

运筹学上机实验指导书

重庆交通大学管理学院

目录

绪论	1
1. 简单的模型表示.....	1
2. 方便的数据输入和输出选择.....	2
3. 强大的求解器.....	2
4. 交互式模型或创建 Turn-key 应用程序	2
5. 广泛的文件和 HELP 功能	2
第一章 运筹学上机实验指导	3
§1.1 中小型线性规划模型的计算机求解	3
§1.2 大型线性规划模型的编程计算机求解	8
§1.3 线性规划的灵敏度分析	11
§1.4 运输问题数学模型的计算机求解	16
§1.5 目标规划数学模型的计算机求解	19
§1.6 整数规划数学模型的计算机求解	21
§1.7 指派问题的计算机求解	23
§1.8 最短路问题的计算机求解	24
§1.9 最大流问题的计算机求解	26
第二章 LINGO 软件基础及应用	29
§2.1 原始集(primitive set)和派生集(derived set)与集的定义	29
§2.2 LINGO 中的函数与目标函数和约束条件的表示	32
§2.3 LINGO 中的数据	44
§2.4 LINDO 简介	48
第三章 运筹学上机实验及要求	53
实验一.中小型线性规划模型的求解与 Lingo 软件的使用	53
实验二.中小型运输问题数学模型的 Lingo 软件求解。	53
实验三.大型线性规划模型的编程求解。	53
实验四.运输问题数学模型的 Lingo 编程求解。	54
实验五.分支定界法上机实验	54
实验六.整数规划、0-1 规划和指派问题的计算机求解	54
实验七: 最短路问题的计算机求解	55
实验八:最大流问题的计算机求解	55
实验九:运筹学综合实验	55
露天矿生产的车辆安排 (CMCM2003B)	56

绪论

运筹学是研究资源最优规划和使用的数量化的管理科学，它是广泛利用现有的科学技术和计算机技术，特别是应用数学方法和数学模型，研究和解决生产、经营和经济管理活动中的各种优化决策问题。

运筹学通常是从实际问题出发，根据决策问题的特征，建立适当的数学模型，研究和分析模型的性质和特点，设计解决模型的方法或算法来解决实际问题，是一门应用性很强的科学技术。运筹学的思想、内容和研究方法广泛应用于工程管理、工商企业管理、物流和供应链管理、交通运输规划与管理等各行各业，也是现代管理科学和经济学等许多学科研究的重要基础。

在解决生产、经营和管理活动中的实际决策问题时，一般都是建立变量多、约束多的大型复杂的运筹学模型，通常都只能通过计算机软件才能求解，因此，学习运筹学的计算机求解和进行上机实验，就是运筹学教学的重要组成部分。

现在求解各类运筹学模型的软件多种，主要有 Microexcel, Matlab, LINDO, LINGO, WinQSB 和英国运筹学软件 Dash-Xpress。Microexcel 主要利用规划求解来解线性规划模型，WinQSB 功能比较齐全，但是主要适合解决规模较小的运筹学模型，英国运筹学软件 Dash-Xpress 现在在中国的使用率不高，Matlab 是通过矩阵的方法解决线性规划，对非线性规划和其它运筹学模型特别是大规模的模型的输入不太方便，。而 LINGO 和 LINDO 是使用最广泛的运筹学专业软件，前者功能强大，能解决几乎所有的运筹学优化模型，后者主要功能是线性规划模型的求解。在 LINGO 中模型的输入和编程都比较方便，可解决大规模的运筹学模型。因此，本课程的教学就是以 LINGO 为主，适当补充 Excel 和 LINDO 作为运筹学上机软件，后者的优势主要在于能获得最优单纯形表以进行更全面地灵敏度分析。

LINGO 是用来求解线性和非线性优化问题的简易工具。LINGO 内置了一种建立最优化模型的语言，可以简便地表达大规模问题，利用 LINGO 高效的求解器可快速求解并分析结果。

LINGO 全称是 Linear Interactive and General Optimizer 的缩写---交互式的线性和通用优化求解器。它是一套设计用来帮助您快速,方便和有效的构建和求解线性,非线性,和整数最优化模型的功能全面的工具.包括功能强大的建模语言,建立和编辑问题的 全功能环境,读取和写入 Excel 和数据库的功能,和一系列完全内置的求解程序.

运行环境： Win9x/NT/2000/XP/2003/Vista/Win7

软件类别： 国外软件/工具软件/计算工具

软件语言： 英文

LINGO 是使建立和求解线性、非线性和整数最佳化模型更快更简单更有效率的综合工具。LINGO 提供强大的语言和快速的求解引擎来阐述和求解最佳化模型。 LINGO 具有如下的优势：

1. 简单的模型表示

LINGO 可以将线性、非线性和整数问题迅速得予以公式表示，并且容易阅读、了解和修改。LINGO 的建模语言允许您使用汇总和下标变量以一种易懂的

直观的方式来表达模型，非常类似您在使用纸和笔。模型更加容易构建，更容易理解，因此也更容易维护。

2. 方便的数据输入和输出选择

LINGO 建立的模型可以直接从数据库或工作表获取资料。同样地，LINGO 可以将求解结果直接输出到数据库或工作表。使得您能够在您选择的应用程序中生成报告。

3. 强大的求解器

LINGO 拥有一整套快速的,内建的求解器用来求解线性的,非线性的(球面&非球面的),二次的,二次约束的,和整数优化问题.您甚至不需要指定或启动特定的求解器,因为 LINGO 会读取您的方程式并自动选择合适的求解器。

4. 交互式模型或创建 Turn-key 应用程序

您能够在 LINGO 内创建和求解模型,或您能够从您自己编写的应用程序中直接调用 LINGO.对于开发交互式模型,LINGO 提供了一整套建模环境来构建,求解和分析您的模型.对于构建 turn-key 解决方案,LINGO 提供的可调用的 DLL 和 OLE 界面能够从用户自己写的程序中被调用.LINGO 也能够从 Excel 宏或数据库应用程序中被直接调用。

5. 广泛的文件和 HELP 功能

安装好了的 LINGO，启动后的界面如下图 1 所示，即可输入求解运筹学模型的程序。

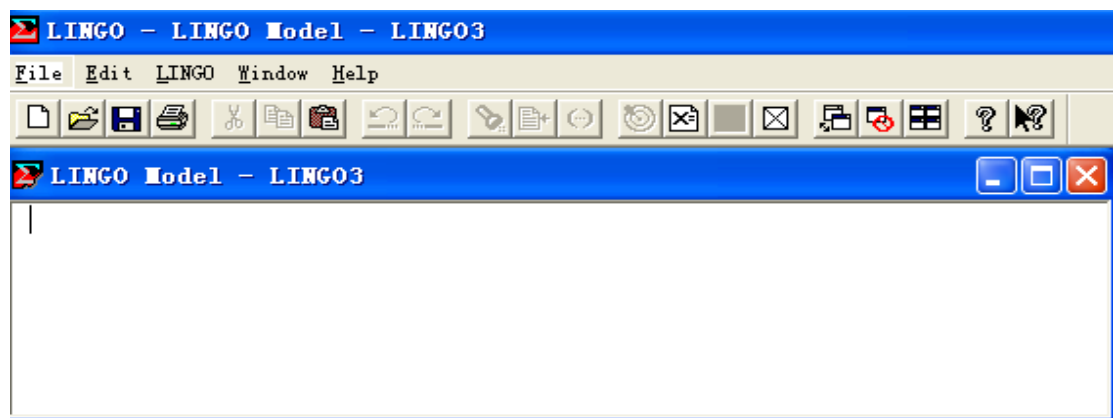


图 1

第一章 运筹学上机实验指导

§1.1 中小型线性规划模型的计算机求解

对于小型线性规划模型的求解, LINGO 中可以用一种与线性规划的数学模型及其类似的方式直接输入模型来求解, 简单方便。

例 1.1 求解下面的线性规划

$$\begin{aligned} \max \quad & z=2x_1+3x_2 \\ & x_1+2x_2 \leq 8 \\ & 4x_1 \leq 16 \\ & 4x_2 \leq 16 \\ & x_1, x_2 \geq 0 \end{aligned}$$

LINGO 中的输入的代码如图 2 所示, 这种输入方式的优势在于适合 LINDO 系统。

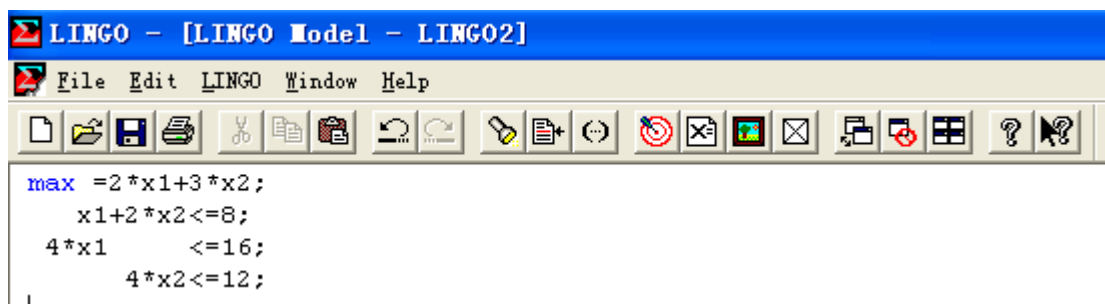


图 2

注 1: LINGO 中输入的代码和线性规划模型的差异如下:

- (1) $\max z \rightarrow \max, \min z \rightarrow \min$;
- (2) 每一行(包括目标函数)用英文的分号结束;
- (3) 数与变量的乘积用*表示;
- (4) 不等号 \leq 和 \geq 用 $<=$ 和 $>=$ 或 $<$ 和 $>$ 表示;
- (5) LINGO 系统默认所有的变量非负, 因此非负变量的约束可省略, 而非正变量和自由变量要用 $x1 \leq 0$ 和 $@free(x2)$ 表示;
- (6) LINGO 中不能输入下标, $x_1 \rightarrow x1$ 。

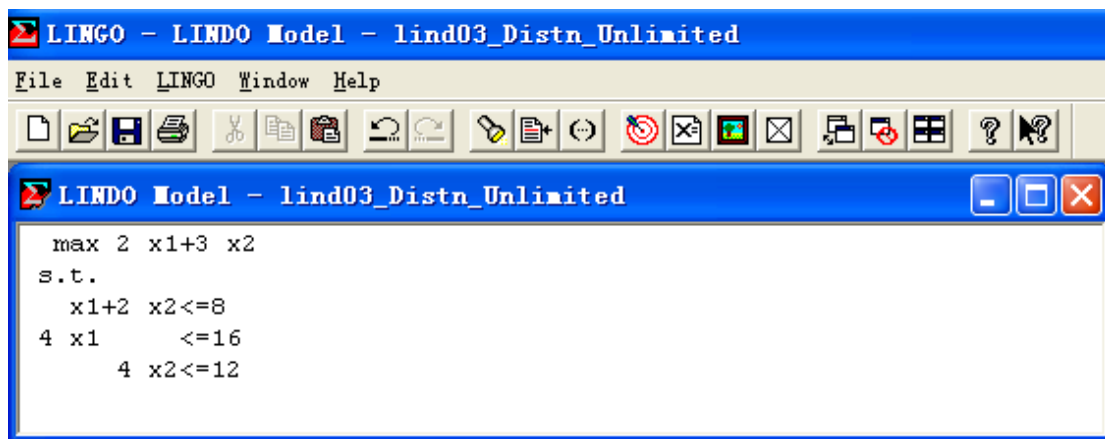


图 3

注 2：例 1.1 的模型求解还可以按图 4 的方式输入代码求解。此时 LINGO 中输入的代码和线性规划模型的除注 1 的相关差异外，还有如下不同：

- (1) 数与变量的乘积，乘号用空格表示；
- (2) 约束条件之前用 s.t.或 subject to 表示后面是约束；
- (3) 每行后面不用分号结束；
- (4) 这种输入法的好处是和 LINDO 的输入一致，可以直接在 LINDO 中求解，做灵敏度分析较方便，也能得到最优单纯形表。

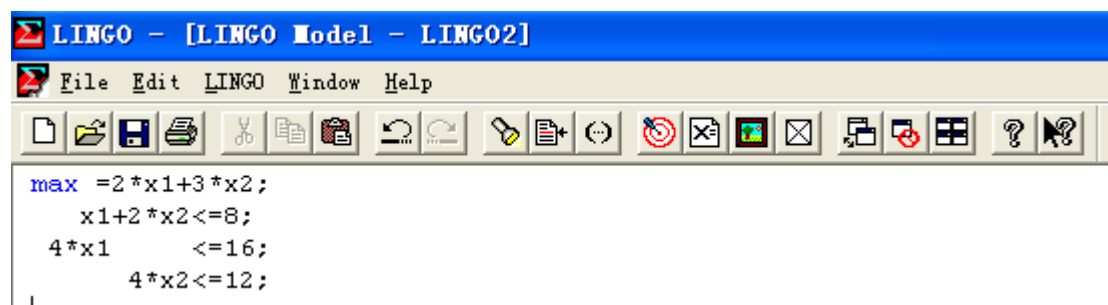



图 4

点菜单栏的 LINGO→Solver，或直接点工具栏上的 ，可得求解结果即解的状况(Solver Status)和解报告(Solution Report):

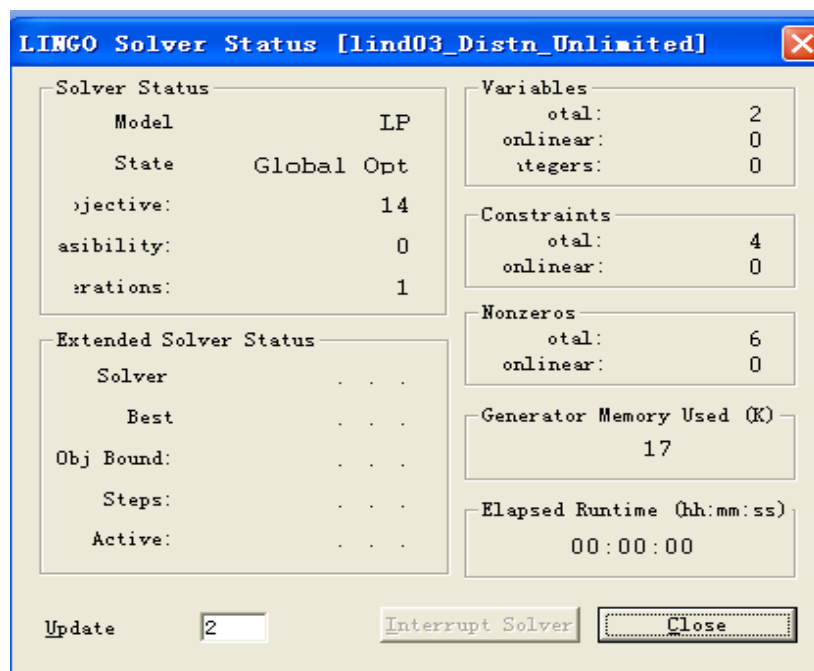


图 5

关于图 5 的 Solver Status 的注释如下：

- (1) Model(模型) LP(线性规划 Linear programming，其它模型还有非线性规划 NLP(Nonlinear programming)，整数线性规划 ILP(Integer)，整数非线性规划 INLP)
- (2) State(状态) Global Opt(整体最优解 Global optimal solution，线性规划的最

- 优解都是整体最优解，非线性规划有局部最优解(Local Opt)和整体最优解之分，其它状态还有无可行解(Infeasible)图 7 和无界解(Unbounded) 图 8)
- (3) Objective, 目标函数值为 14, 由于处于最优解状态, 所以这里表示最优值为 14。
- (4) Infeasibility 0, 不可行性 0, 表示此时有可行解, 否则没有可行解。
- (5) Iteration 1, 表示迭代了 1 步求得最优解。
- (6) Extended Solver Status, 表示扩展的解的状况, 主要用于整数规划和非线性规划。
- (7) Variables, 表示变量, Total 2, 表示总决策变量 2 个, 非线性(Nonlinear)变量和整数(Integer)变量都是 0 个。
- (8) Constraints, 表示约束, Total 4, 表示包括目标函数一共 4 个约束, 非线性(Nonlinear)约束 0 个。
- (9) Nonzeros, 表示非零系数, Total 6, 表示包括目标函数和约束条件中变量的非零系数 6 个, 右端常数项不算。

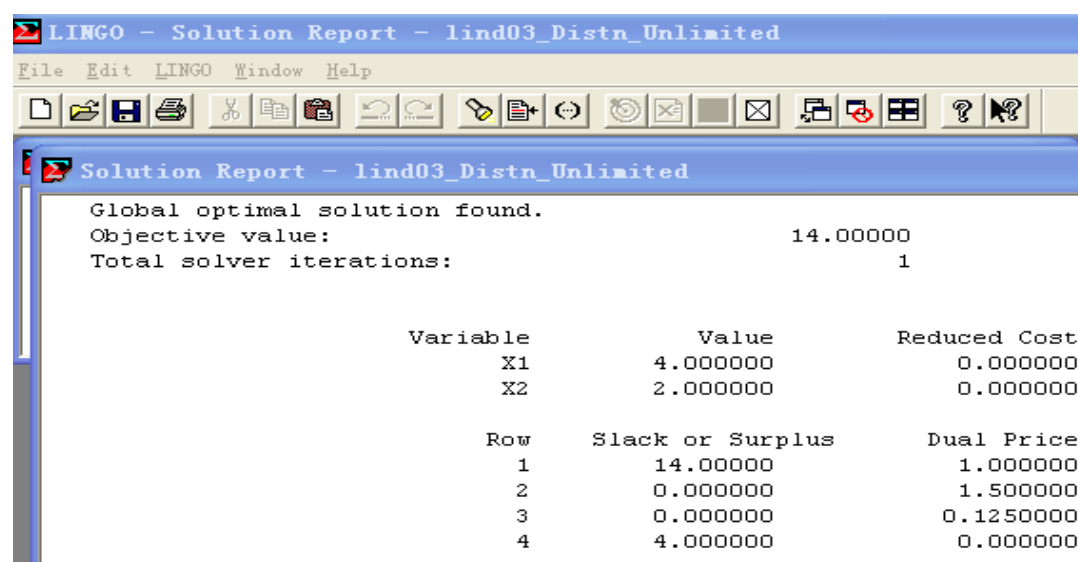


图 6

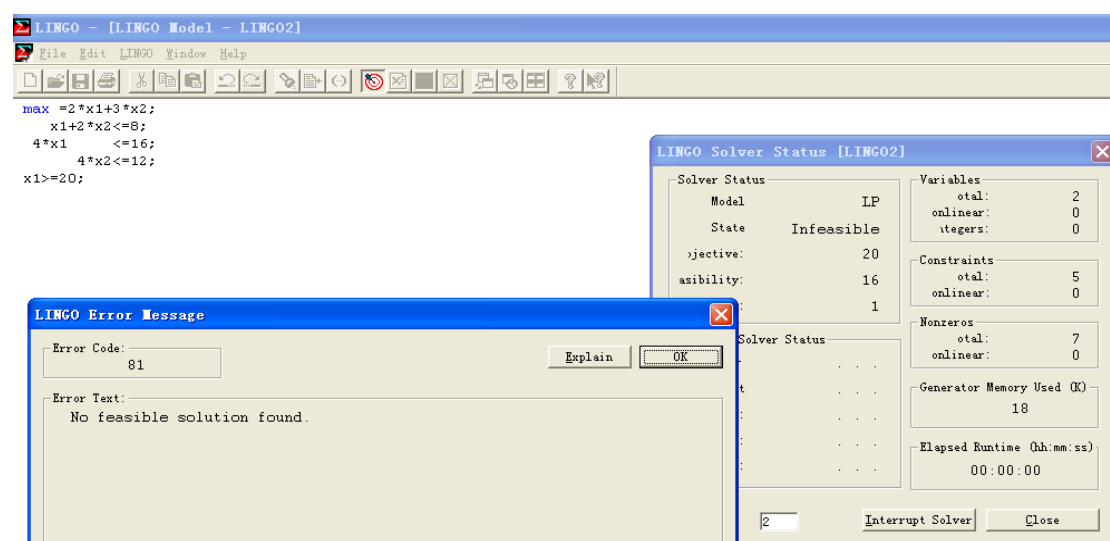


图 7

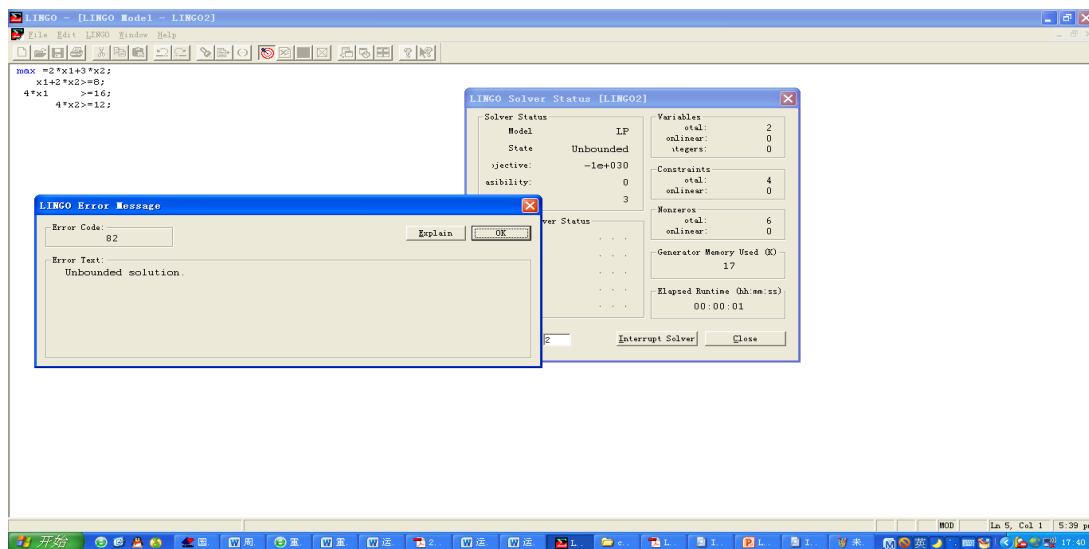


图 8

关于图 6 的 Solution Report 的注释如下:

- (1) Global optimal solution found. 整体最优解被找到。
- (2) Objective value: 14.000000. 最优值为14。
- (3) Total solver iterations: 1. 求解的总迭代步数为1步。

(4) Variable	Value	Reduced Cost
X1	4.000000	0.000000
X2	2.000000	0.000000

最优解的变量 $x_1=4.000000$, $x_2=2.000000$ 。

- (5) Reduced Cost: 表示减少的成本, 即最小化问题的最优目标函数中各变量的检验数, 即在其它变量不变时, 该变量减少一个单位, 目标费用减少的数量如图8。对于最大化问题, 是最优目标函数中各变量的检验数的相反数, 表示当该变量增加一个单位时目标函数减少的数量如图9。这里由于上面 x_1 和 x_2 为取值非零的基变量, 所以检验数为零。Reduced Cost 为在最优解时, 最小化问题中变量的检验数, 最大化问题中变量检验数的相反数。

(6) Row	Slack or Surplus	Dual Price
1	14.000000	1.000000
2	0.000000	1.500000
3	0.000000	0.1250000
4	4.000000	0.000000

Slack or Surplus 表示松弛或剩余变量, 即将最优解带入各个约束条件后, 左边比右边小的或大的数量, 表示在最优方案中, 剩余或超过的资源数量。注意, 这里第一行表示目标函数, 其松弛或剩余变量和对偶价格都没有意义。

- (7) Dual Price, 对偶价格, 即最大化问题中对偶变量的最优解的值。如图 9 所示, 对于最小化问题, 对偶价格为对偶变量的最优解的值的相反数。

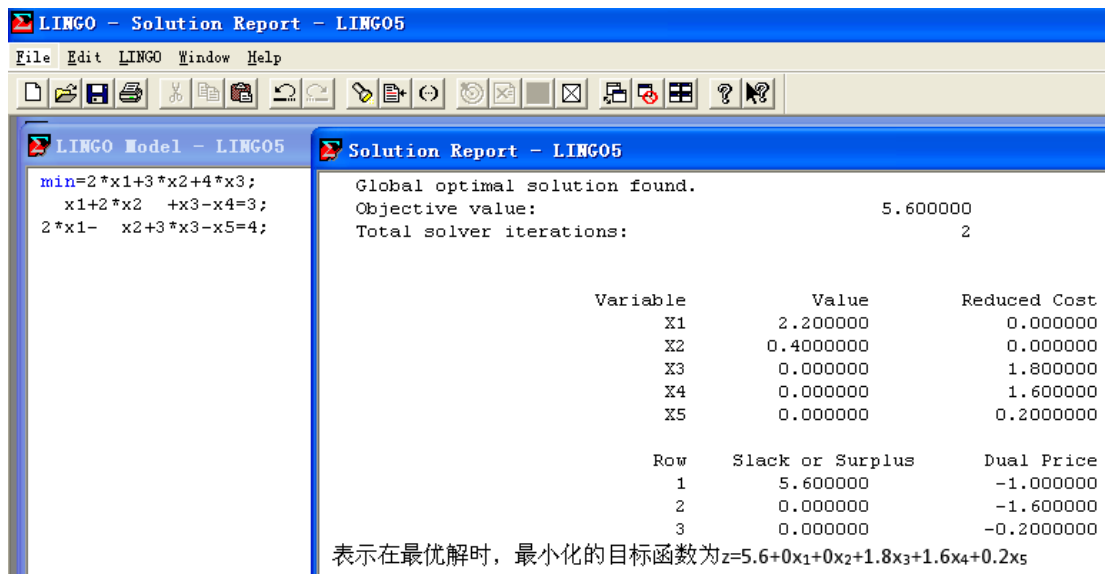


图 9

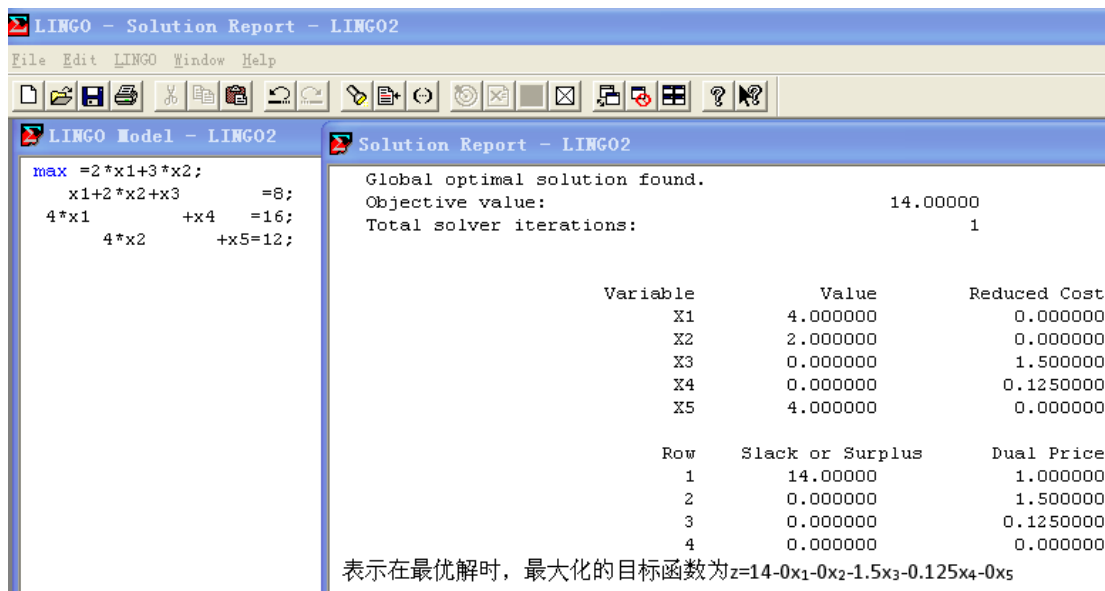


图 10

例1.2 求解下面线性规划的数学模型

$$\min z = -3x_1 + 4x_2 - 2x_3 + 5x_4;$$

$$4x_1 - x_2 + 2x_3 - x_4 = -2;$$

$$x_1 + x_2 + 3x_3 - x_4 \leq 14;$$

$$-2x_1 + 3x_2 - x_3 + 2x_4 \geq 2;$$

$$x_1, x_2, x_3 \geq 0, x_4 \text{ 无约束};$$

LINGO 中输入如下的代码:

$$\min = -3 * x_1 + 4 * x_2 - 2 * x_3 + 5 * x_4;$$

$$4 * x_1 - x_2 + 2 * x_3 - x_4 = -2;$$

$$x_1 + x_2 + 3 * x_3 - x_4 \leq 14;$$

$$-2 * x_1 + 3 * x_2 - x_3 + 2 * x_4 \geq 2;$$

$$@free(x_4);$$

求解可得解报告:

Global optimal solution found.

Objective value: 2.000000

Total solver iterations: 0

Variable	Value	Reduced Cost
X1	0.000000	15.500000
X2	8.000000	0.000000
X3	0.000000	8.500000
X4	-6.000000	0.000000
Row	Slack or Surplus	Dual Price
1	2.000000	-1.000000
2	0.000000	4.500000
3	0.000000	0.500000
4	10.000000	0.000000

§1.2 大型线性规划模型的编程计算机求解

教学过程所见到的运筹学模型大多是小型的,但是,在解决生产和经营管理活动中的实际时,建立的通常是含有很多和变量和约束条件的模型,用前面的方法,经常要花费大量的时间来输入代码或模型,下面介绍编程的方法,对于解决大型复杂的模型,效果显著。

下面是求解例1的线性规划的LINGO程序。

例2.1 用LINGO编程求解例1.1的线性规划模型

!定义变量与常量,给出了值的为常量;

sets:

is/1..3/:b;

js/1..2/:c,x;

links(is,js):a;

endsets

!目标函数;

max=@sum(js(J):c(J)*x(J));

!约束条件;

@for(is(I): @sum(js(J):a(I,J)*x(J))<=b(I));

!指定常量的值;

data:

!直接输入数据;

c=2 3;

b=8 16 12;

a=1 2

4 0

0 4;

end data

end

求解可得Solution Report

Global optimal solution found.

Objective value: 14.000000

Total solver iterations: 1

Variable	Value	Reduced Cost
B(1)	8.000000	0.000000
B(2)	16.000000	0.000000
B(3)	12.000000	0.000000
C(1)	2.000000	0.000000
C(2)	3.000000	0.000000
X(1)	4.000000	0.000000
X(2)	2.000000	0.000000
A(1, 1)	1.000000	0.000000
A(1, 2)	2.000000	0.000000
A(2, 1)	4.000000	0.000000
A(2, 2)	0.000000	0.000000
A(3, 1)	0.000000	0.000000
A(3, 2)	4.000000	0.000000

Row	Slack or Surplus	Dual Price
1	14.000000	1.000000
2	0.000000	1.500000
3	0.000000	0.1250000
4	4.000000	0.000000

这里以!开始和分号结束的语句为注释语句，该程序的求解方法和解报告与小型模型类似，只是编程的解报告会把所有的系数也表述出来而已。

从例3可以看出，一个LINGO的程序由四个部分组成。

1. 以“sets:”开始，以“endsets”结束的语句定义模型中出现的变量集。
2. 以sets中定义的变量和常量来表达目标函数。
3. 以sets中定义的变量和常量来表达全部的约束条件。
4. 以“data:”开始，以“end data”结束的语句给常量指定数值。

第二章详细解释每一部分的含义及如何表达对应的数学模型。

例2.2 求解下面线性规划的数学模型；

min $z = -3x_1 + 4x_2 - 2x_3 + 5x_4$;

$4x_1 - x_2 + 2x_3 - x_4 = -2$;

$x_1 + x_2 + 3x_3 - x_4 \leq 14$;

$-2x_1 + 3x_2 - x_3 + 2x_4 \geq 2$;

$x_1, x_2, x_3 \geq 0$, x_4 无约束;

编程如下：

!定义变量与常量，给出了值的为常量；

```

sets:
is/1..3/:b;
js/1..4/:c,x;
links(is,js):a;
endsets
!目标函数;
min=@sum(js(J):c(J)*x(J));
!约束条件;
@sum(js(J):a(1,J)*x(J))=b(1);
@sum(js(J):a(2,J)*x(J))<=b(2);
@sum(js(J):a(3,J)*x(J))>= b(3);
!自由变量;
@free(x(4));
!指定常量的值;
data:
c=-3 4 -2 5;
b=-2 14 2;
a=4 -1 2 -1
    1 1 3 -1
    -2 3 -1 2;
end data
!结束;
end

```

求解可得解报告:

Global optimal solution found.

Objective value: 2.000000

Total solver iterations: 2

Variable	Value	Reduced Cost
B(1)	-2.000000	0.000000
B(2)	14.00000	0.000000
B(3)	2.000000	0.000000
C(1)	-3.000000	0.000000
C(2)	4.000000	0.000000
C(3)	-2.000000	0.000000
C(4)	5.000000	0.000000
X(1)	0.000000	15.50000
X(2)	8.000000	0.000000
X(3)	0.000000	8.500000
X(4)	-6.000000	0.000000
A(1, 1)	4.000000	0.000000
A(1, 2)	-1.000000	0.000000
A(1, 3)	2.000000	0.000000
A(1, 4)	-1.000000	0.000000
A(2, 1)	1.000000	0.000000

A(2, 2)	1.000000	0.000000
A(2, 3)	3.000000	0.000000
A(2, 4)	-1.000000	0.000000
A(3, 1)	-2.000000	0.000000
A(3, 2)	3.000000	0.000000
A(3, 3)	-1.000000	0.000000
A(3, 4)	2.000000	0.000000
Row	Slack or Surplus	Dual Price
1	2.000000	-1.000000
2	0.000000	4.500000
3	0.000000	0.500000
4	10.00000	0.000000

§1.3 线性规划的灵敏度分析

在求解了一个线性规划的模型的时候，如果是编程输入的模型，还可以通过 LINGO 中的命令显示线性规划的数学模型。

例 3.1 通过图和操作，可显示例 2.1 LINGO 程序的数学模型。

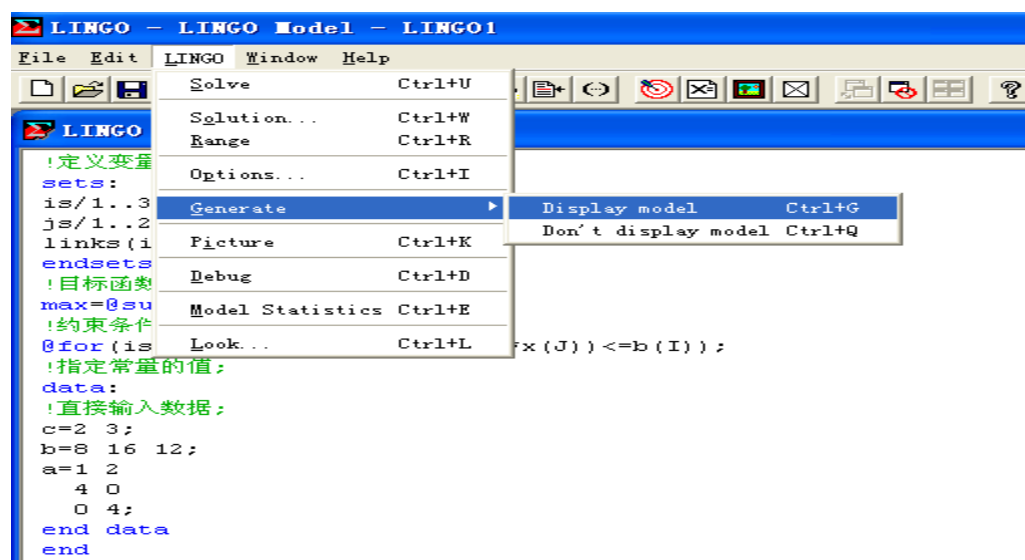


图 11

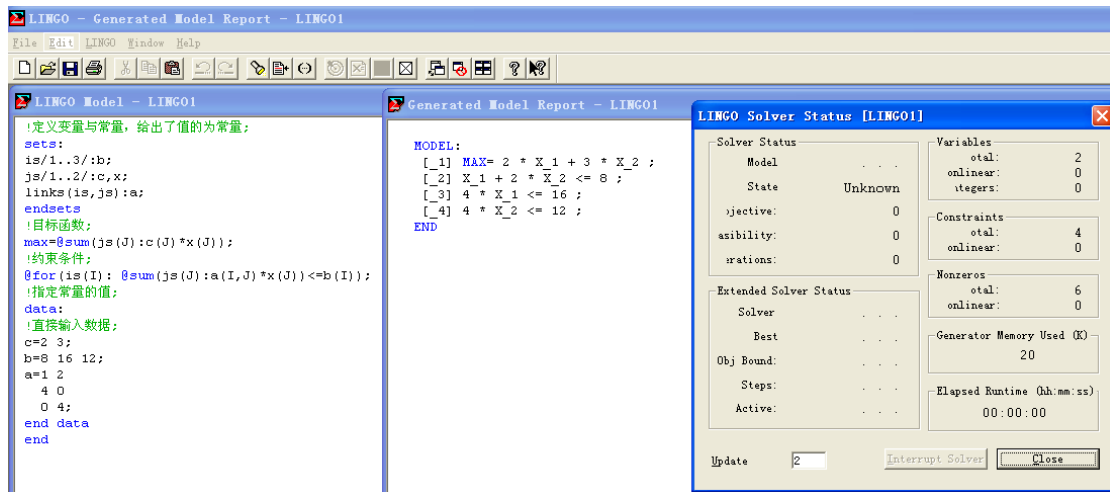


图 12

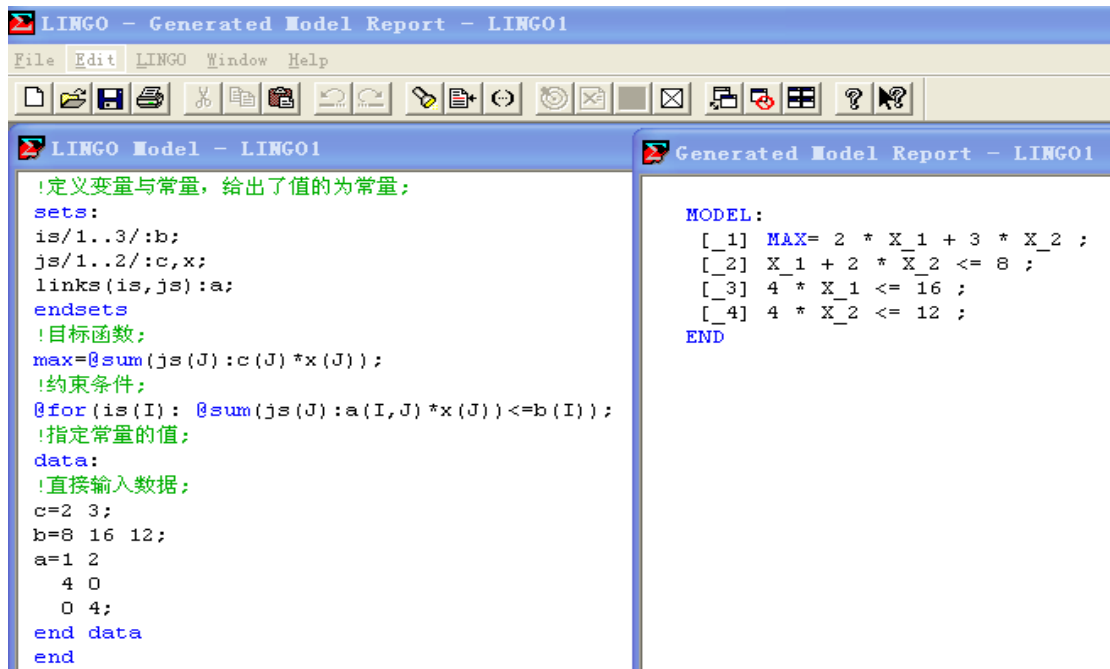


图 13

MODEL:

```

[_1] MAX= 2 * X_1 + 3 * X_2 ;
[_2] X_1 + 2 * X_2 <= 8 ;
[_3] 4 * X_1 <= 16 ;
[_4] 4 * X_2 <= 12 ;

```

END

只是系统默认的非负约束没有显示，下图表明自由变量和非正变量都会显示出来。

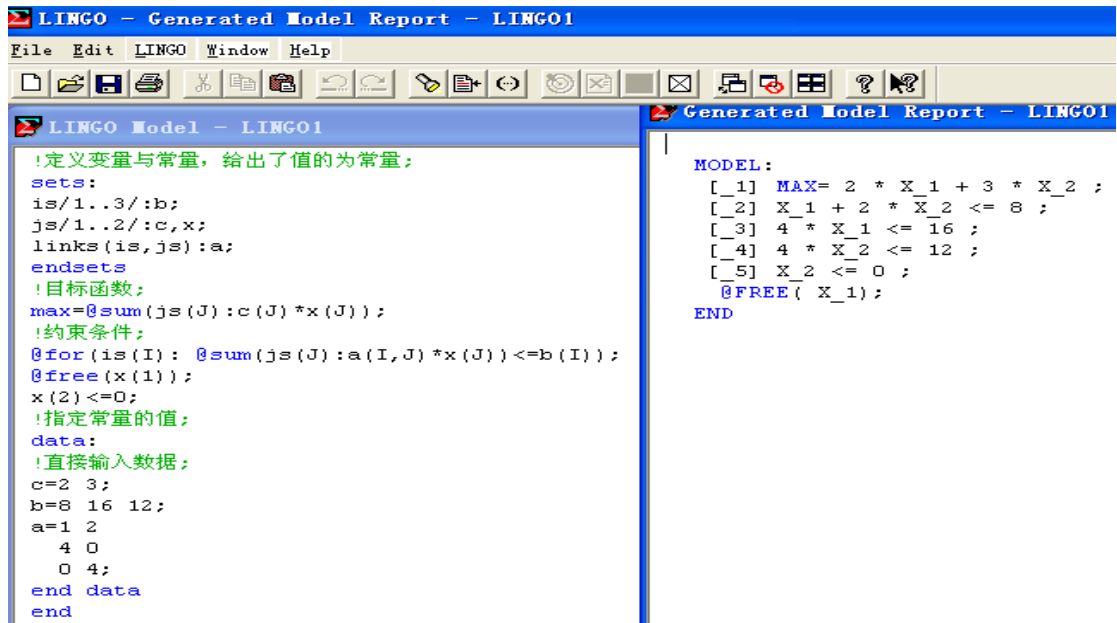


图 14

下面的图演示了对线性规划的灵敏度分析
首先求解一个线性规划模型，然后选中“prices & Ranges”

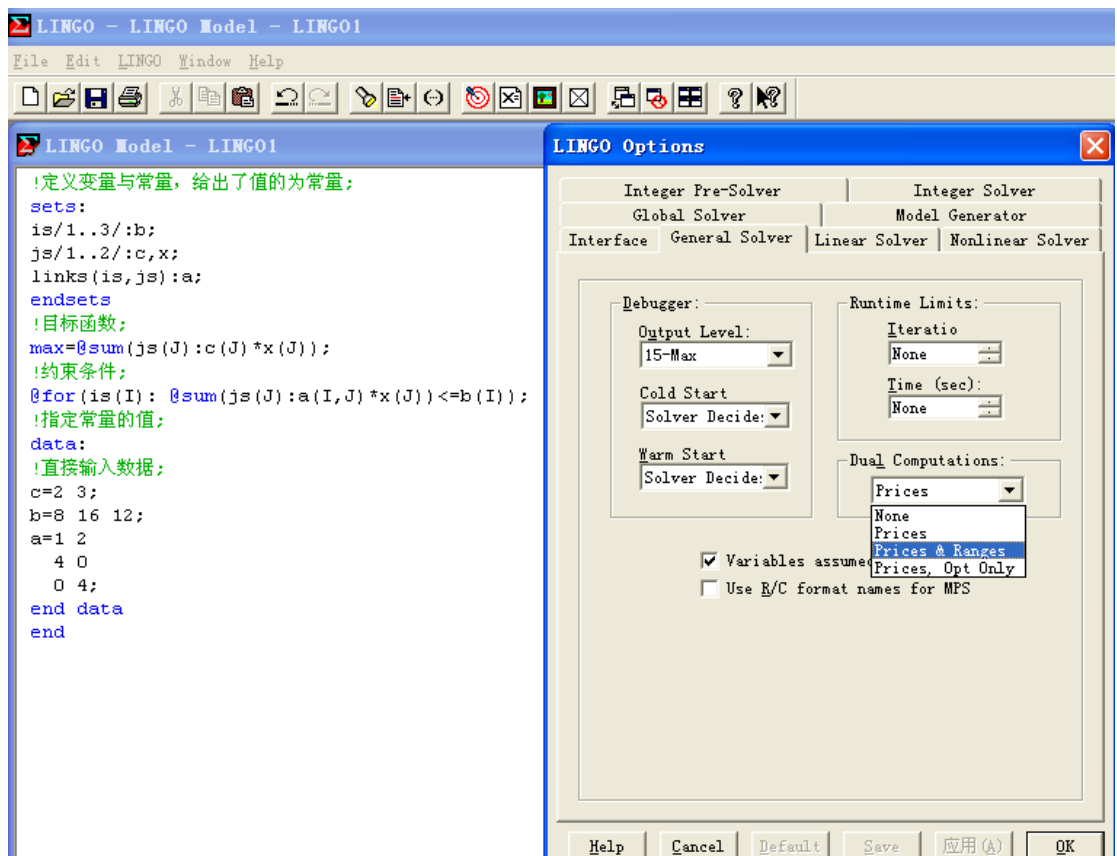


图 15

然后在菜单 LINGO→Ranges



图 16

点击 **Ranges**，得到在最优基或最优解不变时，单个价值系数和右端系数变化范围的灵敏度分析结果。

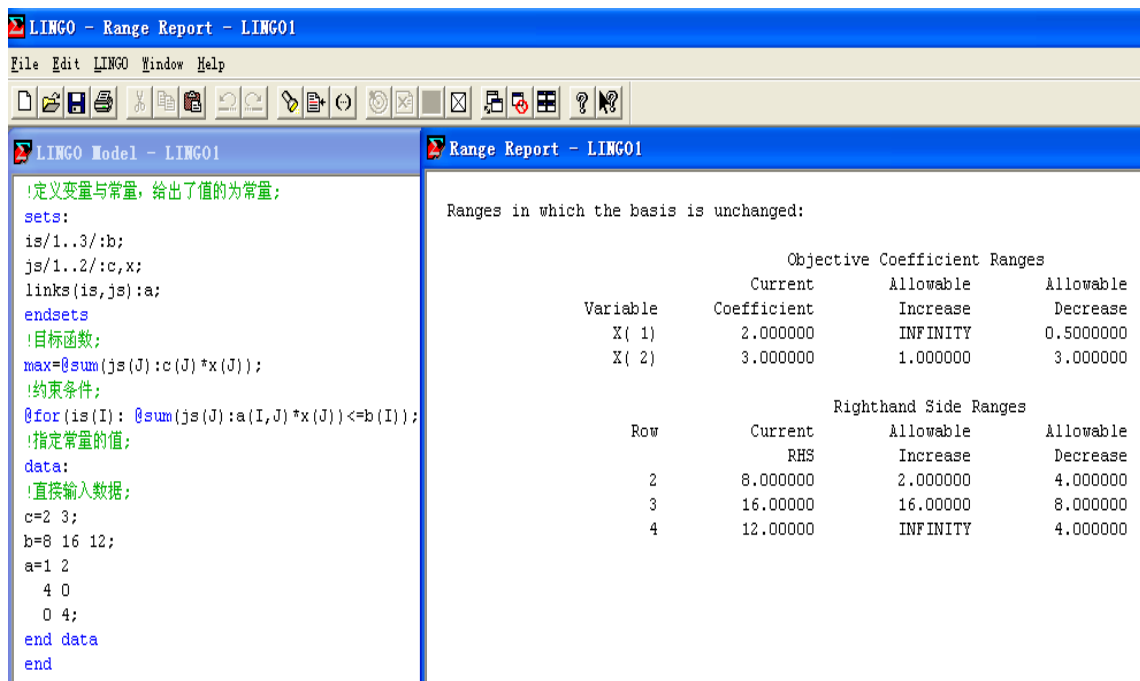


图 17

Ranges in which the basis is unchanged:

Objective Coefficient Ranges			
Variable	Current Coefficient	Allowable Increase	Allowable Decrease

X(1)	2.000000	INFINITY	0.5000000
X(2)	3.000000	1.000000	3.000000

Righthand Side Ranges			
Row	Current RHS	Allowable Increase	Allowable Decrease
2	8.000000	2.000000	4.000000
3	16.00000	16.00000	8.000000
4	12.00000	INFINITY	4.000000

LINDO 中也可以作灵敏度分析，一般在求解了线性规划模型后，自动出现是否进行灵敏度分析的对话框，如图

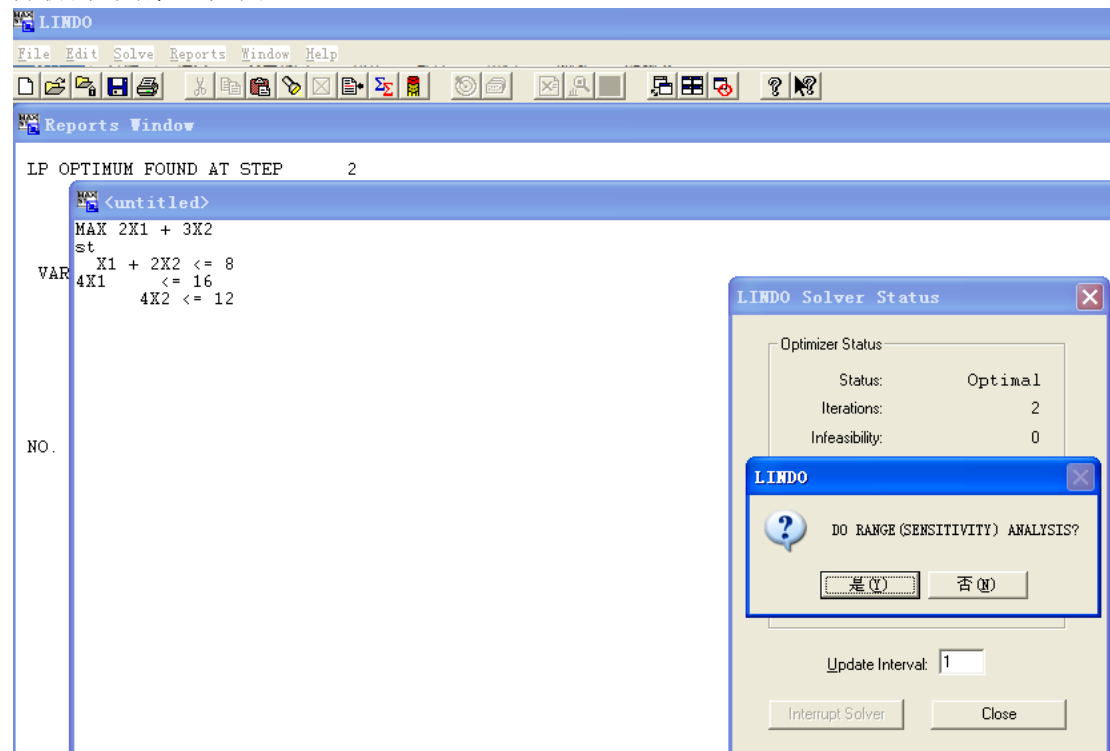


图 18

点击“是”，就可得解和灵敏度分析报告：

LP OPTIMUM FOUND AT STEP 2

OBJECTIVE FUNCTION VALUE

1) 14.00000

VARIABLE	VALUE	REDUCED COST
X1	4.000000	0.000000
X2	2.000000	0.000000

ROW	SLACK OR SURPLUS	DUAL PRICES
2)	0.000000	1.500000
3)	0.000000	0.125000
4)	4.000000	0.000000

NO. ITERATIONS= 2

RANGES IN WHICH THE BASIS IS UNCHANGED:

OBJ COEFFICIENT RANGES			
VARIABLE	CURRENT	ALLOWABLE	
	COEF	INCREASE	DECREASE
X1	2.000000	INFINITY	0.500000
X2	3.000000	1.000000	3.000000

RIGHTHAND SIDE RANGES			
ROW	CURRENT	ALLOWABLE	
	RHS	INCREASE	DECREASE
2	8.000000	2.000000	4.000000
3	16.000000	16.000000	8.000000
4	12.000000	INFINITY	4.000000

§1.4 运输问题数学模型的计算机求解

1. 中小型运输问题的求解

中小型运输问题可以和小型线性规划一样，直接输入运输问题的数学模型代码求解。

例 4.1 求解下面运输问题的数学模型

单位 运 价 产地 \ 销地	B ₁	B ₂	B ₃	B ₄	产量
A ₁	3	11	3	10	7
A ₂	1	9	2	8	4
A ₃	7	4	10	5	9
销量	3	6	5	6	

LINGO 中的输入代码为：

```
min=3*x11+11*x12+3*x13+10*x14+x21+9*x22+2*x23+8*x24+7*x31+4*x32
+10*x33+5*x34;
```

```
x11+ x12+ x13+ x14 = 7;
```

```
x21+ x22+ x23+ x24 = 4;
```

```
x31+ x32+ x33+ x34 = 9;
```

```
x11+ x21+ x31 = 3;
```

```
x12+ x22+ x32 = 6;
```

```
x13+ x23+ x33 = 5;
```

```
x14+ x24+ x34 = 6;
```

求解可得：

Global optimal solution found.

Objective value: 83.00000

Total solver iterations: 0

V Variable	Value	Reduced Cost
x11	0.000000	0.000000

X12	0.000000	2.000000
X13	5.000000	0.000000
X14	2.000000	0.000000
X21	3.000000	0.000000
X22	0.000000	2.000000
X23	0.000000	1.000000
X24	1.000000	0.000000
X31	0.000000	9.000000
X32	6.000000	0.000000
X33	0.000000	12.000000
X34	3.000000	0.000000

Row	Slack or Surplus	Dual Price
1	85.000000	-1.000000
2	0.000000	-3.000000
3	0.000000	-1.000000
4	0.000000	2.000000
5	0.000000	0.000000
6	0.000000	-6.000000
7	0.000000	0.000000
8	0.000000	-7.000000

2. 大规模运输问题的求解

对于变量多和约束条件多的运输问题数学模型，用编程的方法求解，简化了模型的输入，带来了很大的方便。

例 4.2 用编程的方法求解例 4.1

LINGO 编程为

```
MODEL:
!! 定义变量和常量;
SETS:
    As/A1..A3/:a;
    Bs/B1..B4/:b;
    LINKS(As,Bs):c,x;
ENDSETS
! 目标函数;
MIN
=@SUM(LINKS(I,J):C(I,J)*x(I,J));
! 产量约束;
@FOR(As(I): @SUM(Bs(J):x(I,J))=a(I));
! 销量约束;
@FOR(Bs(J): @SUM(As(I):x(I,J))=b(J));
! Here is the data;
DATA:
    a=7 4 9;
    b=3 6 5 6;
```

```

c=3 11 3 10
1 9 2 8
7 4 10 5;
ENDDATA
END

```

上述编程对应的数学模型为如下形式:

$$\min z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

$$\sum_{i=1}^m x_{ij} = b_j, j=1,2,\dots,n$$

$$\sum_{j=1}^n x_{ij} = a_i, i=1,2,\dots,m$$

$$x_{ij} \geq 0, i=1,2,\dots,m, j=1,2,\dots,n$$

求解可得解报告:

Global optimal solution found.

Objective value: 85.00000

Total solver iterations: 7

Variable	Value	Reduced Cost
A(A1)	7.000000	0.000000
A(A2)	4.000000	0.000000
A(A3)	9.000000	0.000000
B(B1)	3.000000	0.000000
B(B2)	6.000000	0.000000
B(B3)	5.000000	0.000000
B(B4)	6.000000	0.000000
C(A1, B1)	3.000000	0.000000
C(A1, B2)	11.00000	0.000000
C(A1, B3)	3.000000	0.000000
C(A1, B4)	10.00000	0.000000
C(A2, B1)	1.000000	0.000000
C(A2, B2)	9.000000	0.000000
C(A2, B3)	2.000000	0.000000
C(A2, B4)	8.000000	0.000000
C(A3, B1)	7.000000	0.000000
C(A3, B2)	4.000000	0.000000
C(A3, B3)	10.00000	0.000000
C(A3, B4)	5.000000	0.000000
X(A1, B1)	0.000000	0.000000
X(A1, B2)	0.000000	2.000000
X(A1, B3)	5.000000	0.000000
X(A1, B4)	2.000000	0.000000
X(A2, B1)	3.000000	0.000000

X(A2, B2)	0.000000	2.000000
X(A2, B3)	0.000000	1.000000
X(A2, B4)	1.000000	0.000000
X(A3, B1)	0.000000	9.000000
X(A3, B2)	6.000000	0.000000
X(A3, B3)	0.000000	12.000000
X(A3, B4)	3.000000	0.000000

Row	Slack or Surplus	Dual Price
1	85.000000	-1.000000
2	0.000000	-3.000000
3	0.000000	-1.000000
4	0.000000	2.000000
5	0.000000	0.000000
6	0.000000	-6.000000
7	0.000000	0.000000
8	0.000000	-7.000000

§1.5 目标规划数学模型的计算机求解

LINGO 中不能直接求解目标规划，但是可用 LINDO 的 Preemptive Goal 命令求解。

例 5.1 求解目标规划

`min z=P1d1++P2 (d2-+d2+)+P3d3-`

`2x1+x2≤11`

`x1-x2+d1--d1+=0`

`x1+2x2+d2--d2+=10`

`8x1+10x2+d3--d3+=56`

`x1, x2, di-, di+≥0, i=1,2,3`

在 LINDO 中输入下面的代码，其中 di1=di⁻, di2=di⁺, i=1,2,3。

`min obj1+ obj2+ obj3`

`s.t.`

`2x1+x2<=11`

`x1-x2+d11-d12=0`

`x1+2x2+d21-d22=10`

`8x1+10x2+d31-d32=56`

`obj1-d12=0`

`obj2-d21-d22=0`

`obj3-d31=0`

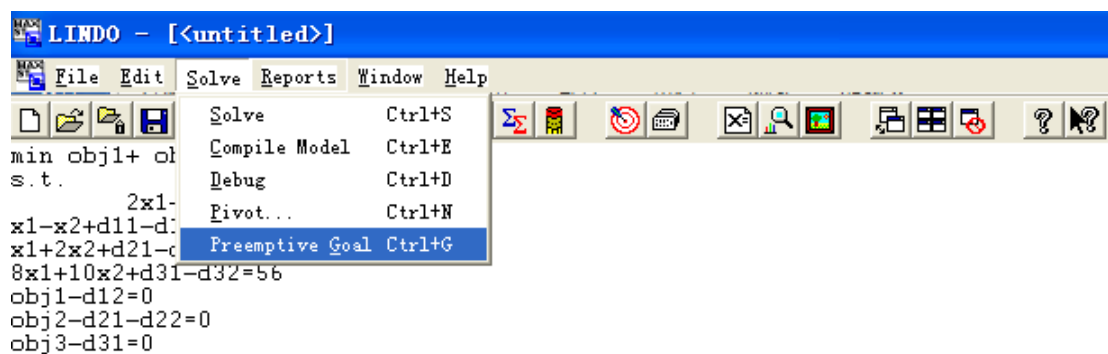


图 19

点击 Solve→Preemptive Goal 可得最优解

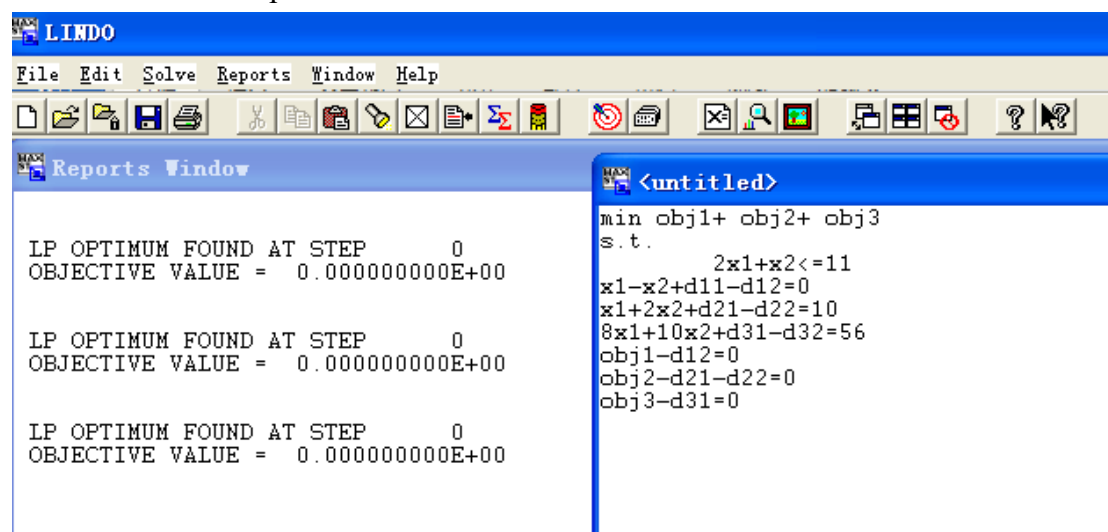


图 20

再点击 Reports→Solution 可找到最优解

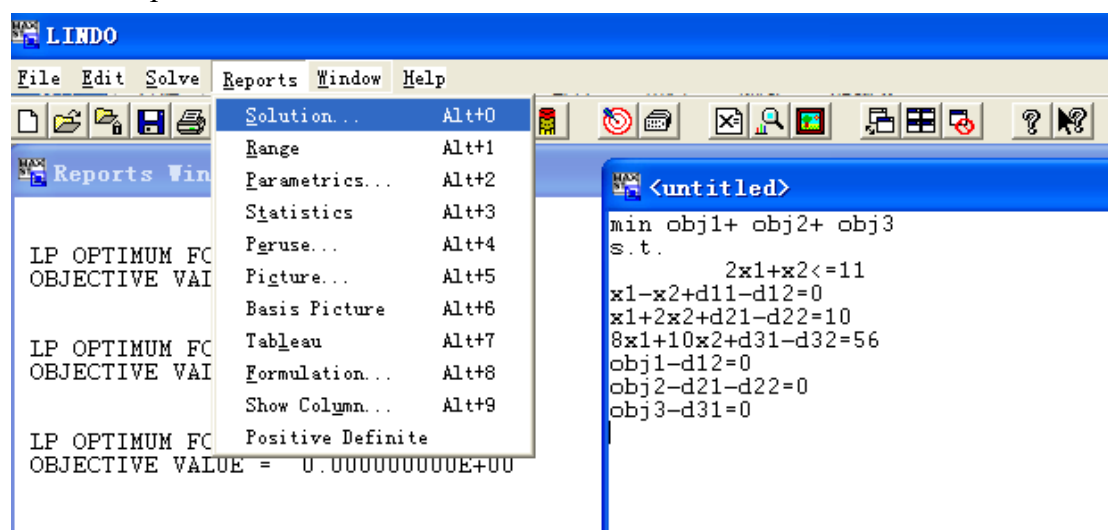


图 21

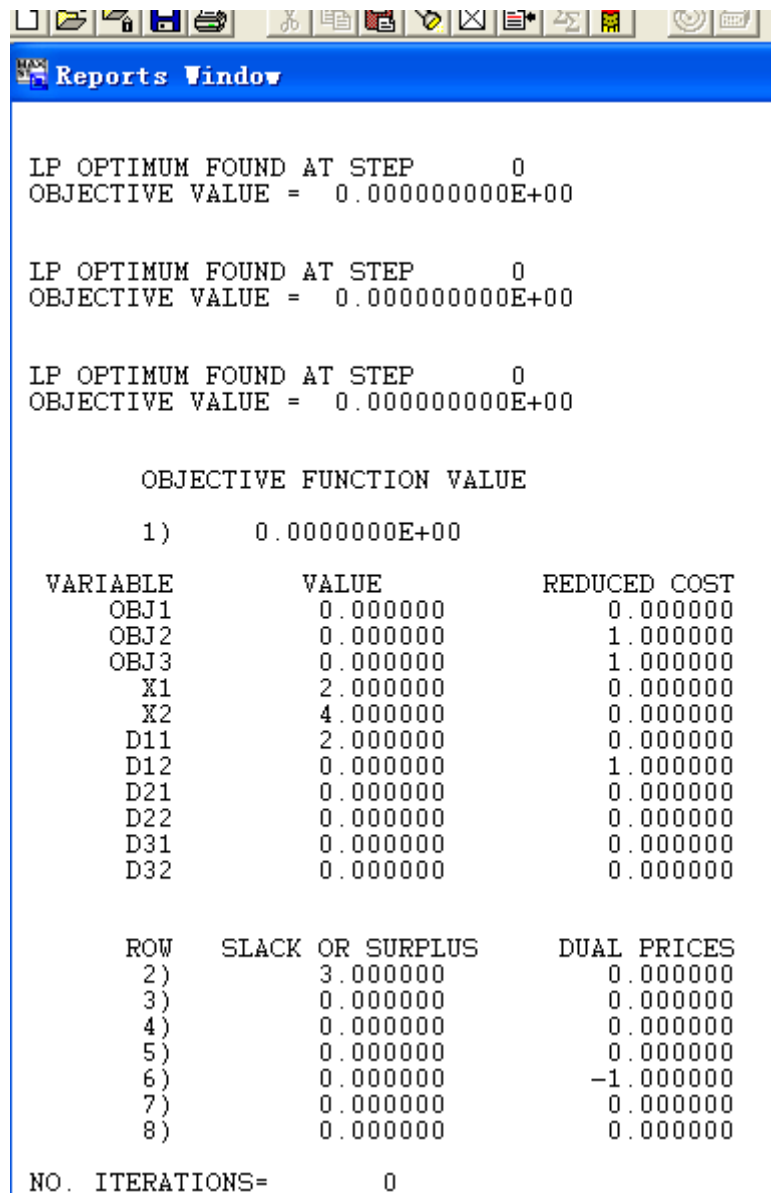


图 22

于是可得最优解 $x_1=2, x_2=4$ 。

§1.6 整数规划数学模型的计算机求解

LINGO 中求解整数规划或 0-1 规划，只需要在一个线性规划的代码或程序中添加整数约束， x_1 为整数可用 `@gin(x1)` 表示， x_2 为 0-1 变量可用 `@bin(x2)` 表示。

例 求解下面混合整数规划

$$\min z = -3x_1 + 4x_2 - 2x_3 + 5x_4;$$

$$4x_1 - x_2 + 2x_3 - x_4 = -2;$$

$$x_1 + x_2 + 3x_3 - x_4 \leq 14;$$

$$-2x_1 + 3x_2 - x_3 + 2x_4 \geq 2;$$

x_1 为整数， $x_2=0$ 或 1 ， $x_3 \geq 0$ ， x_4 无约束；

LINGO 编程为：

```

!定义变量与常量，给出了值的为常量；
sets:
is/1..3/:b;
js/1..4/:c,x;
links(is,js):a;
endsets
!目标函数;
min=@sum(js(J):c(J)*x(J));
!约束条件;
@sum(js(J):a(1,J)*x(J))=b(1);
@sum(js(J):a(2,J)*x(J))<=b(2);
@sum(js(J):a(3,J)*x(J))>= b(3);
!自由变量;
@gin(x(1));@bin(x(2));@free(x(4));
!指定常量的值;
data:
c=-3 4 -2 5;
b=-2 14 2;
a=4 -1 2 -1
   1 1 3 -1
   -2 3 -1 2;
end data
!结束;
End

```

求解可得解报告:

Global optimal solution found.

Objective value: 9.000000

Extended solver steps: 0

Total solver iterations: 0

Variable	Value	Reduced Cost
B(1)	-2.000000	0.000000
B(2)	14.000000	0.000000
B(3)	2.000000	0.000000
C(1)	-3.000000	0.000000
C(2)	4.000000	0.000000
C(3)	-2.000000	0.000000
C(4)	5.000000	0.000000
X(1)	0.000000	17.000000
X(2)	1.000000	-1.000000
X(3)	0.000000	8.000000
X(4)	1.000000	0.000000
A(1, 1)	4.000000	0.000000
A(1, 2)	-1.000000	0.000000
A(1, 3)	2.000000	0.000000

A(1, 4)	-1.000000	0.000000
A(2, 1)	1.000000	0.000000
A(2, 2)	1.000000	0.000000
A(2, 3)	3.000000	0.000000
A(2, 4)	-1.000000	0.000000
A(3, 1)	-2.000000	0.000000
A(3, 2)	3.000000	0.000000
A(3, 3)	-1.000000	0.000000
A(3, 4)	2.000000	0.000000

Row	Slack or Surplus	Dual Price
1	9.000000	-1.000000
2	0.000000	5.000000
3	14.000000	0.000000
4	3.000000	0.000000

§1.7 指派问题的计算机求解

指派问题的数学模型实质就是一个特殊的运输问题，产地个数和销地个数相同，每个产地的产量和每个销地的销量都是1个单位，变量也都为0-1变量。

例7.1 用运输问题的数学模型求解一个指派问题

$$\left\{ \begin{array}{l} \min \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \\ s.t. \\ \sum_{i=1}^n x_{ij} = 1, \quad j = 1, 2, \dots, n \\ \sum_{j=1}^n x_{ij} = 1, \quad i = 1, 2, \dots, n \\ x_{ij} = 0, 1 \end{array} \right.$$

```
model:
    !7个工人，7个工作的分配问题;
sets:
    workers/w1..w7/;
    jobs/j1..j7/;
    links(workers,jobs): cost,volume;
endsets
    !目标函数;
    min=@sum(links: cost*volume);
    !每个工人只能有一份工作;
    @for(workers(I): @sum(jobs(J): volume(I,J))=1; );
    !每份工作只能有一个工人;
    @for(jobs(J): @sum(workers(I): volume(I,J))=1; );
data:
    cost= 6  2  6  7  4  2  5
          4  9  5  3  8  5  8
          5  2  1  9  7  4  3
          7  6  7  3  9  2  7
```

```

2 3 9 5 7 2 6
5 5 2 2 8 11 4
9 2 3 12 4 5 10;
enddata
end
求解得

```

§1.8 最短路问题的计算机求解

最短路问题有很多种算法，包括动态规划算法和 Dijkstra 算法，还有作为运输问题的线性规划算法，即将一单位物品沿着该网络从起点运往终点，满足如下条件：

- (1) 每条弧上的运量只能取 0 或 1；
- (2) 起点的净流出量为 1，终点的净流入量为 1；
- (3) 其余点都作为转运点，总流入等于总流出。

下面就是最短路问题的线性规划解法。

例 8.1

```

sets:
    nodes/1..6/;
    arcs(nodes, nodes) |&1 #lt# &2: c, x; !c(i,j) 表示点i到点j的距离,
x(i,j)=0 or 1 表示点i到点j被选中与不被选中;
endsets
data:
    c = 7 12 21 31 44
        7 12 21 31
        7 12 21
        7 12
        7;
enddata
n=@size(nodes);
min=@sum(arcs: c*x);
@for(nodes(i) | i #ne# 1 #and# i #ne# n:
    @sum(arcs(i,j): x(i,j)) = @sum(arcs(j,i): x(j,i)) );
@sum(arcs(i,j)| i #eq# 1: x(i,j))=1;
求解可得
Global optimal solution found.
Objective value:                31.00000
Total solver iterations:        0

```

Variable	Value	Reduced Cost
N	6.000000	0.000000
C(1, 2)	7.000000	0.000000
C(1, 3)	12.00000	0.000000
C(1, 4)	21.00000	0.000000
C(1, 5)	31.00000	0.000000
C(1, 6)	44.00000	0.000000
C(2, 3)	7.000000	0.000000
C(2, 4)	12.00000	0.000000
C(2, 5)	21.00000	0.000000
C(2, 6)	31.00000	0.000000
C(3, 4)	7.000000	0.000000
C(3, 5)	12.00000	0.000000
C(3, 6)	21.00000	0.000000
C(4, 5)	7.000000	0.000000
C(4, 6)	12.00000	0.000000
C(5, 6)	7.000000	0.000000
X(1, 2)	1.000000	0.000000
X(1, 3)	0.000000	0.000000
X(1, 4)	0.000000	2.000000
X(1, 5)	0.000000	7.000000
X(1, 6)	0.000000	13.00000
X(2, 3)	0.000000	2.000000
X(2, 4)	1.000000	0.000000
X(2, 5)	0.000000	4.000000
X(2, 6)	0.000000	7.000000
X(3, 4)	0.000000	0.000000
X(3, 5)	0.000000	0.000000
X(3, 6)	0.000000	2.000000
X(4, 5)	0.000000	2.000000
X(4, 6)	1.000000	0.000000
X(5, 6)	0.000000	0.000000

Row	Slack or Surplus	Dual Price
1	0.000000	0.000000
2	31.00000	-1.000000
3	0.000000	-31.00000
4	0.000000	-24.00000
5	0.000000	-19.00000
6	0.000000	-12.00000
7	0.000000	-7.000000

容易知道，最短路的权为最小值31，最短路为：
 $v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6$

§1.9 最大流问题的计算机求解

LLINGO 中用标号法求最大流比较困难，最简单的方法还是线性规划方法，即

$$\begin{aligned} \text{Max } z &= v(f) \\ \sum_{(v_i, v_j) \in A} f_{ij} - \sum_{(v_j, v_i) \in A} f_{ji} &= 0, \quad i \neq s, i \neq t \\ \sum_{(v_s, v_j) \in A} f_{sj} - \sum_{(v_j, v_s) \in A} f_{js} &= v(f) \\ \sum_{(v_t, v_j) \in A} f_{tj} - \sum_{(v_j, v_t) \in A} f_{jt} &= -v(f) \\ f_{ij} &\leq c_{ij} \\ f_{ij} &\geq 0, v(f) \geq 0 \end{aligned}$$

例 9.1 用编程求解一个网络的最大流

```
model:
sets:point/v1..v4/;
links(point,point):cost,volume;
endsets
max=vf;
!出入的等于最大流量;
@sum(point(J):volume(1,J))-@sum(point(J):volume(J,1))=vf;
@sum(point(J):volume(4,J))-@sum(point(J):volume(J,4))=-vf;
!其它中间的流出流入相等;
@for(point(I)|I #ne# 1 #and# I #ne# 4:
@sum(point(J):volume(I,J))-@sum(point(J):volume(J,I))=0);
@for(links(I,J):volume(I,J)<cost(I,J));!约束，不能超过最大流量;
data:
cost=
0 12 13 0
0 0 0 100
0 7 0 8
0 0 5 0;
enddata
end
```

求解可得解报告

Global optimal solution found.

Objective value:	25.00000
Total solver iterations:	0
Variable	Value
VF	25.00000
COST(V1, V1)	0.000000
COST(V1, V2)	12.00000
Reduced Cost	0.000000
	0.000000
	0.000000

COST(V1, V3)	13.000000	0.000000
COST(V1, V4)	0.000000	0.000000
COST(V2, V1)	0.000000	0.000000
COST(V2, V2)	0.000000	0.000000
COST(V2, V3)	0.000000	0.000000
COST(V2, V4)	100.0000	0.000000
COST(V3, V1)	0.000000	0.000000
COST(V3, V2)	7.000000	0.000000
COST(V3, V3)	0.000000	0.000000
COST(V3, V4)	8.000000	0.000000
COST(V4, V1)	0.000000	0.000000
COST(V4, V2)	0.000000	0.000000
COST(V4, V3)	5.000000	0.000000
COST(V4, V4)	0.000000	0.000000
VOLUME(V1, V1)	0.000000	0.000000
VOLUME(V1, V2)	12.000000	0.000000
VOLUME(V1, V3)	13.000000	0.000000
VOLUME(V1, V4)	0.000000	0.000000
VOLUME(V2, V1)	0.000000	1.000000
VOLUME(V2, V2)	0.000000	0.000000
VOLUME(V2, V3)	0.000000	0.000000
VOLUME(V2, V4)	19.000000	0.000000
VOLUME(V3, V1)	0.000000	1.000000
VOLUME(V3, V2)	7.000000	0.000000
VOLUME(V3, V3)	0.000000	0.000000
VOLUME(V3, V4)	6.000000	0.000000
VOLUME(V4, V1)	0.000000	1.000000
VOLUME(V4, V2)	0.000000	0.000000
VOLUME(V4, V3)	0.000000	0.000000
VOLUME(V4, V4)	0.000000	0.000000

Row	Slack or Surplus	Dual Price
1	25.00000	1.000000
2	0.000000	-1.000000
3	0.000000	0.000000
4	0.000000	0.000000
5	0.000000	0.000000
6	0.000000	0.000000
7	0.000000	1.000000
8	0.000000	1.000000
9	0.000000	1.000000
10	0.000000	0.000000
11	0.000000	0.000000
12	0.000000	0.000000

13	81.000000	0.000000
14	0.000000	0.000000
15	0.000000	0.000000
16	0.000000	0.000000
17	2.000000	0.000000
18	0.000000	0.000000
19	0.000000	0.000000
20	5.000000	0.000000
21	0.000000	0.000000

第二章 LINGO 软件基础及应用

§2.1 原始集(primitive set)和派生集(derived set)与集的定义

第一章例1.3的程序对应的线性规划数学模型是

$$\begin{aligned}\max \quad & z=2x_1+3x_2 \\ & x_1+2x_2 \leq 8 \\ & 4x_1 \leq 16 \\ & 4x_2 \leq 16 \\ & x_1, x_2 \geq 0\end{aligned}$$

对应于线性规划模型的一般表达形式为:

$$\begin{aligned}\max z &= \sum_{j=1}^n c_j x_j \\ \begin{cases} \sum_{j=1}^n a_{ij} x_j \leq b_i & (i=1,2,\dots,m) \\ x_j \geq 0 & (j=1,2,\dots,n) \end{cases}\end{aligned}$$

例1.3的 $m=3$ 表达约束条件的个数, $n=2$ 表达变量的个数, 模型中出现的变量和常量有以 i 为原始下标(单下标) b_i , 以 j 为原始下标的 c_j 和 x_j , 以及以 ij 为派生或复合下标(双下标)的 a_{ij} 。

在例3 如下的集的定义中:

```
sets:
is/1..3/:b;
js/1..2/:c,x;
links(is,js):a;
endsets
```

首先定义原始下标 i 和 j 的集合 is 和 js , 在“ $is/1..3/:b;$ ”中“ $..$ ”表示省略号, 即 $is=\{1, 2, 3\}$, 这些下标上定义了 b_i , 即该语句就是定义了变量 $\{b_1, b_2, b_3\}$,

同理, “ $js/1..2/:c,x;$ ”

就是定义了 $\{c_1, c_2\}$ 和 $\{x_1, x_2\}$ 。在“ $links(is,js):a;$ ”中, 首先, 借助于原始下标集 is 和 js , $links(is,js)$ 定义了双下标 (i, j) , 并在该双下标上定义了线性规

划的系数矩阵 $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}$

于是就定义了全部的变量。

一般情况下, 集是 LINGO 建模语言的基础, 是程序设计最强有力的基本构件。借助于集, 能够用一个单一的、长的、简明的复合公式表示一系列相似的约束, 从而可以快速方便地表达规模较大的模型。

集是一群相联系的对象，这些对象也称为集的成员。一个集可能是一系列产品、卡车或雇员。每个集成员可能有一个或多个与之有关联的特征，我们把这些特征称为属性。属性值可以预先给定，也可以是未知的，有待于LINGO 求解。例如，产品集中的每个产品可以有一个价格属性；卡车集中的每辆卡车可以有一个牵引力属性；雇员集中的每位雇员可以有一个薪水属性，也可以有一个生日属性等等。一个原始集是由一些最基本的对象组成的。一个派生集是用一个或多个其它集来定义的，也就是说，它的成员来自于其它已存在的集。

定义一个原始集，用下面的语法：

```
setname[/member_list/][:attribute_list];
```

注意：用“[]”表示该部分内容可选。Setname 是你选择的来标记集的名字，最好具有较强的可读性或含义明确、容易记住或与模型中的呼号对应。集名字必须严格符合标准命名规则：以拉丁字母或下划线（_）为首字符，其后由拉丁字母（A—Z）、下划线、阿拉伯数字（0, 1, …, 9）组成的总长度不超过32 个字符的字符串，且不区分大小写。：该命名规则同样适用于集成员名和属性名等的命名。

Member_list 是集成员列表。如果集成员放在集定义中，那么对它们可采取显式罗列和隐式罗列两种方式。如果集成员不放在集定义中，那么可以在随后的数据部分定义它们。

当显式罗列成员时，必须为每个成员输入一个不同的名字，中间用空格或逗号搁开，允许混合使用。

例4 可以定义一个名为students 的原始集，它具有成员John、Jill、Rose 和Mike，属性有sex 和age：

sets:

```
students/John Jill, Rose Mike/: sex, age;
```

endsets

当隐式罗列成员时，不必罗列出每个集成员。可采用如下语法：

```
setname/member1..memberN/[: attribute_list];
```

这里的member1 是集的第一个成员名，memberN 是集的最末一个成员名。LINGO 将自动产生中间的所有成员名。LINGO 也接受一些特定的首成员名和末成员名，用于创建一些特殊的集。

列表如下：

隐式成员列表格式 示例 所产生集成员

1..n 1..5 1, 2, 3, 4, 5

StringM..StringN Car2..car14 Car2, Car3, Car4, ..., Car14

DayM..DayN Mon..Fri Mon, Tue, Wed, Thu, Fri

MonthM..MonthN Oct..Jan Oct, Nov, Dec, Jan

MonthYearM..MonthYearN Oct2001..Jan2002

Oct2001, Nov2001, Dec2001, Jan2002

集成员不放在集定义中，而在随后的数据部分来定义也可。

例 定义集的第种方式

```
!集部分;
sets:
students:sex,age;
endsets
!数据部分;
data:
students,sex,age= John 1 16
                Jill 0 14
                Rose 0 17
                Mike 1 13;
Enddata
```

在这个集部分只定义了一个集students，并未指定成员。在数据部分罗列了集成员John、

Jill、Rose 和Mike，并对属性sex 和age 分别给出了值。

LINGO 内置的建模语言是一种描述性语言，用它可以描述现实世界中的一些问题，然后再借助于LINGO 求解器求解。因此，集属性的值一旦在模型中被确定，就不可能再更改。在LINGO 中，只有在初始部分中给出的集属性值在以后的求解中可更改。这与前面并不矛盾，初始部分是LINGO 求解器的需要，并不是描述问题所必须的。

为了定义一个派生集，必须详细声明：

- 集的名字
- 父集的名字
- 可选，集成员
- 可选，集成员的属性

可用下面的语法定义一个派生集：

```
setname(parent_set_list)/member_list/[:attribute_list];
```

setname 是集的名字。parent_set_list 是已定义的集的列表，多个时必须用逗号隔开。如果没有指定成员列表，那么LINGO 会自动创建父集成员的所有组合作为派生集的成员。派生集的父集既可以是原始集，也可以是其它的派生集。

例 派生集的定义

```
sets:
product/A B/;
machine/M N/;
week/1..2/;
allowed(product,machine,week):x;
endsets
```

LINGO 生成了三个父集的所有组合共八组作为allowed 集的成员。列表如下：

编号 成员

- 1 (A, M, 1)
- 2 2 (A, M, 2)
- 3 3 (A, N, 1)
- 4 4 (A, N, 2)
- 5 5 (B, M, 1)

6 6 (B, M, 2)
 7 7 (B, N, 1)
 8 8 (B, N, 2)

成员列表被忽略时，派生集成员由父集成员所有的组合构成，这样的派生集成为稠密集。如果限制派生集的成员，使它成为父集成员所有组合构成的集合的一个子集，这样的派生集成为稀疏集。同原始集一样，派生集成员的声明也可以放在数据部分。一个派生集的成员列表有两种方式生成：①显式罗列；②设置成员资格过滤器。当采用方式①时，必须显式罗列出所有要包含在派生集中的成员，并且罗列的每个成员必须属于稠密集。使用前面的例子，显式罗列派生集的成员：
 allowed(product, machine, week)/A M 1, A N 2, B N 1/;

如果需要生成一个大的、稀疏的集，那么显式罗列就很讨厌。幸运地是许多稀疏集的成员都满足一些条件以和非成员相区分。我们可以把这些逻辑条件看作过滤器，在LINGO 生成派生集的成员时把使逻辑条件为假的成员从稠密集中过滤掉。

例7 稀疏派生集的定义

sets:

!学生集：性别属性sex, 1 表示男性, 0 表示女性; 年龄属性age. ;

students/John, Jill, Rose, Mike/:sex, age;

!男学生和女学生的联系集：友好程度属性friend, [0, 1]之间的数. ;

linkmf(students, students) | sex(&1) #eq# 1 #and# sex(&2) #eq# 0: friend;

!男学生和女学生的友好程度大于0.5 的集;

linkmf2(linkmf) | friend(&1, &2) #ge# 0.5 : x;

endsets

data:

sex, age = 1 16

0 14

0 17

0 13;

friend = 0.3 0.5 0.6;

enddata

用竖线(|)来标记一个成员资格过滤器的开始。#eq#是逻辑运算符，用来判断是否“相等”，可参考§4. &1 可看作派生集的第1 个原始父集的索引，它取遍该原始父集的所有成员；&2 可看作派生集的第2 个原始父集的索引，它取遍该原始父集的所有成员；&3, &4, ……，以此类推。注意如果派生集B 的父集是另外的派生集A，那么上面所说的原始父集是集A 向前回溯到最终的原始集，其顺序保持不变，并且派生集A 的过滤器对派生集B 仍然有效。因此，派生集的索引个数是最终原始父集的个数，索引的取值是从原始父集到当前派生集所作限制的总和。

§2.2 LINGO 中的函数与目标函数和约束条件的表示

一.LINGO 有9 种类型的函数:

1. 基本运算符：包括算术运算符、逻辑运算符和关系运算符
2. 数学函数：三角函数和常规的数学函数
3. 金融函数：LINGO 提供的两种金融函数

4. 概率函数：LINGO 提供了大量概率相关的函数
5. 变量界定函数：这类函数用来定义变量的取值范围
6. 集操作函数：这类函数为对集的操作提供帮助
7. 集循环函数：遍历集的元素，执行一定的操作的函数
8. 数据输入输出函数：这类函数允许模型和外部数据源相联系，进行数据的输入输出
9. 辅助函数：各种杂类函数

二.LINGO 提供了5 种二元运算符：

\wedge 乘方
 $*$ 乘
 $/$ 除
 $+$ 加
 $-$ 减

LINGO 唯一的一元算术运算符是取反函数 “ $-$ ”。

这些运算符的优先级由高到底为：

高 $-$ （取反）

\wedge

$*$ $/$

低 $+$ $-$

运算符的运算次序为从左到右按优先级高低来执行。运算的次序可以用圆括号“ $()$ ”来改变。

三.LINGO 具有 9 种逻辑运算符：

#not# 否定该操作数的逻辑值，#not# 是一个一元运算符

#eq# 若两个运算数相等，则为true；否则为false

#ne# 若两个运算符不相等，则为true；否则为false

#gt# 若左边的运算符严格大于右边的运算符，则为true；否则为false

#ge# 若左边的运算符大于或等于右边的运算符，则为true；否则为false

#lt# 若左边的运算符严格小于右边的运算符，则为true；否则为false

#le# 若左边的运算符小于或等于右边的运算符，则为true；否则为false

#and# 仅当两个参数都为true 时，结果为true；否则为false

#or# 仅当两个参数都为false 时，结果为false；否则为true

这些运算符的优先级由高到低为：

高 #not#

#eq# #ne# #gt# #ge# #lt# #le#

低 #and# #or#

四. LINGO 有三种关系运算符

“ $=$ ”、“ \leq ”和“ \geq ”。LINGO 中还能用“ $<$ ”表示小于等

于关系，“ $>$ ”表示大于等于关系。LINGO 并不支持严格小于和严格大于关系运算符。然而，如果需要严格小于和严格大于关系，比如让A 严格小于B：

$A < B$,

那么可以把它变成如下的小于等于表达式：

$A + \varepsilon \leq B$,

这里 ε 是一个小的正数，它的值依赖于模型中A 小于B 多少才算不等。

下面给出以上三类操作符的优先级：

高 #not# - (取反)

^

* /

+ -

#eq# #ne# #gt# #ge# #lt# #le#

#and# #or#

低 <= = >=

五. LINGO 提供了大量的标准数学函数:

@abs(x) 返回x 的绝对值

@sin(x) 返回x 的正弦值, x 采用弧度制

@cos(x) 返回x 的余弦值

@tan(x) 返回x 的正切值

@exp(x) 返回常数e 的x 次方

@log(x) 返回x 的自然对数

@lgm(x) 返回x 的gamma 函数的自然对数

@sign(x) 如果x<0 返回-1; 否则, 返回1

@floor(x) 返回x 的整数部分。当x>=0 时, 返回不超过x 的最大整数; 当x<0 时, 返回不低于x 的最大整数。

@smax(x1, x2, ..., xn) 返回x1, x2, ..., xn 中的最大值

@smin(x1, x2, ..., xn) 返回x1, x2, ..., xn 中的最小值

六. 目前 LINGO 提供了两个金融函数:

@fpa(I,n)和@fpl(I,n)

@fpa(I,n)返回如下情形的净现值: 单位时段利率为I, 连续n 个时段支付, 每个时段支付单位费用。若每个时段支付x 单位的费用, 则净现值可用x 乘以 @fpa(I, n)算得。@fpa 的计算公式为:

$$\sum_{k=1}^n \frac{1}{(1+I)^k} = \frac{1-(1+I)^{-n}}{I}$$

@fpl(I,n) 返回如下情形的净现值: 单位时段利率为I, 第n 个时段支付单位费用。@fpl(I, n)的计算公式为 $(1+I)^n$ 。

$$\textbf{@fpa(I,n)} = \sum_{k=1}^n \textbf{fpl(I,k)}$$

七. LINGO 提供了如下的概率函数:

1. @pbn(p,n,x)

二项分布的累积分布函数。当n 和(或)x 不是整数时, 用线性插值法进行计算。

2. @pcx(n,x)

自由度为n 的 χ^2 分布的累积分布函数。

3. @peb(a,x)

当到达负荷为a，服务系统有x 个服务器且允许无穷排队时的Erlang 繁忙概率。

4. @pel(a,x)

当到达负荷为a，服务系统有x 个服务器且不允许排队时的Erlang 繁忙概率。

5. @pfd(n,d,x)

自由度为n 和d 的F 分布的累积分布函数。

6. @pfs(a,x,c)

当负荷上限为a，顾客数为c，平行服务器数量为x 时，有限源的Poisson 服务系统的等待或返修顾客数的期望值。a 是顾客数乘以平均服务时间，再除以平均返修时间。当c 和（或）x 不是整数时，采用线性插值进行计算。

7. @phg(pop,g,n,x)

超几何（Hypergeometric）分布的累积分布函数。pop 表示产品总数，g 是正品数。从所有产品中任意取出n ($n \leq \text{pop}$) 件。pop, g, n 和x 都可以是非整数，这时采用线性插值进行计算。

8. @ppl(a,x)

Poisson 分布的线性损失函数，即返回 $\max(0, z-x)$ 的期望值，其中随机变量z 服从均值为a 的Poisson 分布。

9. @pps(a,x)

均值为a 的Poisson 分布的累积分布函数。当x 不是整数时，采用线性插值进行计算。

10. @psl(x)

单位正态线性损失函数，即返回 $\max(0, z-x)$ 的期望值，其中随机变量z 服从标准正态。

11. @psn(x)

标准正态分布的累积分布函数。

12. @ptd(n,x)

自由度为n 的t 分布的累积分布函数。

13. @qrand(seed)

产生服从(0, 1)区间的拟随机数。@qrand 只允许在模型的数据部分使用，它将用拟随机数填满集属性。通常，声明一个 $m \times n$ 的二维表，m 表示运行实验的次数，n 表示每次实验所需的随机数的个数。在行内，随机数是独立分布的；在行间，随机数是非常均匀的。这些随机数是用“分层取样”的方法产生的。

例

model:

```

data:
M=4; N=2; seed=1234567;
enddata
sets:
rows/1..M/;
cols/1..N/;
table(rows,cols): x;
endsets
data:
X=@qrand(seed);
enddata
end

```

如果没有为函数指定种子，那么LINGO 将用系统时间构造种子。

14. @rand(seed)

返回0 和1 间的伪随机数，依赖于指定的种子。典型用法是 $U(I+1)=@rand(U(I))$ 。注意如果seed 不变，那么产生的随机数也不变。

例 利用@rand 产生15 个标准正态分布的随机数和自由度为2 的t 分布的随机数。

```

model:
!产生一系列正态分布和t 分布的随机数;
sets:
series/1..15/: u, znorm, zt;
endsets
!第一个均匀分布随机数是任意的;
u( 1) = @rand( .1234);
!产生其余的均匀分布的随机数;
@for(series( I) | I #GT# 1:u( I) = @rand( u( I - 1)));
@for( series( I):
!正态分布随机数;
@psn( znorm( I)) = u( I);
!和自由度为2 的t 分布随机数;
@ptd( 2, zt( I)) = u( I);
!ZNORM 和 ZT 可以是负数;
@free( znorm( I)); @free( zt( I)););
End

```

八. LINGO中有四种变量界定函数，用于实现对变量取值范围的附加限制：

@bin(x) 限制x 为0 或1

@bnd(L, x, U) 限制 $L \leq x \leq U$

@free(x) 取消对变量x 的默认下界为0 的限制，即x 可以取任意实数

@gin(x) 限制x 为整数

在默认情况下，LINGO 规定变量是非负的，也就是说下界为0，上界为 $+\infty$ 。@free 取消了默认的下界为0 的限制，使变量也可以取负值。@bnd 用于设定一个变量的上下界，它也可以取消默认下界为0 的约束。

LINGO 提供了几个函数帮助处理集。

1. @in(set_name,primitive_index_1 [,primitive_index_2,...])

如果元素在指定集中，返回1；否则返回0。

例 全集为I，B 是I 的一个子集，C 是B 的补集。

```
sets:
I/x1..x4/;
B(I)/x2/;
C(I) | #not#@in(B,&1) ;;
endsets
```

2. @index([set_name,] primitive_set_element)

该函数返回在集set_name 中原始集成员primitive_set_element 的索引。如果set_name被忽略，那么LINGO 将返回与primitive_set_element 匹配的第一个原始集成员的索引。如果找不到，则产生一个错误。

例 如何确定集成员(B, Y)属于派生集S3。

```
sets:
S1/A B C/;
S2/X Y Z/;
S3(S1,S2)/A X, A Z, B Y, C X/;
endsets
X=@in(S3,@index(S1,B),@index(S2,Y));
```

看下面的例子，表明有时为@index 指定集是必要的。

例

```
sets:
girls/debble,sue,alice/;
boys/bob,joe,sue,fred/;
endsets
I1=@index(sue);
I2=@index(boys,sue);
I1 的值是2，I2 的值是3。我们建议在使用@index 函数时最好指定集。
```

3. @wrap(index,limit)

该函数返回 $j = \text{index} - k * \text{limit}$ ，其中k 是一个整数，取适当值保证j 落在区间[1, limit]内。该函数相当于index 模limit 再加1。该函数在循环、多阶段计划编制中特别有用。

4. @size(set_name)

该函数返回集set_name 的成员个数。在模型中明确给出集大小时最好使用该函数。它的使用使模型更加数据中立，集大小改变时也更易维护。

九. LINGO中也有几个集循环函数集循环函数遍历整个集进行操作。

其语法为

```
@function(setname[(set_index_list)[|conditional_qualifier]]:expression_list);
```

@function 相应于下面罗列的四个集循环函数之一；setname 是要遍历的集；set_index_list 是集索引列表；conditional_qualifier 是用来限制集循环函数的范围，当集循环函数遍历集的每个成员时，LINGO 都要对 conditional_qualifier 进行评价，若结果为真，则对该成员执行@function 操作，否则跳过，继续执行下一次循环。expression_list 是被应用到每个集成员的表达式列表，当用的是@for 函数时，expression_list 可以包含多个表达式，其间用逗号隔开。这些表达式将被作为约束加到模型中。当使用其余的三个集循环函数时，expression_list 只能有一个表达式。如果省略set_index_list，那么在 expression_list 中引用的所有属性的类型都是setname 集。

1. @for

该函数用来产生对集成员的约束。基于建模语言的标量需要显式输入每个约束，不过@for 函数允许只输入一个约束，然后LINGO 自动产生每个集成员的约束。

例 产生序列{1, 4, 9, 16, 25}

```
model:
sets:
number/1..5/:x;
endsets
@for(number(I): x(I)=I^2);
end
```

2. @sum

该函数返回遍历指定的集成员的一个表达式的和。

例 求向量[5, 1, 3, 4, 6, 10]前5 个数的和。

```
model:
data:
N=6;
enddata
sets:
number/1..N/:x;
endsets
data:
x = 5 1 3 4 6 10;
enddata
s=@sum(number(I) | I #le# 5: x);
end
```

3. @min 和@max

返回指定的集成员的一个表达式的最小值或最大值。

例 求向量[5, 1, 3, 4, 6, 10]前5 个数的最小值，后3 个数的最大值。

```
model:
data:
N=6;
```



```

enddata
sets:
number/1..N/:x;
endsets
data:
x = 5 1 3 4 6 10;
enddata
minv=@min(number(I) | I #le# 5: x);
maxv=@max(number(I) | I #ge# N-2: x);
end

```

下面看一个稍微复杂一点儿的例子。

例 职员时序安排模型 一项工作一周7 天都需要有人(比如护士工作), 每天(周一至周日)所需的最少职员数为20、16、13、16、19、14 和12, 并要求每个职员一周连续工作5 天, 试求每周所需最少职员数, 并给出安排。注意这里我们考虑稳定后的情况。

```

model:
sets:
days/mon..sun/: required, start;
endsets
data:
!每天所需的最少职员数;
required = 20 16 13 16 19 14 12;
enddata
!最小化每周所需职员数;
min=@sum(days: start);
@for(days(J):
@sum(days(I) | I #le# 5:
start(@wrap(J+I+2,7))) >= required(J));
end

```

十. 输入和输出函数

输入和输出函数可以把模型和外部数据比如文本文件、数据库和电子表格等连接起来。

1. @file 函数

该函数用从外部文件中输入数据, 可以放在模型中任何地方。该函数的语法格式为@file(' filename')。这里filename 是文件名, 可以采用相对路径和绝对路径两种表示方式。@file 函数对同一文件的两种表示方式的处理和对两个不同的文件处理是一样的, 这一点必须注意。

例 @file 函数的用法。

注意第一个地方是集部分的6 个warehouses集成员和8 个vendors 集成员; 第二个地方是数据部分的capacity, demand 和cost 数据。为了使数据和我们的模型完全分开, 我们把它们移到外部的文本文件中。修改模型代码以便于用@file 函数把数据从文本文件中拖到模型中来。修改后(修改处代码黑体加粗)的模型代码如下:

```

model:
!6 发点8 收点运输问题;
sets:
warehouses/ @file('1_2.txt') /: capacity;
vendors/ @file('1_2.txt') /: demand;
links(warehouses,vendors): cost, volume;
endsets
!目标函数;
min=@sum(links: cost*volume);
!需求约束;
@for(vendors(J):
@sum(warehouses(I): volume(I,J))=demand(J));
!产量约束;
@for(warehouses(I):
@sum(vendors(J): volume(I,J))<=capacity(I));
!这里是数据;
data:
capacity = @file('1_2.txt') ;
demand = @file('1_2.txt') ;
cost = @file('1_2.txt') ;
enddata
end

```

模型的所有数据来自于1_2.txt 文件。其内容如下：

```

!warehouses 成员;
WH1 WH2 WH3 WH4 WH5 WH6 ~
!vendors 成员;
V1 V2 V3 V4 V5 V6 V7 V8 ~
!产量;
60 55 51 43 41 52 ~
!销量;
35 37 22 32 41 32 43 38 ~
!单位运输费用矩阵;
6 2 6 7 4 2 5 9
4 9 5 3 8 5 8 2
5 2 1 9 7 4 3 3
7 6 7 3 9 2 7 1
2 3 9 5 7 2 6 5
5 5 2 2 8 1 4 3

```

把记录结束标记（~）之间的数据文件部分称为记录。如果数据文件中没有记录结束标记，那么整个文件被看作单个记录。注意到除了记录结束标记外，模型的文本和数据同它们直接放在模型里是一样的。

我们来看一下在数据文件中的记录结束标记连同模型中@file 函数调用是如何工作的。当在模型中第一次调用@file 函数时，LINGO 打开数据文件，然后读取第一个记录；第二次调用@file 函数时，LINGO 读取第二个记录等等。文件

的最后一条记录可以没有记录结束标记，当遇到文件结束标记时，LINGO 会读取最后一条记录，然后关闭文件。如果最后一条记录也有记录结束标记，那么直到LINGO 求解完当前模型后才关闭该文件。如果多个文件保持打开状态，可能会导致一些问题，因为这会使同时打开的文件总数超过允许同时打开文件的上限16。

当使用@file 函数时，可把记录的内容（除了一些记录结束标记外）看作是替代模型中@file(' filename')位置的文本。这也就是说，一条记录可以是声明的一部分，整个声明，或一系列声明。在数据文件中注释被忽略。注意在LINGO中不允许嵌套调用@file 函数。

2. @text 函数

该函数被用在数据部分用来把解输出至文本文件中。它可以输出集成员和集属性值。其语法为@text([' filename'])

这里filename 是文件名，可以采用相对路径和绝对路径两种表示方式。如果忽略filename，那么数据就被输出到标准输出设备（大多数情形都是屏幕）。@text 函数仅能出现在模型数据部分的一条语句的左边，右边是集名（用来输出该集的所有成员名）或集属性名（用来输出该集属性的值）。

我们把用接口函数产生输出的数据声明称为输出操作。输出操作仅当求解器求解完模型后才执行，执行次序取决于其在模型中出现的先后。

例 @text 的用法。

```
model:
sets:
days/mon..sun/: required, start;
endsets
data:
!每天所需的最少职员数;
required = 20 16 13 16 19 14 12;
@text('d:\out.txt')=days '至少需要的职员数为' start;
enddata
!最小化每周所需职员数;
min=@sum(days: start);
@for(days(J):
@sum(days(I) | I #le# 5:
start(@wrap(J+I+2, 7))) >= required(J));
end
```

3. @ole 函数

@OLE 是从EXCEL 中引入或输出数据的接口函数，它是基于传输的OLE 技术。OLE 传输直接在内存中传输数据，并不借助于中间文件。当使用@OLE 时，LINGO 先装载EXCEL，再通知EXCEL 装载指定的电子数据表，最后从电子数据表中获得Ranges。为了使用OLE 函数，必须有EXCEL5 及其以上版本。OLE 函数可在数据部分和初始部分引入数据。

@OLE 可以同时读集成员和集属性，集成员最好用文本格式，集属性最好用数值格式。原始集每个集成员需要一个单元(cell)，而对于n 元的派生集每个集

成员需要n 个单元，这里第一行的n 个单元对应派生集的第一个集成员，第二行的n 个单元对应派生集的第二个集成员，依此类推。

@OLE 只能读一维或二维的Ranges（在单个的EXCEL 工作表(sheet)中），但不能读间断的或三维的Ranges。Ranges 是自左而右、自上而下来读。

例

```
sets:
PRODUCT; !产品;
MACHINE; !机器;
WEEK; !周;
ALLOWED (PRODUCT, MACHINE, WEEK):x, y; !允许组合及属性;
endsets
data:
rate=0.01;
PRODUCT, MACHINE, WEEK, ALLOWED, x, y=@OLE('D:\IMPORT.XLS');
@OLE('D:\IMPORT.XLS')=rate;
enddata
```

代替在代码文本的数据部分显式输入形式，我们把相关数据全部放在如下电子数据表中来输入。下面是D:\IMPORT.XLS 的图表。

除了输入数据之外，我们也必须定义Ranges 名：PRODUCT, MACHINE, WEEK, ALLOWED, x, y. 明确的，我们需要定义如下的Ranges 名：

Name	Range
PRODUCT	B3:B4
MACHINE	C3:C4
WEEK	D3:D5
ALLOWED	B8:D10
X	F8:F10
Y	G8:G10
rate	C13

为了在EXCEL 中定义Ranges 名：

- ① 按鼠标左键拖曳选择Range，
- ② 释放鼠标按钮，
- ③ 选择“插入|名称|定义”，
- ④ 输入希望的名字，
- ⑤ 点击“确定”按钮。

我们在模型的数据部分用如下代码从EXECL 中引入数据：

```
PRODUCT, MACHINE, WEEK, ALLOWED, x, y=@OLE('D:\IMPORT.XLS');
@OLE('D:\IMPORT.XLS')=rate;
```

等价的描述为

```
PRODUCT, MACHINE, WEEK, ALLOWED, x, y
=@OLE('D:\IMPORT.XLS', PRODUCT, MACHINE, WEEK, ALLOWED, x, y);
@OLE('D:\IMPORT.XLS', rate)=rate;
```

这一等价描述使得变量名和Ranges 不同亦可。

4. @ranged(variable_or_row_name)

为了保持最优基不变，变量的费用系数或约束行的右端项允许减少的量。

5. @rangeu(variable_or_row_name)

为了保持最优基不变，变量的费用系数或约束行的右端项允许增加的量。

6. @status()

返回LINGO 求解模型结束后的状态：

0 Global Optimum（全局最优）

1 Infeasible（不可行）

2 Unbounded（无界）

3 Undetermined（不确定）

4 Feasible（可行）

5 Infeasible or Unbounded（通常需要关闭“预处理”选项后重新求解模型，以确定模型究竟是不可行还是无界）

6 Local Optimum（局部最优）

7 Locally Infeasible（局部不可行，尽管可行解可能存在，但是LINGO 并没有找到一个）

8 Cutoff（目标函数的截断值被达到）

9 Numeric Error（求解器因在某约束中遇到无定义的算术运算而停止）

通常，如果返回值不是0、4 或6 时，那么解将不可信，几乎不能用。该函数仅被用在模型的数据部分来输出数据。

例

```
model:
min=@sin(x);
data:
@text()=@status();
enddata
end
```

部分计算结果为：

Local optimal solution found at iteration: 33

Objective value: -1.000000

6

Variable Value Reduced Cost

X 4.712388 0.000000

结果中的6 就是@status() 返回的结果，表明最终解是局部最优的。

7. @dual

@dual(variable_or_row_name)返回变量的判别数（检验数）或约束行的对偶（影子）价格（dual prices）。

十一.辅助函数

1. @if(logical_condition,true_result,false_result)

@if 函数将评价一个逻辑表达式logical_condition，如果为真，返回true_result，否则返回false_result。

2. @warn('text',logical_condition)

如果逻辑条件logical_condition 为真, 则产生一个内容为' text' 的信息框。

十二.LINGO 菜单

1. 求解模型 (Solve)

从LINGO 菜单中选用“求解”命令、单击“Solve”按钮或按Ctrl+S 组合键可以将当前模型送入内存求解。

2. 求解结果. . . (Solution. . .)

从LINGO 菜单中选用“Solution. . . ”命令、单击“Solution. . . ”按钮或直接按Ctrl+O组合键可以打开求解结果的对话框。这里可以指定查看当前内存中求解结果的那些内容。

3. 查看. . . (Look. . .)

从LINGO 菜单中选用“Look. . . ”命令或直接按Ctrl+L 组合键可以查看全部的或选中的模型文本内容。

4. 灵敏性分析 (Range, Ctrl+R)

用该命令产生当前模型的灵敏性分析报告: 研究当目标函数的费用系数和约束右端项在什么范围(此时假定其它系数不变)时, 最优基保持不变。灵敏性分析是在求解模型时作出的, 因此在求解模型时灵敏性分析是激活状态, 但是默认是不激活的。为了激活灵敏性分析, 运行LINGO|Options..., 选择General Solver Tab, 在Dual Computations 列表框中, 选择Prices and Ranges 选项。灵敏性分析耗费相当多的求解时间, 因此当速度很关键时, 就没有必要激活它。

十三.LINGO中的目标函数和约束条件的表示

例3的线性规划的LINGO程序中, 目标函数的表达形式为:

```
max=@sum(js(J):c(J)*x(J));
```

恰好和线性规划模型中的表达式 $\max z = \sum_{j=1}^n c_j x_j$ 相对应。

例3程序中的约束条件

```
@for(is(I): @sum(js(J):a(I,J)*x(J))<=b(I));
```

恰好和约束条件 $\sum_{j=1}^n a_{ij} x_j \leq b_i, (i=1, \dots, m)$ 相对应。

§2.3 LINGO 中的数据

以关键字“data:”开始, 以关键字“enddata”结束的数据部分也是 LINGO 程序的重要组成部分。

一. 数据部分入门

数据部分提供了模型相对静止部分和数据分离的可能性。显然, 这对模型的维护和维数的缩放非常便利。其语法如下:

```
object_list = value_list;
```

对象列(object_list)包含要指定值的属性名、要设置集成员的集名, 用逗号或空格隔开。一个对象列中至多有一个集名, 而属性名可以有任意多。如果

对象列中有多个属性名，那么它们的类型必须一致。如果对象列中有一个集名，那么对象列中所有的属性的类型就是这个集。

数值列 (value_list) 包含要分配给对象列中的对象的值，用逗号或空格隔开。注意属性值的个数必须等于集成员的个数。看下面的例子。

例

```
sets:  
set1/A,B,C/: X,Y;
```

```
endsets
```

```
data:
```

```
X=1,2,3;
```

```
Y=4,5,6;
```

```
enddata
```

在集set1 中定义了两个属性X 和Y。X 的三个值是1、2 和3，Y 的三个值是4、5 和6。也可采用如下例子中的复合数据声明 (data statement) 实现同样的功能。

例

```
sets:  
set1/A,B,C/: X,Y;
```

```
endsets
```

```
data:
```

```
X,Y=1 4
```

```
2 5
```

```
3 6;
```

```
enddata
```

看到这个例子，可能会认为X 被指定了1、4 和2 三个值，因为它们是数值列中前三个，而正确的答案是1、2 和3。假设对象列有n 个对象，LINGO 在为对象指定值时，首先在n个对象的第1 个索引处依次分配数值列中的前n 个对象，然后在n 个对象的第2个索引处依次分配数值列中紧接着的n 个对象，……，以此类推。模型的所有数据——属性值和集成员——被单独放在数据部分，这可能是最规范的数据输入方式。

二. 参数

在数据部分也可以指定一些标量变量 (scalar variables)。当一个标量变量在数据部分确定时，称之为参数。看一例，假设模型中用利率8.5%作为一个参数，就可以象下面一样输入一个利率作为参数。

例

```
data:  
interest_rate = .085;
```

```
enddata
```

也可以同时指定多个参数。

例

```
data:  
interest_rate,inflation_rate = .085 .03;
```

```
enddata
```

三. 实时数据处理

在某些情况，对于模型中的某些数据并不是定值。譬如模型中有一个通货膨胀率

的参数，
我们想在2%至6%范围内，对不同的值求解模型，来观察模型的结果对通货膨胀的依赖有多
么敏感。我们把这种情况称为实时数据处理（what if analysis）。LINGO 有一个特征可方便地做到这件事。
在本该放数的地方输入一个问号（?）。

例

`data:`

```
interest_rate, inflation_rate = .085 ?;
```

`enddata`

每一次求解模型时，LINGO 都会提示为参数inflation_rate 输入一个值。在WINDOWS 操作系统下，将会接收到一个类似下面的对话框：

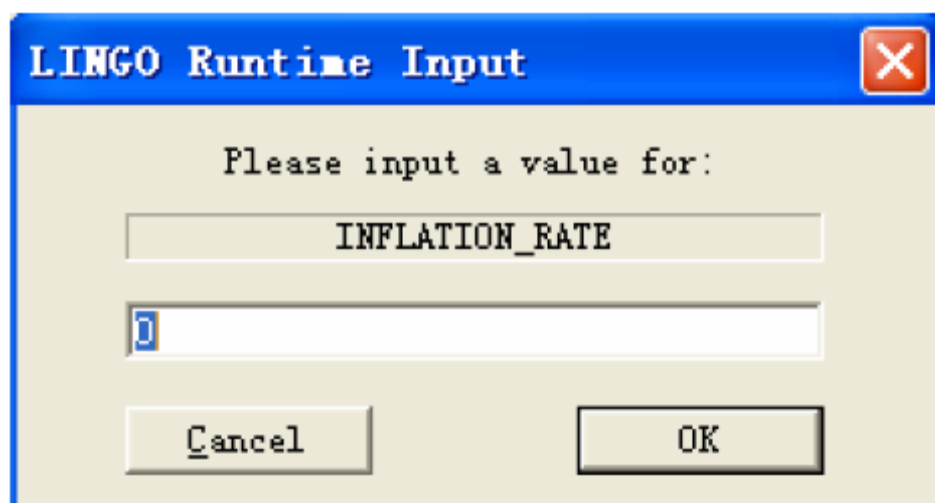


图 10

直接输入一个值再点击OK 按钮，LINGO 就会把输入的值指定给inflation_rate，然后继续求解模型。除了参数之外，也可以实时输入集的属性值，但不允许实时输入集成员名。

四．指定属性为一个值

可以在数据声明的右边输入一个值来把所有的成员的该属性指定为一个值。看下面的例子。

例

`sets:`

```
days /MO, TU, WE, TH, FR, SA, SU/:needs;
```

`endsets`

`data:`

```
needs = 20;
```

`enddata`

LINGO 将用20 指定days 集的所有成员的needs 属性。对于多个属性的情形，见下例。

例


```
sets:
days /MO, TU, WE, TH, FR, SA, SU/:needs, cost;
endsets
data:
needs cost = 20 100;
enddata
```

五. 数据部分的未知数值

有时只想为一个集的部分成员的某个属性指定值，而让其余成员的该属性保持未知，以便让LINGO 去求出它们的最优值。在数据声明中输入两个相连的逗号表示该位置对应的集成

员的属性值未知。两个逗号间可以有空格。

例

```
sets:
years/1..5/: capacity;
endsets
data:
capacity = ,34,20,,;
enddata
```

属性capacity 的第2 个和第3 个值分别为34 和20，其余的未知。

六. 模型的初始部分

初始部分是LINGO 提供的另一个可选部分。在初始部分中，可以输入初始声明（initialization statement），和数据部分中的数据声明相同。对实际问题的建模时，初始部分并不起到描述模型的作用，在初始部分输入的值仅被LINGO 求解器当作初始点来用，并且仅仅对非线性模型有用。和数据部分指定变量的值不同，LINGO 求解器可以自由改变初始部分初始化的变量的值。一个初始部分以“init:”开始，以“endinit”结束。初始部分的初始声明规则和数据部分的数据声明规则相同。也就是说，我们可以在声明的左边同时初始化多个集属性，可以把集属性初始化为一个值，可以用问号实现实时数据处理，还可以用逗号指定未知数值。

例

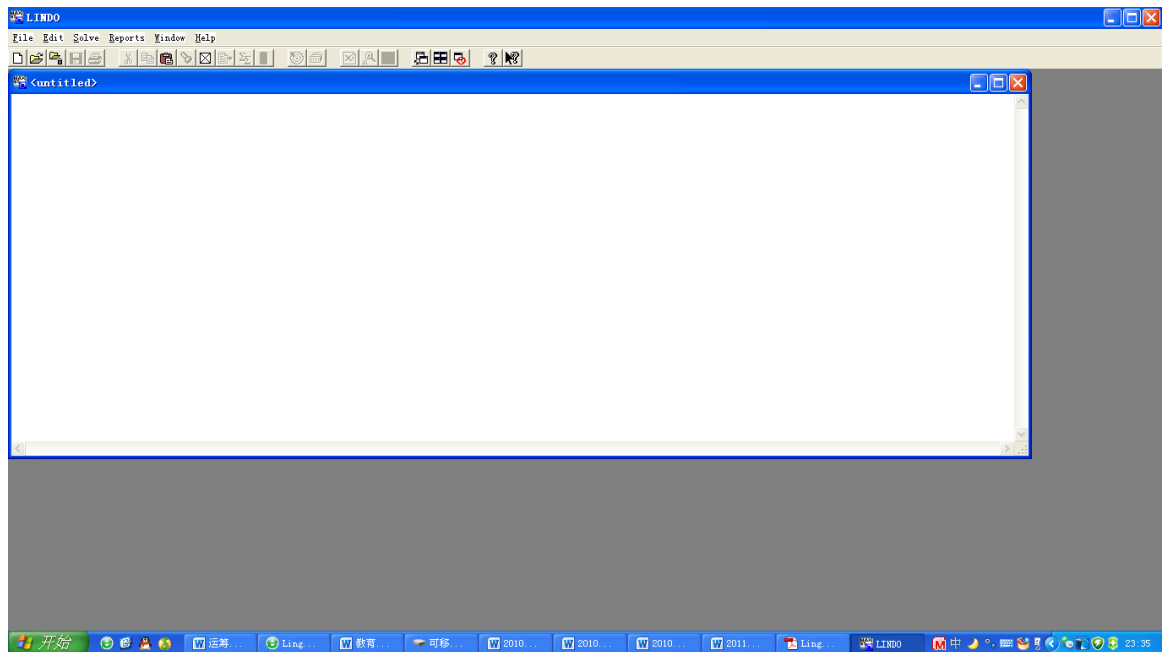
```
init:
X, Y = 0, .1;
endinit
Y=@log(X);
X^2+Y^2<=1;
```

好的初始点会减少模型的求解时间。在这一节中，我们仅带大家接触了一些基本的数据输入和初始化概念，不过现在你应该可以轻松为自己的模型加入原始数据和初始部分啦。

§2.4 LINDO 简介

LINDO 是Linear, INteractive, and Discrete Optimizer的缩写，它是一个便利而有强大的工具软件，常用于求解线性规划（LP——Linear Programming）、整数规划（IP——Integer Programming）和二次规划（QP——Quadratic Programming）问题，这些问题一般在商业、工业、科研和政府工作中都会遇到。LINDO已被证实 在一些特殊领域能发挥巨大作用，如产品配售、配料问题、生产与人员时序安排、库存管理等。

打开 LINDO 的初始界面如下图，要求解线性规划就可以输入相应代码。



例 求解下面的线性规划

$$\begin{aligned}\max \quad & z=2x_1+3x_2 \\ & x_1+2x_2 \leq 8 \\ & 4x_1 \leq 16 \\ & 4x_2 \leq 16 \\ & x_1, x_2 \geq 0\end{aligned}$$

LINGO 中不能获得单纯形表，可以在 LINDO 中获得
首先，打开 LINDO，输入如下代码：

```
MAX 2X1 + 3X2
st
  X1 + 2X2 <= 8
  4X1 <= 16
  4X2 <= 12
```

或者

```
MAX 2 X1 + 3 X2
s.t.
  X1 + 2 X2 <= 8
  4 X1 <= 16
```

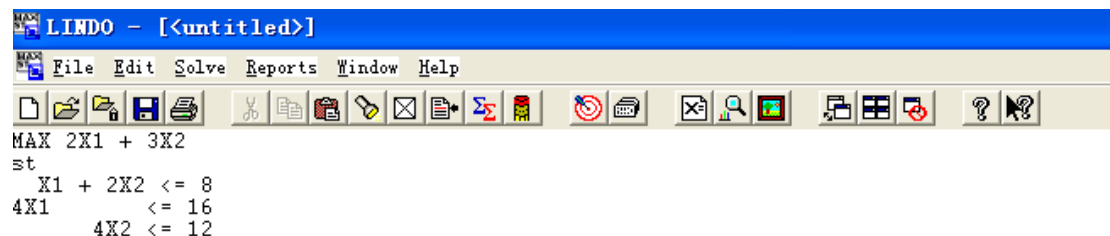
$4 X_2 \leq 12$
 或者
 $\text{MAX } 2 X_1 + 3 X_2$
 s.t.
 $X_1 + 2 X_2 \leq 8$
 $4 X_1 \leq 16$
 $4 X_2 \leq 12$

注：(1) 目标函数之后和约束之前用下述的形式之一来表示：

SUBJECT TO
 SUCH THAT
 S.T.
 ST

约束之后用END来表示结束。

- (2) 目标函数不输 $z=$;
- (3) 数与变量的乘号可以省略，也可用空格代替；
- (4) 系统默认变量非负。



在菜单中 Reports→Tableau，可得初始单纯形表

ROW (BASIS)		X1	X2	SLK 2	SLK 3	SLK 4	
1	ART	-2.000	-3.000	0.000	0.000	0.000	0.000
2	SLK 2	1.000	2.000	1.000	0.000	0.000	8.000
3	SLK 3	4.000	0.000	0.000	1.000	0.000	16.000
4	SLK 4	0.000	4.000	0.000	0.000	1.000	12.000
ART	ART	-2.000	-3.000	0.000	0.000	0.000	0.000

图 18

THE TABLEAU(初始单纯形表)

ROW	(BASIS)	X1	X2	SLK2	SLK 3	SLK4	
1	ART	-2.000	-3.000	0.000	0.000	0.000	0.000
2	SLK 2	1.000	2.000	1.000	0.000	0.000	8.000
3	SLK 3	4.000	0.000	0.000	1.000	0.000	16.000
4	SLK 4	0.000	4.000	0.000	0.000	1.000	12.000
ART	ART	-2.000	-3.000	0.000	0.000	0.000	0.000

求解并作灵敏度分析(Ranges)可得如下结果：

LP OPTIMUM FOUND AT STEP 0

OBJECTIVE FUNCTION VALUE

1) 14.00000

VARIABLE	VALUE	REDUCED COST
----------	-------	--------------

X1	4.000000	0.000000
----	----------	----------

X2	2.000000	0.000000
----	----------	----------

ROW	SLACK OR SURPLUS	DUAL PRICES
-----	------------------	-------------

2)	0.000000	1.500000
----	----------	----------

3)	0.000000	0.125000
----	----------	----------

4)	4.000000	0.000000
----	----------	----------

NO. ITERATIONS= 0

RANGES IN WHICH THE BASIS IS UNCHANGED(最优基不变的灵敏度分析):

OBJ COEFFICIENT RANGES(目标函数价值系数变化)

VARIABLE	CURRENT	ALLOWABLE	ALLOWABLE
----------	---------	-----------	-----------

	COEF	INCREASE	DECREASE
--	------	----------	----------

X1	2.000000	INFINITY	0.500000
----	----------	----------	----------

X2	3.000000	1.000000	3.000000
----	----------	----------	----------

RIGHTHAND SIDE RANGES(右端系数变化)

ROW	CURRENT	ALLOWABLE	ALLOWABLE
-----	---------	-----------	-----------

	RHS	INCREASE	DECREASE
--	-----	----------	----------

2	8.000000	2.000000	4.000000
---	----------	----------	----------

3	16.000000	16.000000	8.000000
---	-----------	-----------	----------

4	12.000000	INFINITY	4.000000
---	-----------	----------	----------

在菜单中 Reports→Tableau, 可得最优单纯形表:

THE TABLEAU(最优单纯形表)

ROW	(BASIS)	X1	X2	SLK2	SLK3	SLK4	
1	ART	0.000	0.000	1.500	0.125	0.000	14.000
2	X2	0.000	1.000	0.500	-0.125	0.000	2.000
3	X1	1.000	0.000	0.000	0.250	0.000	4.000
4	SLK 4	0.000	0.000	-2.000	0.500	1.000	4.000
ART	ART	0.000	0.000	1.500	0.125	0.000	0.000

注: LINDO 的单纯形表中, ART 表示 artificial variable, 实际上是目标函数行, 该行中各变量的系数是最大化问题检验数的相反数, SLK2, SLK3 和 SLK4 表示三个松弛变量。

LINDO变量命名规则:

LINDO中变量名最多可用八个字母, 变量名必须以字母(A到Z)开始, 其后最多可以跟七个其它字符, 这些字符可以包括除下述符号外的任意符号: !) + - = < > 。因此, 作为示例下述名称将被视为有效命名:

XYZ MY_VAR A12 SHIP.LA

而下述名称将被视为无效命名:

THISONESTOOLONG A-HYPHEN 1INFRONT

第一个示例包含字符多于八个, 而第二个示例包含了禁用的符号, 最后一个例子不是以字母开头。

LINDO中约束也可以命名

约束名称使得许多LINDO的输出报告更易理解。约束命名必须遵从和变量命

名一样的规则，要命名一个约束你必须在约束名后加一个右圆括号作为结束标志，然后如前面一样输入约束。例如下面对：

A型配件) $5x_1+2x_2<180$

运算符：

LINDO 只认可五个运算符：加(+)、减(-)、大于(>)、小于(<)和等于(=)。当你输入严格不等号 大于(>) 或小于(<)，LINDO 将把它们解释为非严格不等号 \geq 或 \leq 。这是因为许多键盘没有非严格不等号。在有非严格不等号的系统中，LINDO 将不承认它们。不过如果你愿意，你可以输入“ \geq ”(和“ \leq ”)来代替“>”(和“<”)。

LINDO不接受圆括号作为优先运算的标志，所有运算被安排为从左到右进行。所以所有变量不能重复，变量前的系数必须化简。

注释可以放在模型中任何地方，用感叹号来标识。感叹号后当前行的所有东西会被认为是注释。例如本章前面讨论的小例子用注释重新表示如下：

max $20x_1+15x_2$!求最大利润

s.t.

!下面的约束是现有A型配件和B型配件总数限制

A型配件) $5x_1+2x_2<180$

B型配件) $3x_1+4x_2<135$

end

LINDO对大小写不区分：所有输入被LINDO内核转化为大写格式。

约束和目标函数可以用多行表示，也可以合并到一行中。除了变量名或系数，你可以在任何地方分割一行。

约束方程的格式：

约束方程的右侧只能是约束值而不能含变量，因此一个像下面的输入将被LINDO拒绝

$X > Y$

这条输入可改为如下形式：

$X - Y > 0$

反之，只有变量和它们的系数可以放在约束方程的左侧。例如下面的约束是不允许的

$3X + 4Y - 10 = 0$

因为常数-10被放在了左侧，这个约束可以改为：

$3X + 4Y = 10$

建模命令	功能
FREE <Variable>	消除<Variable>所有界限，允许<Variable>使用任意实数，正数或负数
GIN <Variable>	使<Variable>取一般整数值（即限取非负整数）
INT <Variable>	使<Variable>取二值（即限取0或1）
SLB <Variable> <Value>	对<Variable>置一个简单下界<Value>。用来代替X r 约束
SUB <Variable> <Value>	对<Variable>置一个简单上界<Value>。用来代替X r 约束
QCP <Constraint>	在二次规划模型中标志“实”约束的开头
TITLE <Title>	定义模型名称为<Title>

max $20x_1+15x_2$!求最大利润

s.t.

!下面的约束是现有 A 型配件和 B 型配件总数限制

A 型配件) $5x_1+2x_2<180$

B 型配件) $3x_1+4x_2<135$

end

gin x1

gin x2

LINDO可以求解单纯的或混合的整数规划（IP）问题，所用方法是分枝定界法。

数规划指的是所有变量都是整数型的，混合的整数规划指的是部分变量是整数、部分变量是连续变化的。灵敏度分析在整数规划中用处不大(尚无完善理论)。整数规划（IP）问题的LINDO求解方法与LP问题类似，但是要在END标志后定义整型变量：

GIN var 或 GIN n

前者将变量var定义为整数变量，后者将前n个变量定义为整数变量。

和LINGO一样，用LINDO求出的只是所有最优解（最优解可能有多个）中的一个而已。

LINDO 中解 0-1 规划的用命令是在约束后添加变量定义，INT var 或 INT n (n 指前 n 个变量标识为 0/1 型)

第三章 运筹学上机实验及要求

实验一.中小型线性规划模型的求解与 Lingo 软件的使用

一. 实验目的：了解 Lingo 软件的基本功能和简单线性规划模型的求解的输入和输出结果。

二. 实验内容：

1. 在 Lingo 中求解下面的线性规划数学模型；

$$\max z=2x_1+3x_2;$$

$$x_1+2x_2\leq 8$$

$$4x_1\leq 16$$

$$4x_2\leq 12$$

$$x_1, x_2\geq 0;$$

2. 在 Lingo 中求解教材 P44 习题 1.2(1)的线性规划数学模型；

3. 建立教材 P38 例 10 的数学模型并用 Lingo 求解。

4. 建立教材 P46 习题 1.9 的数学模型并用 Lingo 求解。

三. 实验要求：

1. 给出所求解问题的数学模型；

2. 给出 Lingo 中的输入；

3. 能理解 Solution Report 中输出的四个部分的结果；

4. 能给出最优解和最优值；

5. 能理解哪些约束是取等式和哪些约束取不等式。

四. 写出实验报告。

实验二.中小型运输问题数学模型的 Lingo 软件求解。

一.实验目的：熟悉运输问题的数学模型，掌握简单运输问题数学模型的 Lingo 软件求解的方法，掌握解报告的内容。

二.实验内容：

用 Lingo 求解教材 P79 例 1

三.实验要求：

1.写出数学模型；

2.在 Lingo 中输入求解的程序；

3.求解得到解报告；

4.写出最优解和最优值；

四.写出实验报告。

实验三.大型线性规划模型的编程求解。

一.实验目的：掌握求解大型线性规划模型 Lingo 软件的编程的基本方法。

二.实验内容：

1. 在 Lingo 中编程求解下面的线性规划数学模型；

$$\max z=2x_1+3x_2;$$

$$x_1 + 2x_2 \leq 8$$

$$4x_1 \leq 16$$

$$4x_2 \leq 12$$

$$x_1, x_2 \geq 0;$$

2. 在 Lingo 中编程求解教材 P44 习题 1.2(1)的线性规划数学模型;
3. 建立教材 P38 例 10 的数学模型并用 Lingo 编程求解;
4. 建立教材 P46 习题 1.9 的数学模型并用 Lingo 编程求解。

三.实验要求:

1. 给出所求解问题的数学模型;
2. 给出 Lingo 中的编程程序;
3. 能给出最优解和最优值;
4. 指出哪些约束是取等式和哪些约束取不等式。

四.写出实验报告。

实验四.运输问题数学模型的 Lingo 编程求解。

一.实验目的: 熟悉运输问题的数学模型, 掌握简单运输问题数学模型的 Lingo 软件编程求解的方法, 掌握解报告的内容。

二.实验内容:

1. 用 Lingo 编程求解教材 P79 例 1;
2. 建立教材 P99 习题 3.5 的数学模型并 Lingo 编程求解(两个问题: 一个产销平衡, 另一个产销不平衡 A—2500 套)。

三.实验要求:

1. 写出数学模型;
2. 在 Lingo 中输入求解的程序;
3. 求解得到解报告;
4. 写出最优解和最优值;

四.写出实验报告。

实验五.分支定界法上机实验

一.实验目的: 通过分支定界法的上机实验, 掌握分支定界法的思想和方法和步骤。

二.实验内容:

用分支定界法求解教材 p131 习题 5.2。

三.实验要求:

1. 写出要求解的数学模型;
2. 写出分支和定界的过程;
3. 写出在分支和定界过程中求解的每一个线性规划和 Lingo 程序;
4. 写出最优解和最优值。

四.写出实验报告。

实验六.整数规划、0-1 规划和指派问题的计算机求解

一.实验目的：掌握整数规划、0-1 规划和指派问题的计算机求解方法。

二.实验内容：

- 1.在 Lingo 中求解整数规划(教材)p131 习题 5.1。
2. 在 Lingo 中求解 0-1 规划(教材)p132 习题 5.6(1)。
3. 在 Lingo 中求解指派问题(教材)p132 习题 5.7。

三.实验要求：

- 1.写出求解的每一个问题 Lingo 程序；
- 2.给出解报告；
- 3.写出最优解和最优值。

四.写出实验报告

实验七：最短路问题的计算机求解

一.实验目的：掌握最短路问题的计算机求解方法。

二.实验内容：

- 1.在 Lingo 中求解 (教材)p261 例 10 的最短路问题。
2. 在 Lingo 中求解 (教材)p261 习题 10.6 的最短路问题。

三.实验要求：

- 1.写出求解的 Lingo 程序；
- 2.写出最短路线及其权。

四.写出实验报告。

实验八:最大流问题的计算机求解

一.实验目的：掌握最大流问题的计算机求解方法。

二.实验内容：

- 1.在 Lingo 中求解教材 p283 习题 10.12 的最大流。
2. 在 Lingo 中求解教材 p283 习题 10.12 的最大流。

三.实验要求：

- 1.写出求解的每一个问题的 Lingo 程序；
- 3.写出最优解和最优值。

四.写出实验报告。

实验九:运筹学综合实验

一.实验目的：

运筹学课程是以解决实际的生产和管理问题为基础，通过建立数学模型以及对模型的求解与分析，获得解决问题的最优方案。线性规划是运筹学中最重要的一类数学模型，其应用非常广泛，该实验的目的是使学生通过对实际问题的完整解决的过程，有助于学生熟悉和掌握有关的核心的知识和技术，也能培养学生的动手动脑的思维和习惯，提高学习的兴趣和学习效果。

二.实验要求：

以四人为一组，选择一个生产和管理活动中的实际决策问题，建立线性规划的数学模型，编制和调试计算机程序并求解，对结果进行分析，最后写出一个完

整的论文或实验报告。

三.应用的知识面:

- 1、线性规划的数学模型;
- 2、运输问题的数学模型;
- 3、0-1 规划的数学模型;

四.实验手段:

编制和调试多个计算机程序。由于本实验将涉及到该课程的多个知识点,由建模、编程、调试、求解和分析等多项任务组成,因此,把该实验确定为综合性实验。

五.本实验的资料

露天矿生产的车辆安排 (CMCM2003B)

钢铁工业是国家工业的基础之一,铁矿是钢铁工业的主要原料基地。许多现代化铁矿是露天开采的,它的生产主要是由电动铲车(以下简称电铲)装车、电动轮自卸卡车(以下简称卡车)运输来完成。提高这些大型设备的利用率是增加露天矿经济效益的首要任务。

露天矿里有若干个爆破生成的石料堆,每堆称为一个铲位,每个铲位已预先根据铁含量将石料分成矿石和岩石。一般来说,平均铁含量不低于 25%的为矿石,否则为岩石。每个铲位的矿石、岩石数量,以及矿石的平均铁含量(称为品位)都是已知的。每个铲位至多能安置一台电铲,电铲的平均装车时间为 5 分钟。

卸货地点(以下简称卸点)有卸矿石的矿石漏、2 个铁路倒装场(以下简称倒装场)和卸岩石的岩石漏、岩场等,每个卸点都有各自的产量要求。从保护国家资源的角度及矿山的经济效益考虑,应该尽量把矿石按矿石卸点需要的铁含量(假设要求都为 $29.5\% \pm 1\%$,称为品位限制)搭配起来送到卸点,搭配的量在一个班次(8 小时)内满足品位限制即可。从长远看,卸点可以移动,但一个班次内不变。卡车的平均卸车时间为 3 分钟。

所用卡车载重量为 154 吨,平均时速 28 km/h 。卡车的耗油量很大,每个班次每台车消耗近 1 吨柴油。发动机点火时需要消耗相当多的电瓶能量,故一个班次中只在开始工作时点火一次。卡车在等待时所耗费的能量也是相当可观的,原则上在安排时不应发生卡车等待的情况。电铲和卸点都不能同时为两辆及两辆以上卡车服务。卡车每次都是满载运输。

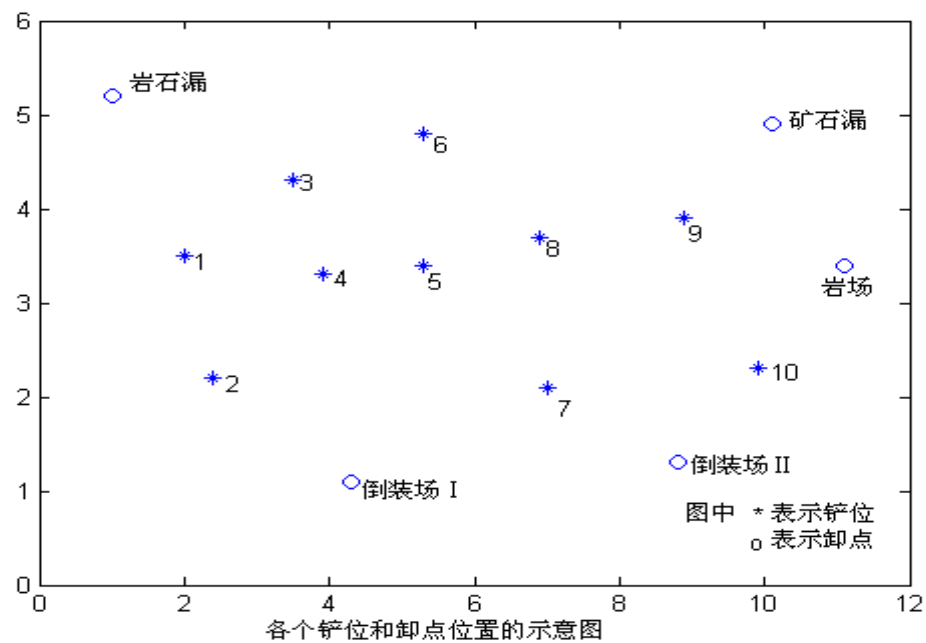
每个铲位到每个卸点的道路都是专用的宽 60 m 的双向车道,不会出现堵车现象,每段道路的里程都是已知的。

一个班次的生产计划应该包含以下内容:出动几台电铲,分别在哪些铲位上;出动几辆卡车,分别在哪些路线上各运输多少次(因为随机因素影响,装卸时间与运输时间都不精确,所以排时计划无效,只求出各条路线上的卡车数及安排即可)。一个合格的计划要在卡车不等待条件下满足产量和质量(品位)要求,而一个好的计划还应该考虑下面两条原则之一:

1. 总运量(吨公里)最小,同时出动最少的卡车,从而运输成本最小;
2. 利用现有车辆运输,获得最大的产量(岩石产量优先;在产量相同的情况下,取总运量最小的解)。

请你就两条原则分别建立数学模型,并给出一个班次生产计划的快速算法。针对下面的实例,给出具体的生产计划、相应的总运量及岩石和矿石产量。

某露天矿有铲位 10 个,卸点 5 个,现有铲车 7 台,卡车 20 辆。各卸点一个班次的产量要求:矿石漏 1.2 万吨、倒装场 I 1.3 万吨、倒装场 II 1.3 万吨、岩石漏 1.9 万吨、岩场 1.3 万吨。



铲位和卸点位置二维示意图如下，各铲位和各卸点之间的距离（公里）如下表：

	铲位 1	铲位 2	铲位 3	铲位 4	铲位 5	铲位 6	铲位 7	铲位 8	铲位 9	铲位 10
矿石漏	5.26	5.19	4.21	4.00	2.95	2.74	2.46	1.90	0.64	1.27
倒装场 I	1.90	0.99	1.90	1.13	1.27	2.25	1.48	2.04	3.09	3.51
岩场	5.89	5.61	5.61	4.56	3.51	3.65	2.46	2.46	1.06	0.57
岩石漏	0.64	1.76	1.27	1.83	2.74	2.60	4.21	3.72	5.05	6.10
倒装场 II	4.42	3.86	3.72	3.16	2.25	2.81	0.78	1.62	1.27	0.50

各铲位矿石、岩石数量(万吨)和矿石的平均铁含量如下表：

	铲位 1	铲位 2	铲位 3	铲位 4	铲位 5	铲位 6	铲位 7	铲位 8	铲位 9	铲位 10
矿石量	0.95	1.05	1.00	1.05	1.10	1.25	1.05	1.30	1.35	1.25
岩石量	1.25	1.10	1.35	1.05	1.15	1.35	1.05	1.15	1.35	1.25
铁含量	30%	28%	29%	32%	31%	33%	32%	31%	33%	31%