基于分组的MPS进近航班着陆调度算法研究

刘 洪[1]，杨红雨[1]，彭莉娟[2]

(1. 国家空管自动化系统技术重点学科实验室，四川大学计算机学院 成都 610064;
2. 西南科技大学计算机科学与技术学院 四川 绵阳 621010)

【摘要】讨论了机场终端区到达航班流的着陆调度规划问题。以航班总延误时间最小为目标函数，考虑了复杂的空中管制约束，提出了基于分组的MPS为1的隐枚举排序算法。该算法考虑了空中交通管制的多种约束条件，首先，分航路对航班进行分组；其次，根据约束条件初始化位置许可矩阵；然后建立解空间树搜索最优解。通过边界条件选取、无效分支判断、次优序列淘汰的设计提高了算法求解速度。结合真实数据，用计算机仿真实验对该算法进行了验证，结果表明，该算法能满足复杂空中交通管制条件下的各种限制约束，有效减少交通延误和提高空域利用率。

关 键 词 着陆调度; 空中交通管制; 隐枚举; MPS
中图分类号 TP391.9 文献标志码 A doi:10.3969/j.issn.1001-0548.2013.04.016

Study of MPS Algorithm Based on Grouped Scheduling to Approach Aircraft Landing Scheduling Problem

LIU Hong[1], YANG Hong-yu[1], and PENG Li-juan[2]

(1. National Key Laboratory of Air Traffic Control Automation System Technology, College of Computer Science, Sichuan University Chengdu 610064;
2. School of Computer Science and Technology, Southwest University of Science and Technology Mianyang Sichuan 621010)

Abstract The approach aircraft landing scheduling problem in the terminal area is discussed in this paper. Taking the flights' minimum total delay as the objective function, this paper proposes an implicit enumeration sorting algorithm with packet-based max position shift (MPS) = 1. It takes into account a variety of air traffic control constraints. Firstly the flights in same route bound to group lock, and then the position-matrix is initialized according to constraints, finally the solution space tree is built to search the optimal solution. The computing speed of the algorithm is improved effectively by selecting the boundary conditions, judging the invalid branch, and eliminating second-best series out. The results indicate that the algorithm can apply to complicated air traffic control (ATC) environment, and can greatly reduce the delay and increase airspaces' availability.

Key words approach aircraft landing scheduling; ATC; implicit enumeration; max position shift

文献[1]首次提出MPS(max position shift)思想，即优化序列中的飞机位置不能是任意的，和最初的FCFS序列相比，其最大位移数不应超过某一值(即MPS为1或者2)对管制员来说才是可行的。文献[2]采用隐枚举算法对能满足现实复杂约束的实时算法进行了研究，并应用在CTAS的验证版本中，通过搜寻进港航班流的类型组合，与预存的组合进行比较，然后确定一个新的序列，使同类飞机在满足MPS=1限定的前提下，得以聚类绑定。文献[3]研究了着陆航班调度MPS算法，在MPS=1的位置约束下，以隐枚举算法为基础，对到港航班进行排序，其研究重点在解空间的搜索和剪枝方式上。在对到港航班排序进行绑定时，通常考虑航班的机型，即同类机型加以绑定。在实际的管制操作中，管制员通常将同一航路、距离较近的航班视为整体，不允许插入另外的航班[4-6]。本文基于此，提出了一种基于分组的MPS=1隐枚举算法。在算法中，首先对参与排序的航班进行分组，综合考虑了航班的机型和航班之间的间距因素，同组航班在排序过程中被视为一个整体，然后在MPS=1的位置约束条件下，采用隐枚举算法对目标函数求解。

1 MPS=1隐枚举算法介绍

MPS=1隐枚举算法是限定飞机的位置只能在初

收稿日期：2011－09－19；修回日期：2012－05－07
基金项目：国家863项目(2012AA011804)
作者简介：刘洪(1978－)，男，博士生，主要从事多源信息融合、空中交通管制理论方面研究.

始位置附近最适当的位置上，它只能与它前面或后面一架飞机交换位置[3]。若用解空间来表示，则可理解为：设原集合 E 中有 n 个元素，其对应的优化集合为 S，图1表示了从集合 E 到集合 S 的映射递归过程。设从集合 E 到集合 S 的解空间为 G_n。若选定 S 中的第1个元素为 E 中的第1个元素，则 S 中的第2个元素可以是 E 中的第2个或第3个元素，这样得到图中所示的 G_{n-1}。若选择 S 中的第1个元素为 E 中的第2个元素，则 S 中的第2个元素只能是 E 中的第1个元素，这样得到图中的 G_{n-2}。

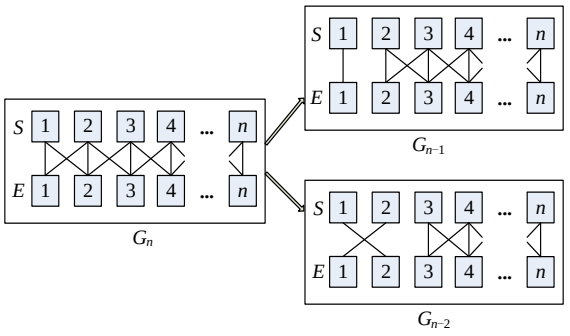

图1 MPS=1解空间示意图

这是一个数学上称为Fibonacci的序列，其递归方程为：

$$G_n = G_{n-1} + G_{n-2} + c \tag{1}$$

其解为 $G_n = c(1+\sqrt{5}/2)^n$，这是一个计算量呈指数增加的递归计算。

2 算法模型

2.1 变量描述

1) 集合 $R=\{i:1\leqslant i\leqslant u\}$：终端区区域内的航路集合。

2) 集合 $G=\{i:1\leqslant i\leqslant m\}$：航班分组的集合。$G_i$ 表示一个航班分组，其详细定义为：G_i = {gNum, fCount, fID[2], route, firstArTime, lastArTime, originalArTime}。其中，gNum 为分组 G_i 的组号；fCount 为 G_i 的组内航班数量；fID[2] 为 G_i 的组内每架航班的ID值；route 为 G_i 的航班所在航路，同一个分组的航班来自一条航路，$\text{route}\in R$；firstArTime 为分组 G_i 中第一架航班到达汇聚点的时间，随着分组在序列中的位置不同而改变；lastArTime 为分组 G_i 中最后一架航班到达汇聚点的时间，随着分组在序列中的位置不同而改变；originalArTime 为分组 G_i 中第一架航班到达汇聚点的原始时间，不随组在序列中的位置不同而发生改变，用于记录原始时间。

3) 集合 $F=\{j:1\leqslant j\leqslant n\}$：参与排序的航班序列。$F_j$ 表示一个航班，定义 F_j ={ID, Eta, Sta, fType, route, gNum, fixArTime, minSta, maxSta}，其中，ID 为航班 F_j 的标识；Eta 为航班 F_j 的预计到达时间；Sta 为航班 F_j 的分配到达时间；fType 为航班 F_j 的机型；route 为 F_j 所在的航路，$\text{route}\in R$；gNum 为航班 F_j 所在的分组号；fixArTime 为航班 F_j 到达汇聚点的时间；$\min\text{Sta}$ 为航班 F_j 允许的最早到达时间；$\max\text{Sta}$ 为航班 F_j 允许的最晚到达时间。

4) ts_{ij} 为航班 F_i 与航班 F_j 的安全间隔要求，其中，航班 F_i 为前机，航班 F_j 为后机。

5) Δs 为安全裕度。管制员的实际操作中，为了安全，会适当地增加两架航班的间隔[7-8]。

6) ε 为航班进行分组加锁绑定的间隔门限值。若同一航路上两架航班间隔小于 $\text{ts}_{ij}+\varepsilon$ (考虑不同机型的不同间隔)，则将其分为一个分组。

7) **POSM** =$[\text{posm}_{ij}]_{m\times 3}$ 是位置许可矩阵。表示分组 G_i 存在的位置。行表示分组，列表示分组可出现的位置，用 posm_{ij} =0表示 G_i 不能出现在该位置，$\text{posm}_{ij}=k$，$k\neq 0$ 表示分组 G_i 许可出现在位置 k。

2.2 约束条件

1) $\forall G_i\in G$，满足$1\leqslant G_i.\text{fCount}\leqslant 2$，$i\in[1,m]$。每个分组包含的航班数量至少为1，且不超过2。分组中的所有航班作为一个整体参与排序，中间不允许插入。

2) $F_j.\min\text{Sta}\leqslant F_j.\text{Sta}\leqslant F_j.\max\text{Sta}$，$j\in[1,n]$。每架航班 F_j 排序后的到达时间应能满足飞机的性能要求[9]，应在其最早和最晚到达时间的范围内。

3) $G_{i+1}.\text{firstArTime} - G_i.\text{lastArTime}\geqslant \text{ts}_{i(i+1)}+\Delta s$，$i\in[1,m]$。$G_{i+1}$ 中第一个航班 G_i 中后一个航班到达汇聚点的时间应满足管制间隔要求。

4) 若 $G_i.\text{route}=G_j.\text{route}$，那么 $G_i.\text{lastArTime} < G_j.\text{firstArTime}$，若前后两个分组在同一航路上，则不允许超越。

2.3 目标函数

隐枚举算法是通过创建一棵序列树，然后搜索解决的方案[3]。树的每一个节点代表一个分组，树的分支是参与排序的下一个分组，树的深度则应等于分组数。隐枚举算法是一种深度优先算法，树首先进行深度搜索，当搜索到结点时，则得到一个序列方案，目标则是寻求最优的序列方案。

排序问题在数学上可表述为某一目标函数的最小化问题[9-10]，对于进近排序，其目的是让所有飞机总延误成本最少[11]，因此选择航班分组在汇聚点处的总延误时间最小作为目标函数：

$$O(t)=\min\sum_{i=1}^{m}(G_i.\text{firstArTime}-G_i.\text{originalArTime}) \tag{2}$$

3　模型求解

3.1　满足约束条件

1) 本文采用位置许可矩阵来标识航班分组可以在排序序列中的位置。若不满足任何约束条件，航班分组生成的序列树将是一个多叉树，第一层节点有 n 个分支，其下一层节点有 $n-1$ 个分支，以此类推。研究和经验表明，MPS为1或2，其排序结果是可以接受的，本文采用MPS=1作为位置约束条件，解空间树为一颗二叉树，产生一个m行3列的位置许可矩阵：

$$\mathbf{POSM}_{ij}=\begin{cases} i+j & \\ 0 & i=0,j=0 \\ 0 & i=m,j=3 \end{cases} \tag{3}$$

矩阵的第1列为各分组的前一个位置，第2列为原始位置，第3列为各航班分组的后一个位置。

2) 若分组 G_i 和 G_{i+1} 同属一条航路，则分组 G_{i+1} 不允许超越分组 G_i，通过修改位置许可矩阵来实现：令 **POSM** [$i\times3+2$]=0，**POSM** [$(i+1)\times3$]=0。

3) 若管制员输入指定的航班分组 G_i 的位置不能改变，则令 **POSM** [$i\times3$]=0，**POSM** [$i\times3+2$]=0。

3.2　提高求解速度

1) 边界条件的选取。

树进行深度搜索，当搜索到层次为分组总数的结点时，则得到一个序列方案，以及该序列对应的目标函数值。最好的情况是开始就能找到一个接近于最优解的目标函数值，因此，生成树时让第一个分支按时间先后的序列，则得到先来先服务的序列方案，并将其目标函数值作为边界条件。

2) 无效分支的判断。

如图2所示，一个m=5的航班分组序列生成的树中，有几个无效的分支：序列1-2-4-5、1-3-4-5和1-3-4，这是由于位置约束MPS=1，对于序列1-2-4-5中，分组3只能出现在树的2、3、4层中，而不能在第5层出现，由此序列1-2-4-5为无效。同理1-3-4-5和1-3-4也无效。随着m值的增大，解空间树中的无效分支树也将大量地增加。因此，考虑无效分支的剪枝，增加判断条件：航班分组 G_i 若在树的第 $i-1$、i、$i+1$ 层未出现，则该分支无效。具体判断方法为：设当前插入的节点值为 i，插入的位置为树的第 k 层，判断 i != $k-1$ 是否成立，若成立，则判断该节点的父节点与祖父节点是否出现了值 $k-1$，若未出现，则为无效分支。

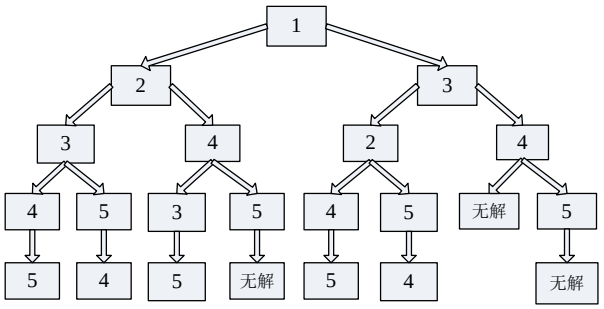

图2　m=5航班分组序列生成树示意图

3) 次优序列的淘汰。

序列的总延误时间在生成树的同时计算，即每插入一个节点就计算这一序列的总延误时间，由于求解的是最小化问题，若某一局部序列的总延误时间大于之前求得的边界值，则该序列只能为次优序列，可以被丢弃掉。

3.3　算法详细步骤

首先，算法根据航班信息生成航班分组 G，接着根据 G 和约束条件，创建位置许可矩阵，然后根据位置许可矩阵生成解空间进行求解。

3.3.1　航班分组子算法

输入：航班集合 F，间隔门限值 ε。输出：航班分组集合 G。

1) 取航班集合 F 中的航班 F_j。

2) 遍历航班 F_j 的后继，找出同一条航路的后继 F_{j+k}，　若 (F_{j+k}. fixArTime − F_j. fixArTime$_j$) < ($\text{ts}_{j(j+k)}+\varepsilon$)，则判断 F_j.gNum；若 F_j.gNum =0，则表示 F_j 未分组，转步骤3)，若 F_j.gNum !=0，则表示 F_j 已分组，转步骤4)。

3) F_j 与 F_{j+k} 分为一个组，加入 G，转步骤5)。

4) 若 F_j.fCount <2，则将航班 F_{j+k} 加入到该分组，否则转到步骤5)。

5) 若 $j+k<n$，令 k ++，转步骤2)，否则到步骤6)。

6) 若 F_j.gNum =0，即遍历完 F_j 的后继，没有可与其一组的航班，那么航班 F_j 单独作为一个分组，加入到集合 G 中，否则转到步骤7)。

7) 若 $j<n-1$，令 j ++，转步骤1)。否则，算法结束。

3.3.2　位置许可矩阵初始化子算法

输入：航班分组集合 G，管制员指定分组 G'（即

G' 的位置不允许改变）。输出：位置许可矩阵 **POSM** 。

1) 初始化变量 i =1，转到步骤2)。

2) 令 **POSM** [$i\times3$]= i –1， **POSM** [$i\times3$ +1]= i , **POSM** [$i\times3$ +2]= i +1；若 i <= m ，则 i ++，转步骤2)；否则，转步骤3)。

3) 令 **POSM** [$m\times3$ +2]=0， i =1，转步骤4)。

4) 若分组 G_i . route = G_{i+1} . route ， 则 **POSM** [$i\times3+2$] =0， **POSM** [$(i+1)\times3$]=0；若分组 G_i . gNum = G' . gNum ， 则 **POSM** [$i\times3$]=0， **POSM** [$i\times3$ +2]=0；若 $i<m$ ， i ++，转到步骤4)；否则，算法结束。

将航班分组和初始化位置许可矩阵后，根据分组信息和位置约束信息生成解空间树，搜索最优序列。生成解空间树是一个递归算法，按深度优先生成树的节点，建立树的同时，计算各个序列的延误值，并记录最优路径，当树生成完毕，输出最优路径。

首先，引入几个变量：

1) 树节点 node ={ data ， levelNum , delayTime , lChildNode， rChildNode， parentNode }，其中，data 为节点值，在文中表示节点代表的分组号；levelNum 为节点所在树的层数；delayTime 为从树的根节点到当前节点的路径上航班分组的总延误时间；lChildNode 为节点的左子节点； rChildNode 为节点的右子节点； parentNode 为节点的父节点；

2) path 为记录最优序列的路径，初始时为先来先服务的顺序；

3) min Delay 为一个完整序列的最小延误值。

3.3.3 子树生成子算法

输入：节点层次树 levelNum ， 父节点 parent ；输出：无。

1) 若 **POSM** [levelNum $\times3$]!=0 且 **POSM** [levelNum $\times3$]< parent . data ， 调用子节点生成子算法，输入参数为(levelNum, **POSM** [levelNum $\times3$], parent ,1)。

2) 若 **POSM** [levelNum $\times3+1$]!=0 且 **POSM** [levelNum $\times3+1$]！ = parent . data ， 判断 parent 节点是否有左儿子。若父节点没有左儿子，调用子节点生成子算法，输入参数为(levelNum， **POSM** [levelNum $\times3+1$], parent ,1)；否则调用子节点生成子算法，输入参数为(levelnum， **POSM** [levelNum× 3+1], parent ,0)。

3) 若 **POSM** [levelNum $\times3+2$]!=0 且 **POSM** [levelNum $\times3+2$]！ = parent . data ， 调用子节点生成子算法，输入参数为(levelNum ,**POSM** [levelNum× 3+1], parent ,0)。

3.3.4 子节点生成子算法

输入：节点层次树 levelNum ， 节点值 data ， 父节点 parent , 左子标识 lChildFlag (lChildFlag =1,生成左儿子; lChildFlag =0,生成右儿子）；输出： path 。

1) 新建一个节点 node ， node . data = data ， node.levelNum=levelNum,node.parent=parent ， node . lChildNode = null ， node . rChildNode = null ，计算从根节点到当前节点的延误 node . delaytime 。

2) 若 data ！ = levelNum –1(无效分支的判断)，转到到步骤3)；否则，转到步骤5)。

3) 判断父亲节点或祖父节点值是否等于 levelNum –1，若相等： lChildFlag =1， 则 parent . lChildNode = node ； lChildFlag =0 ， 则 parent . rChildNode = node ，转到步骤4)。

4) 若 node . levelNum = m (为一个完整序列的末节点)，则判断 node . delayTime < min Delay ，若成立，path =当前路径， min Delay = node . delaytTime ， 转到步骤6)。

5) 若 lChildFlag =1， 则 parent . lChildNode = node ； lChildFlag =0 ， 则 parent . rChildNode = node ，转到步骤4)。

6) 若 node . levelNum < m ， 调用子树生成子算法，输入参数为(levelNum +1, node)。

递归算法执行完毕后， path 记录的就是最优序列。各个子算法的调用流程如图3所示。

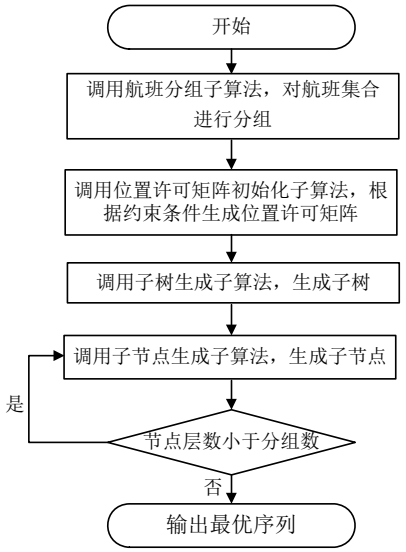

图3 各子算法调用的流程图

4　仿真实验与结果分析

仿真实验的数据根据某机场的到港航班数据得到。该机场进场航路由3条航线组成，在最后进近段汇聚成一点进行降落。对20架航班组成的航班流进行排序，得到的排序结果如表1所示。

表1　20架航班序列的优化排序结果

序号	航路	过点时间	机型	算法1	算法2	算法3
1	1	0	H	1	1	1
2	1	110	H	2	2	2
3	3	170	H	3	3	3
4	3	300	M	4	4	4
5	3	400	M	5	5	5
6	2	440	H	6	6	6
7	2	600	M	8	8	8
8	1	600	H	7	7	9
9	1	700	H	10	10	7
10	3	800	M	9	9	10
11	1	820	M	12	11	11
12	1	900	H	11	12	12
13	2	900	H	14	13	13
14	3	960	M	13	15	15
15	2	1 040	H	15	14	17
16	1	1 050	M	16	16	14
17	2	1 160	M	17	17	16
18	2	1 300	H	19	19	19
19	1	1 300	M	18	18	18
20	1	1 450	H	20	20	20
总延时/s				14 399	15 844	15 658
延误时间标准差				485.25	486.92	540.89
解空间规模				6 765	936	288

表1中3种算法分别是：1) 算法1：满足MPS=1约束的算法[2]；2) 算法2：满足MPS=1约束，且满足同一航路不允许超越约束的算法；3) 算法3：即本文算法。使用算法1对航班序列优化排序，其产生的解空间规模最大。算法2是在算法1的基础上考虑了同一航路不允许超越，其解空间规模大大减少。本文算法在算法2的基础上进行了改进，首先进行分组加锁绑定再排序，可以看出，算法3的解空间规模最小。而且随着航班序列规模的增大，算法的解空间规模呈指数增长。如果参与排序的航班序列为40架次，算法1的解空间将达到102 334 155，算法2的解空间将达到299 520，而算法3仅为71 200，远低于前两者。

假设一次空中等待时间为固定值180 s，如果采用常用的FCFS算法，20架航班总延时为21 418 s，延误时间标准差为833.12。从表中可以看到相对于FCFS，3种算法的总延时和标准差都有大幅下降。算法1排序后的结果较先来先服务顺序变化最大。算法2中，由于约束同一条航路的航班不允许超越，所以其序列变化的幅度较小一些，如航班11和12，它们来自同一条航路，所以不允许换位。算法3中，首先对航班流进行了分组，输入的分组加锁绑定的间隔门限值 ε=20 s，即若两航班间距小于 $ts_{ij}+\varepsilon$，则它们分为一组。其排序过程如图4所示。

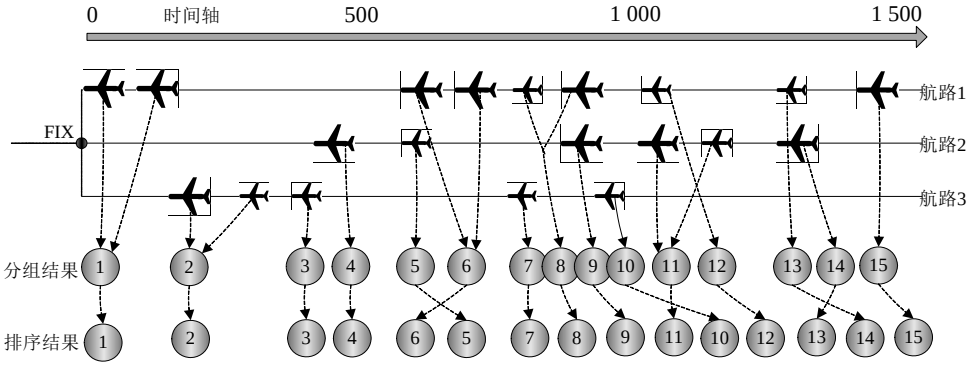

图4　本文算法对航班优化排序的示意图

图4中的横轴表示时间轴，其余3条线分别表示航路1、航路2、航路3上面的航班到达顺序图，下面是分组后的航班分组示意图，以及排序后的航班分组序列图。可以看出，经过分组后，原航班序列的20架航班分成了15个分组参与排序。解空间相对于算法1和算法2明显减少，仅为算法1的4.26%，算法2的30.7%。且随着航班序列规模的增大，解空间大小

和排序优化时间的减少越显著。虽然其总延时和标准差相对较大，但相对总延时最小的算法1，其总延时仅仅超过8.74%。而且从排序的序列结果可以看出，算法1、算法2排序后同一航路连续降落的航班占航班总数的50%，算法3排序后，同一航路连续降落的航班占70%，这样，更有利于管制员的管制和操作，减少了管制员的负荷。

参 考 文 献

[1] DEAR R G. The dynamic scheduling of Aircraft in the near terminal area[R]. [S.l.]: MIT Flight Transportation Lab Rep.R76-9, 1976.

[2] BRINION C R. An implicit enumeration algorithm for arrival aircraft scheduling[C]//Proceeding of the 11th IEEE/AIAA Digital Avionics Systems Confernce. [S.l.]: IEEE, 1992.

[3] 余江, 刘晓明, 蒲云. 飞机着陆调度问题的MPS优化算法研究[J]. 系统工程理论与实践, 2004, 3(3): 119-122.
YU Jiang, LIU Xiao-ming, PU Yun. Research on MPS optimization to landing schedule problem[J]. Systems Engineering-Theory & Practice, 2004, 3(3): 119-122.

[4] BAYEN A M, TOMLIN C J, YE Yin-yu, et al. An approximation algorithm for scheduling aircraft with holding time[C]//Proceeding of 43rd IEEE Conference on Decision and Control. [S.l.]: IEEE, 2004.

[5] HU Xiao-bing, CHEN Wen-hua. Receding horizon control for aircraft arrival sequencing and scheduling[J]. IEEE Transactions on Intelligent Transportation Systems, 2005, 6(2): 189-197.

[6] ANAGNOSTAKIS I, CLARKE J P, BOHME D, et al. Runway operations planning and control: Sequencing and scheduling[J]. AIAA Journal of Aircraft, 2001, 38(6): 988-996.

[7] 牟奇锋, 王慈光. 高度层优化使用问题的指派模型及算法[J]. 电子科技大学学报, 2009, 38(4): 573-577.
MOU Qi-feng, WANG Ci-guang. Assignu use of flight level [J]. Journal of University of Electronic Science and Technology of China, 2009, 38(4): 573-577.

[8] 余静, 杨红雨, 马博敏, 等. 证据理论在机场动态容量预测模型中的研究[J]. 电子科技大学学报, 2010, 39(1): 141-144.
YU Jing, YANG Hong-yu, Ma Bo-min, et al. Study on application of the evidence theory in airdrome’s dynamic capacity prediction model[J]. Journal of University of Electronic Science and Technology of China, 2010, 39(1): 141-144.

[9] ZHANG Xie, ZHANG Xue-jun, ZHANG Jun, et al. Optimization of sequencing for aircraft arrival based on approach routes[C]//Proceedings of the 2007 IEEE Intelligent Transportation Systems Conference Seattle. WA, USA: IEEE, 2007.

[10] LYMPEROPOULOS I, LYGEROS J. Improved multi-aircraft ground trajectory prediction for air traffic control[J]. AIAA Journal of Guidance, Control, and Dynamics, 2010, 33(2): 347-362.

[11] VRANAS P B M, BERTSIMAS D J, ODONI A R. Dynamic ground-holding policies for a network of airports[J]. Transportation Science(S0041-1655), 1994, 28(4): 275-291.

编 辑 漆 蓉