“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　**18102800019**

队员姓名　**1.丁艳红**

2.姚宇巍

3.王江锋

“华为杯”第十五届中国研究生

数学建模竞赛

题 目　机场新增卫星厅对中转旅客影响的评估方法

摘　　要：

本文以最优化理论为基础，研究了机场新增卫星厅对中转旅客影响的评估方法的研究。首先，讨论了不考虑中转旅客的换乘率的境况下尽可能多地分配航班到合适的登机口，并且最小化登机口数量。其次，修改目标函数以符合最小化旅客换乘流程时间。最后，不仅考虑旅客换乘时间，而且考虑捷运时间与步行时间，得出一套最优解。**本文使用贪婪算法求得了上述三种情况下的可行解，并且使用遗传算法逼近全局最优解。**通过两算法的对比，实验证明，**遗传算法能给出更优的登机口分配方案。**并且，提出了一种启发式的多层搜索树算法用于登机口分配，实现优化程度与时间上的折中。

问题一中，忽略中转旅客因素，仅仅考虑航班-登机口分配，针对该问题（AGAP），建立了带约束的数学优化模型。通过贪婪算法寻找局部可行解，并通过遗传算法尝试求解全局最优解。**实验表明，遗传算法的分配方案中，航班分配登机口成功率将近 80%，比贪婪算法所求可行解高出将近 20%。**

问题二中，引入中转旅客最短流程时间的因素，在问题一的基础上，增加了最小化中转旅客总体流程时间的目标，通过加权和的方法将多目标约束最优化问题转化为单目标约束最优化问题。使用贪婪算法和遗传算法求得最优解，并讨论不同适应度权重对求解最优解的影响。**实验表明，旅客换乘成功率达到 100%。**

问题三中，在问题的基础上，重新考虑中转旅客的换乘因素，增加了最小化换乘旅客总体紧张度的目标，且额外考虑最小化步行时间与捷运时间。使用贪婪算法与遗传算法求得最优解。**结果表明，90%旅客的换乘紧张度在 0.2 以内。**并且，提出了一种启发式的多层搜索树算法用于登机口分配，实现优化程度与时间

上的折中，将计算时间降低至少 2/3。

关键字：AGAP，贪婪算法，遗传算法，最优化理论

目录

1、问题重述

1.1 问题背景

随着我国经济的快速发展，我国空中交通运输业蓬勃发展。近几十年来，我国空中的客运量一直持续增长，各个地区的航线也在不断的增加。我国各个航空公司的运输能力日益增强，机场的基础设施建设也越来越完善。机场是我国民航运输业中最重要的基础设施，是保证民航安全和正常运行的重要环节，也是国家航空运输业中的重要组成部分。

由于飞机起降架数和出行旅客的增多，我国空中交通经常出现交通拥挤，延误，登机口资源不足等等问题。现在整个世界的趋势是，机场运营商必须在现有机场配置和容量限制的基础上面对严峻的瓶颈局面，这对航空运输业的整个发展带来重大的影响。根据以往的经验，有些机场通过重新改造当前航站楼或者扩大停机坪区域来增加登机口，这是一个耗资大且不太实际的方法。机场登机口是比较昂贵的设施，而且重新建造登机口不仅要耗费大量的人力、物力，也需要花费很多的时间。

机场登机口是空中交通管理机场运营的重要部分，它涉及到的任务处理是不同航空公司航班到达离开的时间，分配到的登机口以及飞机类型、乘客数目等等信息。目前许多机场面临登机口资源不足的压力。为了缓解这个压力，许多机场通过新建航站厅来增加登机口，这个办法虽然能缓解登机口不足的压力，但是带来了其他的问题。登机口是否能合理地利用不造成空中拥挤、资源浪费，当新建的航站厅离原来的航站厅比较远的时候，中转旅客的换乘时间过长影响登机，以及邻近登机口区的飞机推回或者滑行冲突引起的潜在危险，甚至可能增加废气排放和离开飞机的额外燃料成本[1]。所以目前机场登机口合理分配问题是非常重要的。

自从 19 世纪 70 年代以来，人们研究了许多方法来解决这个问题。Steuart 提出简单的随机模型来找到登机口位置 [2]。之后由于这方面的资料比较少，这个问题的研究逐渐减少。到 20 世纪之后，人们又重新燃起对这个问题研究的兴趣。机场登机口分配的最初目标主要是以乘客为导向或者以机场为导向。关于以乘客为导向，为了提高乘客的满意度，主要注重解决减少旅客的步行时间和中转换乘距离，尽量避免飞机延误。关于以机场为导向，主要是最大化机场的运行效率、最大化登机口的利用率、减少空中拥挤等等问题。

关于以乘客为导向最大化减少乘客的步行距离，Braaks 和 Shortreed 使用定量分析方法描述和模拟最小化乘客步行距离与关键路径的问题[3]。Babic 等人提出了一个 0-1 二进制整数规划模型，并使用分支定界框架来解决问题[4]，但是中转乘客不包括在内。后来，Mangoubi 和 Mathaisel 考虑了根据以前的模型中转乘客的步行距离[5]。此外，他们还尝试将此模型转换成混合模型整数规划（MIP）问题并通过使用线性规划松弛和贪婪启发式方法来解决它。在此之后，出现了另一种尽量减少非门控飞行的考虑因素[6]。此时，最大化减少乘客步行距离的模型已经基本成熟。于是人们考虑使用不同的方法来提高这个问题的计算效率。Xu 和 Bailey 提出混合 0-1 二次分配模型，并使用禁忌搜索算法求解它[6]。

关于以机场为导向合理分配登机口，如果单纯地考虑在实时分配中最小化乘客的步行距离，登机口的利用率是非常低的，也可能导致预先分配到同一个门的

航班登机口出现冲突问题。Yan 和 Chang 认为在飞机进入模型之前增加缓冲时间的重要性，来证明它有助于提高分配到同一个门的两个航班之间的稳健时间表的准时性[7]。Bolat 考虑了最小化空闲时间方差的目标，它的方法提高了登机口的均匀分布利用的可能性，同时保持门分布的鲁棒性[8]。后来也有其他研究者研究提出了比例惩罚，延迟航班无法分配到特定时间窗口的原始位置。

1.2 需要解决的问题

围绕机场新增卫星厅对中转旅客影响的评估方法，本文一次解决如下问题：

问题一：本题只考虑航班-登机口分配。作为分析新建卫星厅对航班影响问题的第一步，首先要建立数学优化模型，尽可能多地分配航班到合适的登机口，并且在此基础上最小化被使用登机口的数量。本问题不需要考虑中转旅客的换乘，但要求把建立的数学模型进行编程，求最优解。

问题二：考虑中转旅客最短流程时间。本问题是在问题一的基础上加入旅客换乘因素，要求最小化中转旅客的总体最短流程时间，并且在此基础上最小化被使用登机口的数量。本题不考虑旅客乘坐捷运和步行时间，但也要求编程并求最优解。

问题三：考虑中转旅客的换乘时间。如前所述，新建卫星厅对航班的最大影响是中转旅客换乘时间的可能延长。因此，数学模型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。本问题可以在问题二的基础上细化，引入旅客换乘连接变量，并把中转旅客的换乘紧张度作为目标函数的首要因素。换乘紧张度定义：

$$\text{换乘紧张度} = \frac{\text{旅客换乘时间}}{\text{航班连接时间}}$$

$$\text{旅客换乘时间} = \text{最短流程时间} + \text{捷运时间} + \text{行走时间}$$

$$\text{航班连接时间} = \text{后一航班出发时间} - \text{前一航班到达时间}$$

其中，行走时间由下列表格查找（单位：分钟。捷运乘坐时间需另行计算）

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

2、模型假设

由于实际航班比较复杂，在整个求解过程中，假设：

假设 1 机场提供数量不限的临时停机位，供分配不到固定登机口的飞机停靠。停在临时停机位的飞机将取消航班。

假设 2 每架飞机转场的到达和出发两个航班必须分配在同一登机口进行，期间不能挪移别处。

假设 3 每个登机口的国内/国际、到达/出发、宽体机/窄体机等功能属性事先给定不能改变。飞机转场计划里的航班智能分配到与之属性相吻合的登机口。

假设 4 分配在同一登机口的两飞机之间的空挡间隔时间必须大于等于 45 分钟。不考虑飞机滑翔时间与其他时间消耗。

假设 5 新建卫星厅对始发旅客和终到旅客的影响不考虑。

假设 6 T 和 S 所有登机口统筹规划分配。

假设 7 T 航站楼与 S 航站楼中间的捷运线，旅客无需等待，随时出发，且单程八分钟。

假设 8 问题 1 中假设换乘旅客在不同航站楼间不花费时间。

假设 9 问题 2 中假设换乘旅客在不同航站楼间仅花费流程时间，而忽略捷运时间与步行时间。

假设 10 问题 3 取消假设 8 与假设 9。

3、符号说明

符号	意义
$\boldsymbol{\rho_{gate}}$	靠桥率:给航班分配登机口的优劣。
$\boldsymbol{G_{success}}$	成功分配登机口的航班。
$\boldsymbol{G_{all\_use}}$	所有使用的登机口。
$\boldsymbol{M}$	登机口的总数。
$\boldsymbol{N}$	航班总数。
$\boldsymbol{x_{ik}}$	0-1 决策变量，$x_{ik}=1$表示第i架飞机被分配到第k个登机口；反之，$x_{ik}=0$
$\boldsymbol{z_{ijk}}$	0-1 决策决策变量，$z_{ijk}=1$表示第i,j架飞机被分配到第k个登机口，且航班j在航班i之后。
$\boldsymbol{A_j}$	第j个航班到达时间。
$\boldsymbol{D_{\mathrm{i}}}$	第i 个航班离开时间。
$\boldsymbol{s_{ik}}$	第i架航班在登机口k之前的空闲时间。
$\mathbf{t_{cost}}$	换乘总消费时间。
$\boldsymbol{t_{pass}}$	换乘流程时间。
$\boldsymbol{t_{vericle}}$	换乘捷运时间。
$\mathbf{t_{walk}}$	换乘步行时间。
$\boldsymbol{t_{cost}^{(q)(k)}}$	乘客q换乘航班被分配到登机口k后所需要的换乘时间。
$\boldsymbol{\rho_{nervice}^{(q)(k)}}$	乘客q的换乘航班被分配到登机口k后的换乘紧张度。
$\boldsymbol{D_{after}^{(q)(k)}}$	乘客q换乘航班被分配到登机口k后，航班的离开时间。
$\boldsymbol{A_{before}^{(q)(k)}}$	乘客q的换乘航班被分配到登机口k前，航班的到达时间。

4、 问题一（航班-登机口分配）

4.1 问题分析与模型建立

问题一只考虑航班登机口分配，要求尽可能多地分配航班到合适的登机口，并且在此基础上最小化登机口的数量。该问题要求是在只考虑航班-登机口分配的情况下，首先将尽可能多的航班分配到合适的登机口，其次在该基础上尽可能最小化使用登机口的数量。这个问题本身就隐藏了两个目标，1.航班被分配到登机口的成功率应尽可能高，尽量减少停到临时停机场的航班数量。2.在航班被分配到登机口成功率保障的情况下，应尽可能地最小化使用登机口地数量。

两个目标本身是有一定的矛盾性的，如果要使登机口使用的数量最小化，那么航班应尽可能多地被分配到临时停机场，其极端情况就是只使用临时停机场。然而这个问题的两个目标是存在优先级的。航班分配到登机口的成功率为高优先级，在这个条件满足的情况下，再考虑目标二。因此本文对两个目标进行分析，将问题转化为首先满足航班分配成功率，即尽可能最大化单个登机口的利用率。

给航班分配登机口的优劣可以用靠桥率ρ_{gate}来表示，其计算公式如公式（1）所示，其中$G_{success}$表示成功分配登机口的航班，G_{all_use}表示所有使用的登机口。

$$\rho_{gate} = \frac{G_{success}}{G_{all\_use}} \tag{1}$$

所有使用的登机口可以由公式（2）表示。M表示登机口的总数，在本问题中，共有 69 个登机口，即$M = 69$。N表示总共的飞机数量。x_{ik}为 0-1 决策变量，$x_{ik} = 1$表示第i架飞机被分配到第k个登机口；反之，$x_{ik} = 0$。$U(\cdot)$为单位阶跃函数，满足$x \geq 0$时，$U(\cdot) = 1$；否则，$U(\cdot) = 0$。G_k则表示第k个登机口。

$$G_{all\_use} = \sum_{k=1}^{M} U\left(\sum_{i=1}^{N} x_{ik}\right) G_k \tag{2}$$

登机口利用率越大，说明登机口被尽可能多地分配到航班。由于航班信息是完全先验已知的，因此登机口的利用率越大，也说明每个登机口被充分利用，可以减少不必要地登机口地使用。因此这是一个单目标的最优化规划问题，是一个NP-hard 的优化问题。

本问题要尽可能多的分配合适的登机口，并且最小化登机口的数量，则可以将问题转化为如下优化问题：机场登机口分配问题（The Airport Gate Assignment Problem，AGAP）。

AGAP：找到最优决策变量x_{ik}与z_{ijk}使得目标方程（3）最小。

$$min \qquad F = \sum_{i=1}^{N}\sum_{k=1}^{M} S_{ik} \tag{3}$$

s. t.

$$x_{ik}, z_{ijk} \in \{0,1\}, \qquad \forall i,j \in N, \forall k \in M \tag{4}$$

$$A_j z_{ijk} - D_i z_{ijk} \leq s_{ik} \leq A_j - D_i z_{ijk} \qquad \forall i,j \in N, \forall k \in M \tag{5}$$

$$A_j - D_i > \beta z_{ijk} \qquad \forall i,j \in N, \forall k \in M \tag{6}$$

$$\sum_{k=1}^{M} x_{ik} = 1 \quad \forall i \in N \tag{7}$$

$$\sum_{i=1}^{N}\sum_{k=1}^{M} z_{ijk} = 1 \quad \forall j \in N \tag{8}$$

$$x_{ik} + x_{jk} - 2z_{ijk} \geq 0 \qquad \forall i,j \in N, \forall k \in M \tag{9}$$

$$x_{jk} - z_{0jk} \geq 0 \qquad \forall j \in N, \forall k \in M \tag{10}$$

$$s_{ik} \geq 0 \quad \forall i,j \in N, \forall k \in M \tag{11}$$

其中，x_{ik}为 0-1 决策变量，$x_{ik} = 1$表示第i架飞机被分配到第k个登机口。反之，$x_{ik} = 0$，该变量已在上文描述。z_{ijk}为 0-1 决策变量，表示第j架飞机在第i架飞机之后被分配到登机口k，若成功分配则为 1，反之则为 0。z_{0jk}表示第j架飞机着陆至登机口k前，并没有其他飞机着陆至该登机口。A_j表示第j架飞机的到达时间，D_i表示第i架飞机的到达时间。β表示两架飞机的间隔时间，由于前一架飞机起飞后，登机口需要进行一系列检查，只有检查完毕后，才允许后一架飞机降落。在本场景中，同一登机口，两架飞机的最小间隔时间为 45 分钟，即$\beta = 45$。

目标函数表示最小化任意两个飞机着陆的间隙，显然要想要使用最少量的登机口，每个登机口的利用率需要达到最大，利用率达到最大，则需要任一两架飞机着陆的间隙时间最小。约束（4）代表求解变量x_{ik}与z_{ijk}为 0-1 决策变量。约束（5）给出了登机口空闲时间的计算方法，若飞机j在飞机i之后分配在同一登机口，即$z_{ijk} = 1$，则$s_{ik} = A_j - D_i$，表示后一架飞机到达时间减去前一架飞机离开时间；反之，$z_{ijk} = 0$，则$0 \leq s_{ik} \leq A_j$即可。约束（6）代表两架飞机间隔的最小值，前文所述，在本问题中，最小间隔是 45 分钟。约束（7）（8）代表每个航班只能分配到一个登机口。约束（9）代表两架飞机的一般关系。而约束（10）指出了第一架被分配到指定登机口的飞机特征。约束（11）代表任意间隔时间的合法性，即负数时间无意义。

通过求解x_{ik}可以获得任意航班安排在哪个登机口为最优情况，求解z_{ijk}可获得任意两个航班的顺序。若$z_{ijk} = 1$，则可知i，j都安排在登机口k，且航班j紧跟着航班i。至此，可以获得分配航班的所有信息。

4.2 问题求解

将航班抽象为物体，将航班在机场转场的时间抽象为物体体积，则该问题就转化为了典型的多背包问题。如何使用最少的背包装入最多的价值，但略有不同

的是，飞机有时刻的性质，只有等飞机降落之后才有可能对其分配某个登机口，否则将毫无意义。求解多背包问题有很多种解法，常用的解法包括贪婪算法，蚁群算法，遗传算法等。贪婪算法能非常快速地求得可行解，但通常不是最优解，遗传算法则能牺牲一定时间的代价下更好获取全局最优解。本章节将分别使用贪婪算法与遗传算法求解，试图获取最优的登机口利用率。

4.2.1 使用贪婪算法求解

贪婪算法是求解最优化问题最简便的方法，贪婪算法虽然只考虑局部的状况，不考虑局部解对于整体最优解未来的影响。但是问题一是一个可分解问题，每架飞机是有顺序到达的。如果针对每架飞机的最优化分配作为局部最优解，而该解又是全局最优解的一部分的话。那么每架飞机的最优化分配问题就是问题一的最优子结构，贪婪算法也可以求得最后的全局最优解，当然并不是每次都能得到。事实上，有些情况离全局最优解相去甚远。本章节使用贪婪算法进行了第一次登机口分配尝试。

贪婪算法用于求解登机口分配问题的基本步骤如算法 1：

算法 1：贪婪算法

Step1：初始化参数。将航班按照到达时间进行排序。先到达的航班优先进行最优化分配。设定所有登机口的空闲时间为上一架飞机的出发时间。

Step2：求解局部最优解。对于每架到来的航班，都寻找其与登机口空闲时间间隔最小的登机口作为局部最优解。

Step3：检验是否为全局最优解。计算航班分配到登机口的成功率，以及登机口的使用数量

从贪婪算法的结果图如图 1 可以看出，宽机型的分配成功率是可观的，可能原因是在本情境中贪婪算法恰好可行，但窄机型的登机口分配率则要欠佳一些。左图则给出了宽机型窄机型以及是否成功占比进行了汇总。可以看出，宽机型占所有机型的 16%，而窄机型占所有机型的 84%，因此从所有机型来看，贪婪算法整体表现性能欠佳，只能说给出了可行解，但并不是最优解。

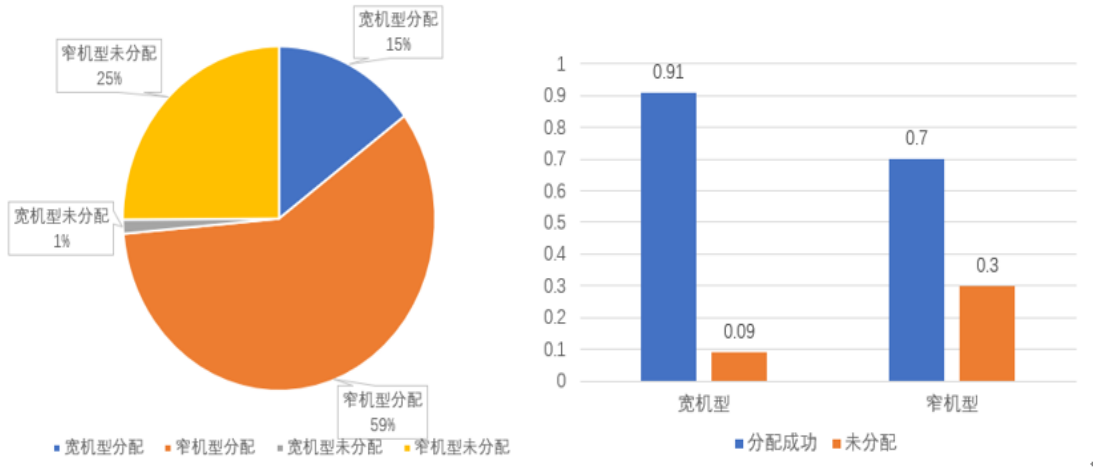

图 1 贪婪算法宽窄体机成功分配到登机口航班数量和比例

4.2.2 使用遗传算法求解

针对问题一的最优化规划问题，其实最直接的方法是枚举法。把所有可能的分配情况都计算其目标函数，最后选取最满足目标需求的一组分配方式，即全局最优解。然而就问题一而言，解空间过于庞大，不可能使用枚举法进行求解（NP难）。而遗传算法就是一种常用的求解最优解或次优解的方法。遗传算法是一种模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，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。为了更好地求解最优解，我们使用带精英策略的遗传算法进行求解。

表 1 给出了本问题中，遗传算法参数配置。

表 1 遗传算法参数配置

Maxgen	Population	corPos	muaPos
500	30	0.7	0.1

遗传算法用于求解登机口分配问题的基本步骤如算法 2：

算法 2：带精英策略的遗传算法

Step1: 初始化参数。设定种群大小，最大迭代次数，交叉算子，变异算子。染色体基因编码过程：直接使用分配给每架飞机的登机口号作为染色体。设定适应度函数为登机口平均占用率。

Step2: 初始化种群。第一次随机分配登机口，计算适应度。并且针对种群按照适应度值进行降序排序。取适应度值最高的个体作为精英并保留。

Step3: 种群自然演化。模拟自然遗传规律，对于染色体随机进行选择，交叉，变异。并对演化后的种群评估其个体适应度。同样按照适应度值进行降序排序，淘汰适应度值最低的个体，使用上一代精英进行补充，并更新精英个体。

选择：根据适应度值占总适应度值的比例，使用轮盘赌法随机选择染色体；

交叉：生成随机数，作为需要截取的染色体片段，并随机选取两条染色体交换；

变异：生成随机数，作为需要变异的基因位置，并随机选择登机口号作为变异；

Step4: 直到达到最大迭代次数，停止演化。取演化最后适应度值最高的个体作为最后遗传算法求得的全局最优解。

遗传算法伪代码为：

```
Genetic Algorithm :
Input: Flight, Gate, Ticket
Output: Flight-Gate-Table
Start:
Initialize parameters; (Maxgen, Population, croPos, mutPos,
weight(1,2))
Initialize variables:
```

```
  chroms: (FlightSeNum, Gate, unappropriated, fitness1, fitness2,
fitness3, fitness)
  Distribute Gate for first time;
  Calculate Fitness for first time;
Start iterating until Maxgen:
  Selection;  (Random)
  Crossover;  (Random)
  Mutation;  (Random)
  Calculate Fitness;
  Sort By Fitness; (choose the Best for Elite strategy)
End iterating;
Display Result;
```

采用贪婪算法，直接禁忌搜索算法，带精英策略的遗传算法对登机口分配问题进行求解的结果见图 2、3、4 所示。

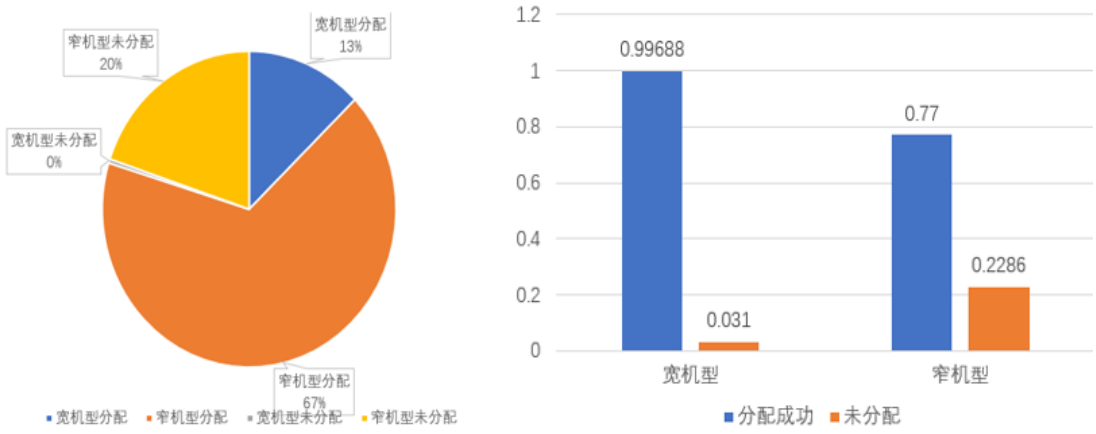

图 2 遗传算法宽窄体机成功分配到登机口航班数量和比例

由图 2 可以看出使用遗传算法求解最优解后，宽机型的成功分配率几乎达到 100%。事实上只有一架宽机型需要停在临时停机位，窄机型的航班分配率也大幅度提升。

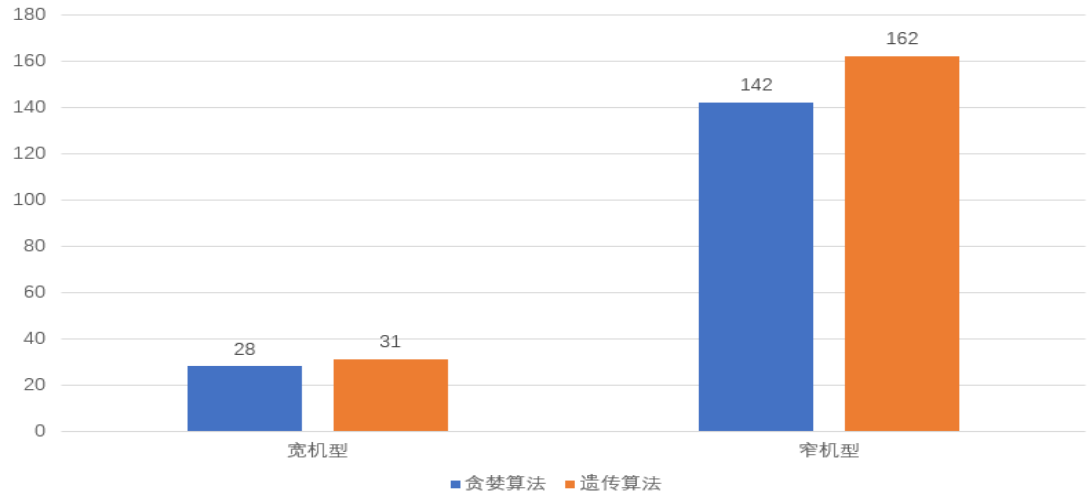

图 3 贪婪算法与遗传算法成功分配登机口航班的比例对比图

由图 3 贪婪算法与遗传算法成功分配登机口航班的比例对比图可以看出，遗传算法在搜索全局最优解的时候更有优势。

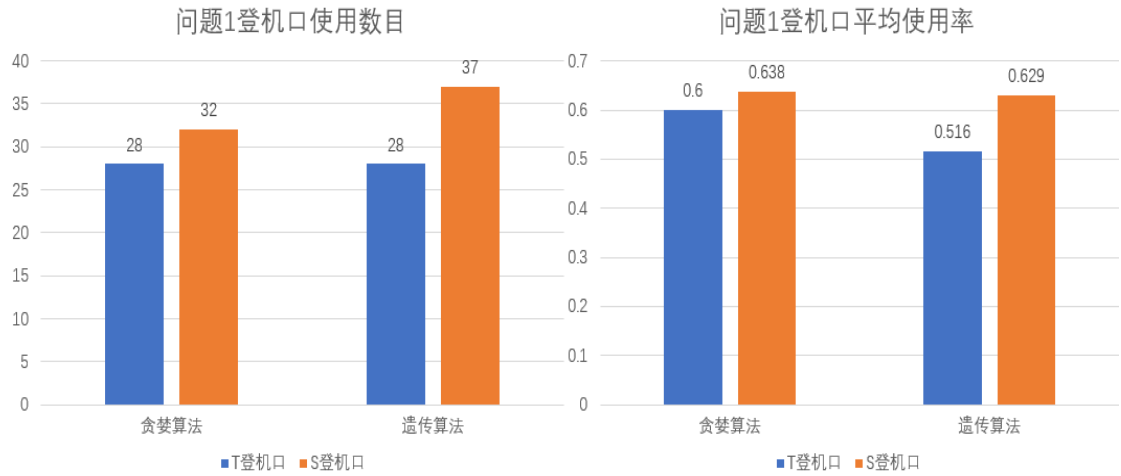

图 4 贪婪算法与遗传算法在最少登机口使用情况下登机口平均使用率的对比

图 4 展示了贪婪算法与遗传算法在最少登机口使用情况下登机口平均使用率的对比。虽然遗传算法似乎使用了更多的登机口，但同时，在平均利用率上也要优秀得多，遗传算法在 T 航站楼与 S 航站楼均有 65%的利用率，即约 16 小时的利用率。

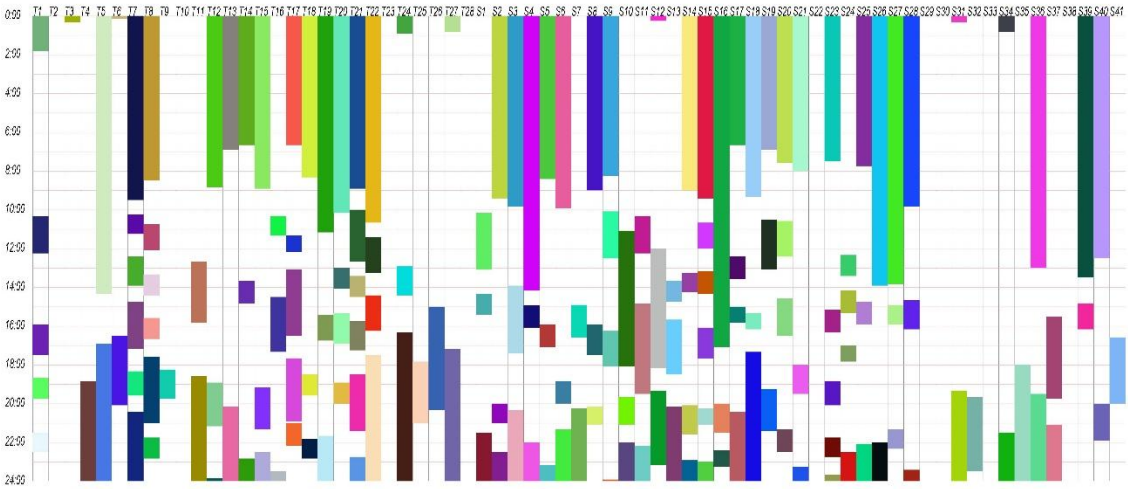

图 5 航班登机口分配甘特图

如图 5 所示展示了航班登机口分配的可视化场景图，横轴表示 T 航站厅的 28 个登机口和新建卫星厅 S 的 41 个登机口，纵轴表示 2018 年 9 月 20 号 0 点-24 点时间。由图 5 我们可以直观的看出飞机在某个登机口停留的时间。

5、问题二（考虑中转旅客最短流程时间航班-登机口分配）

5.1 问题分析与模型建立

本问题在问题一的基础加入了旅客换成因素，要求在最小化中转旅客流程时间的前提下，最小使用登机口的数量。在机场换乘中，旅客需要经过一系列的流程，在本题中，机场航站楼分为主航站楼与卫星厅，每个航站楼又包含国内与国际两个通道，旅客从一个航站楼到另一个航站楼需要一定的时间。如图 6 所示给出了部分航站楼之间换乘所需要的流程时间。

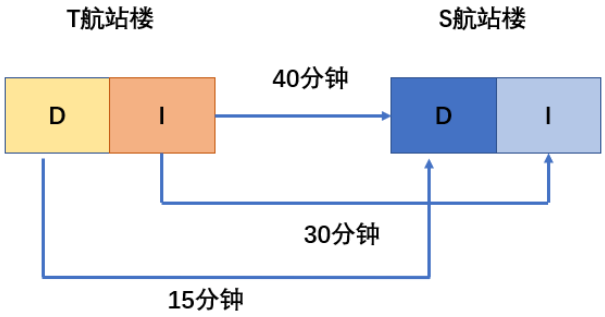

图 6 部分航站楼之间换乘所需流程时间

可以看出，从 T 航站楼国际换乘到 S 航站楼国内需要花费 40 分钟的流程时间，而在本问题中，最长的流程时间为 45 分钟。

图 6 可以使用图论的知识进行分析。航站楼为节点，流程为边，而流程时间则为边上的权重，本问题的目标是寻找从一点到另一点的最短路径。如公式（12）（13）（14）定义图：

$$V = \{DT, IT, DS, IS\} \tag{12}$$

$$E = \{e_{11}, e_{12}, .., e_{44}\} \tag{13}$$

$$G = (V, E) \tag{14}$$

其中V为顶点集，包括国内 T 航站楼（DT），国际 T 航站楼(IT)，国内 S 航站楼(DS)，国际 S 航站楼(IS)。E为边集，为四个位置之间换乘所需要的流程时间，由于到达某航站楼与出从某航站楼出发所需要的时间不同，因此该图为有向图。求解旅客的最短流程时间，即为寻找两节点之间的最短距离。

因此，本问题的总目标函数则可以有以下两部分组成。第一项与问题一一致，即最小化登机口的利用率，第二项为本问题中的最小化换乘旅客的流程时间(15)。

$$min \qquad F = \sum_{i=1}^{N}\sum_{k=1}^{M} S_{ik} + T_{pass} \tag{15}$$

$$T_{pass} = \sum_{i=1}^{Q}\sum_{k=1}^{M} t_{ik} \tag{16}$$

将公式（15）第二项展开为公式（16）所示。t_{ij}表示第i个换乘乘客换乘时，航班位置被分配到第k个登机口所需要的流程时间。

5.2 问题求解

本章节依旧使用贪婪算法与遗传算法进行求解。该过程分为两步，首先最小化旅客的总体流程，其次最小化登机口的使用数量。在求解本问题的过程中，注意到由于换乘飞机航班时间的异步性，使得贪婪算法搜索到的当前最优与全局最优相差甚远，因此本章节提出一种多层搜索树的启发式算法，快速求得优化解。

5.2.1 使用贪婪算法求解

贪婪算法是求解的局部最优解。因此在问题二的求解过程中，贪婪算法除了要考虑到航班到达时间和登机口空闲开始时间的时间间隔之外，还要考虑中转旅客的最短流程时间计算。贪婪算法的流程图如图 7 所示：

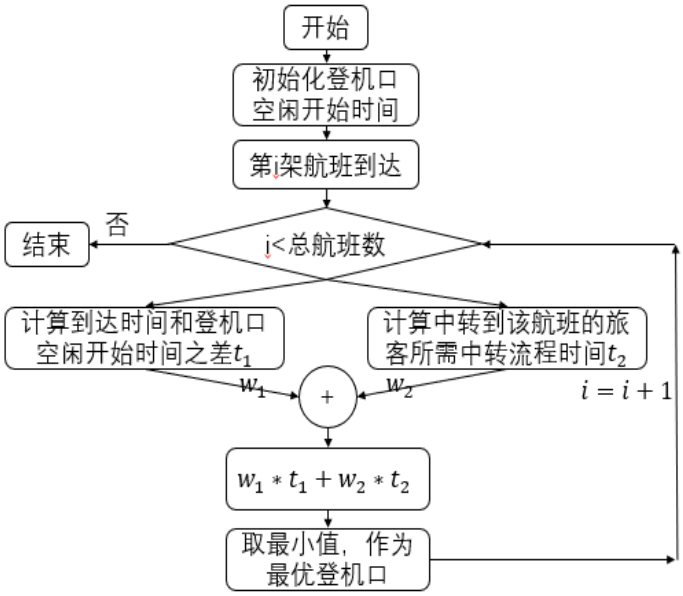

图 7 贪婪算法流程图

从图 7 贪婪算法流程图中可以看出，问题二和问题一的最大区别在于，多目标函数优化。本文使用加权和的方式，将多目标优化问题转化为单目标优化问题。同时可以通过对于w_1和w_2的选择，来突出两个目标函数之间的优先级关系，关系如表 2。

表 2 权重设置与目标优先级的关系

(w_1, w_2)	$(1,0)$	$w_1 > w_2$	$(0.5,0.5)$	$w_1 < w_2$	$(0,1)$
目标优先级	仅有目标 1	目标 1 起主要影响	两目标同等重要	目标 2 起主要影响	仅有目标 2

在该问题中，本文设置w_1为 0.6，w_2为 0.4。由于$w_1 > w_2$，登机口分配成功率的目标优先级就会高于航班中转旅客流程时间最小化的目标优先级，这也符合问题二的原意。

5.2.2 使用多层搜索树求解

由于涉及到换乘因素，贪婪算法并不能很好搜索到与当前目标较远的空间中，因此贪婪算法在本问题中表现欠佳。本文提出了一种多层搜索树的结构，如图 8 多层搜索树流程图所示，来搜索到更远的空间中去，已获得更好的可行解。

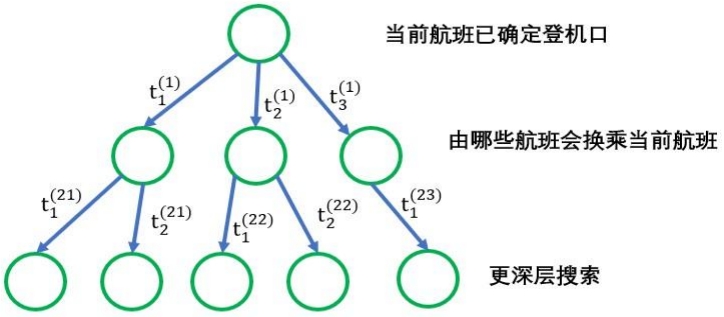

图 8 多层搜索树流程图

当前航班已经确定登机口位置后，向下搜索旅客换乘名单，寻找换乘当前航班的旅客，计算其时间，并寻找最短时间所对应的登机口。该过程可由公式（17）描述，其中$Floor$为搜索层数，R为换乘至根节点所在航班的旅客数。

$$F = \min_{q \in Q} \sum_{f=1}^{Floor} \sum_{i=1}^{R} t_i^{(q)} \tag{17}$$

实际上贪婪算法仅仅是该搜索算法的一个特例，即当只搜索一层的时候；而当层数达到最大时，则该算法表现为枚举法。显然，该算法属于 NP 难问题，不可能穷举所有情况，且每层搜索空间会随着层数呈指数增长，因此只能适当选取层数，在本文中，全选层数为三层。

5.2.3 使用遗传算法求解

同样的，本文在使用遗传算法处理问题二时，也将多目标优化问题通过加权和的方式转化为单目标优化问题，其流程图如图 9 所示：

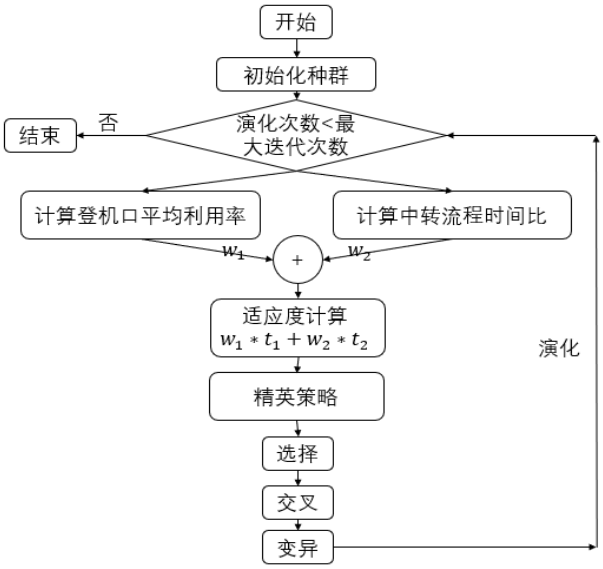

图 9 遗传算法流程图

在遗传算法的具体实施过程中，本文同样使用加权和的方法将多目标优化问题转化为单目标优化问题。使用登机口平均利用率和中转流程时间比作为两个目标函数，登机口平均利用率是为了使得航班分配到登机口的成功率最大化。中转流程时间比是中转流程所用时间除以中转最大停留时间，如公式（18）所示：

$$rate_{pass} = \frac{t_{pass}}{t_m} = \frac{t_{pass}}{t_{depart} - t_{arrive}} \tag{18}$$

为了满足加权和时，两目标函数的影响力相同，因此代表中转流程所需时间的目标函数应尽量满足 0-1 之间。同时，由于出发航班的出发时间和到达航班的到达时间是固定的，因此中转流程时间比和中转流程所需时间成正比。因此中转流程时间比非常适合作为第二目标函数。

采用贪婪算法和遗传算法的计算结果如下：

我们针对 20 号到达或 20 号出发的共 242 架航班和 1585 位中转旅客进行测试。分别使用贪婪算法，遗传算法对这些航班进行登机口分配。分配完后，统计了成功分配到登机口的航班数量和比例，如图 10、11 所示：

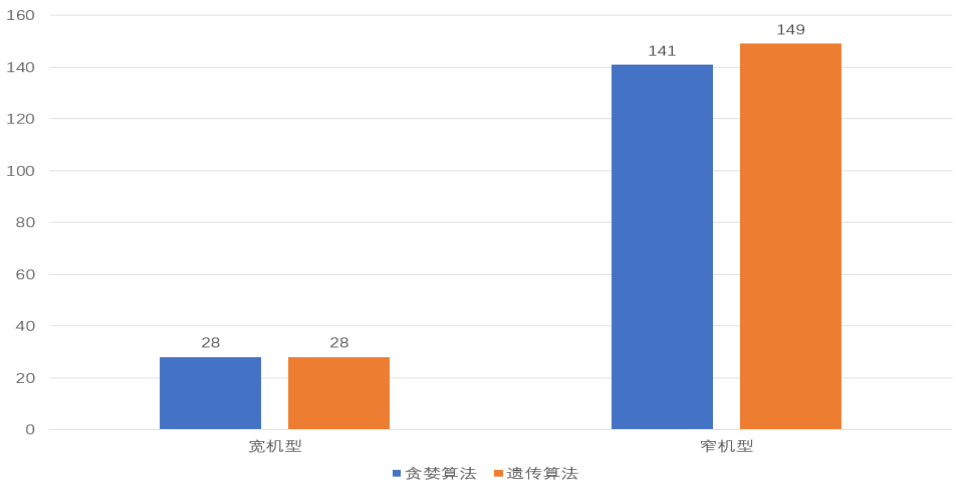

图 10 贪婪算法和遗传算法下宽窄体机成功分配到登机口航班数量

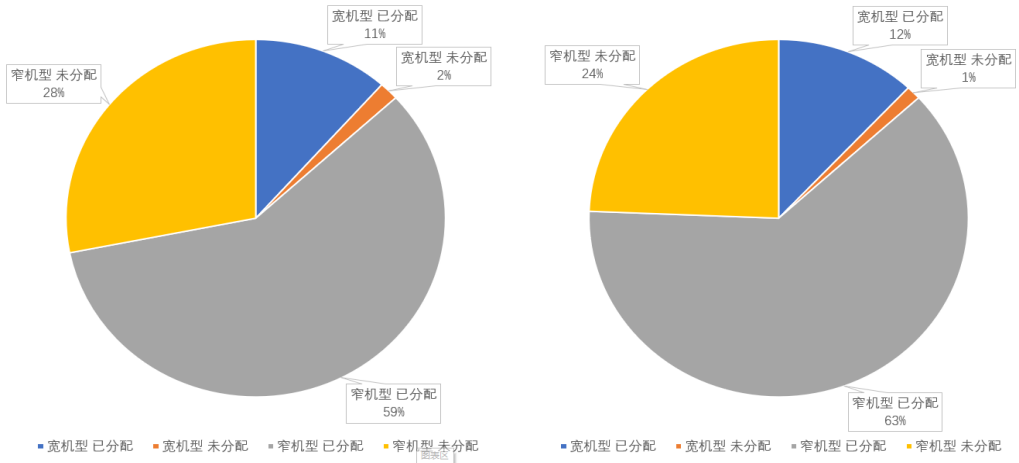

图 11 贪婪算法和遗传算法下宽窄体机成功分配到登机口航班比例

由图可得，遗传算法在登机口分配上比贪婪算法略胜一筹。遗传算法拥有

更高的登机口分配成功率。因为在宽体机和窄体机的分配问题上，遗传算法的结果未分配比率都比较低。同时，我们对于贪婪算法和遗传算法统计了 T 和 S 登机口的使用数目和被使用登机口的平均使用率，如图 12、13 所示：

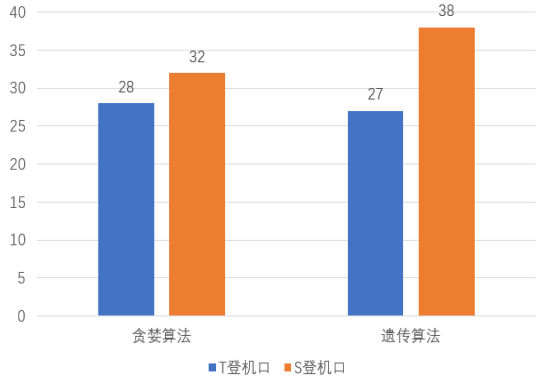

图 12 T 和 S 登机口的使用数目

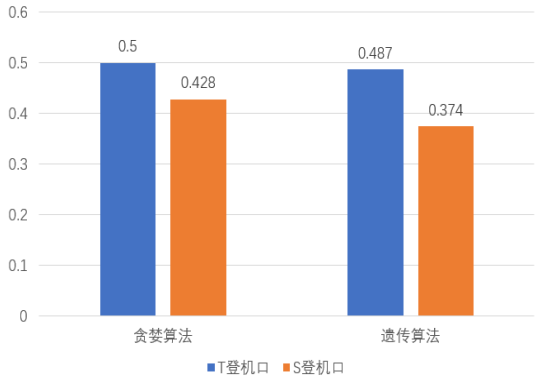

图 13 T 和 S 登机口的平均利用率

从图 12，13 中，可以看到遗传算法的登机口利用率并不高，但是从图 11 中可知，遗传算法能达到更高的登机口分配成功率，这满足更高优先级的目标。因此遗传算法开放更多的登机口，导致了登机口利用率的下降。

贪婪算法和遗传算法最后的分配结果都能达到 100%换乘成功率。为了更好展示结果，我们将给出旅客换乘时间分布图和换乘紧张度分布图

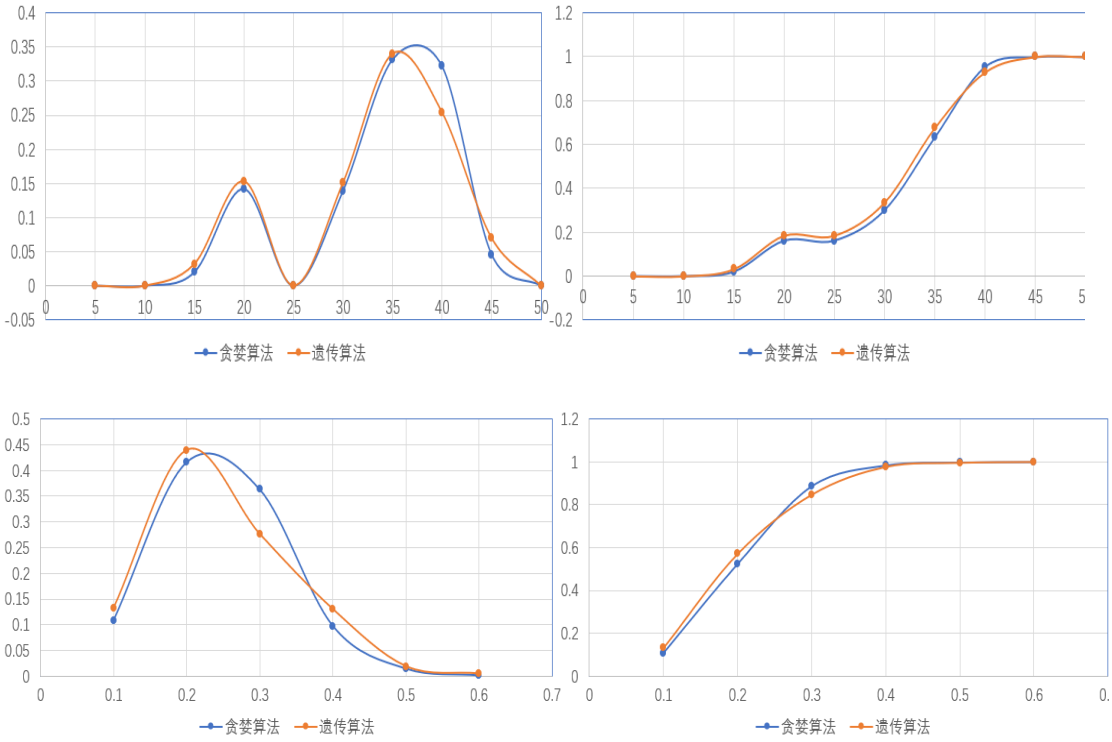

图 14 旅客换乘时间分布比率图及累计分布图(上图左为比率图，右为累计分布)

旅客换乘紧张度分布比率图及累计分布图(下图左为比率图，右为累计分布)

由图 14 得到，在仅考虑最小化总换乘流程时间的目标下，贪婪算法和遗传算法的分配结果对于旅客换乘时间上的调整非常相似，因此旅客换乘紧张度分布也十分相近。从旅客换乘紧张度分布比率图可以看出，遗传算法还是比贪婪

算法略胜一筹。

综上所述，针对于问题二，目标优先级为最大化航班分配登机口成功率>最小化换乘旅客总流程时间>最小化登机口使用数量。遗传算法虽然在登机口利用率上没有明显优势，但是具有较高的航班分配登机口成功率，满足了最高优先级的同时，同时最小化换成旅客的总流程时间，换乘旅客的紧张度也同样较低。

6、问题三（考虑中转旅客换乘航班-登机口分配）

6.1 问题分析与模型建立

本问题要求将考虑换乘乘客的所有换乘时间而不仅仅是流程时间，此外，还要考虑换乘乘客的紧张度。即要在问题二的基础上，不仅仅最小化登机口的使用数量，还要使得乘客换乘的紧张度达到最小，且后者的优先级更高。

旅客的换乘时间不仅仅包含问题二中的流程时间，还要考虑不同航站楼间的捷运时间与航站楼内部的步行时间。因此，整个换乘时间$\mathrm{t_{cost}}$定义为如公式（19）所示，其中t_{pass}为流程时间，$t_{vericle}$为捷运时间，t_{walk}为步行时间。

$$t_{cost} = t_{pass} + t_{vericle} + t_{walk} \tag{19}$$

紧张度定义为换乘时间与航班间隔时间的比值，如公式（20）所示，$t_{interval}$为换乘两个航班之间的时间间隔。将其展开则得到公式（21），D_{after}为后一个航班起飞的时间，A_{before}为前一个航班到达的时间。加入换乘两架飞机的间隔是 1 小时，而换乘所需要的时间是 10 分钟，则紧张度为 10/60=0.17；若两个航站楼之间距离比较远，可能换乘需要 40 分钟，则紧张度为 0.67，可见紧张度越小意味着越换乘时间越充裕；反之则意味着危险，若紧张度大于 1，意味着换乘失败，飞机已经飞走。

$$\rho_{nervice} = \frac{t_{cost}}{t_{interval}} \tag{20}$$

$$\rho_{nervice} = \frac{t_{pass} + t_{vericle} + t_{walk}}{D_{after} - A_{before}} \tag{21}$$

因此，在问题二的基础上，进一步修改目标函数，使得换乘紧张度最优的情况下最优化登机口利用率。目标函数定义如公式（22）所示：

$$min \qquad F = \sum_{i=1}^{N}\sum_{k=1}^{M} S_{ik} + \sum_{q=1}^{Q}\sum_{k=1}^{M} t_{cost}^{(q)(k)} + \sum_{q=1}^{Q}\sum_{k=1}^{M} \rho_{nervice}^{(q)(k)} \tag{22}$$

该该目标函数中，第一项表示最大化航班利用率，第二项表示最小化乘客换乘时间，第三项表示最小化乘客换乘紧张度。第二第三项中，Q表示所有换乘的人数，M表示所有登机口数量。$t_{cost}^{(q)(k)}$表示换乘乘客q所换乘的航班被分配到登机口k所需要换乘的时间。$\rho_{nervice}^{(q)(k)}$表示换乘乘客q所换乘的航班被分配到登机口k后他换乘的紧张度。

将最小化紧张度这一项展开，其分母代表后一架飞机起飞前与前一架飞机到达后的空闲时间。$D_{after}^{(q)(k)}$表示用户q将要换乘的飞机被安排在登机口k后的起飞时间。显然，飞机的起飞时间不会随着分配到不同登机口而改变，因此$D_{after}^{(q)(k)} =$

$D_{after}^{(q)}$。同理$A_{before}^{(q)(k)} = A_{before}^{(q)}$。因此，下式进一步化简。由于飞机航班是事先定好的，且换乘人数是定好的，因此所有飞机起飞与降落间隔是常数，该项可以去除。则，最小化乘客换乘紧张度与最小化换乘时间等价。整个推导流程如公式（23）~（27）所示。

$$\min \sum_{q=1}^{Q}\sum_{k=1}^{M} \rho_{nervice}^{(q)(k)} \tag{23}$$

$$= \min \sum_{q=1}^{Q}\sum_{k=1}^{M} \frac{t_{cost}^{(q)(k)}}{D_{after}^{(q)(k)} - A_{before}^{(q)(k)}} \tag{24}$$

$$= \min \sum_{q=1}^{Q}\sum_{k=1}^{M} \frac{t_{cost}^{(q)(k)}}{D_{after}^{(q)} - A_{before}^{(q)}} \tag{25}$$

$$\text{as: } \sum_{q=1}^{Q}\sum_{k=1}^{M} D_{after}^{(q)} - A_{before}^{(q)} = \text{constant} \tag{26}$$

$$= \min \sum_{q=1}^{Q}\sum_{k=1}^{M} t_{cost}^{(q)(k)} \tag{27}$$

因此，该优化模型的目标函数为公式（28）所示：

$$min \qquad F = \sum_{i=1}^{N}\sum_{k=1}^{M} S_{ik} + \sum_{q=1}^{Q}\sum_{k=1}^{M} t_{cost}^{(q)(k)} \tag{28}$$

s. t.

$$x_{ik}, z_{ijk} \in \{0,1\}, \qquad \forall i,j \in N, \forall k \in M \tag{29}$$

$$A_j z_{ijk} - D_i z_{ijk} \le s_{ik} \le A_j - D_i z_{ijk} \qquad \forall i,j \in N, \forall k \in M \tag{30}$$

$$A_j - D_i > \beta z_{ijk} \qquad \forall i,j \in N, \forall k \in M \tag{31}$$

$$\sum_{k=1}^{M} x_{ik} = 1 \quad \forall i \in N \tag{32}$$

$$\sum_{i=1}^{N}\sum_{k=1}^{M} z_{ijk} = 1 \quad \forall j \in N \tag{33}$$

$$x_{ik} + x_{jk} - 2z_{ijk} \ge 0 \qquad \forall i,j \in N, \forall k \in M \tag{34}$$

$$x_{jk} - z_{0jk} \ge 0 \qquad \forall j \in N, \forall k \in M \tag{35}$$

$$s_{ik} \ge 0 \quad \forall i,j \in N, \forall k \in M \tag{36}$$

$$t_{cost}^{(q)(k)} = c_{ijq} x_{jk} (t_{pass}^{ik} + t_{vericle}^{ik} + t_{walk}^{ik}) \tag{37}$$

其中c_{ijq}为 0-1 变量，若$c_{ijq} = 1$，则乘客q从航班i换乘至航班j；反之则不换成该航班。优化该目标函数求解其最优解，即可获得在最小乘客换乘紧张度的情况下，登机口利用率达到最高。

6.2 问题求解

问题三相较于问题二，更贴近于现实，增加了航站楼之间的捷运时间消耗和旅客步行时间。而这两个时间并非简单地在问题二流程时间的基础上进行简单加权。问题二的第二目标是最小化总流程时间，其目标是直接由到达航班登机口位置和出发航班登机口位置决定的。而问题三的目标是最小化旅客的紧张程度，其由捷运时间，旅客步行时间，流程时间三者共同决定，而这三者又都是由到达航班登机口位置和出发航班登机口位置决定的，其目标与变量的关系图如图 15 所示。

虽然看上去问题三中遗传算法的适应度函数旅客平均紧张程度计算上复杂。但分析其本质，在问题二中，为了使得目标函数之间的影响力相当，将问题二的目标函数由最小化总流程时间转化为最小化流程时间比。在问题三中，将捷运时间，步行时间和流程时间之和看作变量，定义为消耗时间。那么旅客紧张度也就是消耗时间/中转最大停留时间，即消耗时间比。

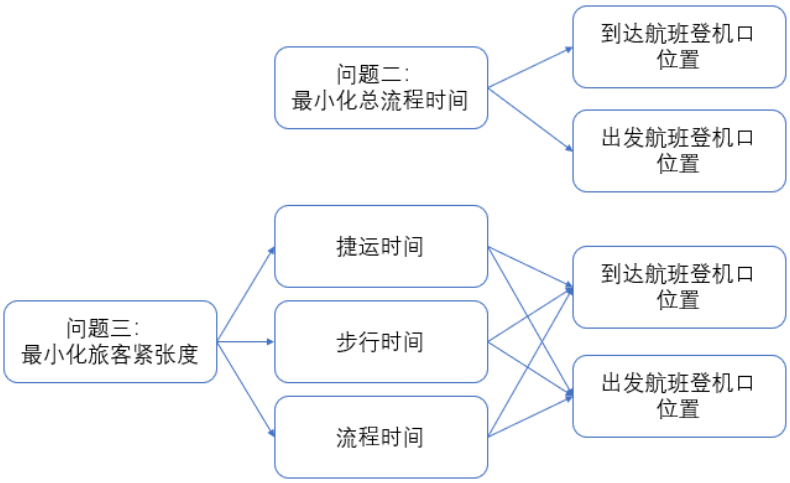

图 15 最小化旅客紧张度与相关变量的关系

因此适应度函数的伪代码表示：

```
Fitness Function:
Input: Flight, Gate, Ticket
Output: chroms
Start:
Calculate GateUSEDPercent:
   For i = 1:size(Flight)
       GateUSEDPercent = sum(Flight_Leave_Time - Flight_Arrive_Time)/(24*60);
   End
   AverageGateUSEDPercent = sum(GateUSEDPercent)/GateNum;
Calculate StressPercent:
   For i = 1:size(Ticket)
       According to Flight-Gate, Calculate BusTime, WalkTime, ProcessTime;
       Flight_Stop_Time = Flight_Leave_Time-Flight_Arrive_Time;
       StressPercent = (BusTime+WalkTime+ProcessTime)/( Flight_Stop_Time);
```

```
End
AverageStress  = sum(StressPercent)/People_Num;
Chroms.fitness  = w1*AverageGateUSEDPercent+w2*AverageStress;
```

与贪婪算法不同，遗传算法不需要考虑航班的先后顺序，因此遗传算法也不存在贪婪算法中由于出发航班还未分配登机口而导致无法计算时间的问题。根据问题三的题意，三个目标的优先级为：最大化航班分配到登机口的成功率>最小化中转旅客的紧张程度>最小化登机口使用数量。因此，问题三的目标权重设置：$w1>w2$。由此取得最优解才符合题目要求。

6.3 结果分析

因此，我们针对 20 号到达或 20 号出发的共 242 架航班和 1585 位中转旅客进行测试。分别使用贪婪算法，搜索树算法，遗传算法对这些航班进行登机口分配。分配完后，统计了成功分配到登机口的航班数量，如图 16 所示：

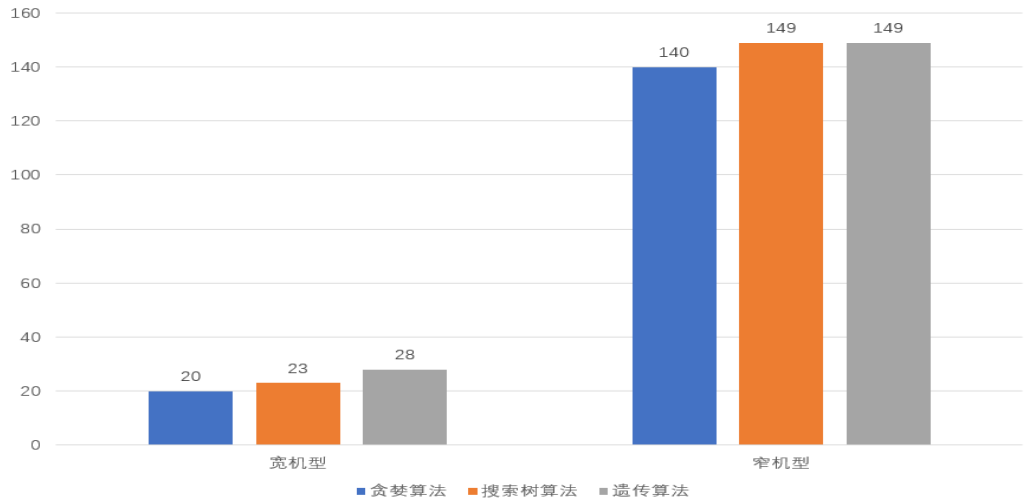

图 16 贪婪算法、搜索树算法、遗传算法成功分配到登机口航班数量

从图 16 中可以清楚地看到，贪婪算法航班分配到登机口的成功率很低。大量飞机都停靠在临时停机场。而迭代 1000 次后，遗传算法能求的解明显优于贪婪算法和搜索树算法所求的解。这是因为贪婪算法和搜索树算法都只能求解当前局部最优解，而遗传算法通过模拟自然选择生物进化的过程，可以求得最优解或者次优解。在之后的实验中，我们尝试遗传算法迭代 2000 次，其结果与迭代 1000 次相近。但是计算耗时巨大，因此我们选取迭代 1000 次为我们的最优结果。

同时，我们对于贪婪算法和遗传算法统计了 T 和 S 登机口的使用数目和被使用登机口的平均使用率，如图 17、18 所示：

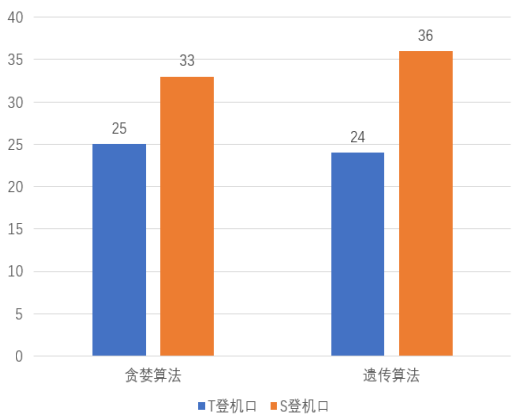

图 17 T 和 S 登机口的使用数目

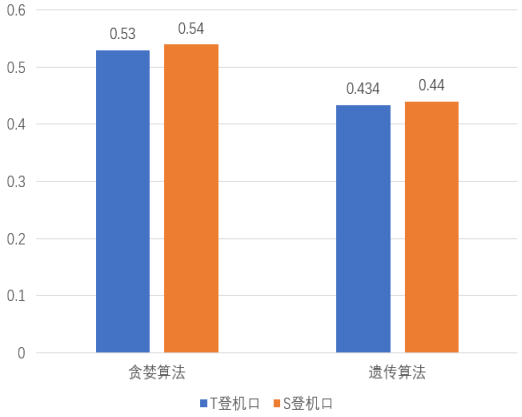

图 18 T 和 S 登机口的平均利用率

由 17, 18 可得，贪婪算法使用的登机口数比较少，因此，虽然其被使用的登机口平均利用率较高，但是由图 16 可得，贪婪算法所求的分配方式，使得飞机停靠登机口的成功率较低，未满足高优先级目标。因此，利用率并不是很高的遗传算法的解更为合理。

三种算法最后的分配结果都不会出现换乘失败旅客，成功换乘率为 100%。为了更好展示结果，我们将给出旅客换乘时间分布图和换乘紧张度分布图。

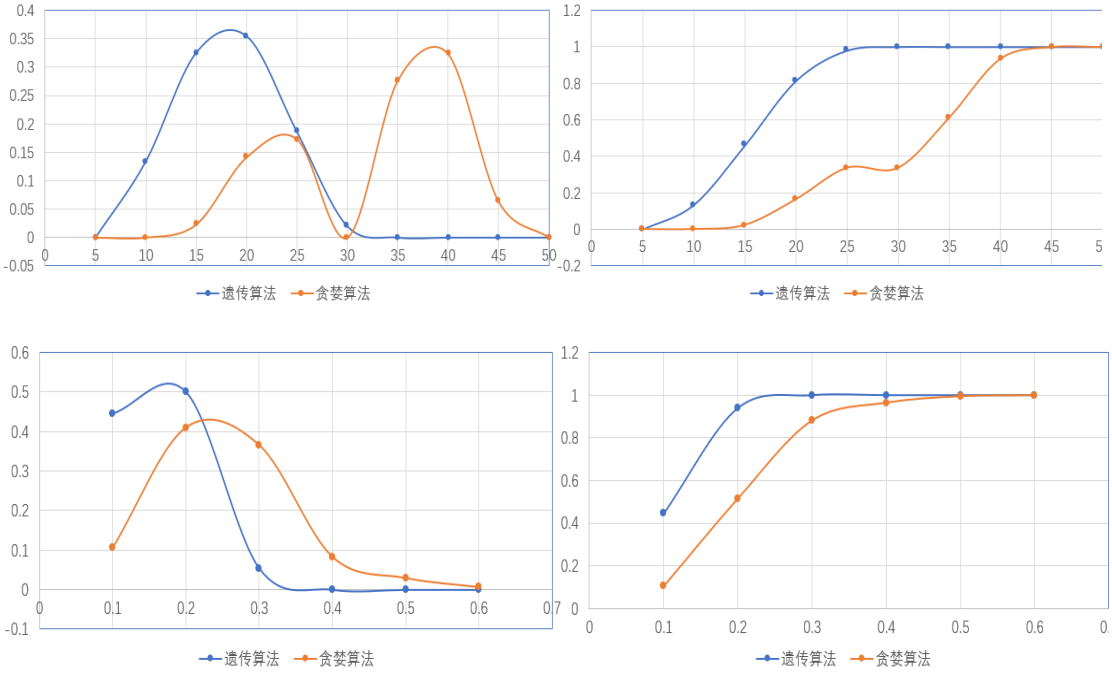

图 19 旅客换乘时间分布比率图及累计分布图(上图左为比率图，右为累计分布)
旅客换乘紧张度分布比率图及累计分布图(下图左为比率图，右为累计分布)

从图 19 中可以得出，贪婪算法虽然能使所有中转旅客换乘成功，但是基本都聚集于 30-45 分钟之间，这样就会导致用户的紧张度上升。而遗传算法中优先考虑了用户紧张度的目标函数，因此，将用户换乘成功时间聚集到 5-30 分钟，因此用户的平均紧张度也较低。

综上所述，针对于问题三，在增加了换乘旅客紧张度的目标后，贪婪算法

出现了由于换乘出发航班还未分配登机口而不能计算时间的问题。并且由于先到先得的机制，导致登机口分配存在不合理性，换乘旅客虽然全部换乘成功，但大多在 30-45 分钟内换乘成功，这就导致换乘旅客的紧张度上升。而遗传算法虽然在登机口利用率上没有明显优势，但是他提高了航班分配到登机口的成功率，同时根据紧张度的目标对登机口分配进行了调整，使用户换乘大都聚集在 5-30 分钟，使换乘旅客的紧张度降低。

7、模型的评价

7.1 模型的优点

问题一中，利用最优化理论，尽可能多地分配航班到合适的登机口，并且在此基础上最小化被使用登机口的数量。通过贪婪算法与遗传算法，将问题转化为带约束的多背包问题，简化了模型计算的复杂程度，提高了模型的可行性，给出了最优的航班登机口分配方案。

问题二中，在问题一的基础上加入旅客换乘因素，增加了最小化中转旅客的总体最短流程时间的约束条件。通过加权和的方法将多目标优化问题转化为单目标优化问题，并使用贪婪算法和遗传算法进行求解，给出了最优的登机口分配方案。

问题三中，引入中转旅客换乘时间的变量。在原问题上增加换成最小化旅客总体紧张度的约束条件。通过贪婪算法，搜索树算法，遗传算法三种算法，分别求解该问题，对比三种算法给出的最终结果，实验证明遗传算法能够求解出全局最优解或全局次优解，并给出最优的分配方案。

7.2 模型的缺点

对于这三个问题，无论是贪婪算法，搜索树算法还是遗传算法，都不能保证能给出全局最优解。

遗传算法虽然能给出全局最优解或全局次优解，但是需要其迭代次数的增加。迭代次数的增加会导致算法计算速度的下降。而贪婪算法虽然能在线性时间求得可行解，但有时离最优解相差甚远。多层搜索树，则可以在优化与时间开销上进行了折中。图 20 给出三种算法的时间开销，该测试使用笔记本 CPU i7-6700 内存 16G GTX1060,多层搜索树采用四层搜索，遗传算法迭代次数 1000 次，种群大小 30。

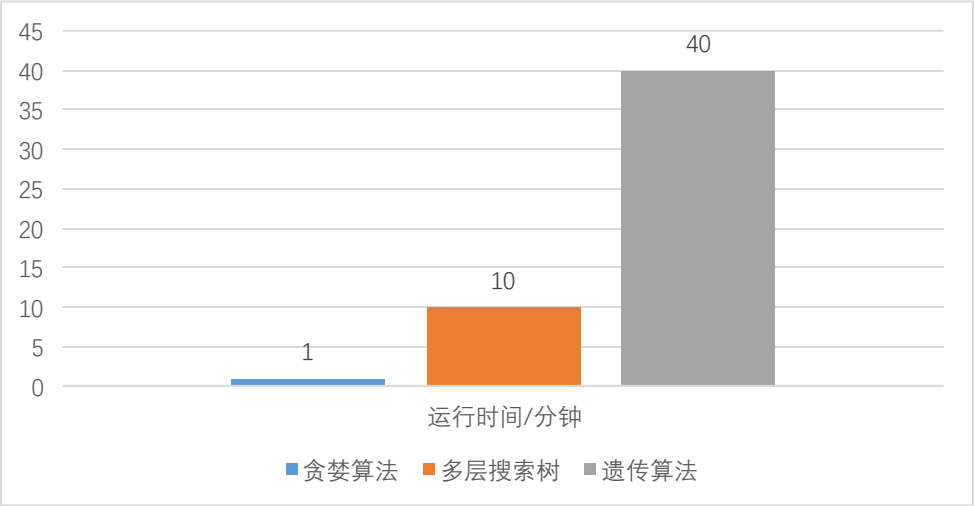

图 20 三种算法的时间开销

8、参考文献

[1] Liu S, Chen W, Liu J. Optimizing airport gate assignment with operational safety constraints[C]// International Conference on Automation and Computing. IEEE, 2014:61-66.

[2] Bouras A, Ghaleb M A, Suryahatmaja U S, et al. The Airport Gate Assignment Problem: A Survey[J]. The Scientific World Journal,2014,(2014-11-20), 2014, 2014(6):9165-9172.

[3] Zhang D, Klabjan D. Optimization for gate re-assignment[J]. Transportation Research Part B, 2017, 95:260-284.

[4] Şeker M, Noyan N. Stochastic optimization models for the airport gate assignment problem[J]. Transportation Research Part E, 2012, 48(2):438-459.

[5] Cheng C H, Ho S C, Kwan C L. The use of meta-heuristics for airport gate assignment[J]. Expert Systems with Applications, 2012, 39(16):12430-12437.

[6] Li W, Xu X. Optimized Assignment of Airport Gate Configuration Based on Immune Genetic Algorithm[M]// Measuring Technology and Mechatronics Automation in Electrical Engineering. Springer US, 2012:347-355.

[7] Dorndorf U, Jaehn F, Pesch E. Flight gate scheduling with respect to a reference schedule[J]. Annals of Operations Research, 2012, 194(1):177-187.

[8] Neuman U M, Atkin J A D. Airport Gate Assignment Considering Ground Movement[J]. Lecture Notes in Computer Science, 2013, 8197(8197):184.

9、附录

9.1 数据结果

附件中包含三个问题的航班登机口计算结果，为 csv 格式。
本文中所有数据均实验所得。

9.2 部分程序

限于篇幅，本文附录只给出部分遗传算法主程序。所有源程序可参见附件中程序文件夹下整理好的文件。

GA_main.m：以下代码为遗传算法的主程序

```
close all;
clear all;
clc;

%% 数据准备
%{
数据格式：
    chroms 为一个存储 struct 的 cells。struct 的结构为：
        1.FlightSeNum          1*303
        2.Gate                 1*303
        3.unappropriated       1*303
        4.fitness              1*1
    Flight 为存储航班信息的 cells，其结构为：
        1.航班序列号(修改为 1-m 数字)
        2.到达日期
        3.到达时间!!!
        4.到达航班号
        5.到达类型!!!
        6.转场时间!!!
        7.出发日期
        8.出发时间!!!
        9.出发航班号
        10.出发类型!!!
        11.飞机大小!!!
    Gate 为存储登机口信息的 cells，其结构为：
        1.登机口型号!!!
        2.登机口号!!!
        3.到达类型!!!
        4.出发类型!!!
        5.飞机大小!!!
        6.空闲开始时间(需要增添)
```

```
    Ticket 为存储用户信息的 cells，其结构为：
        1.旅客记录号
        2.乘客数
        3.到达航班号
        4.到达日期
        5.出发航班号
        6.出发日期
%}
load 'Flight.mat';
load 'Gate.mat';
load 'Ticket_Final.mat'
load 'Gate_Place.mat';
%load 'Ticket.mat';
Flight_origin  = Flight;
Gate_origin = Gate;
Ticket = Ticket_Final;
%Ticket_origin = Ticket;
[Flight,Gate] = DataPrepare(Flight_origin,Gate_origin);
Gate = [Gate,Gate_Place];
[m,n]=size(Flight);
[p,q]=size(Gate);

%% 解结构,参数设定=================================================
% 种群，染色体组成
% 最大迭代次数
Maxgen = 100;
% 种群大小
Y = 30;
% 交叉算子
croPos = 0.7;
% 变异算子
mutPos = 0.1;
% 权重
% w1 = 0.7;
% w2 = 0.3;
% 单目标 or 多目标
% goal = 0;
chroms = cell(1,Y);
%=================================================================
======

%% 计时开始
tic;
% 初始化，构建初始方案/种群
```

```
for i = 1:Y
    structchroms.FlightSeNum = Flight(1:m,1)';
    structchroms.Gate = zeros(1,m);
    structchroms.unappropriated = zeros(1,m);
    structchroms.fitness1 = 0;
    structchroms.fitness2 = 0;
    structchroms.fitness3 = 0;
    structchroms.fitness = 0;
    chroms{1,i} = structchroms;
end

disp('分配登机口');
% 登机口分配!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
chroms = position(chroms, 'first', Flight, Gate);

%chroms{1,1}.Gate
disp('适应度计算')
chroms = fitness3(chroms, Gate, Flight,Ticket);
%chroms{1,1}.Gate

% 适应度值排序!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
disp('适应度排序')
chroms = sortByFitness(chroms);

% 每代精英策略
chromBest = chroms{1,1};
% 历史纪录
trace = zeros(1, Maxgen);
disp('迭代开始');
% 迭代开始!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
k=1;
while(k<=Maxgen)
    STR = sprintf('%s%d','进化代数',k);
    disp(STR);

    % 选择
    chroms = selection(chroms);
    % 交叉
    chroms = crossover(chroms, croPos);
    % 变异
    chroms = mutation(chroms, Gate, mutPos);
    % 计算 fitness
    chroms = position(chroms, 'else', Flight, Gate);
    chroms = fitness3(chroms, Gate, Flight, Ticket);
```

```
    % 适应度值排序
    chroms = sortByFitness(chroms,chromBest);
    % 统计，去除精英
    chromBest = chroms{1,1};
    trace(1,k) = chroms{1,1}.fitness;
    %trace(2,k) = chroms{1,1}.fitness2;
    %trace(3,k) = chroms{1,1}.fitness3;
    k=k+1;
end
% 迭代结束!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
% 计时结束
toc;

%% 输出结果
% 格式：1 最优个体行：2 航班序列号：3 登机口号：4 适应度值 1：5 适应度值 2：6 适应
度
disp('航班序列号');
chroms{1,1}.FlightSeNum
disp('登机口号');
Gate_output = chroms{1,1}.Gate';
%{
switch(goal)
    case 1
        disp('单目标：等待时间');
        chroms{1,1}.fitness
        % 画出迭代图目标函数 1
        figure(1)
        plot(trace(1,:))
        hold on, grid;
        xlabel('进化代数');
        ylabel('最优解变化');
        title('单目标：等待时间')

    case 2
        disp('单目标：换乘流程时间')
        chroms{1,1}.fitness2
        % 画出迭代图目标函数 2
        figure(2)
        plot(trace(2,:))
        hold on,grid;
        xlabel('进化代数');
        ylabel('最优解变化');
        title('单目标：换乘时间')
end%}
```