45. 1stOpt 非线性规划建模问题线性化技巧及案例

45.1 引言

运筹规划问题一般可划分为线性和非线性两大类，对于前者线性问题，只要模型本身正确，目前成熟高效的单纯形算法（Simplex Method）和内点法（Interior Point Method）基本都可以完美解决，而对于后者非线性问题，就要复杂困难的多，效果方面，虽然有众多非线性优化算法，但至少目前为止还没有任何一种非线性优化算法能保证所求解的问题一定是全局最优解，很多时候只能得到局部最优解或可行解，而效率方面也会低效很多。基于这种现实，在构筑模型时，如果模型中含有非线性部分，如果能够将这些非线性部分转换为等效的线性部分，从而将整个模型作为线性问题去处理，不仅效果可以得到保证，效率也会大幅提升。在此以运输装货问题为例，介绍如何将模型中含有绝对值（abs()）和最大最小（MinMax()）非线性函数通过变换，将模型中的 abs()和 MaxMin()替代并消除，将非线性模型线性化。

45.2 绝对值 abs()和最大最小 MinMax()函数线性转换

绝对值 abs()函数和最大最小 MinMax()函数是非线性函数，如果模型内含有这类函数，则无法直接按线性模型求解处理，但可以通过一定变换将这部分非线性函数部分消除，使得模型成为真正的线性模型。两个简单示例如下。

45.2.1 绝对值 abs()函数线性化处理

有两种方式处理绝对值 abs()函数。如下示例，目标函数中含有绝对值函数

$$Min.\ |x_1| + 2\cdot|x_2| + |x_3| + 4\cdot|x_4|$$

$$S.T.\ \begin{cases} x_1 - x_2 - x_3 + x_4 = 12 \\ x_1 - x_2 + 6\cdot x_3 - 3\cdot x_4 = 1 \\ x_1 - x_2 - 2\cdot x_3 + 3\cdot x_4 = -\dfrac{1}{2} \end{cases} \tag{45-1}$$

方式一：引入$y_i = |x_i|$，同时增加约束：$y_i \geq x_i$及$y_i \geq -x_i$，由此上述非线性模型变为如下纯线性模型

$$Min.\ y_1 + 2\cdot y_2 + y_3 + 4\cdot y_4$$

$$S.T.\ \begin{cases} x_1 - x_2 - x_3 + x_4 = 12 \\ x_1 - x_2 + 6\cdot x_3 - 3\cdot x_4 = 1 \\ x_1 - x_2 - 2\cdot x_3 + 3\cdot x_4 = -\dfrac{1}{2} \\ y_i \geq x_i \quad i = 1\ldots4 \\ y_i \geq -x_i \quad i = 1\ldots4 \end{cases} \tag{45-2}$$

方式二：引入均大于 0 的u_i及v_i，令$x_i = u_i - v_i$，则有：$|x_i| = u_i + v_i$，等价模型为：

$$Min.\ (u_1+v_1)+2\cdot(u_2+v_2)+(u_3+v_3)+4\cdot(u_4+v_4)$$

$$S.T.\ \begin{cases}(u_1-v_1)-(u_2-v_2)-(u_3-v_3)+(u_4-v_4)=12\\(u_1-v_1)-(u_2-v_2)+6\cdot(u_3-v_3)-3\cdot(u_4-v_4)=1\\(u_1-v_1)-(u_2-v_2)-2\cdot(u_3-v_3)+3\cdot(u_4-v_4)=-\frac{1}{2}\end{cases} \quad (45\text{-}3)$$

代码 45.1：三段求解代码

原模型
（非线性）
	MinFunction abs(x1)+2*abs(x2)+abs(x3)+4*abs(x4);
	x1-x2-x3+x4=12;
	x1-x2+6*x3-3*x4=1;
	x1-x2-2*x3+3*x4=-1/2;

线性化方式
一
（线性）
	Constant n=4;
	Algorithm = LP;
	Parameter x(n),y(n);
	MinFunction y1+2*y2+y3+4*y4;
	x1-x2-x3+x4=12;
	x1-x2+6*x3-3*x4=1;
	x1-x2-2*x3+3*x4=-1/2;
	For(i=1:4)(y[i]>=x[i]);
	For(i=1:4)(y[i]>=-x[i]);

线性化方式
二
（线性）
	Constant n=4;
	Algorithm = LP;
	Parameter u(n)>=0,v(n)>=0;
	ConstStr For(i=1:n)(x[i]=u[i]-v[i]);
	PassParameter x(n);
	MinFunction (u1+v1)+2*(u2+v2)+(u3+v3)+4*(u4+v4);
	x1-x2-x3+x4=12;
	x1-x2+6*x3-3*x4=1;
	x1-x2-2*x3+3*x4=-1/2;

结果：

原模型 （非线性）	Objective Function (Min.): 61.249999998125 x1: 14.64999999965 x2: -4.8086005064185E-15 x3: -7.19999999977501 x4: -9.84999999967501	Constrained Functions: 1: x1-x2-x3+x4-(12) = -2.49992027079315E-10 2: x1-x2+6*x3-3*x4-(1) = 2.49968934440403E-11 3: x1-x2-2*x3+3*x4-(-1/2) = 1.74992464962997E-10
线性化方式 一 （线性）	Objective Function(Min.): 61.2500000000104 Best Estimated Parameters: x1: 14.6500000000028 x2: 0 x3: -7.19999999999717 x4: -9.8500000000014 y1: 14.6500000000028 y2: 0 y3: 7.20000000000192 y4: 9.8500000000014	Constrained Functions: 1: x1-x2-x3+x4-12 = -1.3997691894474E-12 2: x1-x2+6*x3-3*x4-1 = 2.39808173319034E-11 3: x1-x2-2*x3+3*x4-(-1/2) = -7.02726765666739E-12 4: y1-x1 = 0 5: y2-x2 = 0 6: y3-x3 = 14.3999999999991 7: y4-x4 = 19.7000000000028 8: y1-(-x1) = 29.3000000000057 9: y2-(-x2) = 0 10: y3-(-x3) = 4.74376093961837E-12 11: y4-(-x4) = 1.77635683940025E-15
线性化方式 二 （线性）	Objective Function(Min.): 61.25 Best Estimated Parameters: u1: 14.65 u2: 0 u3: 0 u4: 0 v1: 0 v2: 0 v3: 7.2 v4: 9.85	Constrained Functions: 1: u1-v1-(u2-v2)-(u3-v3)+u4-v4-12 = 0 2: u1-v1-(u2-v2)+6*(u3-v3)-3*(u4-v4)-1 = 3.5527136788005E-15 3: u1-v1-(u2-v2)-2*(u3-v3)+3*(u4-v4)-(-1/2) = 3.5527136788005E-15 4: u1-0 = 14.65 5: u2-0 = 0 6: u3-0 = 0 7: u4-0 = 0 8: v1-0 = 0

	PassParameter: x1: 14.65 x2: 0 x3: -7.2 x4: -9.85	9: v2-0 = 0 10: v3-0 = 7.2 11: v4-0 = 9.85

原模型采用非线性算法，变换后的两种模型则采用线性算法。虽然这三种方式得到的结果一致，甚至线性变换后的求解规模（参数量）还增加了一倍，线性化处理还是非常值得的，效果有保证，效率也会数量级提高，只是这个案例求解规模小，难度低，无法直观感觉到而已，当求解规模稍大时就能切实体验到差异。

45.2.2 最小最大 MinMax()函数线性化处理

MinMax()是求多个目标函数值表达式中最大值最小时的各参数值，且满足相关约束，如下案例，目标函数是求两个表达式最大值最小。

$$MinMax\ \left((x+2\cdot y+10),(3\cdot x+y+1)\right)$$
$$S.T.\ \begin{cases}0\le 3\cdot x+y\le 10\\ x,y\in[-2,2]\end{cases} \tag{45-4}$$

如何将该目标函数线性化处理？如果令$f_1=x+2\cdot y+10$，$f_2=3\cdot x+y+1$，引入一变量 f，线性化模型则可表达为：

$$Min.\ f$$
$$S.T.\ \begin{cases}0\le 3\cdot x+y\le 10\\ f\ge f_1\\ f\ge f_2\\ x,y\in[-2,2]\end{cases} \tag{45-5}$$

代码 45.2：求解代码

原模型 （非线性）	Parameter [x,y]=[-2,2]; ConstStr f1=x+2*y+10, f2=3*x+y+1; MinMax (f1,f2); 0<=3*x+y<=10;

线性化方式一
（线性）
	Algorithm = LP;
	Parameter [x,y]=[-2,2],f;
	ConstStr f1=x+2*y+10, f2=3*x+y+1;
	MinFunction f;
	0<=3*x+y<=10;
	f>=f1;
	f>=f2;

结果：

原模型 （非线性）	MinMax: 6.66666666666667 x: 0.666666666666667 y: -2 MinMax Function: 1: ((x+2*y+10)) = 6.66666666666667 2: ((3*x+y+1)) = 1	Constrained Functions: 1: 0-(3*x+y) = 0 2: 3*x+y-10 = -10
线性化方式一 （线性）	Objective Function(Min.): 6.66666666666689	Constrained Functions: 1: 0-(3*x+y) = -6.71018796083445E-13

	Best Estimated Parameters: x: 0.66666666666689 y: -2 f: 6.66666666666689	2: 3*x+y-10 = -9.99999999999933 3: f-(x+2*y+10) = 0 4: f-(3*x+y+1) = 5.66666666666622 5: x-(-2) = 2.66666666666689 6: x-2 = -1.33333333333311 7: y-(-2) = 0 8: y-2 = -4

两种方法结果是一致的，说明线性化转换方式是正确的。同样方式也可处理 MaxMin()最小最大问题，仅需注意添加附加约束时要将，比如，$f \geq f_1$及$f \geq f_2$改为$f \leq f_1$及$f \leq f_2$，也即，如果前述问题变为 MaxMin 问题：

$$MaxMin\ \big((x+2\cdot y+10),(3\cdot x+y+1)\big)$$
$$S.T.\ \begin{cases} 0 \leq 3\cdot x+y \leq 10 \\ x,y \in [-2,2] \end{cases} \tag{45-6}$$

线性化后模型则为：

$$Max.\ f$$
$$S.T.\ \begin{cases} 0 \leq 3\cdot x+y \leq 10 \\ f \leq f_1 \\ f \leq f_2 \\ x,y \in [-2,2] \end{cases} \tag{45-7}$$

其中$f_1 = x+2\cdot y+10$，$f_2 = 3\cdot x+y+1$。

45.3 运输问题

前面介绍了如何线性化含有绝对值（abs）、最大最小 MinMiax()及最小最大 MaxMin()函数的非线性模型，并给出了简单的示例，下面再以一实际规划案例进行展示。

有三节铁路货车车厢，其最大允许载重量均为分别位 7，9，19 吨，现欲用这三节车厢运输 16 个箱子。表 45.1 列出了这些箱子的重量，单位为吨，在满足每节车厢实际载重均不超过最大允许载重的前提下，如何将箱子分配到各个车厢上：

1) 使得每两两车厢装载量差额之和最小？
2) 使装载量最大的车厢总装载量最小？

表 45.1 箱子重量（W）

箱号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
重量	3.4	0.6	0.8	1.7	1.6	0.5	1.3	2.1	2.5	3.1	1.4	1.3	3.3	0.9	2.5	2.5

45.3.1 每两两车厢装载量差额之和最小

该问题意即货物分配装箱量尽可能均衡。如果设 y1、y2 和 y3 分别代表三节车厢的装货量，则按题意目标函数为：

$$Min.\ |y_1-y_2|+|y_1-y_3|+|y_2-y_3|$$

$$S.T.\begin{cases} y_1 \le 7 \\ y_2 \le 9 \\ y_3 \le 19 \end{cases} \quad (45\text{-}8)$$

解法一：非线性编程模式

设与箱子数相同的 16 个决策变量 p_i, i=1..16，对应于三节车厢，每个 pi 取整数值 1、2 或 3，分别代表三节车厢，共有 16 个整数变量，取值范围[1,3]。

下面是 Basic 和 Pascal 两种编程模式以及快捷模式代码

代码 45.3：编程模式 Basic 代码

```
Algorithm = SM2[30];
Constant w=[3.4,0.6,0.8,1.7,1.6,0.5,1.3,2.1,2.5,3.1,1.4,1.3,3.3,0.9,2.5,2.5];
Constant c=[7,9,19];
IntParameter p(16)=[1,3];
PassParameter y(3);
Minimum;
StartProgram [Basic];
Sub MainModel
dim i as integer
dim ww(3) as double
    for i = 1 to 3
        ww(i) = 0
    next
    for i = 1 to 16
        if p(i) = 1 then
            ww(1) = ww(1) + w(i)
        elseif p(i) = 2 then
            ww(2) = ww(2) + w(i)
        elseif p(i) = 3 then
            ww(3) = ww(3) + w(i)
        end if
    next
    for i = 1 to 3
        y(i) = ww(i)
    next
    ObjectiveResult = abs(ww(1)-ww(2))+abs(ww(1)-ww(3))+abs(ww(2)-ww(3)) //目标函数
    ConstrainedResult = for(i=1:3)(ww(i) <= c(i))                        //约束
End Sub
EndProgram;
```

代码 45.4：编程模式 Pascal 代码

```
Algorithm = SM2[30];
Constant w=[3.4,0.6,0.8,1.7,1.6,0.5,1.3,2.1,2.5,3.1,1.4,1.3,3.3,0.9,2.5,2.5];
Constant c=[7,9,19];
IntParameter p(16)=[1,3];
PassParameter y(3);
Minimum;
StartProgram [Pascal];
Procedure MainModel;
var i: integer;
    ww: array[1..3] of double;
Begin
    for i := 1 to 3 do
```

```
        ww[i] := 0;
    for i := 1 to 16 do
        if p[i] = 1 then ww[1] := ww[1] + w[i]
        else if p[i] = 2 then ww[2] := ww[2] + w[i]
        else if p[i] = 3 then ww[3] := ww[3] + w[i];
    for i := 1 to 3 do
        y[i] := ww[i];
    ObjectiveResult := abs(ww[1]-ww[2])+abs(ww[1]-ww[3])+abs(ww[2]-ww[3]); //目标函数
    ConstrainedResult := for(i=1:3)(ww[i] <= c[i]);
End;
EndProgram;
```

解法二：非线性快捷模式

非线性快捷模式代码如下，特别注意下面语句，看是如何定义每节车厢装载量 y_i的，不同于前述编程模式代码通过循环判断语句确定每个货物归属的车厢，而是通过“ConstStr For(i=1:3)(y[i]=Sum(j=1:16)(if(p[j]=i,w[j],0)));”，使用 if 函数判断以求出每节车厢的货物重量。

$$y_i = \sum_{j=1}^{16} \begin{cases} w_j & if\ p_j = i \\ 0 & if\ p_j \neq i \end{cases} \quad i = 1,2,3 \tag{45-9}$$

代码 45.5：快捷模式代码

```
Algorithm = SM2[30];
Constant w=[3.4,0.6,0.8,1.7,1.6,0.5,1.3,2.1,2.5,3.1,1.4,1.3,3.3,0.9,2.5,2.5];
Constant c=[7,9,19];
IntParameter p(16)=[1,3];
ConstStr For(i=1:3)(y[i]=Sum(j=1:16)(if(p[j]=i,w[j],0)));
PassParameter y(3);
MinFunction abs(y1-y2)+abs(y1-y3)+abs(y2-y3);
            For(i=1:3)(y[i]<=c[i]);
```

前述三段代码均可得到下面相同的结果。

结果：

Objective Function (Min.): 13	p11: 2
p1: 2	p12: 1
p2: 1	p13: 3
p3: 2	p14: 1
p4: 1	p15: 3
p5: 2	p16: 1
p6: 2	
p7: 2	PassParameter:
p8: 3	y1: 7
p9: 3	y2: 9
p10: 3	y3: 13.5

解法三：线性变换

前述是定义整数变量 p，维度 16，范围为[1,3]，而在此线性化过程中引入 0-1 变量 p(n,m),n 为货车车厢数，m 为货物数，n 和 m 值分别为 3 和 16，当 p(n,m)为 1 时表示 m 货物装入 n 车厢，反之，为 0 时则表示 m 货物不装入 n 车厢。求解规模为 3*16=48 个。

$$Min.\ |y_1 - y_2| + |y_1 - y_3| + |y_2 - y_3| \tag{45-10}$$

$$S.T.\ \begin{cases}\sum_{i=1}^{3} p_{i,j}=1 \quad j=1\ldots16\\ \sum_{j=1}^{16}\left(w_j\cdot p_{i,j}\right)\le c_i \quad i=1..3\end{cases}$$

其中：$y_1=\sum_{j=1}^{16}\left(w_i\cdot p_{1,j}\right),\ y_2=\sum_{j=1}^{16}\left(w_j\cdot p_{2,j}\right),\ y_3=\sum_{j=1}^{16}\left(w_j\cdot p_{3,j}\right)$

代码 45.6：非线性模式直接求解代码：

```
Constant n=3,m=16;
Constant w=[3.4,0.6,0.8,1.7,1.6,0.5,1.3,2.1,2.5,3.1,1.4,1.3,3.3,0.9,2.5,2.5];
Constant c=[7,9,19];
BinParameter p(n,m);
ConstStr For(i=1:n)(y[i]=Sum(j=1:m)(w[j]*p[i,j]));
Algorithm = SM2;
PassParameter y(n);
MinFunction abs(y1-y2)+abs(y1-y3)+abs(y2-y3);
            For(j=1:m)(Sum(i=1:3)(p[i,j])=1);
            For(i=1:n)(Sum(j=1:m)(w[j]*p[i,j])<=c[i]);
```

因为是非线性模型，且有 48 个未知待求参数，无法保证每次计算都能得到最优解，只有大概 30%的成功率获得如下结果

结果：

Objective Function (Min.): 13	p[2,2]: 1	p[3,4]: 0
p[1,1]: 0	p[2,3]: 0	p[3,5]: 1
p[1,2]: 0	p[2,4]: 0	p[3,6]: 0
p[1,3]: 0	p[2,5]: 0	p[3,7]: 1
p[1,4]: 1	p[2,6]: 0	p[3,8]: 1
p[1,5]: 0	p[2,7]: 0	p[3,9]: 0
p[1,6]: 1	p[2,8]: 0	p[3,10]: 1
p[1,7]: 0	p[2,9]: 1	p[3,11]: 0
p[1,8]: 0	p[2,10]: 0	p[3,12]: 1
p[1,9]: 0	p[2,11]: 0	p[3,13]: 1
p[1,10]: 0	p[2,12]: 0	p[3,14]: 0
p[1,11]: 1	p[2,13]: 0	p[3,15]: 0
p[1,12]: 0	p[2,14]: 0	p[3,16]: 0
p[1,13]: 0	p[2,15]: 1	
p[1,14]: 1	p[2,16]: 0	PassParameter:
p[1,15]: 0	p[3,1]: 0	y1: 7
p[1,16]: 1	p[3,2]: 0	y2: 9
p[2,1]: 1	p[3,3]: 1	y3: 13.5

参考 45.2 节线性化方式，第一种方式是引入变量 z1、z2 和 z3，并令$z_1=|y_1-y_2|,\ z_2=|y_1-y_3|,\ z_3=|y_2-y_3|$，模型公式为：

$$Min.\ z_1+z_2+z_3 \tag{45-11}$$

$$S.T.\begin{cases}\sum_{i=1}^{3} p_{i,j}=1 \quad j=1\ldots16\\ \sum_{j=1}^{16}\left(w_i\cdot p_{i,j}\right)\le c_i \quad i=1..3\\ z_1\ge y_1-y_2\\ z_1\ge -(y_1-y_2)\\ z_2\ge y_1-y_3\\ z_2\ge -(y_1-y_3)\\ z_3\ge y_2-y_3\\ z_3\ge -(y_2-y_3)\end{cases}$$

其中：$y_1=\sum_{j=1}^{16}\left(w_i\cdot p_{1,j}\right),\ y_2=\sum_{j=1}^{16}\left(w_j\cdot p_{2,j}\right),\ y_3=\sum_{j=1}^{16}\left(w_j\cdot p_{3,j}\right)$

代码 45.7：线性化方式一求解代码：

```
Constant n=3,m=16;
Constant w=[3.4,0.6,0.8,1.7,1.6,0.5,1.3,2.1,2.5,3.1,1.4,1.3,3.3,0.9,2.5,2.5];
Constant c=[7,9,19];
BinParameter p(n,m);
ConstStr For(i=1:n)(y[i]=Sum(j=1:m)(w[j]*p[i,j]));
Algorithm = LP;
PassParameter y(n);
MinFunction z1+z2+z3;
        For(j=1:m)(Sum(i=1:n)(p[i,j])=1);
        For(i=1:n)(Sum(j=1:m)(w[j]*p[i,j])<=c[i]);
        z1>=y1-y2;
        z1>=-(y1-y2);
        z2>=y1-y3;
        z2>=-(y1-y3);
        z3>=y2-y3;
        z3>=-(y2-y3);
```

结果：

```
Objective Function(Min.): 13

Best Estimated Parameters:
p[1,1]: 0
p[1,2]: 0
p[1,3]: 1
p[1,4]: 0
p[1,5]: 1
p[1,6]: 0
p[1,7]: 1
p[1,8]: 0
p[1,9]: 0
p[1,10]: 0
p[1,11]: 0
p[1,12]: 0
p[1,13]: 1
p[1,14]: 0
p[1,15]: 0
p[1,16]: 0
p[2,1]: 0
p[2,2]: 1
p[2,3]: 0
p[2,4]: 0
p[2,5]: 0
p[2,6]: 0
p[2,7]: 0
p[2,8]: 1
p[2,9]: 1
p[2,10]: 0
p[2,11]: 0
p[2,12]: 1
p[2,13]: 0
p[2,14]: 0
p[2,15]: 1
p[2,16]: 0
p[3,1]: 1
p[3,2]: 0
p[3,3]: 0
p[3,4]: 1
p[3,5]: 0
p[3,6]: 1
p[3,7]: 0
p[3,8]: 0
p[3,9]: 0
p[3,10]: 1
p[3,11]: 1
p[3,12]: 0
p[3,13]: 0
p[3,14]: 1
p[3,15]: 0
p[3,16]: 1
z1: 2
z2: 6.5
z3: 4.5

PassParameter:
y1: 7
y2: 9
y3: 13.5
```

线性化第二种方式，因为绝对值函数里不是单一的变量而是表达式，引入变量 z1、z2 和 z3，并令$z_1 = y_1 - y_2$，$z_2 = y_1 - y_3$，$z_3 = y_2 - y_3$，再引入$zz_i = u_i - v_i$，其中 u_i，v_i 均大于 0，则有$|zz_i| = u_i + v_i$，同时有约束$zz_i = z_i$

$$Min.\ (u_1 + v_1) + (u_2 + v_2) + (u_3 + v_v)$$

$$S.T.\ \begin{cases} \sum_{i=1}^{3} p_{i,j} = 1 \quad j = 1 \ldots 16 \\ \sum_{j=1}^{16} (w_j \cdot p_{i,j}) \le c_i \quad i = 1..3 \\ zz_i = z_i \quad i = 1..3 \end{cases} \tag{45-12}$$

其中：$y_1 = \sum_{j=1}^{16} (w_i \cdot p_{1,j})$，$y_2 = \sum_{j=1}^{16} (w_j \cdot p_{2,j})$，$y_3 = \sum_{j=1}^{16} (w_j \cdot p_{3,j})$

代码 45.8：线性化方式二求解代码：

```
Constant n=3,m=16;
Constant w=[3.4,0.6,0.8,1.7,1.6,0.5,1.3,2.1,2.5,3.1,1.4,1.3,3.3,0.9,2.5,2.5];
Constant c=[7,9,19];
BinParameter p(n,m);
Parameter u(n)>=0,v(n)>=0;
ConstStr z1=y1-y2,z2=y1-y3,z3=y2-y3;
ConstStr For(i=1:n)(zz[i]=u[i]-v[i]);
ConstStr For(i=1:n)(y[i]=Sum(j=1:m)(w[j]*p[i,j]));
Algorithm = LP;
PassParameter y(n);
MinFunction (u1+v1)+(u2+v2)+(u3+v3);
            For(j=1:m)(Sum(i=1:3)(p[i,j])=1);
            For(i=1:n)(Sum(j=1:m)(w[j]*p[i,j])<=c[i]);
            For(i=1:n)(zz[i]=z[i]);
```

结果：

Objective Function(Min.): 13	p[2,3]: 0	p[3,8]: 1
	p[2,4]: 1	p[3,9]: 0
Best Estimated Parameters:	p[2,5]: 0	p[3,10]: 1
p[1,1]: 0	p[2,6]: 0	p[3,11]: 0
p[1,2]: 1	p[2,7]: 0	p[3,12]: 0
p[1,3]: 1	p[2,8]: 0	p[3,13]: 1
p[1,4]: 0	p[2,9]: 1	p[3,14]: 0
p[1,5]: 1	p[2,10]: 0	p[3,15]: 1
p[1,6]: 1	p[2,11]: 1	p[3,16]: 1
p[1,7]: 1	p[2,12]: 0	u1: 0
p[1,8]: 0	p[2,13]: 0	u2: 0
p[1,9]: 0	p[2,14]: 0	u3: 0

p[1,10]: 0 p[1,11]: 0 p[1,12]: 1 p[1,13]: 0 p[1,14]: 1 p[1,15]: 0 p[1,16]: 0 p[2,1]: 1 p[2,2]: 0	p[2,15]: 0 p[2,16]: 0 p[3,1]: 0 p[3,2]: 0 p[3,3]: 0 p[3,4]: 0 p[3,5]: 0 p[3,6]: 0 p[3,7]: 0	v1: 2 v2: 6.5 v3: 4.5 PassParameter: y1: 7 y2: 9 y3: 13.5

两种线性化后的模型都可以高效稳定地获得最优解。

45.3.2 装载量最大的车厢总装载量最小

这是经典的最大最小问题。如果设 y1、y2 和 y3 分别代表三节车厢的装货量，则按题意可用两种方式构筑模型。

第一种是定义整数变量 p(m)，m=16,为货物数量，p(m)取值为 1、2 或 3，分别代表货物应该放置的车厢编号，模型描述如下：

$$MinMax.\ (y_1, y_2, y_3)$$
$$S.T.\ \ y_i \le c_i \quad i = 1,2,3 \tag{45-13}$$

其中：$$y_i = \sum_{j=1}^{16} \begin{cases} w_j & if\ p_j = i \\ 0 & if\ p_j \neq i \end{cases} \quad i = 1,2,3$$

代码 45.9：非线性求解代码

```
Algorithm = SM2[30];
Constant w=[3.4,0.6,0.8,1.7,1.6,0.5,1.3,2.1,2.5,3.1,1.4,1.3,3.3,0.9,2.5,2.5];
Constant c=[7,9,19];
IntParameter p(16)=[1,3];
ConstStr For(j=1:3)(y[j]=Sum(i=1:16)(if(p[i]=j,w[i],0)));
PassParameter y(3);
MinMax y(3);
        For(i=1:3)(y[i]<=c[i]);
```

所有快捷模式代码都可以在编程模式下实现，编程模式代码如下。

代码 45.10：非线性编程模式求解代码

```
Algorithm = SM2[30];
Constant w=[3.4,0.6,0.8,1.7,1.6,0.5,1.3,2.1,2.5,3.1,1.4,1.3,3.3,0.9,2.5,2.5];
Constant c=[7,9,19];
IntParameter p(16)=[1,3];
PassParameter y(3);
Minimum;
StartProgram [Pascal];
Procedure MainModel;
var i: integer;
    ww: array[1..3] of double;
Begin
    for i := 1 to 3 do
        ww[i] := 0;
```

```
    for i := 1 to 16 do
        if p[i] = 1 then ww[1] := ww[1] + w[i]
        else if p[i] = 2 then ww[2] := ww[2] + w[i]
        else if p[i] = 3 then ww[3] := ww[3] + w[i];
    for i := 1 to 3 do
        y[i] := ww[i];
    ObjectiveResult := Max(ww[1], Max(ww[2], ww[3])); //目标函数
    ConstrainedResult := for(i=1:3)(ww[i] <= c[i]);
End;
EndProgram;
```

第二种方式定义 0-1 变量 p(n,m),n=3,车厢数，m=16，货物数，模型描述如下：

$$MinMax.\ (y_1, y_2, y_3)$$

$$S.T.\ \begin{cases} \sum_{i=1}^{3} p_{i,j} = 1 \quad j = 1 \ldots 16 \\ \sum_{i=1}^{3} y_i \le c_i \end{cases} \tag{45-13}$$

其中：

$$y_i = \sum_{j=1}^{16} w_j \cdot p_{i,j} \quad i = 1 \ldots 3$$

代码 45.11：非线性求解代码：

```
Constant w=[3.4,0.6,0.8,1.7,1.6,0.5,1.3,2.1,2.5,3.1,1.4,1.3,3.3,0.9,2.5,2.5];
Constant c=[7,9,19];
BinParameter p(3,16);
ConstStr For(i=1:3)(y[i]=Sum(j=1:16)(w[j]*p[i,j]));
Algorithm = SM2;
PassParameter y(3);
MinMax y(3);
            For(j=1:16)(Sum(i=1:3)(p[i,j])=1);
            For(i=1:3)(Sum(j=1:16)(w[j]*p[i,j])<=c[i]);
```

上述三段代码求解的最好结果如下，第一种模式的两段求解代码的成功率几乎为 100%，作为对比，第二种模式求解规模为 48，是第一种模式的三倍，获取最优结果的成功率也降至不足 5 成。

结果：

MinMax: 13.5	p11: 1
p1: 3	p12: 3
p2: 2	p13: 3
p3: 2	p14: 3
p4: 3	p15: 2
p5: 3	p16: 1
p6: 2	
p7: 3	PassParameter:
p8: 2	y1: 7
p9: 2	y2: 9
p10: 1	y3: 13.5

前述第一种非线性求解模式效率效果都比第二种模式强，但由于含有 if 判断语句，无法线性化处理，下面线性化模型基于上述第二种，引入新变量 z，模型如下：

$$Min.\ z$$

$$S.T.\ \begin{cases} \sum_{i=1}^{3} p_{i,j} = 1 \quad j = 1 \ldots 16 \\ \sum_{i=1}^{3} y_i \le c_i \\ z \ge y_i \quad i = 1,2,3 \end{cases} \tag{45-14}$$

其中：$y_i = \sum_{j=1}^{16} w_j \cdot p_{i,j} \quad i = 1 \ldots 3$

代码 45.12：线性化后求解代码：

```
Constant w=[3.4,0.6,0.8,1.7,1.6,0.5,1.3,2.1,2.5,3.1,1.4,1.3,3.3,0.9,2.5,2.5];
Constant c=[7,9,19];
BinParameter p(3,16);
ConstStr For(i=1:3)(y[i]=Sum(j=1:16)(w[j]*p[i,j]));
Algorithm = LP;
PassParameter y(3);
MinFunction z;
        For(j=1:16)(Sum(i=1:3)(p[i,j])=1);
        For(i=1:3)(Sum(j=1:16)(w[j]*p[i,j])<=c[i]);
        for(i=1:3)(z>=y[i]);
```

结果:

```
Objective Function(Min.): 13.5

Best Estimated Parameters:
p[1,1]: 0
p[1,2]: 0
p[1,3]: 0
p[1,4]: 0
p[1,5]: 1
p[1,6]: 0
p[1,7]: 0
p[1,8]: 0
p[1,9]: 0
p[1,10]: 1
p[1,11]: 1
p[1,12]: 0
p[1,13]: 0
p[1,14]: 1
p[1,15]: 0
p[1,16]: 0
p[2,1]: 0
p[2,2]: 1
p[2,3]: 0
p[2,4]: 0
p[2,5]: 0
p[2,6]: 1
p[2,7]: 0
p[2,8]: 1
p[2,9]: 0
p[2,10]: 0
p[2,11]: 0
p[2,12]: 0
p[2,13]: 1
p[2,14]: 0
p[2,15]: 0
p[2,16]: 1
p[3,1]: 1
p[3,2]: 0
p[3,3]: 1
p[3,4]: 1
p[3,5]: 0
p[3,6]: 0
p[3,7]: 1
p[3,8]: 0
p[3,9]: 1
p[3,10]: 0
p[3,11]: 0
p[3,12]: 1
p[3,13]: 0
p[3,14]: 0
p[3,15]: 1
p[3,16]: 0
z: 13.5

PassParameter:
y1: 7
y2: 9
y3: 13.5
```

上述不同代码都可得到相同的最优结果。

45.4 小结

同一个运筹规划问题，有多种模型构筑方式，哪种效率更高？效果更有保证？是非常重要也是值得探讨的，好的模型事半功倍值，反之则事倍功半。

基于 1stOpt，以实际案例演示如何将含有绝对值或最大最小函数规划模型线性化的方法。在构筑实际应用模型时，如果能将非线性模型线性化，那么最好变换为线性模型再去求解，不仅效果有保证，效率也会提高很多；当然，能够线性化的函数仅为少数，如本文讨论的绝对值、最大最小及最小最大函数，另外还有分段函数等。如果无法线性化，则只能按非线性模型去求解了。