Modeling and Simulation 建模与仿真, 2022, 11(1), 40-50
Published Online January 2022 in Hans. http://www.hanspub.org/journal/mos
https://doi.org/10.12677/mos.2022.111004

Hans汉斯

基于蚁群算法在航班着陆调度方面的研究

霍运通，罗　颖，林　津，朱弈洁，刘颜东，冯小荣*

中国民航大学空中交通管理学院，天津

收稿日期：2021年11月12日；录用日期：2021年12月29日；发布日期：2022年1月6日

摘　要

随着我国民航运输业的高速发展，我国各大枢纽机场航班容量接近饱和，如何优化调度现有资源来对枢纽终端区待降落航班进行调度优化以减少航班延误成为重要任务。本文对终端区航班着陆问题进行建模，采用蚁群算法对问题进行求解，运用Python程序对蚁群算法进行仿真。本文对实验参数进行测试以寻求最优解，并将最终数据与先到先服务进行分析。

关键词

航班落地排序，蚁群算法，Python

Research on Flight Landing Scheduling Based on Ant Colony Algorithm

Yuntong Huo, Ying Luo, Jin Lin, Yijie Zhu, Yandong Liu, Xiaorong Feng*

School of Air Traffic Management, Civil Aviation University of China, Tianjin

Received: Nov. 12th, 2021; accepted: Dec. 29th, 2021; published: Jan. 6th, 2022

Abstract

With the rapid development of China's civil aviation transportation industry, the flight capacity of major hub airports in China is close to saturation. How to optimize the flight landing scheduling in the terminal area of the hub to reduce flight delay based on existing resources has become an important task. In this paper, the terminal area flight landing problem is modeled, and the ant colony algorithm is used to solve the problem, while the Python program is used to simulate the ant colony algorithm. In this paper, experimental parameters are tested to find the optimal solution, and the final data is compared with the data of first-come, first-served.

*通讯作者。

文章引用: 霍运通, 罗颖, 林津, 朱弈洁, 刘颜东, 冯小荣. 基于蚁群算法在航班着陆调度方面的研究[J]. 建模与仿真, 2022, 11(1): 40-50. DOI: 10.12677/mos.2022.111004

Keywords

Flight Landing Sequencing, Ant Colony Algorithm, Python

1. 引言

近年来随着我国航空业的快速发展，空中交通流量急剧增长，空中交通流量需求与有限的枢纽机场空域资源间的矛盾日益尖锐，高峰时刻造成大量航班在机场上空等待降落，严重影响了落地航班的调度效率。因此对终端区航班调度顺序进行优化提高枢纽机场终端区航班调度自动化水平已经成为民航业内人士的共识。国内外对终端区航班的调度问题都有着各自的进展与突破，但是根据目前的形势，航班落地优化调度仍然存在着很大的提升空间，这成为近几年在空域领域优化方面研究的热点。Briskom 和 Stolletz [1]通过局部探索进一步实现了航班的优化排序，两位学者采用用元胞自动机模型来模拟飞机着陆调度中的使用方法，以此更加直观的寻找较好的的飞机降落顺序，然后再通过局部优化的算法，从而计算得出一定数量航班排序的最优解。Gilbo 等人[2]，考虑到了单一机场跑道、空域容量的限制以及出入境交通战略管理的随机性等约束问题。文中绘制出了芝加哥奥黑尔机场的运力曲线，并进行分析，提出了有限固定容量和预期容量的概念。孟欣[3]等人为探究终端区航班调度的优化，开发出了机场终端区航班计算机排序辅助系统，该系统基于最大位置限制的遗传算法和免疫思想的粒子群算法，对终端区航班调度进行优化。

值得关注的是，在分析机场着陆调度问题时可以运用仿真算法的方式，基于该类算法构建出目标函数，从而计算求出优化解。马卫民，杨文娟，徐博[4]提出了一种基于最大位移限制的蚁群算法，该算法利用了蚁群算法高效的全局探索能力，较强的收敛能力等性质，并且结合了最大受限位置的约束，以此来确保在航班调度排序中存在的公平性和灵活性，能够为现实生活的空中交通流量管理提供理论依据和研究方法。Abdul (2014) [5]结合蜂群算法针对单跑道静态飞机的着陆调度问题进行了研究。王莉莉，胡畔[6]考虑航空器起飞所需的空域和机场容量限制等约束，通过遗传算法，在满足机场容量限制的前提下，探求能够降低总延误时间的优化解等。冯小荣、冯兴杰、王兴隆[7]提出了一种基于动态计算启发式信息的蚁群优化算法，该算法在一定程度上是对传统蚁群算法的一种优化与改进，此实验有效克服了传统蚁群优化算法实验中存在的不足，且得到的结果与基于时域的遗传算法和传统的蚁群算法得出的结果相比，具有较强的鲁棒性、搜索能力和较快的收敛速度。本文运用两个相邻航班中的后续航班最佳降落时间与前序航班的降落时间差定义启发式信息，采用设置了最大位移限制的蚁群算法对终端区航班着陆调度问题进行建模，在标准数据集，运用 python 语言对算法进行仿真实验，并将所得实验数据与先到先服务方案得到的数据进行比较分析。

2. 蚁群算法在航班着陆调度上的研究

2.1. 蚁群算法概述

蚁群系统是由意大利学者 Dorigo，Maniezzo 等人在 20 世纪 90 年代首先提出来的，在研究蚂蚁觅食的过程中，发现蚁群在路径寻觅方面似乎存在着一种很智能的行为。蚁群算法是模拟自然界中蚂蚁寻找

路径的方式，从而得出的一种仿生算法。

蚂蚁在运动过程中，能够在它所经过的路径上留下一种名叫信息素的化学物质，通过这种物质来给其他蚂蚁进行信息传递，并且蚁群在运动过程中也能够接收到这种物质，通过这样的方式来指引各自的运动方向。这种蚁群集体的行为是由大量的蚂蚁组成的，表现出蚂蚁在运动过程中，一种对信息的正反馈现象，在某一路径上走过的蚂蚁越多，后方的的蚂蚁选择该路径的概率也就越大。蚁群算法有着分布计算、信息正反馈和启发式搜索的特征，其本质上就是一种进化算法中的启发式全局优化算法。该算法同传统的路由算法比较起来，具有信息分布式性、动态性、随机性和异步性等特点，而这些特点正好能满足网络路由的需要及优化的需要。

2.2. 基于蚁群算法的航班着陆调度问题的数学模型

航空器产生间隔的主要原因是根据空气动力学原理，航空器在飞行过程中产生尾流，由于航空器的机型不同、轻重不同，尾流对其的影响的也随之不同。所以在研究落地航班的优化排序中，对尾流的考虑是必不可少的。根据尾流的不同得到不同类型飞机之间应保持的最小安全间隔(MSI, Minimum Security Interval)如表 1 所示：(本文所涉及的参数及航班间的最小间隔都是参考了 Beasley 等人[8]在民航飞机到达排序与调度问题(ASS)中定义的航班最小间隔)。

Table 1. Minimum safety interval for flights
表 1. 航班的最小安全间隔

MSI (*i*, *j*) (s)		后机着陆类型 *j*			
		1	2	3	4
前机着陆类型 *i*	1	96	200	181	228
	2	72	80	70	110
	3	72	100	70	130
	4	72	80	70	90

1 = 波音 747，2 = 波音 727，3 = 波音 707，4 = Mc Donnel Douglas DC9

建立基于蚁群算法的航班着陆模型所需的实验参数如表 2 所示。

Table 2. Experimental parameters
表 2. 实验参数

实验参数	
n	飞机的数量
Π	调度 n 架飞机的所有潜在调度序列的集合
|Π|	π 中包含的调度序列数
π	集合的调序序列
PLT (π_i)	航班预计着陆时间(Expected Landing Time)
ALT (π_i)	航班指定着陆时间(Appoint Landing Time)
MSI (i, j)	最小安全间隔
Ant_max	蚂蚁个数

Continued

Nc_max	最大迭代次数
Alpha	启发式信息素重要程度
Beta	启发因子重要程度
RHO	挥发因子
Q	矩阵增量
MPS	最大位移偏量(Maximum Displacement Offset)

定义一：启发式信息

启发式信息与第 i 架航班到第 j 架航班的延误间隔有关(假设航班不能提前到达)；

设航班 j 在航班 i 后落地，则航班 j 的指定着陆时间为：

$$\mathrm{ALT}\left(\pi_j\right)=\mathrm{Max}\left(\mathrm{PLT}\left(\pi_j\right),\mathrm{ALT}\left(\pi_i\right)+\mathrm{MSI}\left(\pi_i,\pi_j\right)\right) \tag{1}$$

两架落到航班之间的间隔应大于他们的 LTI，即：

$$\mathrm{ALT}\left(\pi_j\right)-\mathrm{ALT}\left(\pi_i\right)\geq\mathrm{MSI}\left(\pi_i,\pi_j\right) \tag{2}$$

启发式信息的计算为

$$\eta\left(\pi_i,\pi_j\right)=\frac{1}{\mathrm{ALT}\left(\pi_j\right)-\mathrm{ALT}\left(\pi_i\right)} \tag{3}$$

定义二：航班的选择概率的实现公式：

$$p_{ij}^{k}=\begin{cases}\dfrac{\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}}{\sum_{t\in N_i^k}\tau_{it}^{\alpha}\cdot\eta_{it}^{\beta}}, & j\in N_i^k\\ 0, & \text{其他}\end{cases} \tag{4}$$

定义三：对信息素矩阵的更新：

信息素 τ_{ij} 表示航班 i 降落后，对航班 j 降落的期望度；

我们在本次实验中选取与蚁群的信息素同步更新的方法。假设有 m 只蚂蚁，当这些蚂蚁各自都完成了所有的搜索之后，再统一对信息素矩阵进行更新。

$$\tau_{ij}=\left(1-\mathrm{RHO}\right)\cdot\tau_{ij}+\sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{5}$$

根据上述定义，对实验步骤进行描述：

步骤一：对最大蚂蚁数量，最大迭代次数，算法中所需要的排序的航班量，各种类型的航班之间降落应保持的尾流间隔，启发式信息等各类参数进行定义；

步骤二：定义一个虚拟的航班，与其他航班之间的距离为−10；

步骤三：计算出第一架航班与虚拟航班的启发式信息；

步骤四：根据启发式信息和信息素浓度计算出航班的选择概率；

步骤五：根据轮盘赌的方法选择下一架需要落地的航班，并记录选择的航班及延误时间；

步骤六：重复步骤三，直至所有的航班都被选择完成，从而计算出蚂蚁所经过路径对应的落地航班顺序以及总延误时间，航空器的总延误时间是每架航空器的实际降落时间与预计降落时间差的求和：

$$\mathrm{TTD}_{\pi}=\arg\min_{\pi\in\Pi}\sum_{i=1}^{n}\left[\mathrm{ALT}\left(\pi_i\right)-\mathrm{PLT}\left(\pi_j\right)\right] \tag{6}$$

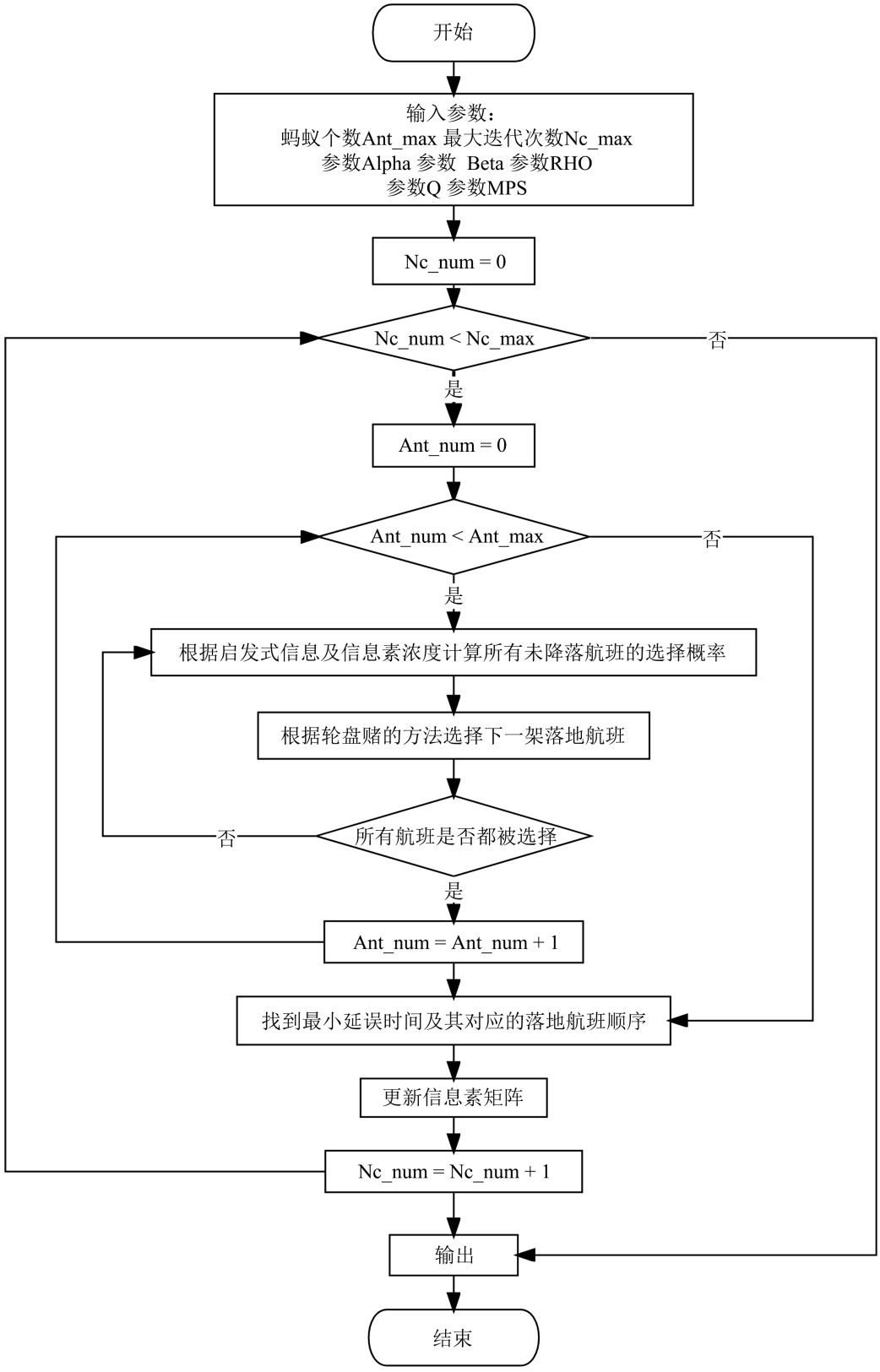

Figure 1. Program framework
图 1. 程序框架

步骤七：更新信息素矩阵 τ_{ij}，选择第二只蚂蚁重复步骤二，以此类推计算出每只蚂蚁所对应的总延误时间，继续迭代，直到达到最大迭代次数；

$\Delta\tau_{ij}^{k}$ 表示第 k 只蚂蚁在(i, j)上所释放的信息量，同时也代表着航空器 j 跟着航空器 i 降落，更新的信息素矩阵由：

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k}, & \text{航班}j\text{紧接航班}i\text{降落}\\ 0, & \text{其他}\end{cases} \tag{7}$$

计算出。L_k 表示第 k 只蚂蚁构造的降落顺序总的延迟时间。

步骤八：对比得到的结果，输出最小延误时间以及对应的落地航班顺序；

落地航班的前后的调动还应不大于设置的最大位移偏量 MPS，当 MPS 为 0 时，即先到先服务。

$$\text{abs}\left(\pi_i - L\pi_i\right)\le \text{MPS} \tag{8}$$

$L\pi_i$ 为调动后的航班位次。

通过蚁群算法，及设置的各类参数，结合对航班优化排序调度问题的研究，进行如图 1 所示的框架建立，可以更直观的展现出该算法在所研究问题中的应用。并通过 Python 程序对框架功能进行实现。

3. 实验数据分析

3.1. 实验参数的确定

本文所应用的 30 架航班先到先服务的航班降落顺序、机型以及着陆航班的预计着陆时间等数据选取自 Zhan ZH，Zhang J，Liu O 等人[9]的实验数据。本文的所有实验皆通过 Python 编程的方式得以实现。

首先控制变量法定义进行实验时所要使用各类参数的取值。对 Alpha 的定义采用的方法是：任意给定参数 Beta = 1.0，蚂蚁数量为 200，迭代次数为 50，参数 RHO = 0.1 和参数 Q = 10,000 的值，再对 Alpha 分别取 0.1、0.2、0.3、……、1.0 时，进行十次实验，计算每个取值下十次实验中所获得的总延误时间的最小值，求其平均，再对所求得的十个平均值进行比较，选平均总延误时间最小时对应的 Alpha 为继续进行实验选取参数 Alpha 的最优解。

根据计算汇总出来的数据，对每个 Alpha 取值下对应的十次实验得到的结果求平均延误时间，对应的平均值绘制如图 2 所示的折线统计图。

根据图 2 中折线的走势，可以清晰看出，Alpha 的取值对延误时间的影响是趋于单调的，在 0.1~1.0 之间，Alpha 的取值越大，计算得到的总延误时间反而越小，而当 Alpha = 1.0 时，总延误时间的平均值最小，获得的实验结果最优，故选取参数 Alpha = 1.0 继续进行试验。

确定好参数 Alpha 的值，随即对参数 Beta 进行确定，实验采用和确定参数 Alpha 相同的方式，给定参数 Alpha = 1.0，蚂蚁数量为 200，迭代次数为 50，参数 RHO = 0.1 和 Q = 10,000，对 Beta 分别取 0.5、1.0、……、5.0 时，进行十次实验。根据实验数据得到如图 3 所示的折线统计图。

根据如图 3 中所示的内容，在 0.5~5.0 之间，当 Beta 取 1.5 时，图中折线到达谷点，即表示所计算到的总延误时间越小，而当 Beta 继续增大，平均总延误时间反而开始逐渐变大，表明在参数 Beta 的选择中，至少存在一个最优值，所以在接下来的实验中选取参数 Beta 最优值 1.5 进行计算。

确定了 Alpha 及 Beta 的值，其次再对挥发因子 RHO 的选择进行讨论，在已给定参数 Alpha = 1.0、Beta = 1.5、蚂蚁数量为 200、迭代次数为 50、Q = 10,000 的条件下，RHO 分别取 0.1、0.2、0.3、……、1.0 时，进行十次实验。对得到的数据进行汇总，取十次实验的平均值得到图 4 所示的折线图。

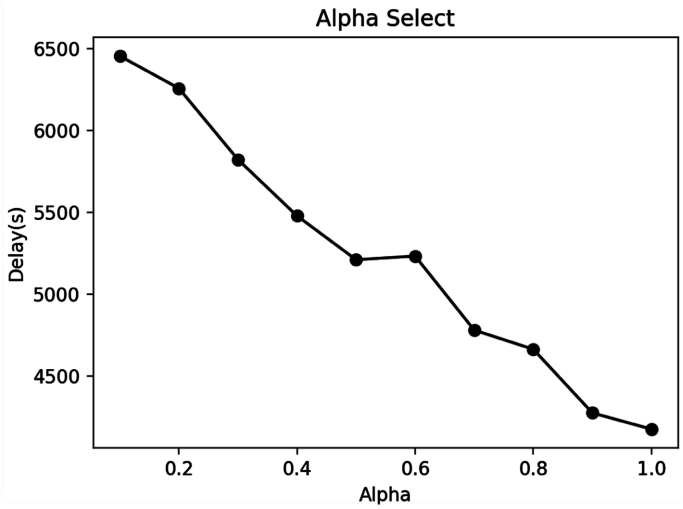

Figure 2. Determination of the Alpha
图 2. Alpha 的测定

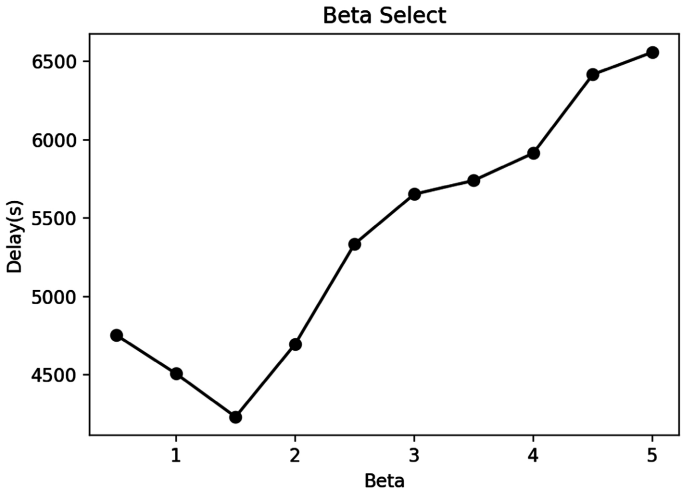

Figure 3. Determination of Beta
图 3. Beta 的测定

参数 RHO 是挥发因子，当 RHO 等于 0 时，表示不会遗忘信息素矩阵遗传信息，而当 RHO 等于 1 时，表示在更新信息素矩阵中遗传信息被完全忘记，相当于蚂蚁在航班排序的选择中做随机选择。根据图 4 所示折线图，可以直观的看出，除了 RHO = 1.0 时计算得到的总延误时间偏大以外，其他数值对延误时间的结果影响并不显著，趋于稳定水平，所以对于参数 RHO 的取值，只要不是 1.0，其余数值均可，在本次实验中，选取参数 RHO = 0.1 进行计算。

确定了 Alpha、Beta 及 RHO 的值之后，下面对不同最大位移限制下的收敛性进行讨论。在给定参数 Alpha = 1.0、Beta =1 .5、蚂蚁数量为 200、Q = 10,000 的条件下，MPS 分别取 1、2、3、4 时，进行十次实验。对得到的数据进行汇总，取十次实验的平均值得到图 5 所示的折线图。由图 5 可知不同的 MPS 条件下得到平均延误时间最优解所对应的迭代次数相应减小，计算得到最优解在越发快捷，且蚂蚁数量一定时，迭代次数不断增加，模拟仿真得到的最小延误时间也逐渐减小，慢慢收敛，当迭代次数到达 200 以上，最小延误时间变化逐渐趋于一个稳定水平。这充分体现出了蚁群算法的收敛性和稳定性。

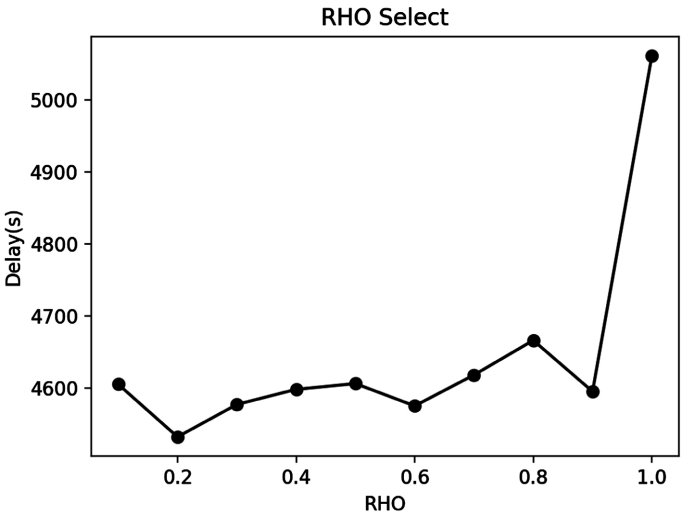

Figure 4. Determination of the RHO
图 4. RHO 的测定

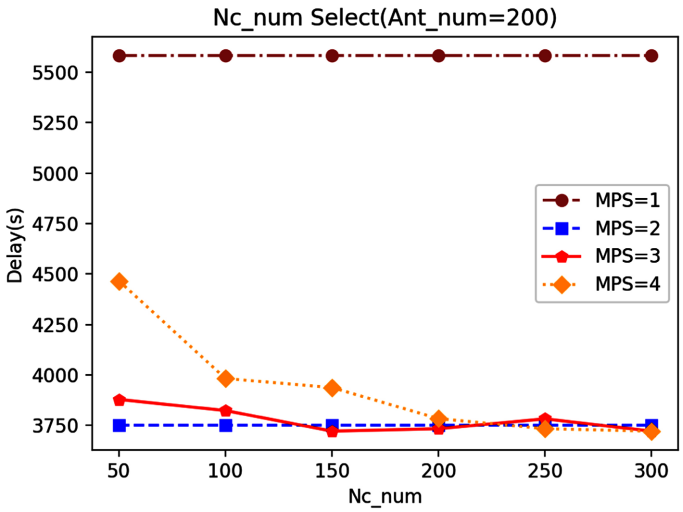

Figure 5. The rate of convergence under the different MPS
图 5. 不同 MPS 下的收敛速度

3.2. 实验数据对比

利用与先到先服务相同的航班顺序对实验进行测试，实验参数的选择为：Alpha = 1.0、Beta = 1.5、RHO = 0.1、Q = 10,000、蚂蚁数量为 200、迭代次数为 200、最大位移限制分别选取 MPS = 1、2、3、4。每个最大位移限制下实验十次所得数据的平均值如表 3 所示并由表 3 中数据制成如图 6 所示的折线图。

根据表 3 中数据和图 6 中的折线图分布，发现对 30 架航班，当 MPS = 1 时，通过蚁群算法计算得到的总延误时间比先到先服务方式得到的总延误时间减少了 30.4%；当 MPS = 2 时，总延误时间减少了 53.3%；当 MPS = 3 时，总延误时间减少了 53.6%且 ALT 的分布与 MPS = 2 时无太大波动；当 MPS = 4 时，ALT 分布与 MPS = 3 时完全相同。由此可以得出结论，MPS 的取值在一定范围内不断增大时，得到的优化结果会越来越好，优化效果的增长率在不断减小，这表现出当 MPS 达到一定的值后，总延误时间会趋于一个稳定的范围，但整体效果依然会优于先到先服务的结果。

Table 3. Experimental parameters under different MPS were compared with first-come service
表 3. 不同 MPS 下的实验参数与先到先服务对照

FCFS			MPS = 1			MPS = 2			MPS = 3			MPS = 4		
编号	种类	PLT	编号	种类	ALT	编号	种类	ALT	编号	种类	ALT	编号	种类	ALT
1	1	0	1	1	0	1	1	0	1	1	0	1	1	0
2	1	79	2	1	96	2	1	96	2	1	96	2	1	96
3	1	144	3	1	192	3	1	192	3	1	192	3	1	192
4	2	204	5	1	288	5	1	288	5	1	288	5	1	288
5	1	264	4	2	488	6	1	384	6	1	384	6	1	384
6	1	320	7	2	568	4	2	584	7	2	584	7	2	584
7	2	528	6	1	640	7	2	664	4	2	664	4	2	664
8	1	635	8	1	736	9	2	744	9	2	744	9	2	744
9	2	730	10	2	936	10	2	824	10	2	824	10	1	824
10	2	766	9	2	1016	8	1	896	8	1	896	11	1	896
11	1	790	11	1	1088	11	1	992	11	1	992	12	1	992
12	1	920	12	1	1184	12	1	1088	12	1	1088	8	2	1088
13	3	1046	13	3	1365	13	3	1269	13	3	1269	13	3	1269
14	4	1106	15	2	1465	16	2	1369	15	2	1369	16	2	1369
15	2	1136	14	4	1575	15	2	1449	16	2	1449	15	2	1449
16	2	1166	17	2	1655	14	4	1559	17	2	1529	17	2	1529
17	2	1233	16	2	1735	17	2	1639	14	4	1639	14	4	1639
18	1	1642	18	1	1807	18	1	1711	18	1	1711	18	1	1711
19	1	1715	19	1	1903	19	1	1807	19	1	1807	19	1	1807
20	3	1770	20	3	2084	20	3	1988	20	3	1988	20	3	19
21	1	2074	21	1	2156	21	1	2074	21	1	2074	21	1	2074
22	1	2168	22	1	2252	22	1	2170	22	1	2170	22	1	2170
23	4	2259	24	2	2452	23	4	2398	23	4	2398	23	4	2398
24	2	2427	23	4	2562	24	2	2478	24	2	2478	24	2	2478
25	1	2481	25	1	2634	25	1	2550	25	1	2550	25	1	2550
26	2	2679	26	2	2834	26	2	2750	26	2	2750	26	2	2750
27	3	2883	27	3	2904	27	3	2883	27	3	2883	27	3	2883
28	2	2982	28	2	3004	28	2	2983	28	2	2983	28	2	2983
29	1	3046	29	1	3076	29	1	3055	29	1	3055	29	1	3055
30	1	3091	30	1	3172	30	1	3151	30	1	3151	30	1	3151
总延误：8027(*s*)			总延误：5583(*s*)			总延误：3751(*s*)			总延误：3721(*s*)			总延误：3721(*s*)		

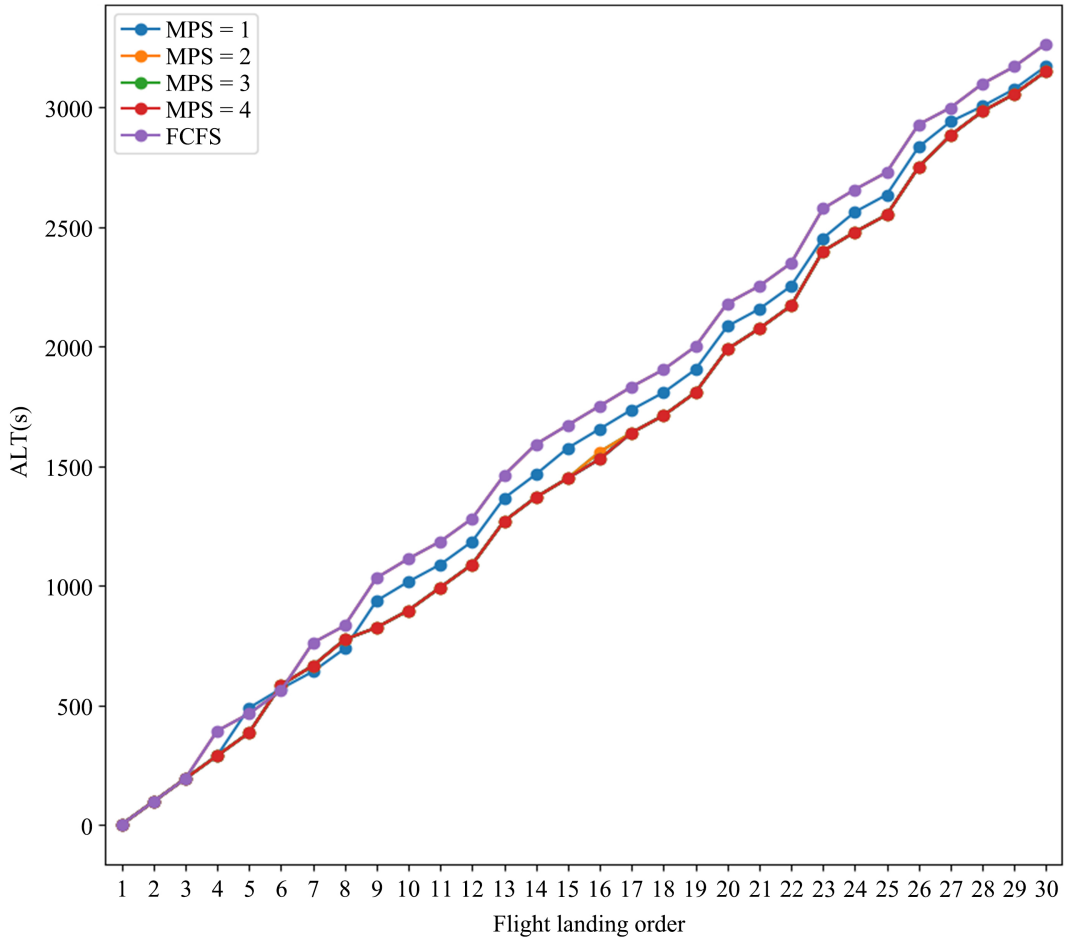

Figure 6. Distribution of ALT under the different MPS
图 6. 不同 MPS 下 ALT 的分布情况

4. 结论与展望

本文是通过蚁群算法较强的全局探索能力和较好的收敛能力来研究解决航班着陆调度优化的问题。首先对蚁群算法各项参数的值进行确定，再将蚁群算法与最大受限位移相互结合，计算得到的优化结果与先到先服务的结果进行比较，观察分析优化后的结果。跟先到先服务相比，能够有效地减少航班的总延误时间，其算法得出的优化后的航班降落顺序可以给管制员提供一个参考，有效减小了管制员的管制压力，同时降低了航空公司的经济损失。

在本文中只利用蚁群算法对航班着陆调度进行了简单的研究，但航班着陆调度问题充满了不确定性，是一个复杂多变的问题，需要考虑到的条件是多方面的，因此对于未来关于此问题的研究，还需要进行更深层次的研究与探索。本文只针对单跑道情况进行研究，未曾考虑多跑道以及各跑道构型的情况。因此之后的研究可以针对多跑道的情况以及不同的跑道构型改进该问题。本文在对该问题进行研究时提出了一个前提条件，航班实际着陆时间不允许早于预计着陆时间，但在实际情境中是允许提前于预计着陆时间的，因此未来研究中需要考虑到这一因素的影响。

参考文献

[1] Briskom, D. and Stolletz, R. (2014) Aircraft landing problems with aircraft classes. *Journal of Scheduling*, **17**, 31-45.

https://doi.org/10.1007/s10951-013-0337-x

[2] Gilbo, E.P. (1997) Optimization of Air Traffic Management Strategies at Airports with Uncertainty in Airport Capacity. *IFAC Proceedings Volumes*, **30,** 35-40. https://doi.org/10.1016/S1474-6670(17)43797-9

[3] 孟欣. 智能算法在动态航班着陆调度中的应用[D]: [硕士学位论文]. 天津: 中国民航大学, 2012.

[4] 马卫民, 杨文娟, 徐博. 基于受限位移约束的蚁群算法在航班着陆调度问题中的应用研究[J]. 管理工程学报, 2016, 30(1): 191-196.

[5] Abdul-Razaq, T.S. and Ali, F.H. (2014) Hybrid Bees Algorithm to Solve Aircraft Landing Problem. *Journal of Zankoy Sulaimani*, **2014**, 157-161. https://doi.org/10.1504/IJADS.2016.081394

[6] 王莉莉, 胡畔. 基于容流匹配的进离场航班调度优化模型和算法[J]. 南京航空航天大学学报, 2015, 47(6): 827-832.

[7] Feng, X.R., Feng, X.J. and Wang, X.L. (2016) An Ant Colony Optimisation Method Based on Pruning Technique for the Aircraft Arrival Sequencing and Scheduling Problem. *International Journal of Applied Decision Sciences*, **9**, 333-347. https://doi.org/10.1504/IJADS.2016.081394

[8] Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M., *et al*. (2000) Scheduling Aircraft Landings—The Static Case. *Transportation Science*, **34**, 180-197. https://doi.org/10.1287/trsc.34.2.180.12302

[9] Zhan, Z.H., Zhang, J., Liu, O., *et al*. (2010) An Efficient Ant Colony System Based on Receding Horizon Control for the Aircraft Arrival Sequencing and Scheduling Problem. *IEEE Transactions on Intelligent Transportation Systems*, **11**, 399-412. https://doi.org/10.1109/TITS.2010.2044793