28. 1stOpt 运筹学案例演示

28.1 介绍

针对两个经典运筹学问题，基于1stOpt进行建模求解，展示了三维参数的定义及求解，运用Wrap函数，For()及Sum()关键字，使得求解代码与模型公式完全吻合的同时兼顾直观易懂，建模效率大幅提高。

28.2 工厂人员排班问题

一工厂某生产线为 7 天 24 小时生产，分为早、中、晚三班倒，需要工人值班， 目前有 12 个员工，轮换值班，编号为 1,2,…,12。要求：

1) 每人每天只能值一个班， 无论何时都不能连续值两个班；
2) 每个人一周至少休息 2 天（从 00:00~24:00）；
3) 每天每个班次满足所需人数， 如下表所示。

表 28.1 一周每天每个班次所需人数

	周一	周二	周三	周四	周五	周六	周日
早班 (06:00~14:00)	4	3	3	3	4	2	3
中班 (14:00~22:00)	4	3	3	2	3	2	2
晚班 (22:00~06:00)	3	2	2	3	3	1	2

表 18.2 12 名员工每个班次的薪酬

员工	1	2	3	4	5	6	7	8	9	10	11	11
薪酬	300	280	350	250	250	330	350	250	200	300	250	200

请给出未来一周的最佳排班表，需满足上述所有条件下，费用最省？

模型建立：

工人编号：i = 1,2,3,…12；

星期编号：j = 1,2,3,…7；

班次编号：k = 1,2,3；

班次所需人数（表28.1）：d(j,k)

员工每班次薪酬（表28.2）：c(i)

决策变量：0-1三维变量

$$x_{i,j,k}=\begin{cases}1 & \text{工人i被安排到第j天第k个班次}\\ 0 & \text{其它}\end{cases}$$

目标函数：费用最少

$$\text{Min.}\sum_{i=1}^{12}\sum_{j=1}^{7}\sum_{k=1}^{3}x_{i,j,k}\cdot c_i$$

对应代码：

```
Sum(i=1:12)(Sum(j=1:7)(Sum(k=1:3)(x[i,j,k]*C[i])));
```

约束条件：

1）每人每天最多工作一个班次

$$\sum_{k=1}^{3}x_{i,j,k}\le 1\quad i=1,\dots 12, j=1,\dots 7$$

对应代码：

```
For(i=1:12)(For(j=1:7)(Sum(k=1:3)(x[i,j,k])<=1))
```

2）每人任何时候都不能连续两个班次工作

$$\begin{cases}x_{i,j,3}+x_{i,j+1,1}\le 1\\ x_{i,7,3}+x_{i,1,1}\le 1\end{cases}\quad \text{i = 1,…12, j = 1,…6}$$

对应代码：

```
For(i=1:n)(For(j=1:6)(x[i,j,3]+x[i,j+1,1]<=1));
For(i=1:n)(x[i,7,3]+x[i,1,1]<=1);
```

上述两段代码实际上可以合二为一，使用“Wrap”函数

对应代码：

```
For(i=1:n)(For(j=1:7)(x[i,j,3]+x[i,Wrap(j+1,7),1]<=1));
```

Wrap(i,n)函数：当i小于等于n时返回i，当i大于n时返回i-n；

Wrap0(i,n)函数：当i小于等于n时返回i，当i大于n时返回i-n-1。

3）每人每周至少休息两天

$$\sum_{j=1}^{7}\sum_{k=1}^{3}x_{i,j,k}\le 5\quad i=1,\dots 12, j=1,\dots 7$$

对应代码：

```
For(i=1:n)(Sum(j=1:7)(Sum(k=1:3)(x[i,j,k]))<=5);
```

4）每人每周至少工作一个班次天

$$\sum_{j=1}^{7}\sum_{k=1}^{3} x_{i,j,k} \geq 1 \quad i = 1, \dots 12, j = 1, \dots 7$$

对应代码：

```
For(i=1:n)(Sum(j=1:7)(Sum(k=1:3)(x[i,j,k]))>=1);
```

5）各班次人数满足需求

$$\sum_{i=1}^{12} x_{i,j,k} \leq d_{j,k} \quad i = 1, \dots 12, j = 1, \dots 7$$

对应代码：

```
For(j=1:7)(for(k=1:3)(Sum(i=1:12)(x[i,j,k])>=d[j,k]));
```

最终完整代码：

```
Constant n=12;
Constant D(7,3)=[4,4,3,
                 3,3,2,
                 3,3,2,
                 3,2,3,
                 4,3,3,
                 2,2,1,
                 3,2,2];
Constant C(n)=[300,280,350,250,250,330,350,250,200,300,250,200];
BinParameter x(n,7,3);//工人i被安排到第j天第k个班次
Algorithm = LP;
MinFunction Sum(i=1:n)(Sum(j=1:7)(Sum(k=1:3)(x[i,j,k]*C[i])));
        For(i=1:n)(For(j=1:7)(Sum(k=1:3)(x[i,j,k])<=1));//每人每天最多工作一个班次
        For(i=1:n)(For(j=1:7)(x[i,j,3]+x[i,Wrap(j+1,7),1]<=1)); //每人在任何时候都不能连续两个班次工作
        For(i=1:n)(Sum(j=1:7)(Sum(k=1:3)(x[i,j,k]))<=5);//每人每周至少休息两天
        For(i=1:n)(Sum(j=1:7)(Sum(k=1:3)(x[i,j,k]))>=1);//每人每周至少工作一个班次
        For(j=1:7)(for(k=1:3)(Sum(i=1:12)(x[i,j,k])>=d[j,k]));//每个班次人数满足需求
```

运行结果：

```
Iterations: 363
Elapsed Time (Hr:Min:Sec:Msec): 00:00:00:203
Algorithms: Simplex Linear Program
Objective Function(Min.): 15500

Best Estimated Parameters:
x[1,1,1]: 0
x[1,1,2]: 1
x[1,1,3]: 0
x[1,2,1]: 0
x[1,2,2]: 0
x[1,2,3]: 1
x[1,3,1]: 0
x[1,3,2]: 0
```

x[1,3,3]: 1
x[1,4,1]: 0
x[1,4,2]: 0
x[1,4,3]: 1
…..

28.3 生产计划安排问题

某工厂加工四种零件供企业使用，每种零件的生产能力和成本如表28.3，最近公司承接了五笔加工订单，各订单签订的收费标准及对零件数量要求分别如表28.4和表28.5

表28.3 某工厂生产能力及成本

	零件1	零件2	零件3	零件4
生产能力（万件）- C	28	23	18	12
成本（元）- D	10	21	13	8

表28.4 各订单收费标准（元/件）

P	零件1	零件2	零件3	零件4
订单A	110	95	72	54
订单B	103	88	68	50
订单C	100	92	72	60
订单D	98	86	70	62
订单E	105	94	78	65

表28.5 各订单对零件数量（万件）的要求：

	订单A	订单B	订单C	订单D	订单E
零件1	1~3	≥ 3	3	1~3	≥ 2
零件2	≥ 3	≥ 3	≥ 4	≥ 3	≤ 6
零件3	≥ 3	1~4	≥ 3	≤ 3	4~6
零件4	≥ 1	≥ 1	≤ 4	0	≥ 2
总数量（F）	≤ 13	≤ 16	≤ 12	≤ 14	≤ 10

试建立数学模型，对公司现有生产能力进行合理配置，使公司收益最大。

模型建立：

订单编号：i = 1,…5；

零件编号：j = 1,…4；

决策变量：x(i,j)，生产件数（万件），整数型参数；

p(i,j)：收费标准（元/件）；

c(j)：生产能力（万件）；

d(j)：零件单位成本（元/件）；

F(i)：各订单总数量约束；

L(i,j) ：各订单各零件下限约束；

U(i,j) ：各订单各零件上限约束；

目标函数：收益最大

$$Max.\ \sum_{i=1}^{5}\sum_{j=1}^{4}\left(\left(p_{i,j}-d_j\right)\cdot x_{i,j}\cdot 10000\right)$$

约束条件：

1）生产能力约束

$$\sum_{i=1}^{5} x_{i,j} \le c_j \ \ j=1,\dots 4$$

2）各订单零件总数量约束

$$\sum_{j=1}^{4} x_{i,j} \le F_i \ \ i=1,\dots 5$$

3）各订单零件上下限约束

参考表28.5，各订单对各零件有不同的上下限要求，有的只有上限要求，有的则只有下限要求，还有的两者均有，为方便代码编写，引入一个足够大的数M=10000，将表28.5分解成两个表格，分别代码上下限约束矩阵，注意下限数据矩阵中的“-M”也可以用“0”替代表示，因为生产的零件数最少只能为0，无论用“-M”还是“0”，计算结果是相同的。

表28.6 上限矩阵U

	零件 1	零件 2	零件 3	零件 4
订单 A	3	M	M	M
订单 B	M	M	4	M
订单 C	3	M	M	4
订单 D	3	M	3	0
订单 E	M	6	6	M

表28.7 下限矩阵L

	零件 1	零件 2	零件 3	零件 4
订单 A	1	3	3	1
订单 B	3	3	1	1
订单 C	3	4	3	-M

订单 D	1	3	-M	0
订单 E	2	-M	4	2

$$L_{i,j} \le x_{i,j} \le U_{i,j} \quad i = 1, \dots 5, j = 1, \dots 4$$

最终完整代码：

```
Constant M=10000;
Constant C=[28,23,18,12],          //生产能力(万件)
          D=[10,21,13,8],          //零件单位成本（元/件）
          P(5,4)=[110,95,72,54, //收费标准
                  103,88,68,50,
                  100,92,72,60,
                  98,86,70,62,
                  105,94,78,65],
          L(5,4)=[1,3,3,1,         //下限需求矩阵
                  3,3,1,1,
                  3,4,3,-M,
                  1,3,-M,0,
                  2,-M,4,2],
          U(5,4)=[3,M,M,M,         //上限需求矩阵
                  M,M,4,M,
                  3,M,M,4,
                  3,M,3,0,
                  M,6,6,M],
          F=[13,16,12,14,10];    //各订单总数量约束
IntParameter x(5,4);
Algorithm = LP;
MaxFunction Sum(i=1:5)(Sum(j=1:4)((P[i,j]-D[j])*x[i,j]*10000));
              For(j=1:4)(Sum(i=1:5)(x[i,j])<=C[j]);
              For(i=1:5)(Sum(j=1:4)(x[i,j])<=F[i]);
              For(i=1:5)(For(j=1:4)(x[i,j]>=L[i,j]));
              For(i=1:5)(For(j=1:4)(x[i,j]<=U[i,j]));
```

运行结果：

```
Iterations: 26
Elapsed Time (Hr:Min:Sec:Msec): 00:00:00:15
Algorithms: Simplex Linear Program
Objective Function(Max.): 48700000

Best Estimated Parameters:
x[1,1]: 3
x[1,2]: 6
x[1,3]: 3
x[1,4]: 1
x[2,1]: 11
x[2,2]: 3
x[2,3]: 1
x[2,4]: 1
x[3,1]: 3
x[3,2]: 6
x[3,3]: 3
x[3,4]: 0
x[4,1]: 3
x[4,2]: 8
```

```
x[4,3]: 3
x[4,4]: 0
x[5,1]: 4
x[5,2]: 0
x[5,3]: 4
x[5,4]: 2
```

28.4 小结

本案例运筹学问题均属线性规划问题，算法因此均采用线性算法“Algorithm = LP;”，虽然非线性算法也能求解线性问题，但效率相差甚远，同时也无法保证得到最优解。

结合线性算法及简洁直观的模型描述语言，1stOpt可以非常方便处理各式各样的运筹学问题。

最后注意，三维、四维或五维参数定义及应用需 10.0 版方可正常使用。