


“华为杯”第十五届中国研究生数学建模竞赛

学　　校	山东财经大学
参赛队号	18104560009
队员姓名	1.李建苹
	2.陈检
	3.王钰晴

“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

随着旅游业的快速发展，老百姓对交通的要求逐渐增高，导致乘坐飞机的旅客数量越来越大，机场航站楼的旅客流量已达饱和，现通过增加卫星厅来缓解原有航站楼登机口不足的压力，对中转旅客的航班衔接显然具有一定的负面影响。本文通过建立数学模型，优化分配登机口，分析中转旅客的换乘紧张程度，为航空公司航班规划的调整提供参考依据。

针对问题一，本文首先对转场记录和登机口记录进行了研究，主要分两个方面考虑：第一个方面考虑尽可能多的分配航班到合适的登机口，考虑建立 0-1 规划模型，利用 python 程序语言和 Gurobi 语言进行编程求解，得出最多有 251 个飞机转场被分配，所以有 502 个航班次数被分配到对应的登机口，最终计算结果显示有 60 个登机口被使用。

关键词：航站楼扩增，0-1 规划模型，登机口分配，多目标规划

目 录

1、 问题重述

1.1 问题背景

由于旅游业的快速发展，某航空公司在某机场的现有航站楼 T 的旅客流量已达饱和状态，为了应对未来的发展，现正增设卫星厅 S。但引入卫星厅后，虽然可以缓解原有航站楼登机口不足的压力，对中转旅客的航班衔接显然具有一定的负面影响。本题希望参赛选手建立数学模型，优化分配登机口，分析中转旅客的换乘紧张程度，为航空公司航班规划的调整提供参考依据。

飞机在机场廊桥（登机口）的一次停靠通常由一对航班（到达航班和出发航班，也叫“转场”）来标识，如下图所示。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。所谓的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-登机口的优化分配问题已经被很好地解决，Sabre Airline Solutions 有非常成熟的产品满足航空公司和机场地勤服务公司的需求。但在优化分配登机口的同时考虑最小化旅客行走时间，学界研究有限，市场上产品一般也不具备此功能。

1.2 需要解决的问题

评估一个系统必须首先掌握该系统如何运行。真实的航班-登机口分配调度方案可能受到各种制约因素的影响，但作为评估，我们可以不妨假设其是在一定约束条件下的优化方案。也就是说，本赛题要求就以下三种情形对航班-登机口分配问题建立数学优化模型，并通过求解这些模型，进行数据评估分析。

需要通过建立数学模型，解决以下几个问题：

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。和前面两个问题一样，本问题也要求把建立的数学模型进行编程，并求最优解。

2 、模型假设

为了便于问题的研究，对题目中某些条件进行简化及合理的假设。

假设 1：航站楼 T 和卫星厅 S 同时使用。

假设 2：T 和 S 之间有捷运线相通，可以快速往来运送国内、国际旅客。假定旅客无需等待，随时可以发车，单程一次需要 8 分钟。

假设 3：机场另有简易临时机位，供分配不到固定登机口的飞机停靠。假定临时机位数量无限制。

假设 4：新建卫星厅对始发旅客和终到旅客的流程时间没有影响。

3 、符号说明

符号	含义
Ω_{st}	所有的登机口
Ω_{pk}	所有的转场空间
ST_j	登机口是否被使用，被使用为 1，否则为 0
$PK_i\_ST_j$	是否分配转场 i 到登机口 j，是为 1，否为 0

4 、问题一:模型的建立与求解

4.1 问题描述与分析

问题一是研究航班-登机口分配问题，问题中所有登机口统筹规划分配，并且每个登机口的国内/国际、到达/出发、宽体机/ 窄体机等功能属性事先给定，飞机转场计划里的航班只能分配到与之属性相吻合的登机口，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，其间不能挪移别处，且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，考虑在这种情况下航班-登机口的最优分配策略，主要考虑如何在尽可能多的分配航班到合适的登机口的前提下，保证最小化被使用登机口的数量。

4.2 数据处理

根据附件中提供的数据对航班情况进行处理，要求只选取 20 日到达或 20 日出发的航班和旅客进行分析。经过处理，共筛选出 303 个转场记录号，本文基于这 303 个转场记录号进行分析。

4.3 模型准备

0-1 规划是一类特殊的整数规划，这种规划的决策变量仅取值 0 或 1 ，故称为 0-1 变量或二进制变量。0-1 变量可以数量化的描述诸如开与关、取与舍、有与无等现象所反映的离散变量间的逻辑关系、顺序关系以及互斥的约束条件，因此

0-1 规划非常适合描述解决如线路设计、工厂选址、生产计划安排、旅行购物、背包问题、人员安排、代码选取、可靠性等人们所关心的多种问题。实际上，凡是有界变量的整数规划都可以转化为 0-1 规划来处理。0-1 规划主要用来解决互斥的计划问题、约束条件互斥问题、固定费用问题和分派问题等方面。

4.4 模型建立与求解

4.4.1 模型建立

在航班与登机口协同任务规划中，采用 0-1 规划模型，基于一定的环境要素和任务需求，在保证最大化分配航班到合适的登机口的前提下，最小化被使用登机口的数量。所以，问题一可以看作是一个基于 0-1 规划模型的最佳路径问题。

根据前期数据准备可以看出，问题一需考察 303 个转场航班数据，且问题一需分两个维度进行。用文字描述为：

第一个维度：计算转场航班的最大值。

第二个维度：计算最少登机口数量。

具体变量定义如下：

Ω_{st}为所有的登机口，Ω_{pk}为所有的转场空间

$$ST_j，\ j \in \Omega_{st} \begin{cases} 1，\text{使用登机口 j} \\ 0，\text{其他} \end{cases}$$

$$PK_i\_ST_j\ ，\ i \in \Omega_{pk}，\ j \in \Omega_{st} \begin{cases} 1，\text{分配转场 i 到登机口 j} \\ 0，\text{其他} \end{cases}$$

则转化后的目标函数为：

$$\min \sum_{i=0,j=0} (ST_j - PK_i\_ST_j)$$

约束条件解释如下：

① $PK_i\_ST_j \leq ST_j, i \in \Omega_{pk}, j \in \Omega_{st}$ ex:若飞机第 j 个登机口转场的到达航班和出发航班存在，则此登机口存在

② $\sum_{j=0} PK_i\_ST_j \leq 1, i \in \Omega_{pk}, j \in \Omega_{st}$ ex: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

③ $t_1 * PK_i\_ST_j - t_2 * PK_{i'}\_ST_{j'} \geq 45 * ST_j$ ex: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

第二步的目标函数为：

$$\min \sum_{j=0} ST_j$$

④ $\sum_{i=0,j=0} PK_i\_ST_j \geq \mathrm{n}, i \in \Omega_{pk}, j \in \Omega_{st}$ ex:最多容纳 n 个转场，n 需要枚举确定

4.4.2 模型结果分析

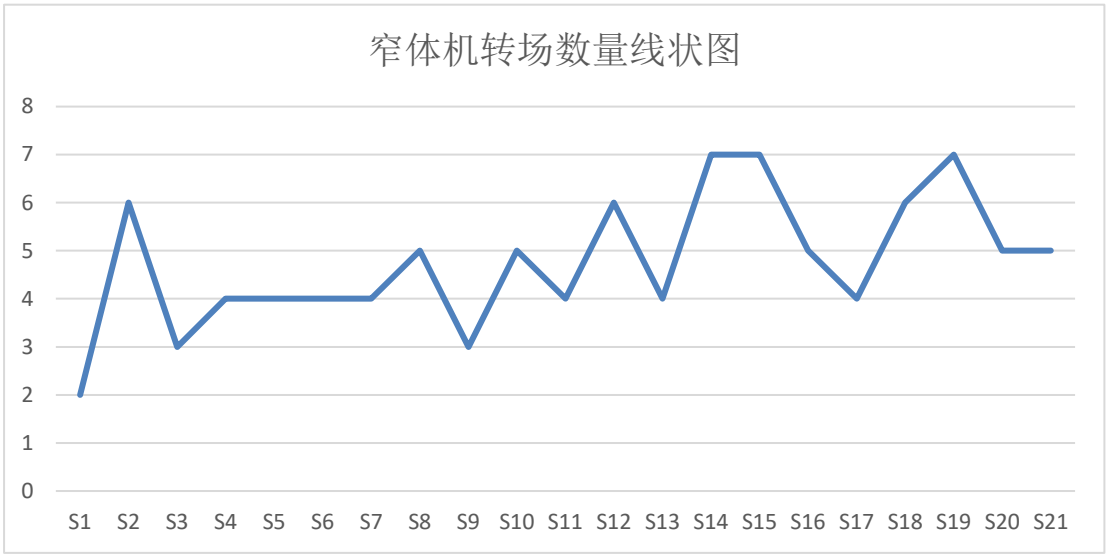
经过对问题的分析，运用 Python 与 gurobi 编程语言对模型进行运算求解，分析得出，共使用了 60 登机口，分配了 251 次转场，共涉及不同时间段的 502 次航班，其中航站楼 T 共使用 24 个登机口，卫星厅 S 共使用 36 了个登机口。具体成功分配到登机口的航班数量和比例见下表。

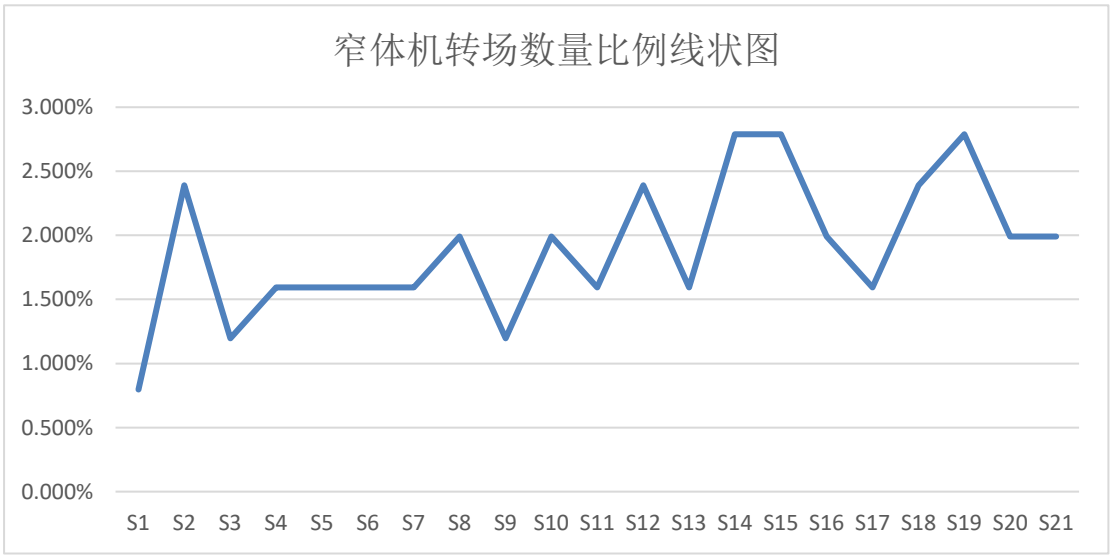
（1） 窄体机

登机口	转场数量	比例	登机口	转场数量	比例
S1	2	0.797%	S22	6	2.390%
S2	6	2.390%	S23	6	2.390%
S3	3	1.195%	S24	5	1.992%
S4	4	1.594%	S25	6	2.390%
S5	4	1.594%	S26	6	2.390%
S6	4	1.594%	S27	5	1.992%
S7	4	1.594%	S28	4	1.594%
S8	5	1.992%	T1	7	2.789%
S9	3	1.195%	T7	8	3.187%
S10	5	1.992%	T8	6	2.390%

S11	4	1.594%	T9	6	2.390%
S12	6	2.390%	T10	4	1.594%
S13	4	1.594%	T11	3	1.195%
S14	7	2.789%	T12	3	1.195%
S15	7	2.789%	T13	2	0.797%
S16	5	1.992%	T15	3	1.195%
S17	4	1.594%	T16	2	0.797%
S18	6	2.390%	T20	7	2.734%
S19	7	2.789%	T21	6	2.344%
S20	5	1.992%	T22	8	3.125%
S21	5	1.992%			

线状图如下：



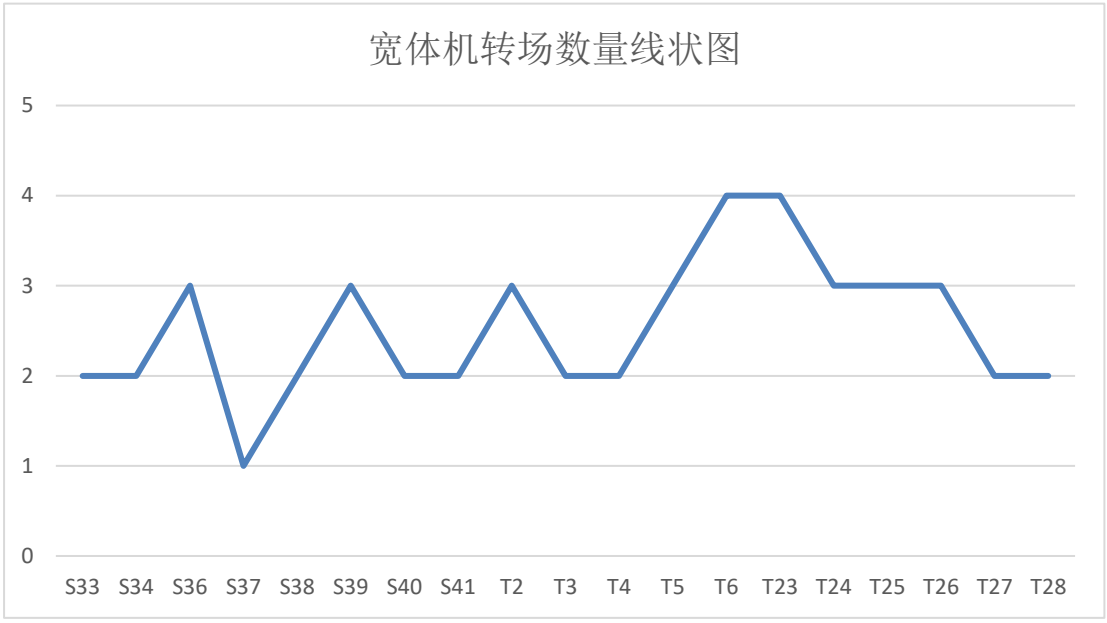


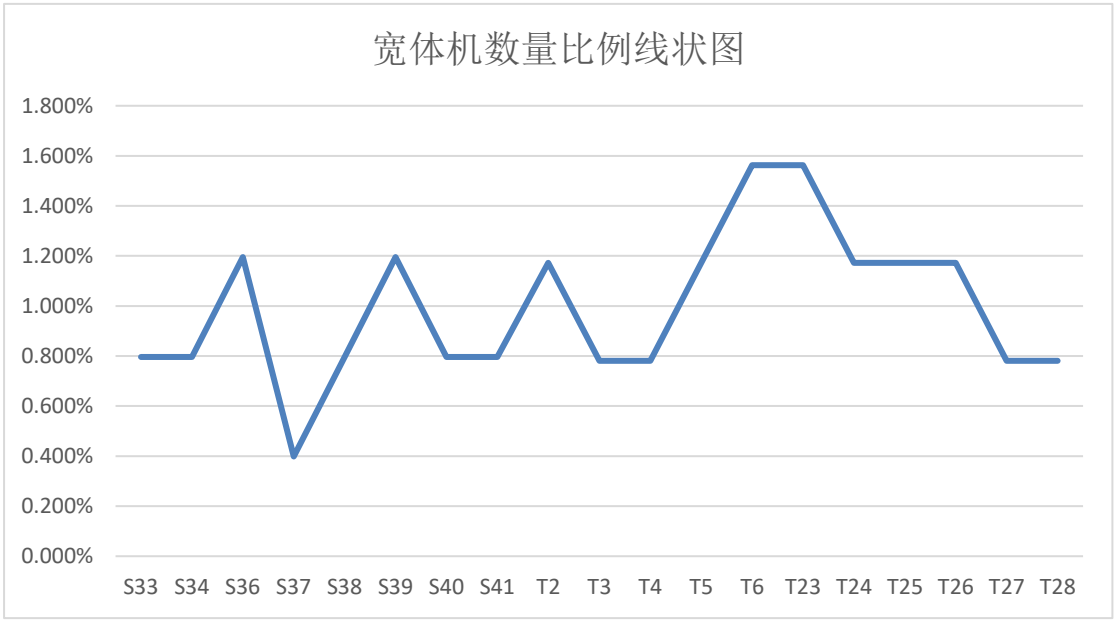
（2） 宽体机

登机口	航班数量	比例	登机口	航班数量	比例
S31	3	1.172%	T2	2	0.781%
S32	1	0.391%	T4	3	1.172%

S33	3	1.172%	T5	1	0.391%
S34	2	0.781%	T6	4	1.563%
S36	2	0.781%	T23	3	1.172%
S37	3	1.172%	T24	5	1.953%
S38	2	0.781%	T25	2	0.781%
S39	1	0.391%	T26	2	0.781%
S40	3	1.172%	T27	2	0.781%
S41	3	1.172%	T28	2	0.781%

线状图如下：





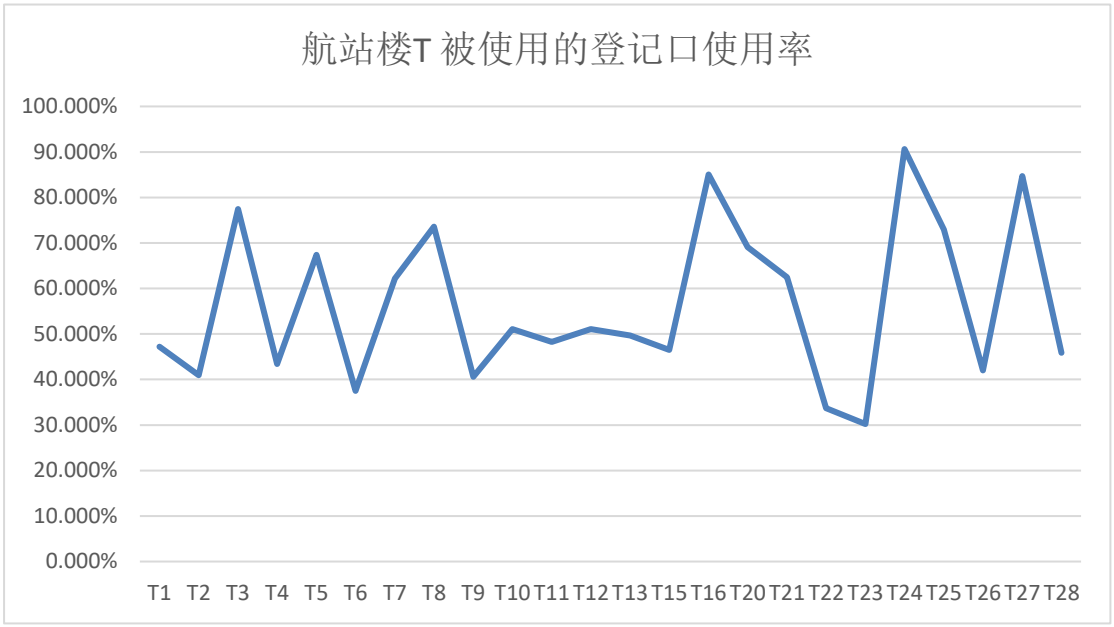
另外，T 和 S 登机口的使用数目和被使用登机口的平均使用率见下表。

（1） 航站楼 T

登机口	转场数	比例	机体类别	登机口	转场数	比例	机体类别
T1	7	47.222%	N	T13	2	49.653%	N
T2	3	40.972%	W	T15	3	46.528%	N

T3	2	77.431%	W	T16	2	85.069%	N
T4	2	43.403%	W	T20	7	69.097%	N
T5	3	67.361%	W	T21	6	62.500%	N
T6	4	37.500%	W	T22	8	33.681%	N
T7	8	62.153%	N	T23	4	30.208%	W
T8	6	73.611%	N	T24	3	90.625%	W
T9	7	40.625%	N	T25	3	72.917%	W
T10	4	51.042%	N	T26	3	42.014%	W
T11	3	48.264%	N	T27	2	84.722%	W
T12	3	51.042%	N	T28	2	45.833%	W

被使用登机口的平均使用率线状图如下：

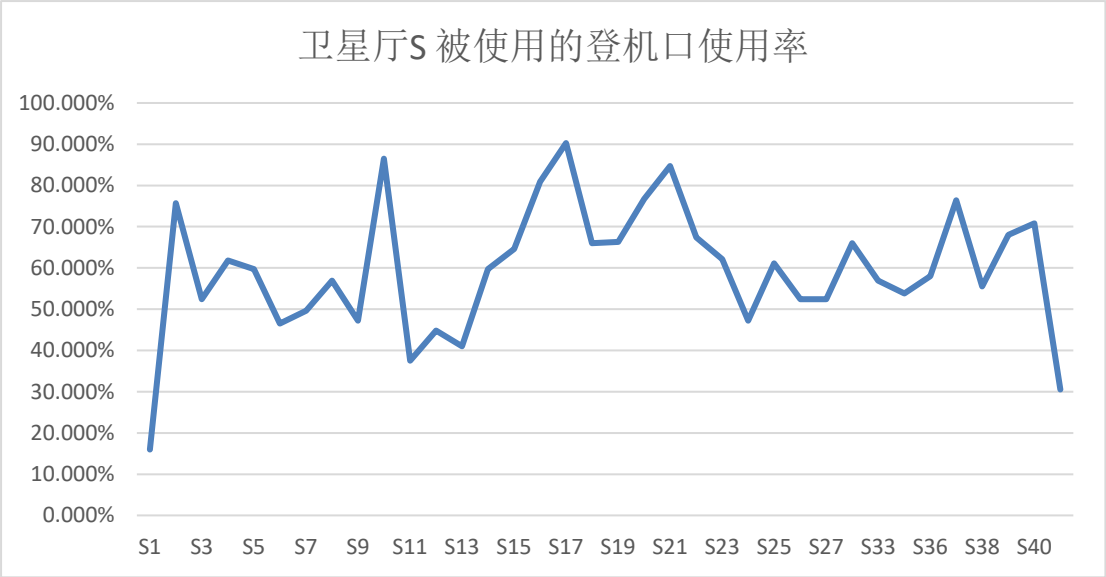


（2） 卫星厅 S

登机口	转场数	使用率	机体类别	登机口	转场数	使用率	机体类别
S1	2	15.972%	N	S19	7	66.319%	N
S2	6	75.694%	N	S20	5	76.736%	N
S3	3	52.431%	N	S21	5	84.722%	N
S4	4	61.806%	N	S22	6	67.361%	N
S5	4	59.722%	N	S23	6	62.153%	N
S6	4	46.528%	N	S24	5	47.222%	N
S7	4	49.653%	N	S25	6	61.111%	N
S8	5	56.944%	N	S26	6	52.431%	N
S9	3	47.222%	N	S27	5	52.431%	N
S10	5	86.458%	N	S28	4	65.972%	N
S11	4	37.500%	N	S33	2	56.944%	W
S12	6	44.792%	N	S34	2	53.819%	W
S13	4	40.972%	N	S36	3	57.986%	W
S14	7	59.722%	N	S37	1	76.389%	W
S15	7	64.583%	N	S38	2	55.556%	W

S16	5	80.903%	N	S39	3	68.056%	W
S17	4	90.278%	N	S40	2	70.833%	W
S18	6	65.972%	N	S41	2	30.556%	W

被使用登机口的平均使用率线状图如下：



4.5 模型评价

4.5.1 模型的优点

（1）本文充分考虑了模型的现实条件，对模型的多种可能情况进行了综合分析，以确保算法具有现实可行性。另外，对模型进行了多次调试，确保结果的真实准确。

（2）对输出结果进行图表化处理，使输出结果更加直观明了。

（3）算法计算复杂度小，收敛速度快，易于理解。

4.5.2 模型的缺点

（1）模型有些条件采取了理想化的处理，实际中，应该更加深入严谨的进行探讨。

（2）对某些信息考虑不足，模型还有待完善。

5、问题二：模型的建立与求解

5.1 问题描述与分析

问题二是研究旅客总体最短流程时间的问题，问题二在问题一的基础上加入了旅客换乘因素，在不考虑旅客乘坐捷运和步行时间的前提下，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。

5.2 数据处理

根据附件中提供的数据对航班情况进行处理，要求只选取 20 日到达或 20 日出发的航班和旅客进行分析。经过处理，共筛选出 1733 个旅客记录号，问题二基于这 1733 个旅客记录号和问题一的相关数据进行分析。

5.3 模型建立与求解

通过旅客记录号与登机口的关系，可以确定旅客是国内出发还是国际出发，或者是到达。

本问题在问题一的基础上加上旅客换乘因素，考虑最小化中转旅客的总体最短流

程时间，变量设为：a 代表国内或国外，b 表示航站楼 T 还是卫星厅 S，变量S_{ab}表示流程时间，我们希望流程时间越短越好。

建立目标函数为：$\min \sum(ST_j - PK_i\_ST_j + S_{ab})$

参考文献

【1】夏蔷薇.计算机仿真技术在枢纽机场中转流程研究中的应用[D].南京航空航天大学,2008.
【2】袁潘峰,孙震.基于仿真角度的中转旅客再值机流程优化[C].中国仓储物流创新与发展高峰论坛.2012.
【3】周覃龙.解决铁路中转旅客站内换乘问题的探讨[J].高速铁路技术,2016,7(3):16-18.
【4】潘佳月.白云机场二号航站楼国际中转流程优化研究[D].兰州大学,2016.
【5】袁姗.民航中转效率的现状与信息化改进策略研究——以南航客运中转调度中心为例[D].中山大学,2009.

附录

问题一：Python 程序

```
import csv
import pandas as pd
from pulp import LpProblem, LpMinimize, LpVariable, LpBinary, lpSum, value, LpStatus
gates_df = pd.read_csv('gates.csv', encoding='gbk')
pucks_df = pd.read_csv('input_data.csv', encoding='gbk')
tickets_df = pd.read_csv('tickets.csv', encoding='gbk')
gates_df['st_var'] = gates_df['登机口'].map(lambda x: LpVariable(x, cat=LpBinary))
pucks_df.index = pucks_df['飞机转场记录号']
gates_df.index = gates_df['登机口']
pucks_df = pucks_df.sort_values(by='到达时间')
wn = {'332': 'W', '333': 'W', '33E': 'W', '33H': 'W', '33L': 'W', '773': 'W', '319': 'N', '320': 'N', '321': 'N', '323': 'N', '325': 'N', '738': 'N', '73A': 'N', '73E': 'N', '73H': 'N', '73L': 'N'}
conn = []
for _, p_row in pucks_df.iterrows():
    for _, g_row in gates_df.iterrows():
        """匹配登机口和转场记录的到达和出发模型"""
        if p_row['到达类型'] in [v.strip() for v in g_row['到达类型'].split(',')] and p_row['出发类型'] in [v.strip() for v in g_row['出发类型'].split(',')] and wn[
                p_row['飞机型号']] == g_row['机体类别']:
                conn.append(
                    {'PK': p_row['飞机转场记录号'], 'ST': g_row['登机口'], 'key': "{}_{}".format(p_row['飞机转场记录号'], g_row['登机口']),
                     'con_var': LpVariable("{}_{}".format(p_row['飞机转场记录号'], g_row['登机口']), cat=LpBinary)})
conn_df = pd.DataFrame(conn)
conn_df.index = conn_df['key']
def opt(num):
    print("*********{}*************".format(num))
    prob = LpProblem("The coal opt", LpMinimize)
    print("***opt")
    """目标函数：求解最小值，登机口数减去转场记录数"""
    prob += lpSum(gates_df['st_var'])-lpSum(conn_df['con_var'])
    print("***st1")
    """限制条件 1"""
    for _, row in conn_df.iterrows():
        prob += row['con_var'] <= gates_df['st_var'][row['ST']], "st1/2/{}_{}".format(row['PK'], row['ST'])
    print("***st2")
    """限制条件 2"""
    for pk in pucks_df['飞机转场记录号']:
        prob += lpSum(conn_df[conn_df['PK'] == pk]['con_var']) <= 1, "st2/{}".format(pk)
```

```
    print("***st3")
    """限制条件 3"""
    prob += lpSum(conn_df['con_var']) <=303, "st3"
    print("***st4")
    """限制条件 4"""
    for ST in gates_df['登机口']:
        print(ST)
        pks = list(conn_df[conn_df['ST'] == ST]['PK'])
        for i in range(1, len(pucks_df)):
            if pucks_df.iloc[i]['飞机转场记录号'] not in pks:
                continue
            conn1 = conn_df['con_var']["{}_{}".format(pucks_df.iloc[i]['飞机转场记录号'], ST)]
            for j in range(0, i):
                if pucks_df.iloc[j]['飞机转场记录号'] not in pks:
                    continue
                if pucks_df.iloc[i]['到达时间'] - pucks_df.iloc[j]['出发时间'] < 45 * 60:
                    conn2 = conn_df['con_var']["{}_{}".format(pucks_df.iloc[j]['飞机转场记
录号'], ST)]
                    prob += conn1 + conn2 <= gates_df['st_var'][ST], "st4/{}/{}_{}".format(ST,
pucks_df.iloc[i]['飞机转场记录号'],
    pucks_df.iloc[j]['飞机转场记录号'])
    prob.writeLP("res/test_{}.lp".format(num))
    print("开始优化")
    prob.solve()
    status = str(LpStatus[prob.status])
    print("Status:", LpStatus[prob.status])
    print("正将优化结果写入文件。")
    def pre_data_write(prob):
        def write_res(res):
            with open('res/result_{}_{}.csv'.format(num, status), 'w', newline='') as f1:
                writer = csv.writer(f1)
                writer.writerows(res)
        res = []
        for v in prob.variables():
            # print(str(v.name) + " = " + str(v.varValue))
            if v.varValue is None:
                print(str(v.name) + " = " + str(v.varValue))
            if v.varValue is not None and int(v.varValue) > 1:
                print(str(v.name) + " ================== " + str(v.varValue))
            # if v.varValue is not None and int(v.varValue) > 0:
            res.append([str(v.name), str(v.varValue)])
        write_res(res)
    pre_data_write(prob)
    print("Total Cost of Transportation = ", value(prob.objective))
```

```
opt('1529')
# st_lp_var = LpVariable.dict("L", self.st_df['st_lp'], cat=LpBinary)
```