飞机座舱分配问题的虚拟嵌套控制模型

杨馨雨、田瑞峰、任泽华

（西安交通大学，电信学部，西安，710049）

摘要：在航空收益管理的问题中，座舱分配关系着航空公司的收益与乘客的满意度。我们针对单腿问题的座舱分配问题，采用了虚拟嵌套(virtual nesting)模型，将座位的级别映射到虚拟类中，并采用随机梯度下降法对问题进行求解，得到全局最优解。经过测试与比对，模型能够较好的计算最大收益目标下的座舱分配问题。

关键词：航班收益管理，O-D 对，虚拟嵌套模型，随机梯度下降法

0 问题描述（单腿问题）

在航空公司的收益管理问题研究中，我们关注的是在固定的舱位价格下，将航段容量分配给客户请求。而在前人的研究工作中，常常把一个航班分为多个航段，比如北京转机西安飞昆明，那么北京到西安即为一个航段（flight-leg）。可以看出，若是研究一个需要转机多次的航班的座位分配收益管理问题时，总是可以将整个航程划分为多个航段，每个航段都有一个源（origin）和一个目的地（destination），而在这个过程中不停靠其他机场。那么这个航段就被称为“单一航段”，所研究的问题也就可以被分解为基于单一航段的收益管理优化问题（single flight-leg problem），即单腿问题。我们调研了几篇相关的论文与他们建立的模型，发现当前比较成熟的模型有基于 OD-pair 的收益管理模型和基于虚拟嵌套（virtual nesting）原理的收益管理模型。而它们都是从最基础的单腿问题开始研究，并且逐步完善起来的。

下面我们将简要介绍两种模型，并且以虚拟嵌套为例进行了实现，选取北京到上海一天之内的所有航班，通过随机数生成客户请求（假设每单最多订 5 张机票），运用该模型进行了最优化设计。经过验证，这种方法可以使航空公司收益最大化，为飞机的座位分配提供了一种优化方案。

1 基于 OD-pair 的模型

如问题描述中所示，一个航班根据起止机场的不同可以看作是一个源-目的地对（OD-pair）。而此模型正是基于这种思想而建立起来的。如图一所示，两个城市机场间的飞机可以看作是一个单腿，而航空公司可以利用这些单腿组合成不同的航班销售给客户。每个航班都可以看作是一个 OD-pair，而图中可以看出有最多 10 个 OD-pair，由此可以进行建模求解。

1.1 基础模型

先定义两个下标集：

$$\varphi = \{1,\dots,S\} 为 S\ 个不同\ OD-pair\ 的下表集$$

$$j = \{1,\dots,J\} 为 J\ 个不同\ flight-leg\ 的下表集$$

再定义一个 0-1 变量：

$$a_{js} = \begin{cases} 1，\ flight-leg\ 在\ OD-pair\ 中 \\ 0，\ flight-leg\ 不在\ OD-pair\ 中 \end{cases}$$

此时，该模型可表示为：

$$maxmize \sum_{s=1}^{S} \Phi_s(x_s) \qquad (1)$$

$$subject\ to \sum_{s=1}^{S} a_{js} x_s \le C_j,\quad j \in j \qquad (2)$$

$$x_s \in \mathbb{Z}_+,\quad s \in \varphi \qquad (3)$$

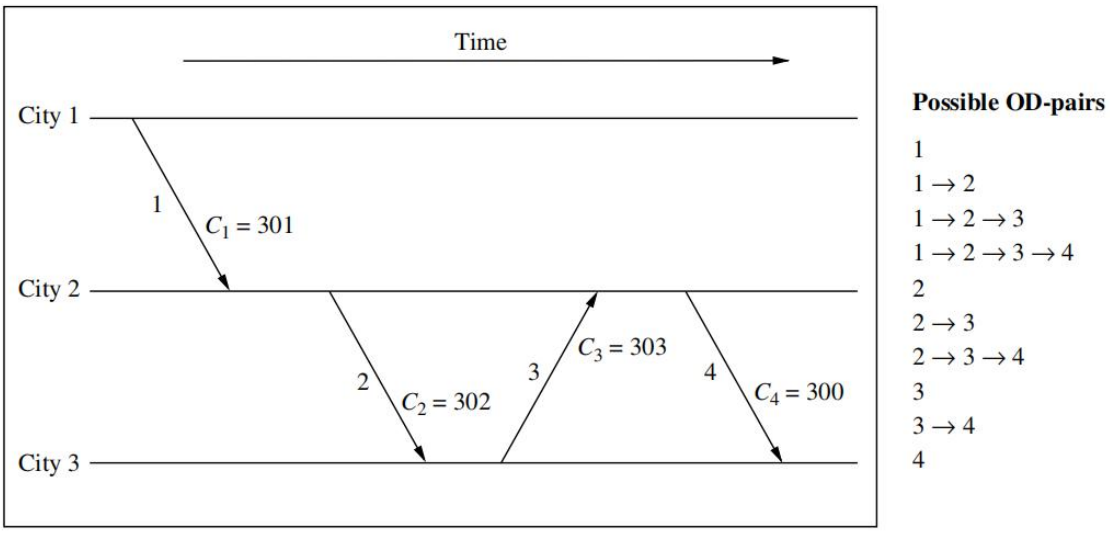
其中$\sum_{s=1}^{S} \Phi_s(x_s)$为收益函数，$C_j$为每个单腿航班上的座位数。

此时定义一些其他变量：

$$B_s = \min_{j \epsilon j} \{C_j | a_{js} = 1\} \qquad (4)$$

其中B_s表示每个 OD-pair 中最小的单腿航班容量，即此 OD-pair 的瓶颈。

$$0 < r_{1s} < r_{2s} < \cdots < r_{I_s s} \qquad (5)$$



图一 一个有 4 个 flight-leg 和 10 个可能的 OD-pair 的飞行网络

其中r_{is}表示第 s 个 OD-pair 在第 i 个座位等级中的收益，此处假定它们是从低到高排列（嵌套）的，这在文章第二部分将重点阐述。

此时，上述模型（2）式约束可以表示为：

$$\sum_{s=1}^{S} a_{js} \sum_{i=1}^{I_s} x_{is} \le C_j, j \in j$$

而每个 OD-pair 的容量显然满足如下约束：

$$x_s = \sum_{i=1}^{I_s} x_{is} \le B_s$$

1.2 静态模型

此模型将 1.1 中的$\sum_{s=1}^{S} \Phi_s(x_s)$修改为$\sum_{s=1}^{S} f_s(x_s)$

$$maxmize \sum_{s=1}^{S} f_s(x_s) \qquad (6)$$

$$subject\ to \sum_{s=1}^{S} a_{js} x_s \le C_j, \quad j \in j \quad (7)$$

$$x_s \in \mathbb{Z}_+, \quad s \in \varphi \qquad (8)$$

其中：

$$f_s(x_s) = max\left\{ \sum_{i=1}^{I_s} r_{is} \mathbb{E}[min\{x_{is}, D_{is}\}] \,|\, \sum_{i=1}^{I_s} x_{is} \le x_s, x_{is} \in \mathbb{Z}_+, i \in \varphi_s \right\} (9)$$

里面r_{is}即为上面提到的收益，$\mathbb{E}[min\{x_{is}, D_{is}\}]$是预期售出的票数的期望，其中$D_{is}$是一个随机量。

可以证明此模型是离散凹的，但却仍然是 NP 难的。

1.3 动态模型

将售票过程细分为多个小时刻，可以认为所有请求都是依次到来的，将其划分为$\hat{T}$个阶段（此时可与时刻等价）。建立动态规划模型：

此模型将 1.1 中的$\sum_{s=1}^{S} \Phi_s(x_s)$修改为$\sum_{s=1}^{S} g_s^1(x_s)$

$$maxmize \sum_{s=1}^{S} g_s^1(x_s) \qquad (10)$$

$$subject\ to \sum_{s=1}^{S} a_{js} x_s \le C_j, \quad j \in j \quad (11)$$

$$x_s \in \mathbb{Z}_+, \quad s \in \varphi \qquad (12)$$

其中：

$$g_s^t(x_s) = \mathbb{E}\left[max\{\xi_t + g_s^{t+1}(x_s - 1), g_s^{t+1}(x_s)\}\right] \quad (13)$$

$g_s^t(x_s)$是指从第 t 时刻到最后时刻$\hat{T}$的收益，其值由此时刻售出票与不售出两种情况的收益最大值的期望表示。其中ξ_t为售出时的票价，这是一个随机数。

这是一个嵌套的动态规划模型，其边界条件为：

$$g_s^{\hat{T}}(x_s) = \begin{cases} \mathbb{E}[\xi_{\hat{T}}], & if\ x_s > 0; \\ 0, & if\ x_s > 0. \end{cases}$$

而这个模型也面临着问题，要求解此模型需要提前规划好 OD-pair 的容量，而在考虑容量随时间的变化时又面临着状态空间爆炸的问题，因此也不好求解。

1.4 简化的求解模型

由于上述模型虽然结构简单却求解复杂，我们将其进行简化，以得到可以求解的模型。

将静态模型（6）~（8）变为：

$$maxmize \sum_{s=1}^{S} h_s(x_s), \quad (14)$$

$$subject\ to \sum_{s=1}^{S} a_{js}x_s \le C_j, \quad j \in j \quad (15)$$

$$x_s \in \mathbb{Z}_+, \quad s \in \varphi \quad (16)$$

$$h_s(x_s) = max\left\{\sum_{i=1}^{I_s} r_{is}min\{x_{is},d_{is}\} \mid \sum_{i=1}^{I_s} x_{is} \le x_s, x_{is} \in \mathbb{Z}_+, i \in \varphi_s\right\} \quad (17)$$

其中d_{is}是总的 OD-pair 容量，由于存在以下关系：

$$\mathbb{E}[min\{x_{is},D_{is}\}] \le min\{x_{is},d_{is}\}$$

所以此模型的解为（6）~（8）静态模型的下界。

将$h_s(x_s)$放入式中，（14）~（17）式可化简为：

$$maxmize \sum_{s=1}^{S}\sum_{i=1}^{I_s} r_{is}min\{x_{is},d_{is}\}, \quad (18)$$

$$subject\ to \sum_{s=1}^{S} a_{js}x_s \le C_j, \quad j \in j \quad (19)$$

$$x_s \in \mathbb{Z}_+, \quad s \in \varphi, i \in \varphi_s \quad (20)$$

再把取最小值函数替换为x_{is}，此时需要在约束中添加$x_{is} \le d_{is_+}$这一约束

此时模型变为：

$$maxmize \sum_{s=1}^{S}\sum_{i=1}^{I_s} r_{is}x_{is}, \quad (21)$$

$$subject\ to \sum_{s=1}^{S} a_{js}\sum_{i=1}^{I_s} x_{is} \le C_j, j \in j \quad (22)$$

$$x_{is} \le d_{is_+}, \quad s \in \varphi, i \in \varphi_s \quad (23)$$

$$x_s \in \mathbb{Z}_+, \quad s \in \varphi, i \in \varphi_s \quad (24)$$

而这就是一个凹的整数线性规划问题，可以用单纯形等方法来求解

1.5 示例

为进一步说明模型，选取图一当中的 1→4，1→2，2→3→4 三个 OD-pair，此时模型可以写为简单的线性规划问题：

$$\begin{aligned} maxmize\ & x_1 + x_2 + x_3, \\ subject\ to\ & x_1 + x_2 \le 301 \\ & x_2 + x_3 \le 302 \\ & x_3 \le 303 \\ & x_1 + x_3 \le 300 \\ & x_1 \ge 0,\ x_2 \ge 0,\ x_3 \ge 0 \end{aligned}$$

此时解得该模型的最优解为$(x_1^*,\ x_2^*,\ x_3^*) = (149.5,151.5,150.5)$

虽然不是整数解，但提供了整数最优解的一个上界。

2. 虚拟嵌套控制

虚拟嵌套是网络收益管理中流行的容量控制策略。在虚拟嵌套中，产品（行程-票价类组合）被映射到网络的每个资源（飞行航线）上相对少量的“虚拟类”中。然后使用嵌套的保护级别来控制这些虚拟类的可用性，也就是说，在保护级别的限制下，当且仅当其对应的虚拟类在所需的每个资源上可用时，用户的产品请求才被接受。在一定的保护级别下，被接收的产品请求组成了此航班的收益。

2.1 网络收益管理

在航班网络中，共有m个航班，该m个航班共提供了n种行程-票价组合。此网络的邻接矩阵为

$$A_{m\times n} = [a_{ij}] \in \{0,1\}^{m\times n},$$

其中：

$$a_{ij} = \begin{cases} 1, & 行程-票价组合i包含j航班 \\ 0, & 行程-票价组合i不包含j航班 \end{cases}$$

第j个行程-票价组合包含的第i个航班被映射到虚拟类$c_i(j)$中，$c_i(j)$用来检索在第 i 个航班中此行程-票价组合为处于哪个保护级别。

图二 虚拟嵌套控制示意图

保护级别 y 表示一个虚拟类被分配的座舱数量。$y_i = (y_{i1},y_{i2},...,y_{ic})$表示第 i 个航班的保护级别，该航班共有 c+1 个保护级别，为满足虚拟嵌套的要求，此模型有如下约束：

$$0 \le y_{i1} \le y_{i2} \le ... \le y_{ic} \le x_i(1),\ \ i = 1,...,m$$

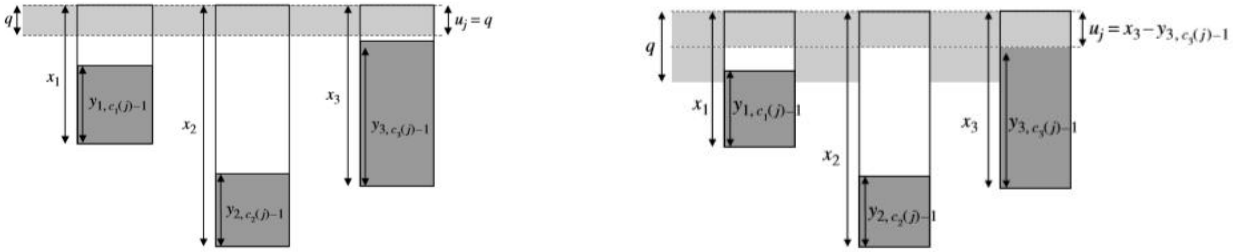
其中$x_i(1)$表示第 i 个航班的初始容量。虚拟嵌套控制可用图二解释。

假设一个航班被分为 3 个虚拟类，则最大虚拟类的保护级别为该航班的初始容量。

用户的需求用序列$\omega = \{\delta_t{:}t = 1,...,T\}$表示，其中 t 表示第 t 个用户，第 t 个用户的需求包括产品的种类和数量，即$\delta_t = (j_t,q_t)$，j_t为行程-票价组合的种类，q_t为该产品的数量。一个用户的需求的某类产品被接受的订单数量用$u_j(x(t),y,q)$表示，则

$$u_j(x(t),y,q) = \min\{q,\left(x_i(t) - y_{i,c_i(j)-1}\right)^+ {:} i \in A_j\}$$

该需求被接受当且仅当该航班剩余容量大于此产品所在类的前一类保护级别，用图三解释：



(a) (b)

图三 需求被接收的情况

设置保护级别的本意是，当卖第 c 类座位时，要为第 c-1 类座位保留足够的位置，为之后对第 c-1 类座位有需求的用户提供服务。一个需求能否被接受取决于航班剩余容量与前一类保护级别的关系，而能接受多少数量的需求取决与需求的数量 q 与航班剩余容量和前一类保护级别的差值熟大熟小。图 2(a)中航班剩余容量均大于前一类保护级别，因此需求可被接受；而需求的数量 q 均小于二者的差值，因此可全部满足此需求。同理，图 2(b)中需求可被接受，但在此行程-票价组合包含的三个航班中，第三个航班的航班剩余容量与前一类保护级别的差值小于 q，因此此需求只能卖出差值数量而不能全部满足需求。

航班的剩余容量在每有一个乘客的需求来临时，都会更新，剩余容量的递推公式为：

$$x(t+1) = x(t) - A_{j_t}u_{j_t}(x(t),y,q_t)$$

航班的收益可用被接受的需求和产品的价格计算，r_{j_t}表示第 t 个乘客的预订的产品价格，则总收益、收益的递推公式及边界条件为：

$$R(y,\omega) = R_1(x(1),y,\omega)$$

$$R_t(x(t),y,\omega) = r_{j_t}u_{j_t}(x(t),y,q_t) + R_{t+1}(x(t+1),y,\omega)$$

$$R_{T+1}(x,y,\omega) = 0$$

综上所述，航班网络收益管理问题的模型如下：

$$\begin{cases} \max R(t,\omega) \\ 0 \le y_{i1} \le y_{i2} \le ... \le y_{ic} \le x_i(1),\ \ i = 1,...,m \end{cases}$$

2.2 单一航班收益管理

单一航班是一种特殊的网络，仅有两个节点和一条边（航线）。为解决单一航班的座舱分配问题，在网络收益管理的基础上可简化为单腿问题模型。对于一个航班，该航班被分为 c+1 个虚拟类，行程-票价组合仅为航班中的虚拟类。因此模型可简化为：

$$\begin{cases} \max R(t,\omega) \\ 0 \le y_1 \le y_2 \le ... \le y_c \le x(1) \end{cases}$$

其中收益的递推式：

$$R(y,\omega) = R_1(x(1),y,\omega)$$

$$R_t(x(t),y,\omega) = ru(x(t),y,q_t) + R_{t+1}(x(t+1),y,\omega)$$

$$R_{T+1}(x,y,\omega) = 0$$

航班剩余容量的递推式：

$$x(t+1) = x(t) - u(x(t),y,q_t)$$

需求能否被接受仅与需求的种类也即虚拟类有关：

$$u_{j_t}(x(t),y,q) = \min\{q,\left(x(t) - y_{j_t-1}\right)^+\}$$

单腿问题较复杂网络问题而言，运算量和复杂度大幅度减少，可以利用下降梯度法和有约束非线性规划进行有效求解。下面以航班初始容量$x(1)=8$，用户序列$\omega=\{(3,1),(3,1),(3,1),(3,1),(2,1),(1,1)\}$，保护级别$y=(2,4)$为例，列出机票售卖过程，在后续的代码实现中，使用的是相同的步骤。

表1.机票售卖过程

用户	种类	数量	接受数量	决策	剩余容量
1	3	1	8－4 ＞ 1	接受	8−1=7
2	3	1	7－4 ＞ 1	接受	7−1=6
3	3	1	6－4 ＞ 1	接受	6−1=5
4	3	1	5－4 ＞ 1	接受	5−4=1
5	2	1	1－2 ＜ 1	拒绝	1
6	1	1	1－1 ≥ 0	接受	1−1=0

当 3 类座位的收益为$r=(25,19,10)$时，假设航班的初始容量增加 1，即$x(1)=9$，则容量增加会带来边际价格 10；假设航班的保护级别y_1增加 1，即$y_1=3$，则不影响收益；若保护级别y_2增加 1，即$y_2=5$，则会多拒绝一名用户的需求，即带来边际价格-10。因此偏导在此模型中有明显的物理意义，在后续的模型实现中，利用偏导向边际成本最大的方向更新，从而获得最大收益。

3．数据准备

在我们的模型中，由于是较为简单的单腿模型，所以需要的乘客模型就只需要三个：乘客的编号 t（即买票的顺序），乘客的买票数量q_t，乘客买票的优先级j_t（即买票的等级）。乘客的买票数量以及买票的优先级在数量较多情况下可以相当于随机产生，所以我们采用产生随机数的方式产生了一定数量的乘客。其中$0\le q_t\le 5$，即一名乘客只能购买 0-5 张票，$1\le j_t\le 3$，即票的等级有三种，最高级为 3，最低级为 1，并且q_t与j_t均为整数，随机数产生方式为均匀分布。

4. 算法

4.1 动态规划

我们以客户买票的需求和购买这样一个过程去看待这个问题。令 T 表示样本路径（如果$\{x_t;t\in[1,T]\}$是一个随机的过程,过程的样本路径为每一个 t 对应的$x(\omega):t\in[1,T]$,其中 w 是过程域中先前给定的固定点）上的客户数量，且对某个有限常数τ有 P（T≤τ）= 1。需求表征为一系列客户请求$\omega=\{\delta_t;t\in[1,T]\}$。索引 t 在时间上向前移动（即 t = 1 代表第一个客户，t = 2 第二个客户，t = T 最后一个）。序列中的每个元素δ_t是一个数对$\delta_t=(j_t,q_t)$，其中j_t为随机变量，代表客户 t 所请求的产品类型，而q_t也为随机变量，表示所请求的数量。

用户的需求用序列$\omega=\{\delta_t:t=1,...,T\}$表示，其中 t 表示第 t 个用户，第 t 个用户的需求包括产品的种类和数量，即$\delta_t=(j_t,q_t)$，j_t为行程-票价组合的种类，q_t为该产品的数量。一个用户的需求的某类产品被接受的订单数量用$u_j(x(t),y,q)$表示，则

$$u_j(x(t),y,q)=\min\{q,(x_i(t)-y_{i,c_i(j)-1})^+:i\in A_j\}$$

该需求被接受当且仅当该航班剩余容量大于此产品所在类的前一类保护级别。

航班的剩余容量在每有一个乘客的需求来临时，都会更新，剩余容量的递推公式为：

$$x(t+1)=x(t)-A_{j_t}u_{j_t}(x(t),y,q_t)$$

航班的收益可用被接受的需求和产品的价格计算，r_{j_t}表示第 t 个乘客的预订的产品价格，则总收益、收益的递推公式及边界条件为：

$$R(y,\omega)=R_1(x(1),y,\omega)$$

$$R_t(x(t),y,\omega)=r_{j_t}u_{j_t}(x(t),y,q_t)+R_{t+1}(x(t+1),y,\omega)$$

$$R_{T+1}(x,y,\omega)=0$$

所以在计算每一位乘客需求的时候，都是一个动态规划的过程，剩余容量随着上一个状态在更改，而收益的递推公式表明其为一个动态规划的计算，最终的收益为最后一个乘客处理完成后的收益。

4.2 非线性规划

由于价格为一个非线性规划函数，所以可以采用计算非线性规划的一些算法去计算这个问题的最优解。在虚拟嵌套问题中，Stochastic Gradient Algorithm（随机梯度下降法）较为简单实用，我们决定使用这种方法。算法的计算过程如下：

Step1.计算或者输入一组初始的保护等级$y^{(0)}$。

Step2.输入一个迭代次数，对 k:=1 to N 有

(a)计算上一次迭代得到的价格函数的梯度$\nabla_y R(y^{(k-1)},\omega^{(k)})$

(b)计算新步长$\rho^{(k)}:=a/k$。

(c)为下一次迭代更新保护等级

$$y^{(k)}=\Pi_\Theta(y^{(k-1)}+\rho^{(k)}\nabla_y R(y^{(k-1)},\omega^{(k)}))$$

Π_Θ是投影到可行集上的正交投影。

Step3.返回$y^{(N)}$,结束计算。

其中关于 Step2(b)中的一个投影形式为 $y=\arg\min||y'-z||$,这个投影是由一个带有线性约束的二次规划给出的，并且可以使用标准的方法来有效地解决。我们采用 MATLAB 中提供的二次规划工具包，将计算变换为二次型进行运算。此处注意到，由于我们在将 $y=\arg\min||y'-z||$变换为二次型的时候势必要将变量进行代换，所以在代换完成并且计算同样完成的时候再将变量代换回原来的形式。

4.3 梯度计算

关于随机梯度下降法，由于其必须得到梯度的计算结果，但又由于价格的梯度函数很难计算，所以我们通过查找文献得到了计算虚拟嵌套问题中价格函数梯度计算的方法。

在前面模型的结果中，我们可以经过数学推导得到$R_t(x(t),y,\omega)$的关于 x 与 y 的梯度表示：

Step1.计算$R_t(x(t),y,\omega)$的关于 x 的梯度表示

$$\frac{\partial}{\partial x_i(t)}R_t(x(t),y,\omega)$$

$$=\left(r_{jt}-\sum_{k\in A_{jt}}\frac{\partial}{\partial x_k(t+1)}R_{t+1}(x(t+1),y,\omega)\right)$$

$$\times\frac{\partial}{\partial x_i(t)}u_{jt}(x(t)-\xi_t,y,q_t)$$

$$+\frac{\partial}{\partial x_i(t+1)}R_{t+1}(x(t+1),y,\omega),\forall i,c,t$$

Step2.依赖前一步结果对$R_t(x(t),y,\omega)$的关于 y 的梯度表示进行计算，使用 EMSR-b 启发式算法通过关于$R_t(x(t),y,\omega)$的关于 x 的梯度表示计算下一步的嵌套结果表示。

$$\frac{\partial}{\partial y_{ic}}R_t(x(t),y,\omega)$$

$$=\left(r_{jt}-\sum_{k\in A_{jt}}\frac{\partial}{\partial x_k(t+1)}R_{t+1}(x(t+1),y,\omega)\right)$$

$$\times\frac{\partial}{\partial y_i(t)}u_{jt}(x(t)-\xi_t,y,q_t)$$

$$+\frac{\partial}{\partial y_{ic}}R_{t+1}(x(t+1),y,\omega),\forall i,c,t$$

5. 模型结果

我们通过随机数产生 100 名乘客，50 名乘客与 30 名乘客，将其导入到 MATLAB 中进行算法分析并迭代出优化结果，输出嵌套计算得到的结果，并与将全部情况遍历一遍后找到的结果进行对比分析，发现优化结果较为理想，价格设定在当前的客户需求下较为合适。

表2.模型求解结果

T	100			50			30		
座舱	一等	二等	三等	一等	二等	三等	一等	二等	三等
	159	14	4	112	33	32	4	87	86
收益	166550			137540			72120		

T=100 时，即有 100 个客户需求，我们假定每一个客户只能买同一种价位的票，初始的价位设定为 500 元，900 元以及 1490 元（对比北京到上海情况设定），总座位数为 177，乘客需求如 need3.xlsx 中所示，计算座位在这种情况下的分配。

程序计算得座位的分配为一等舱 159，二等舱 14，三等舱 4（与实际情况偏差较大是因为我们的数据产生是随机产生的，所以一等舱和三等舱购买意愿是相同的，在这种情况下系统偏向于将座位分配给一等舱，所以与实际有一定偏差）。在这种情况下，我们可以计算得收益为 166550 元，而将系统遍历一遍之后，我们发现最多的收益也为 166550 元。

T=50 时，即有 50 个客户需求，我们假定每一个客户只能买同一种价位的票，初始的价位设定为 500 元，900 元以及 1490 元（对比北京到上海情况设定），总座位数为 177，乘客需求如 need2.xlsx 中所示，计算座位在这种情况下的分配。

程序计算得座位的分配为一等舱 112，二等舱 33，三等舱 32。在这种情况下，我们可以计算得收益为 137540 元，而将系统遍历一遍之后，我们发现最多的收益也为 137540 元。

T=30 时，即有 30 个客户需求，我们假定每一个客户只能买同一种价位的票，初始的价位设定为 500 元，900 元以及 1490 元（对比北京到上海情况设定），总座位数为 177，乘客需求如 need.xlsx 中所示，计算座位在这种情况下的分配。

程序计算得座位的分配为一等舱 4，二等舱 87，三等舱 86。在这种情况下，我们可以计算得收益为 72120 元，而将系统遍历一遍之后，我们发现最多的收益也为 72120 元。

在计算分析之后，我们发现通过输入我们预先产生的乘客数据，我们可以模拟出一个较佳的座位分配情况，并且实际上价格分配与座位分配互为对偶问题，我们也可以通过解出这个问题的对偶问题来确定价格的定价问题。并且如果数据量足够大并且接近客户需求的实际分布情况，我们可以通过这

个模型来解出一个相对较佳的座位分配结果。

参考文献

[1] Garrett van Ryzin, Gustavo Vulcano, (2008) Simulation-Based Optimization of Virtual Nesting Controls for Network Revenue Management. Operations Research56(4):865-880. https://doi.org/10.1287/opre.1080.0550

[2] Ş. İlker Birbil, J. B. G. Frenk, Joaquim A. S. Gromicho, Shuzhong Zhang (2014) A Network Airline Revenue Management Framework Based on Decomposition by Origins and Destinations. Transportation Science 48(3):313-333. https:// doi.org/10.1287/trsc.2013.0469

[3] 运筹学，刁在筠等，高等教育出版社，ISBN:9787040201987

[4] MATLAB 帮助文档，https://ww2.mathworks.cn/help/matlab/