = 'N';
            end
        end
        i=i+1;
    end
    PUCK(:,2)=[];
    PUCK(:,6)=[];
    %TICKET
    TICKET(1,:)=[]; i=1;
    [r1,~] = size(TICKET);
    while(i<=r1)
```

TICKET{i,4}=str2double(TICKET{i,4}(end-1:end))-19;%到达时间：0 表示 19 号的 0 点，20 号就是 1.0 开始

TICKET{i,6}=str2double(TICKET{i,6}(end-1:end))-19;%出发时间：同上

```
        %删除时间为 19 和 21 号的航班信息
        if((TICKET{i,4}<1 && TICKET{i,6}<1) ||
(TICKET{i,4}>=2 && TICKET{i,6}>=2))
            TICKET(i,:)=[];
        else
            i=i+1;
        end
        [r1,~] = size(TICKET);
```

```
    end

    %GATE
    GATE(1,:)=[];
End
```

A2.4 FIFO.m

功能：根据“先进先出”原则将 indI 到 indJ 之间的航班安排到登机口上

输入	输出
优化前机位安排情况 Arrangement	最新机位安排情况 Arrangement
所有航班信息 flightData	每个登机口何时可用 LastDepart
置换航班 indI	每个登机口已安排航班数量 SizeOfPort
置换航班 indJ	位于停机坪的航班队列 Stack
登机口信息 GATE	
每个登机口何时可用 LastDepart	
每个登机口已安排航班数量 SizeOfPort	

```
    function [Arrangement,LastDepart,SizeOfPort,Stack] =
FIFO(Arrangement,LastDepart,SizeOfPort,flightData,indI,in
dJ,GATE,skip)
        n_ports = [];w_ports = [];
        [r,~] = size(GATE);
        for i=1:r
            if(GATE{i,6}=='W')
                w_ports = [w_ports i];
            else
                n_ports = [n_ports i];
```

```
            end
        end
        Stack = [];
        for i=indI:indJ
            if(skip==i)
                  continue;
            end
            debug_kind = flightData{i,5};debug_arriveType =
flightData{i,4};debug_departType = flightData{i,8};
            debug_arrive = flightData{i,2};debug_depart =
flightData{i,6};
            if(debug_kind=='W')
                [best_ports,qualified_ports,full_ports] =
qualify_port(debug_arriveType,debug_departType,GATE,w_por
ts);
            else
                [best_ports,qualified_ports,full_ports] =
qualify_port(debug_arriveType,debug_departType,GATE,n_por
ts);
            end
            if(isempty(qualified_ports) &&
isempty(best_ports) && isempty(full_ports))
                Stack = [Stack i];
            else
                %找最佳解
                MIN = min(LastDepart(best_ports));
                if(MIN<=debug_arrive)
                    portInd =
find(LastDepart(best_ports)==MIN,1,'first');
                    port = best_ports(portInd);
                else
```

```
                %找次佳解
                MIN = min(LastDepart(qualified_ports));
                if(MIN<=debug_arrive)
                    portInd =
find(LastDepart(qualified_ports)==MIN,1,'first');
                    port = qualified_ports(portInd);
                else
                    %找最差解
                    MIN = min(LastDepart(full_ports));
                    if(MIN<=debug_arrive)
                        portInd =
find(LastDepart(full_ports)==MIN,1,'first');
                        port = full_ports(portInd);
                    end
                end
            end
            %此时如果 MIN>flightData{i,2}，就刷新
Arrangement,SizeOfPort 和 LastDepart，otherwise 加入 Stack
            if(MIN>debug_arrive)
                Stack = [Stack i];
            else
                Arrangement(port,SizeOfPort(port)) = i;
                SizeOfPort(port) = SizeOfPort(port)+1;
                LastDepart(port) = debug_depart;
            end
        end
    end
end
```

A2.5 flightAlter.m

功能：判断位于停机坪的航班 indJ 是否可以替换位于登机口的航班 indI

输入	输出
待换航班信息	是否可换 canAlt
被换航班信息	

```
    function canAlt =
flightAlter(alter_kind,alter_arrive,alter_depart,GATE_kin
d,GATE_arrive,GATE_depart,alter_time,LastDepart)
        if(alter_kind==GATE_kind)
            if((strcmp(GATE_arrive,'D, I')==1 ||
strcmp(GATE_arrive,alter_arrive)==1) &&
(strcmp(GATE_depart,'D, I')==1 ||
strcmp(GATE_depart,alter_depart)==1))
                if(alter_time>LastDepart)
                    canAlt = 1;
                else
                    canAlt = 0;
                end
            else
                canAlt = 0;
            end
        else
            canAlt = 0;
        end
    end
```

A2.6 flightAssignment.m

```
    close all;clc;
    %flightData = xlsread('myData.xls');
    [~,~,PUCK] = xlsread('datarev.xlsx');
    [~,~,TICKET] = xlsread('InputData.xlsx','Tickets');
```

```
    [~,~,GATE] = xlsread('InputData.xlsx','Gates');
    [flightData,~,GATE] =
datasetPreprocessing(PUCK,TICKET,GATE);
    %Test:100 samples
    %flightData = flightData(1:300,:);
    %flightData = load('myData.txt');
    [N,~]=size(flightData);[K,~]=size(GATE);%large_ports =
6;
    skip = 0;
    Arrangement = zeros(K,200);LastDepart =
zeros(K,1);SizeOfPort = ones(K,1);iteration = 1;
    %Initialize
    %flightData,indI,indJ,K,Size
    [Arrangement,LastDepart,SizeOfPort,Stack] =
FIFO(Arrangement,LastDepart,SizeOfPort,flightData,1,N,GAT
E,skip);
    SIZE = length(Stack);max_size = SIZE;
    GLOBE_SIZE = [SIZE];MAX_SIZE = [SIZE];
    while(iteration<=40)
       valueStack = zeros(5,4);
    %Arrangement,flightData,indI,indJ,N,K,Size
      for ite = 1:5
       alter_flight = Stack(iteration+ite-1);record_i =
0;msg = 'Nothing was changed.';
       curr_max = 100;
       for i=1:N
          [~,c] = find(Stack==i);
          if(isempty(c))
             [r,COL] = find(Arrangement==i);
             alter_time = flightData{alter_flight,2};
             alter_flight_kind =
```

```
flightData{alter_flight,5};
            alter_flight_arrive =
flightData{alter_flight,4};
            alter_flight_depart =
flightData{alter_flight,8};
            %判断是否可以交换:机型一样、DI 一致、时间不冲突
            if(COL==1)
                window = 0;
            else
                window =
flightData{Arrangement(r,COL-1),6};
            end
            canAlt =
flightAlter(alter_flight_kind,alter_flight_arrive,alter_f
light_depart,GATE{r,6},GATE{r,4},GATE{r,5},alter_time,win
dow);
            if(canAlt)

[fakeArrange,LastDepart,SizeOfPort,tmpStack] =
arrange_ite(Arrangement,flightData,i,alter_flight,GATE);
                tmpc = length(tmpStack);
                if(tmpc<curr_max)
                    curr_max = tmpc;
                end
                if(tmpc<max_size)
                    record_i = i;curr_max = tmpc;
                    %msg = 'New Length:' +
num2str(max_size)+'.';
                end
            end
        end
```

```
        end
        valueStack(ite,:) = [ite curr_max alter_flight
record_i];
      end
        [rIte,~] =
find(valueStack(:,2)==min(valueStack(:,2)),1,'first');
        bIte = valueStack(rIte,1);newLow =
valueStack(rIte,2);alt = valueStack(rIte,3);rec =
valueStack(rIte,4);
        curr_max = newLow;
        if(newLow~=100)
            [Arrangement,LastDepart,SizeOfPort,Stack] =
arrange_ite(Arrangement,flightData,rec,alt,GATE);
        end
        SIZE = length(Stack);
        if(rec~=0)
            max_size = newLow;
        end
fprintf('Iteration:%d,GLOBE_SIZE:%d,CURR_SIZE:%d,record:%
d\n',iteration,max_size,curr_max,record_i);
        if(bIte==1 || rec==0)
            iteration = iteration+1;
        end
        GLOBE_SIZE = [GLOBE_SIZE,max_size];MAX_SIZE =
[MAX_SIZE curr_max];
    end
```

A2.7 flightPort.m

功能：返回三个优先级下的准入登机口(最优/次优/全能登机口)

输入	输出

进场飞机业务 Arrive/Depart	最优登机口 best_ports
登机口信息 GATE	次优登机口 qualified_ports
登机口 ports	全能登机口 full_ports

```
    function [best_ports,qualified_ports,full_ports] =
qualify_port(Arrive,Depart,GATE,ports)
        qualified_ports = [];full_ports = [];best_ports = [];
        for i=1:length(ports)
            Allowed_Arrive = GATE{ports(i),4};Allowed_Depart
= GATE{ports(i),5};
            %全能机场
            if(strcmp(Allowed_Arrive,'D, I')==1 &&
strcmp(Allowed_Depart,'D, I')==1 )
                full_ports = [full_ports ports(i)];
            %有一个是两用的机场
            elseif((strcmp(Allowed_Arrive,'D, I')==1 &&
strcmp(Allowed_Depart,Depart)==1) ||
(strcmp(Allowed_Depart,'D, I')==1 &&
strcmp(Allowed_Arrive,Arrive)==1))
                qualified_ports = [qualified_ports ports(i)];
            %唯一匹配的机场
            elseif (strcmp(Allowed_Arrive,Arrive)==1 &&
strcmp(Allowed_Depart,Depart)==1)
                best_ports = [best_ports ports(i)];
            end
        end
    end
```

A2.8 transferCalc.m

功能：返回某乘客捷运、行走所需时间

输入	输出
到达业务 typeA	某乘客捷运、行走所需时间 walk
出发业务 typeB	
到达厅 locA	
出发厅 locB	

```
function walk = transferCalc(typeA,typeB,locA,locB)
    comb = [typeA,typeB,locA,locB];walk=0;
    %不需要捷运的情况：typeA==typeB
    %2 次捷运：comb = {'I,D,S,S','D,I,S,S'}
    %1 次捷运：rest
    switch comb
        case {'DDSS','IISS','DDTT','IITT'}
            jieyunTimes = 0;
        case {'DISS','IDSS'}
            jieyunTimes = 2;
        otherwise
            jieyunTimes = 1;
    end
    jieyun = jieyunTimes*8;
    walk = walk+jieyun;
    %厅信息
    if(locA(1)==locB(1))
            walk = walk+10;
    else
        walk = walk+20;
    end
    loc = [locA(2),locB(2)];
    switch loc
        case {'NS','SN','WE','EW'}
            walk = walk+10;
```

```
        case {'NN','SS','WW','EE'}
        otherwise
            walk = walk+5;
    end
end
```