43. 1stOpt 运筹学与数学规划应用案例演示

43.1 引言

运筹学与数学规划是把实际问题通过构建成相关的数学模型并进而求解获得最优解决方案的科学决策方法，在实际生产中有着广泛的应用，把控掌控好这一方法将会领略体验到“运筹帷幄，决胜千里”的独特魅力！

1stOpt 不仅在数值优化科学计算方面优势明显，在处理运筹规划问题方面也有不俗的表现，尤其是使用非常直观简单。基于六个实际案例，介绍展示 1stOpt 如何高效快捷处理运筹规划问题。

43.2 案例问题

43.2.1 土方调配问题

土方调配指在土方施工中对挖土和填土方进行综合规划，以确定挖土方的数量及运输调配方向，目标是在满足填土方需求的前提下土方总运输量最小。

已知有 m=4 个挖方点，各挖方点最大可取土方量分别为 A=[450,800,600,350]，n=3 个填方点，所需土方量分别为 B=[800,600,500]，挖填方距离见表 42-1.

表 42-1 挖填方距离 C

填方 B / 挖方 A	B1	B2	B3	挖土方可供量 B
A1	50	70	100	450
A2	70	40	90	800
A3	60	110	70	600
A4	80	100	40	350
填土方需求量 A	800	600	500	

如果用 C[i,j]表示从挖土方 A[i]到填土方 B[j]的距离，X[i,j]为从挖土方 A[i]调配到填土方 B[j]的土方量，则目标函数为：

$$Min.\ \sum_{i=1}^{m}\sum_{j=1}^{n}\left(C_{i,j}\cdot X_{i,j}\right)$$

其中 X[i,j]为大于 0 的待求变量。

约束条件一：每个挖方点实际挖取提供的土方量必须小于等于其可供挖土量：

$$\sum_{j=1}^{n}\left(x_{i,j}\right)\le a_i \quad i=1\ldots4$$

约束条件二：必须满足每个填土方的需求量：

$$\sum_{i=1}^{m}(x_{i,j}) = b_j \quad j = 1\ldots3$$

由上述分析，对应相应的目标函数和约束条件，求解代码如下

代码-1：

```
Constant m=4, n=3;
Constant A(m)=[450,800,600,350],B(n)=[800,600,500],
C(m,n)=[50,70,100,
        70,40,90,
        60,110,70,
        80,100,40];
Algorithm = LP;
Parameter x(m,n)>=0;
MinFunction Sum(i=1:m)(Sum(j=1:n)(C[i,j]*x[i,j]));
            For(i=1:m)(Sum(j=1:n)(x[i,j])<=a[i]);
            For(j=1:n)(Sum(i=1:m)(x[i,j])=b[j]);
```

结果：

```
Objective Function(Min.): 92000

Best Estimated Parameters:
x[1,1]: 450
x[1,2]: 0
x[1,3]: 0
x[2,1]: 0
x[2,2]: 600
x[2,3]: 0
x[3,1]: 350
x[3,2]: 0
x[3,3]: 150
x[4,1]: 0
x[4,2]: 0
x[4,3]: 350
```

即 A1 挖土量 450，全部运送至 B1；A2 挖土量 600，全部运送至 B2；A3 挖土量 350 及 150，分别运送至 B1 和 B3；A4 挖土量 350，全部运送至 B3.目标函数总运输量为 92000.

43.2.2 人力资源分配问题

某昼夜服务的公交线路每天各时间段内所需司机和乘务人员人数如表 43-2 所示。假如司机和乘务人员分别在各时间段开始时上班，并连续工作 8 小时，问该公交线路应怎样安排司机和乘务人员，既能满足工作需要，又使配备司机和乘务人员的人数最少?

表 43-2 公交运行计划

班次	时间	所需人数
14	6:00 – 10.00	60
2	10:00 – 14.00	70
3	14:00 – 18.00	60
4	18:00 – 22.00	50
5	22:00 – 2.00	30
6	2:00 – 6.00	20

设 x[i]表示第 i 班次时开始上班的司机和乘务人员数，一共 6 个班次点，目标函数：

$$Min. \quad \sum_{i=1}^{6}(x_i)$$

约束条件：每个班次时长都为 4 小时，满足连续工作 8 小时，则有

$$\begin{cases} x_1 + x_6 \geq 60 \\ x_1 + x_2 \geq 70 \\ x_2 + x_3 \geq 60 \\ x_3 + x_4 \geq 50 \\ x_4 + x_5 \geq 30 \\ x_5 + x_6 \geq 20 \end{cases}$$

求解代码如下。

代码-2：

```
Constant n=6, p(6)=[60,70,60,50,30,20];
IntParameter x(6);
Algorithm = LP;
MinFunction Sum(i=1:n)(x[i]);
            x1+x6>=p1;
            x1+x2>=p2;
            x2+x3>=p3;
            x3+x4>=p4;
            x4+x5>=p5;
            x5+x6>=p6;
```

上述代码中约束条件的写法虽然直观但略显繁琐，在 1stOpt 中可使用“for”语句并结合“Wrap”函数简化有规律的约束条件。对于“Wrap”函数，比如 Wrap(i,n)，i 和 n 均为整数，如果整数 i 小于等于 n，则返回 i，反之如果大于 n，则返回 i-n。简化代码如下。

代码-3：

```
Constant n=6, p(6)=[60,70,60,50,30,20];
IntParameter x(6);
Algorithm = LP;
MinFunction Sum(i=1:n)(x[i]);
            For(i=1:6)(x[i]+x[Wrap(i+1,6)]>=p[Wrap(i+1,6)]);
```

上述两段代码均可得相同结果如下。

结果：

```
Objective Function(Min.): 150

Best Estimated Parameters:
x1: 60
x2: 20
x3: 40
x4: 10
x5: 20
x6: 0
```

43.2.3 工厂配料问题

某工厂要用三种原料 1、2、3 混合调配出三种不同规格的产品甲、乙、丙，数据如表所示。问:该厂应如何安排生产，使利润收入为最大?

表 43-3 原材料信息

原材料名称	每天最多供应量（公斤）- B	单价（元/公斤）- A
1	150	65
2	70	25

3	80	35

表 43-4 产品信息

产品名称	数量（公斤）- D	单价（元/公斤）- C
1	100	80
2	100	65
3	60	50

设 X[i,j]表示第 i 种产品中的原材料 j 含量，其目标函数为利润最大，即产品总价减去所用材料总价：

$$Max. \quad \begin{matrix} \left(x_{1,1}+x_{1,2}+x_{1,3}\right)\cdot c_1+\left(x_{2,1}+x_{2,2}+x_{2,3}\right)\cdot c_2+\left(x_{3,1}+x_{3,2}+x_{3,3}\right)\cdot c_3 \\ -\left(x_{1,1}+x_{2,1}+x_{3,1}\right)\cdot a_1-\left(x_{1,2}+x_{2,2}+x_{3,2}\right)\cdot a_2-\left(x_{3,1}+x_{3,2}+x_{3,3}\right)\cdot a_3 \end{matrix}$$

也可以简写为：

$$Max. \quad \sum_{i=1}^{3}\sum_{j=1}^{3}\left(x_{i,j}\right)\cdot c_i-\sum_{j=1}^{3}\sum_{i=1}^{3}\left(x_{i,j}\right)\cdot a_i$$

产量约束：

$$\sum_{j=1}^{3}\left(x_{i,j}\right)=d_i \quad i=1\ldots3$$

材料供应量约束：

$$\sum_{i=1}^{3}\left(x_{i,j}\right)\le b_j \quad j=1\ldots3$$

求解代码如下。

代码-4：

```
Constant A=[65,25,35],//材料价格
         B=[150,70,80],//材料供应量
         C=[80,65,55],//产品价格
         D=[100,100,60];//产品产量
IntParameter x(3,3);
Algorithm = LP;
MaxFunction Sum(i=1:3)(Sum(j=1:3)(x[i,j])*C[i])-Sum(j=1:3)(Sum(i=1:3)(x[i,j])*A[j]);
            For(i=1:3)(Sum(j=1:3)(x[i,j])=D[i]);
            For(j=1:3)(Sum(i=1:3)(x[i,j])<=B[j]);
```

结果：

```
Objective Function(Max.): 6100

Best Estimated Parameters:
x[1,1]: 100
x[1,2]: 0
x[1,3]: 0
x[2,1]: 10
x[2,2]: 10
x[2,3]: 80
x[3,1]: 0
x[3,2]: 60
x[3,3]: 0
```

43.2.4 下料问题

某钢管零售商从钢管厂进货，将钢管按照顾客的需求切割后售出。从钢管厂进货时得到原料钢管都是 19m 长。现有一客户需要 50 根 4m 长，20 根 6m 长和 15 根 8m 长的钢管，应如何下料最节省?

首先需要确定每根钢管会有多少种可能的切割模式或切割组合，如可以将 19 米长的钢管切割成 3 根 4 米及一根 6 米长的钢管，余料为 1 米，也可以切割成 3 根 6 米的钢管，余料也是 1 米，以此类推，共有 7 种切割模式如表 43-5.

表 43-5 钢管切割模式

模式	4 米钢管根数	6 米钢管根数	8 米钢管根数	余料（米）- P
模式 1	4	0	0	3
模式 2	3	1	0	1
模式 3	2	0	1	3
模式 4	1	2	0	3
模式 5	1	1	1	1
模式 6	0	3	0	1
模式 7	0	0	2	3

用 X[i]表示按照第 i 种模式切割的原料钢管根数，目标函数为余料之和最少：

$$Min. \quad \sum_{i=1}^{7}(x_i \cdot p_i)$$

约束一：4 米钢管数 50 根：

$$4 \cdot x_1 + 3 \cdot x_2 + 2 \cdot x_3 + x_4 + x_5 \geq 50$$

约束二：6 米钢管数 20 根：

$$x_2 + 2 \cdot x_4 + x_5 + 3 \cdot x_6 \geq 20$$

约束三：8 米钢管数 15 根：

$$x_3 + x_5 + 2 \cdot x_7 \geq 15$$

代码-5：

```
Constant P=[3,1,3,3,1,1,3],        //每种模式的余料
         M=[50,20,15],             //所需切割后钢管数
         C(7,3)=[4,0,0,
                 3,1,0,
                 2,0,1,
                 1,2,0,
                 1,1,1,
                 0,3,0,
                 0,0,2];
IntParameter x(7);
Algorithm = LP;
MinFunction Sum(i=1:7)(x[i]*P[i]);
          For(j=1:3)(Sum(i=1:7)(C[i,j]*x[i])>=M[j]);
```

结果：

```
Objective Function(Min.): 27

Best Estimated Parameters:
x1: 0
x2: 12
x3: 0
x4: 0
```

x5: 15
x6: 0
x7: 0

即共需要 19 米长钢管 27 根，其中 12 根按模式 2 切割，其余 15 根按模式 5 切割，余料之和 27 米。

43.2.5 最短路线问题

最短路径问题旨在寻找图中节点与节点之间的最短路径，如图 42-1 示，A、B、C、D、E、F、G、H、K 表示 9 个不同的村庄，村庄之间的连线表示两个村庄之间的距离，用 D[i,j]表示，无连接节点及自身节点用一很大数表示，比如 1000。现在有一个居民需要从村庄 A 到村庄 K，请求出两村庄之间的最短路线。

表 43-6 路径距离 D[i,j]

	1-A	2-B	3-C	4-D	5-E	6-F	7-G	8-H	9-K
1-A	1000	16	14	11	1000	1000	1000	1000	1000
2-B	16	1000	1000	1000	9	7	10	1000	1000
3-C	14	1000	1000	1000	18	1000	14	19	1000
4-D	11	1000	1000	1000	17	10	16	13	1000
5-E	1000	9	18	17	1000	1000	1000	1000	30
6-F	1000	7	1000	10	1000	1000	1000	1000	26
7-G	1000	10	14	16	1000	1000	1000	1000	21
8-H	1000	1000	19	13	1000	1000	1000	1000	22
9-K	1000	1000	1000	1000	30	26	21	22	1000

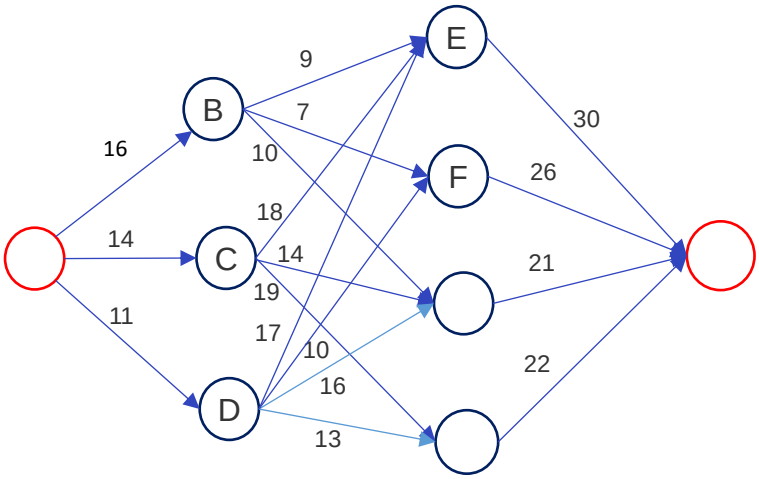

图 43-1 村庄之间网络图

设起点为 A，终点为 K，共 9 个位置点，引入 0-1 型决策变量 X[i,j]，如果路径（i,j）在最短路上，则 X[i,j]=1，否则 X[i,j]=0。

如果定义：

$$s_1 = \sum_{j=1}^{9}(x_{i,j}) \qquad\qquad s_2 = \sum_{j=1}^{9}(x_{j,i})$$

对于某一条最短路而言，除了起点 A 和终点 K，任意一个中间路径节点 i 必有 s1=1 或 s1=0，即节点 i 要么在最短路上，要么不在最短路。当 s=1 时，表明从 i 经过的所有路径中，必有一条路径在最短路上，即最短路经过此节点 i，此时，在所有从其他顶点到达顶点 i 的路径必然也有一条路径在最短路上，因此必有 s2=1；当 s1=0 时，说明最短路不经过顶点 i，则必有 s2=0.

综上分析，对于某一条最短路而言，必有：

$$\sum_{j=1}^{9}(x_{i,j}) = \sum_{j=1}^{9}(x_{j,i}) \quad 1 < i < 9$$

即中间点的流进量等于流出量。对于某一条最短路而言，对于起点 A（i=1）和终点 K（i=9），必有：

起点 A：$$\sum_{j=1}^{9}(x_{1,j}) = 1$$

终点 K：$$\sum_{j=1}^{9}(x_{j,9}) = 1$$

目标函数是最短路上所有路径里程之和最小，因而最短路问题可以用 0-1 规划法进行描述：

$$Min. \quad \sum_{i=1}^{9}\sum_{j=1}^{9}(d_{i,j} \cdot x_{i,j})$$

约束一：$$\sum_{j=1}^{9}(x_{i,j}) = \sum_{j=1}^{9}(x_{j,i}) \quad i = 2\ldots8$$

约束二：$$\sum_{j=1}^{9}(x_{1,j}) = 1$$

约束三：$$\sum_{j=1}^{9}(x_{j,9}) = 1$$

约束四：$$x_{i,j} + x_{j,i} \leq 1 \quad i = 1\ldots9, j = 1\ldots9$$

求解代码如下。

代码-6：

```
Constant n=9;
Constant d(n,n)=[1000,16,14,11,1000,1000,1000,1000,1000,
                 16,1000,1000,1000,9,7,10,1000,1000,
                 14,1000,1000,1000,18,1000,14,19,1000,
                 11,1000,1000,1000,17,10,16,13,1000,
                 1000,9,18,17,1000,1000,1000,1000,30,
                 1000,7,1000,10,1000,1000,1000,1000,26,
                 1000,10,14,16,1000,1000,1000,1000,21,
```

```
                    1000,1000,19,13,1000,1000,1000,1000,22,
                    1000,1000,1000,1000,30,26,21,22,1000];
BinParameter x(n,n);
Algorithm = LP;
MinFunction Sum(i=1:n)(Sum(j=1:n)(d[i,j]*x[i,j]));
            For(i=2:n-1)(Sum(j=1:n)(x[i,j])=Sum(j=1:n)(x[j,i]));
            Sum(j=1:n)(x[1,j])=1;
            Sum(j=1:n)(x[j,n])=1;
            For(i=1:n)(For(j=1:n)(x[i,j]+x[j,i]<=1));
```

上述代码中的距离矩阵也可以用“MDataSet”表示，更简单些

代码-7：

```
Constant n=9;
MDataSet[1000];
    i,j,d=
            1,2,16
            1,3,14
            1,4,11
            2,5,9
            2,6,7
            2,7,10
            3,5,18
            3,7,14
            3,8,19
            4,5,17
            4,6,10
            4,7,16
            4,8,13
            5,9,30
            6,9,26
            7,9,21
            8,9,22
EndMDataSet;
BinParameter x(n,n);
Algorithm = LP;
MinFunction Sum(i=1:n)(Sum(j=1:n)(d[i,j]*x[i,j]));
            For(i=2:n-1)(Sum(j=1:n)(x[i,j])=Sum(j=1:n)(x[j,i]));
            Sum(j=1:n)(x[1,j])=1;
            Sum(j=1:n)(x[j,n])=1;
            For(i=1:n)(For(j=1:n)(x[i,j]+x[j,i]<=1));
```

均可得到相同的结果。

结果：

Objective Function(Min.): 46	x[3,1]: 0	x[5,4]: 0	x[7,7]: 0
	x[3,2]: 0	x[5,5]: 0	x[7,8]: 0
Best Estimated Parameters:	x[3,3]: 0	x[5,6]: 0	x[7,9]: 0
x[1,1]: 0	x[3,4]: 0	x[5,7]: 0	x[8,1]: 0
x[1,2]: 0	x[3,5]: 0	x[5,8]: 0	x[8,2]: 0
x[1,3]: 0	x[3,6]: 0	x[5,9]: 0	x[8,3]: 0
x[1,4]: 1	x[3,7]: 0	x[6,1]: 0	x[8,4]: 0
x[1,5]: 0	x[3,8]: 0	x[6,2]: 0	x[8,5]: 0
x[1,6]: 0	x[3,9]: 0	x[6,3]: 0	x[8,6]: 0
x[1,7]: 0	x[4,1]: 0	x[6,4]: 0	x[8,7]: 0
x[1,8]: 0	x[4,2]: 0	x[6,5]: 0	x[8,8]: 0
x[1,9]: 0	x[4,3]: 0	x[6,6]: 0	x[8,9]: 1
x[2,1]: 0	x[4,4]: 0	x[6,7]: 0	x[9,1]: 0
x[2,2]: 0	x[4,5]: 0	x[6,8]: 0	x[9,2]: 0
x[2,3]: 0	x[4,6]: 0	x[6,9]: 0	x[9,3]: 0
x[2,4]: 0	x[4,7]: 0	x[7,1]: 0	x[9,4]: 0
x[2,5]: 0	x[4,8]: 1	x[7,2]: 0	x[9,5]: 0
x[2,6]: 0	x[4,9]: 0	x[7,3]: 0	x[9,6]: 0
x[2,7]: 0	x[5,1]: 0	x[7,4]: 0	x[9,7]: 0
x[2,8]: 0	x[5,2]: 0	x[7,5]: 0	x[9,8]: 0
x[2,9]: 0	x[5,3]: 0	x[7,6]: 0	x[9,9]: 0

最终结果是通过：A->D->H->K，总距离为 46.

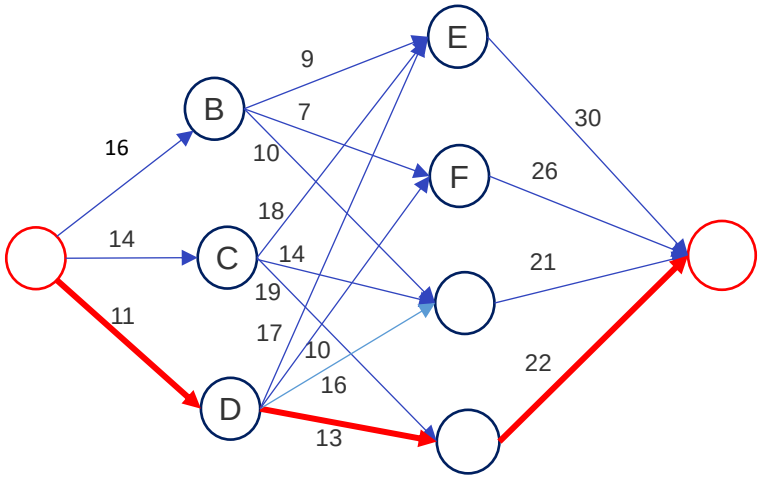

图 43-2 最优路径结果图

这种从某点到某点最短距离问题也可也采用动态规划方法。第 1 点 A 和最终第 9 点 K 为确定值，问题既是找出从 1 点到 9 点的行进次序使得该路径距离最短，其目标函数为：

$$Min. \quad \sum_{i=1}^{9} d_{i,i+1}$$

其中，di, i+1：节点（i, i+1）间距离。

以 P 代表某一方案的路径节点组合号，P ∈ [1,9]，且为整数，因为出发点和终点号为固定已知，即 P1 = 1，P9 = 9，则仅需确定 P2 至 P8 的值，即有 7 个决策变量，取值范围也变为[2,8]。对于一特定的随机路径，P2 至 P8 为 2 至 8 间任意整数值，其目标函数是路径总长最小。如果两节点间没有连接，则采用数据一较大的数值如 1000，该数值实际上起到罚函数的作用，根据具体情况可更改，本例中有连接的最大节点间距离不超过 10，因此取 1000 作为罚函数足可满足要求。该动态模型为非线性，只能采用非线性算法，在此采用“SM2”. 计算结果与前述一致：1（固定）-> 4 -> 8 -> 9（固定），也即：A->D->H->K，最短距离为 46.

代码-8：

```
Constant n=9;
MDataSet[1000];
      i,j,d=
                1,2,16
                1,3,14
                1,4,11
                2,5,9
                2,6,7
                2,7,10
                3,5,18
                3,7,14
                3,8,19
                4,5,17
                4,6,10
                4,7,16
                4,8,13
                5,9,30
                6,9,26
                7,9,21
                8,9,22
EndMDataSet;
Constant p[1]=1, p[9]=9;
IntParameter P(2:n-1)=[2,n-1];
Algorithm = SM2[50];
```

```
MinFunction Sum(i=2:n-2)(D[P[i],P[i+1]])+D[1,P[2]] +D[P[8],9];
```

结果：

```
Objective Function (Min.): 46
p2: 4
p3: 4
p4: 4
p5: 4
p6: 4
p7: 4
p8: 8
```

43.2.6 购书最优问题

某学校需要集体采购“计算机应用基础”、“高等数学”两种教材各 200 本。现有三家书店有这两种教材供应，但每家书店两种教材的库存量都无法满足该学校的需求量，学校需要分别从这三家书店采购。三家书店承诺若一次购买量超过 50 本可享受价格优惠。三家书店教材的库存量和价格折扣如下表所示：

表 43-7 书店教材及折扣

j / i	书店一 B[i,1]	书店二 B[i,2]	书店三 B[i,3]	价格（元/本）（P）
计算机应用	120	98	116	28
高等数学	132	117	165	25
折扣（D）	0.85	0.88	0.84	

已知“计算机应用基础”的单价为 28 元，“高等数学”的单价为 25 元。如何确定如何安排采购计划，可使学校购买这两种教材的费用最小？

引入决策变量 X[i,j]表示第 i 本书在第 j 书店购买的书本量，因为同一个书店购买量超过 50 会有不同的折扣优惠，目标函数为不连续的分段函数：

$$Min. \quad f_1 + f_2 + f_3$$

其中：

$$f_1 = \begin{cases} x_{1,1} \cdot p_1 + x_{2,1} \cdot p_2 & if\ x_{1,1} + x_{2,1} \le 50 \\ (x_{1,1} \cdot p_1 + x_{2,1} \cdot p_2) \cdot d_1 & if\ x_{1,1} + x_{2,1} > 50 \end{cases}$$

$$f_2 = \begin{cases} x_{1,2} \cdot p_1 + x_{2,2} \cdot p_2 & if\ x_{1,2} + x_{2,2} \le 50 \\ (x_{1,2} \cdot p_1 + x_{2,2} \cdot p_2) \cdot d_2 & if\ x_{1,2} + x_{2,2} > 50 \end{cases}$$

$$f_3 = \begin{cases} x_{1,3} \cdot p_1 + x_{2,3} \cdot p_2 & if\ x_{1,3} + x_{2,3} \le 50 \\ (x_{1,3} \cdot p_1 + x_{2,3} \cdot p_2) \cdot d_3 & if\ x_{1,3} + x_{2,3} > 50 \end{cases}$$

约束一：计算机应用及高等数学各 200 本

$$\sum_{j=1}^{3} (x_{i,j}) = 200 \quad i = 1 \ldots 2$$

约束二：每个书店每种书的订购量不得大于库存量 B[i,j]

$$x_{i,j} \le b_{i,j} \quad i = 1 \ldots 2, j = 1 \ldots 3$$

求解代码如下，因为目标函数为分段函数，无法直接采用线性算法（LP），缺省的算法处理这类问题效果并不理想，在此采用 DE 算法，大部分情况下可以得到如下最优解。

代码-9：

```
Algorithm = DE[200];
```

```
Constant B(2,3)=[120,98,116,
                 132,117,165];
Constant D=[0.85,0.88,0.84], P=[28,25];
IntParameter x(2,3)=[0,200];
ConstStr f1=if(x[1,1]+x[2,1]<=50,(x[1,1]*p1+x[2,1]*p2),(x[1,1]*p1+x[2,1]*p2)*D[1]),
          f2=if(x[1,2]+x[2,2]<=50,(x[1,2]*p1+x[2,2]*p2),(x[1,2]*p1+x[2,2]*p2)*D[2]),
          f3=if(x[1,3]+x[2,3]<=50,(x[1,3]*p1+x[2,3]*p2),(x[1,3]*p1+x[2,3]*p2)*D[3]);
MinFunction f1+f2+f3;
            For(i=1:2)(Sum(j=1:3)(x[i,j])=200);
            For(i=1:2)(For(j=1:3)(x[i,j]<=B[i,j]));
```

结果：

```
Objective Function (Min.): 8936.27
x[1,1]: 84
x[1,2]: 0
x[1,3]: 116
x[2,1]: 35
x[2,2]: 0
x[2,3]: 165
```

结果表明，在书店一和书店三分别购买计算机应用 84 和 116 本，同样在书店一和书店三购买高等数学各 35 和 165 本，总费用 8936.27 元。

采用非线性算法，无法保证每次都能计算得到最优解，是否能转换模型形式并进而能采用线性算法，从而保证得到稳定最优解？一个简单的方法是将三个书店打折与否分解，构成了总共 8 种不同的组合方式，使模型线性化，通过关键字“LoopConstant”及“FullLoopModel”，同时引入一个足够大的数 M，保证每种组合下约束条件都会得到满足，代码如下。

代码-10：

```
Constant M=1000000;
Algorithm = LP;
Constant B(2,3)=[120,98,116,
                 132,117,165];
Constant P=[28,25], C=[0.85,0.88,0.84];
LoopConstant D1=[1,0.85],D2=[1,0.88],D3=[1,0.84];
FullLoopModel;
IntParameter x(2,3)=[0,200];
ConstStr f1=(x[1,1]*p1+x[2,1]*p2)*D[1],
          f2=(x[1,2]*p1+x[2,2]*p2)*D[2],
          f3=(x[1,3]*p1+x[2,3]*p2)*D[3];
MinFunction f1+f2+f3;
            x[1,1]+x[2,1]<=50+(1-D[1])*M;
            x[1,1]+x[2,1]>50-(D[1]-C[1])*M;
            x[1,2]+x[2,2]<=50+(1-D[2])*M;
            x[1,2]+x[2,2]>50-(D[2]-C[2])*M;
            x[1,3]+x[2,3]<=50+(1-D[3])*M;
            x[1,3]+x[2,3]>50-(D[3]-C[3])*M;
            For(i=1:2)(Sum(j=1:3)(x[i,j])=200);
            For(i=1:2)(For(j=1:3)(x[i,j]<=B[i,j]));
```

上述代码采用线性（LP）算法，虽然要自动循环计算 8 次，但几乎瞬间完成，8 次计算结果小结如下表，很明显，目标函数为 1E30 的 4 次组合模型无解，剩余的四次中取目标函数值最小的为最优解，即 8936.27，与前面非线性计算的最好结果相同。

表 43-8 线性模型自动循环计算结果

Loop constant d1	Loop constant d2	Loop constant d3	Objective Function
1	1	1	1E30
1	1	0.84	1E30
1	0.88	1	1E30
1	0.88	0.84	9033.08
0.85	1	1	1E30
0.85	1	0.84	8936.27

0.85	0.88	1	9128.2
0.85	0.88	0.84	8975.12

43.4 小结

运筹规划问题，实际生活当中随处可见，解决的好事半功倍，反之则事倍功半。“工欲善其事，必先利其器”，通过对实际案例的建模求解，1stOpt 无疑是解决运筹规划问题的“利器“之一。