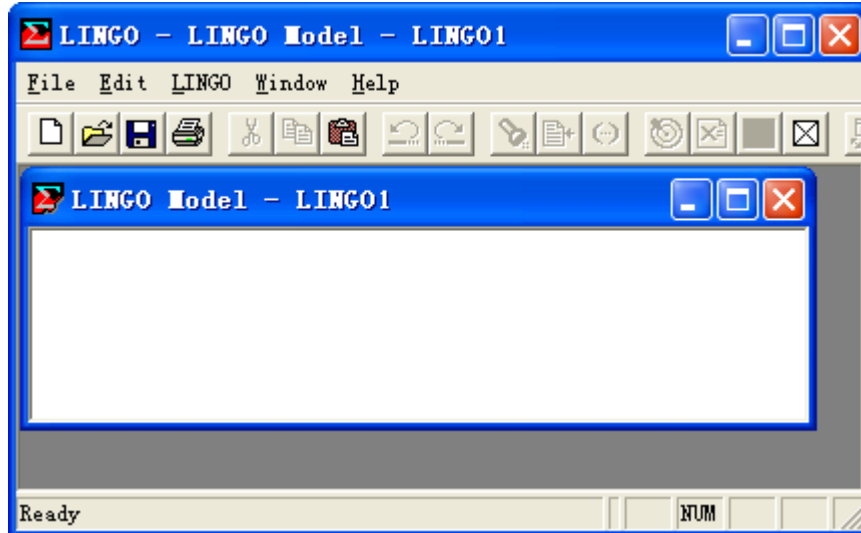


LINGO 是用来求解线性和非线性优化问题的简易工具。LINGO 内置了一种建立最优化模型的语言，可以简便地表达大规模问题，利用 LINGO 高效的求解器可快速求解并分析结果。

## § 1 LINGO 快速入门

当你在 windows 下开始运行 LINGO 系统时，会得到类似下面的一个窗口：



外层是主框架窗口，包含了所有菜单命令和工具条，其它所有的窗口将被包含在主窗口之下。在主窗口内的标题为 LINGO Model - LINGO1 的窗口是 LINGO 的默认模型窗口，建立的模型都都要在该窗口内编码实现。下面举两个例子。

**例 1.1** 如何在 LINGO 中求解如下的 LP 问题：

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & x_1 + x_2 \geq 350 \\ & x_1 \geq 100 \\ & 2x_1 + x_2 \leq 600 \\ & x_1, x_2 \geq 0 \end{aligned}$$


在模型窗口中输入如下代码：

```
min=2*x1+3*x2;
```

```
x1+x2>=350;
```

```
x1>=100;
```

```
2*x1+x2<=600;
```

然后点击工具条上的按钮  即可。

**例 1.2** 使用 LINGO 软件计算 6 个发点 8 个收点的最小费用运输问题。产销单位运价如下表。

| 单位运价<br>产地 \ 销地 | B <sub>1</sub> | B <sub>2</sub> | B <sub>3</sub> | B <sub>4</sub> | B <sub>5</sub> | B <sub>6</sub> | B <sub>7</sub> | B <sub>8</sub> | 产量 |
|-----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----|
| A <sub>1</sub>  | 6              | 2              | 6              | 7              | 4              | 2              | 5              | 9              | 60 |
| A <sub>2</sub>  | 4              | 9              | 5              | 3              | 8              | 5              | 8              | 2              | 55 |

|                |    |    |    |    |    |    |    |    |    |
|----------------|----|----|----|----|----|----|----|----|----|
| A <sub>3</sub> | 5  | 2  | 1  | 9  | 7  | 4  | 3  | 3  | 51 |
| A <sub>4</sub> | 7  | 6  | 7  | 3  | 9  | 2  | 7  | 1  | 43 |
| A <sub>5</sub> | 2  | 3  | 9  | 5  | 7  | 2  | 6  | 5  | 41 |
| A <sub>6</sub> | 5  | 5  | 2  | 2  | 8  | 1  | 4  | 3  | 52 |
| 销量             | 35 | 37 | 22 | 32 | 41 | 32 | 43 | 38 |    |


使用 LINGO 软件，编制程序如下：

```

model:
!6 发点 8 收点运输问题;
sets:
    warehouses/wh1..wh6/: capacity;
    vendors/v1..v8/: 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=60 55 51 43 41 52;
    demand=35 37 22 32 41 32 43 38;
    cost=6 2 6 7 4 2 9 5
          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;
enddata
end

```

然后点击工具条上的按钮  即可。

为了能够使用 LINGO 的强大功能，接着第二节的学习吧。

## § 2 LINGO 中的集

对实际问题建模的时候，总会遇到一群或多群相联系的对象，比如工厂、消费者群体、交通工具和雇工等等。LINGO 允许把这些相联系的对象聚合成**集** (sets)。一旦把对象聚集成集，就可以利用集来最大限度的发挥 LINGO 建模语言的优势。

现在我们将深入介绍如何创建集，并用数据初始化集的属性。学完本节后，你对基于建模技术的集如何引入模型会有一个基本的理解。

### 2.1 为什么使用集

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。

## 2.2 什么是集

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。

LINGO 有两种类型的集：**原始集**(primitive set)和**派生集**(derived set)。

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。

## 2.3 模型的集部分

**集部分**是 LINGO 模型的一个可选部分。在 LINGO 模型中使用集之前，必须在集部分事先定义。集部分以关键字“sets:”开始，以“endsets”结束。一个模型可以没有集部分，或有一个简单的集部分，或多个集部分。一个集部分可以放置于模型的任何地方，但是一个集及其属性在模型约束中被引用之前必须定义了它们。

### 2.3.1 定义原始集

为了定义一个原始集，必须详细声明：

- 集的名字
  - 可选，集的成员
  - 可选，集成员的属性

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。下同，不再赘述。*

Setname 是你选择的来标记集的名字，最好具有较强的可读性。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。

*注意：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。

① 当显式罗列成员时，必须为每个成员输入一个不同的名字，中间用空格或逗号隔开，允许混合使用。

**例 2.1** 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

② 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 |

③ 集成员不放在集定义中，而在随后的**数据部分**来定义。

### 例 2.2

!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

*注意：开头用感叹号 (!)，末尾用分号 (;) 表示注释，可跨多行。*

在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。

集成员无论用何种字符标记，它的索引都是从 1 开始连续计数。在 attribute\_list 可以指定一个或多个集成员的属性，属性之间必须用逗号隔开。

可以把集、集成员和集属性同 C 语言中的结构体作个类比。如下图：

|     |   |       |
|-----|---|-------|
| 集   | ↔ | 结构体   |
| 集成员 | ↔ | 结构体的域 |
| 集属性 | ↔ | 结构体实例 |

LINGO 内置的建模语言是一种描述性语言，用它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。

### 2.3.2 定义派生集

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 2.3

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。幸运的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 2.4

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;
```

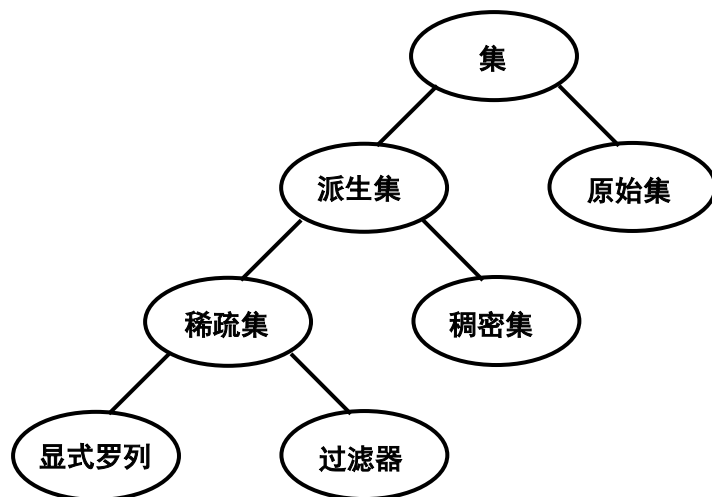
enddata

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。

总的来说,LINGO 可识别的集只有两种类型:原始集和派生集。

在一个模型中,原始集是基本的对象,不能再被拆分成更小的组分。原始集可以由显式罗列和隐式罗列两种方式定义。当用显式罗列方式时,需在集成员列表中逐个输入每个成员。当用隐式罗列方式时,只需在集成员列表中输入首成员和末成员,而中间的成员由 LINGO 产生。

另一方面,派生集是由其它的集来创建。这些集被称为该派生集的父集(原始集或其它的派生集)。一个派生集既可以是稀疏的,也可以是稠密的。稠密集包含了父集成员的所有组合(有时也称为父集的笛卡尔乘积)。稀疏集仅包含了父集的笛卡尔乘积的一个子集,可通过显式罗列和成员资格过滤器这两种方式来定义。显式罗列方法就是逐个罗列稀疏集的成员。成员资格过滤器方法通过使用稀疏集成员必须满足的逻辑条件从稠密集成员中过滤出稀疏集的成员。不同集类型的关系见下图。



LINGO 集类型

### § 3 模型的数据部分和初始部分

在处理模型的数据时，需要为集指派一些成员并且在 LINGO 求解模型之前为集的某些属性指定值。为此，LINGO 为用户提供了两个可选部分：输入集成员和数据的**数据部分**（Data Section）和为决策变量设置初始值的**初始部分**（Init Section）。

#### 3.1 模型的数据部分

##### 3.1.1 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。

数据部分以关键字“data:”开始，以关键字“enddata”结束。在这里，可以指定集成员、集的属性。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 3.1

```
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 3.2

```
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。

### 3.1.2 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 3.3

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

也可以同时指定多个参数。

#### 例 3.4

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

### 3.1.3 实时数据处理

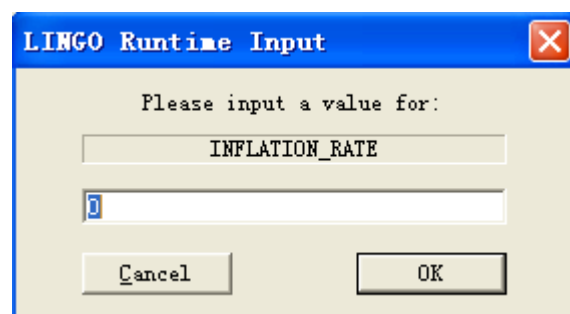
在某些情况，对于模型中的某些数据并不是定值。譬如模型中有一个通货膨胀率的参数，我们想在 2% 至 6% 范围内，对不同的值求解模型，来观察模型的结果对通货膨胀的依赖有多么敏感。我们把这种情况称为**实时数据处理** (what if analysis)。LINGO 有一个特征可方便地做到这件事。

在本该放数的地方输入一个问号 (?)。

#### 例 3.5

```
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：



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。

#### 3.1.4 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 3.6

```
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 3.7

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

#### 3.1.5 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 3.8

```
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。

### 3.2 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 3.9

```
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的为自己的模型加入原始数据和初始部分啦。



## § 4 LINGO 函数

有了前几节的基础知识，再加上本节的内容，你就能够借助于 LINGO 建立并求解复杂的优化模型了。

LINGO 有 9 种类型的函数：

1. 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. 2. 数学函数：三角函数和常规的数学函数
3. 3. 金融函数：LINGO 提供的两种金融函数
4. 4. 概率函数：LINGO 提供了大量概率相关的函数
5. 5. 变量界定函数：这类函数用来定义变量的取值范围
6. 6. 集操作函数：这类函数为对集的操作提供帮助
7. 7. 集循环函数：遍历集的元素，执行一定的操作的函数
8. 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. 9. 辅助函数：各种杂类函数

### 4.1 基本运算符

这些运算符是非常基本的，甚至可以不认为它们是一类函数。事实上，在 LINGO 中它们是非常重要的。

#### 4.1.1 算术运算符

算术运算符是针对数值进行操作的。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。

**例 4.1** 算术运算符示例。

$2 - 5 / 3$ ,  $(2 + 4) / 5$  等等。

#### 4.1.2 逻辑运算符

在 LINGO 中，逻辑运算符主要用于集循环函数的条件表达式中，来控制函数中哪些集成员被包含，哪些被排斥。在创建稀疏集时用在成员资格过滤器中。

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#

#### 例 4.2 逻辑运算符示例

2 #gt# 3 #and# 4 #gt# 2, 其结果为假 (0)。

#### 4.1.3 关系运算符

在 LINGO 中, 关系运算符主要是被用在模型中, 来指定一个表达式的左边是否等于、小于等于、或者大于等于右边, 形成模型的一个约束条件。关系运算符与逻辑运算符 #eq#、#le#、#ge# 截然不同, 前者是模型中该关系运算符所指定关系的为真描述, 而后者仅仅判断一个该关系是否被满足: 满足为真, 不满足为假。

LINGO 有三种关系运算符: “=”、“<=”和“>=”。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  $\varepsilon$  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 <= = >=

#### 4.2 数学函数

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

**例 4.3** 给定一个直角三角形, 求包含该三角形的最小正方形。

解: 如图所示。

$$CE = a \sin x, \quad AD = b \cos x, \quad DE = a \cos x + b \sin x,$$

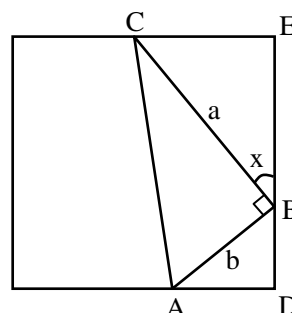
求最小的正方形就相当于求如下的最优化问题:

$$\min_{0 \leq x \leq \frac{\pi}{2}} \max\{CE, AD, DE\}$$

LINGO 代码如下:

model:

sets:



```

object/1..3/: f;
endsets
data:
  a, b = 3, 4; !两个直角边长, 修改很方便;
enddata
f(1) = a * @sin(x);
f(2) = b * @cos(x);
f(3) = a * @cos(x) + b * @sin(x);
min = @smax(f(1), f(2), f(3));
@bnd(0, x, 1.57);
end

```

在上面的代码中用到了函数@bnd, 详情请见 4.5 节。

### 4.3 金融函数

目前 LINGO 提供了两个金融函数。

#### 1. @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

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

净现值就是在一定时期内为了获得一定收益在该时期初所支付的实际费用。

**例 4.4 贷款买房问题** 贷款金额 50000 元, 贷款年利率 5.31%, 采取分期付款的方式 (每年年末还固定金额, 直至还清)。问拟贷款 10 年, 每年需偿还多少元?

LINGO 代码如下:

```
50000 = x * @fpa(.0531, 10);
```

答案是  $x=6573.069$  元。

#### 2. @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

$$(1+I)^{-n}。$$

细心的读者可以发现这两个函数间的关系:

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

### 4.4 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  $\chi^2$  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  $z$  服从标准正态分布。

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 4.5

```
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  $U(I+1)=\text{@rand}(U(I))$ 。注意如果 seed 不变, 那么产生的随机数也不变。

**例 4.6** 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

```

#### 4.5 变量界定函数

变量界定函数实现对变量取值范围的附加限制，共 4 种：

|               |                                  |
|---------------|----------------------------------|
| @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。

#### 4.6 集操作函数

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 4.7** 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 匹配的原始集成员的索引。如果找不到，则产生一个错误。

**例 4.8** 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 4.9**

```

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  $[1, \text{limit}]$  内。该函数相当于  $\text{index} \bmod \text{limit}$  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  $\text{set\_name}$  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。

## 4.7 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;  $\text{setname}$  是要遍历的集;  $\text{set\_index\_list}$  是集索引列表;  $\text{conditional\_qualifier}$  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  $\text{conditional\_qualifier}$  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。 $\text{expression\_list}$  是被应用到每个集成员的表达式列表, 当用的是@for 函数时,  $\text{expression\_list}$  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,  $\text{expression\_list}$  只能有一个表达式。如果省略  $\text{set\_index\_list}$ , 那么在  $\text{expression\_list}$  中引用的所有属性的类型都是  $\text{setname}$  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 4.10** 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 4.11** 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 4.12** 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 4.13 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

```

计算的部分结果为

```

Global optimal solution found at iteration:          0
Objective value:                                  22.00000

```

| Variable       | Value    | Reduced Cost |
|----------------|----------|--------------|
| REQUIRED( MON) | 20.00000 | 0.000000     |
| REQUIRED( TUE) | 16.00000 | 0.000000     |
| REQUIRED( WED) | 13.00000 | 0.000000     |
| REQUIRED( THU) | 16.00000 | 0.000000     |
| REQUIRED( FRI) | 19.00000 | 0.000000     |
| REQUIRED( SAT) | 14.00000 | 0.000000     |
| REQUIRED( SUN) | 12.00000 | 0.000000     |
| START( MON)    | 8.000000 | 0.000000     |
| START( TUE)    | 2.000000 | 0.000000     |
| START( WED)    | 0.000000 | 0.3333333    |
| START( THU)    | 6.000000 | 0.000000     |
| START( FRI)    | 3.000000 | 0.000000     |
| START( SAT)    | 3.000000 | 0.000000     |
| START( SUN)    | 0.000000 | 0.000000     |

从而解决方案是：每周最少需要 22 个职员，周一安排 8 人，周二安排 2 人，周三无需安排人，周四安排 6 人，周五和周六都安排 3 人，周日无需安排人。

#### 4.8 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 4.14** 以例 1.2 来讲解@file 函数的用法。

注意到在例 1.2 的编码中有两处涉及到数据。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 4.15** 借用例 4.12，说明@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 4.16**

```
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。

|    | A    | B                  | C    | D | E | F                | G  | H |
|----|------|--------------------|------|---|---|------------------|----|---|
| 1  |      |                    |      |   |   |                  |    |   |
| 2  |      | 产品                 | 机器   | 周 |   |                  |    |   |
| 3  |      | A                  | M    | 1 |   |                  |    |   |
| 4  |      | B                  | N    | 2 |   |                  |    |   |
| 5  |      |                    |      | 3 |   |                  |    |   |
| 6  |      |                    |      |   |   | 集ALLOWED的属性x和y的值 |    |   |
| 7  |      | 允许的组合 (ALLOWED集成员) |      |   |   | x                | y  |   |
| 8  |      | A                  | M    | 1 |   | 1                | 22 |   |
| 9  |      | A                  | N    | 2 |   | 2                | 10 |   |
| 10 |      | B                  | N    | 1 |   | 0                | 14 |   |
| 11 |      |                    |      |   |   |                  |    |   |
| 12 | 输出结果 |                    |      |   |   |                  |    |   |
| 13 |      | RATE               | 0.01 |   |   |                  |    |   |

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 4.17**

```
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）。

**4.9 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。

**例 4.18 求解最优化问题**

$$\begin{aligned}
& \min f(x) + g(y) \\
& s.t. \\
& f(x) = \begin{cases} 100 + 2x, & x > 0 \\ 2x, & x \leq 0 \end{cases} \\
& g(y) = \begin{cases} 60 + 3y, & y > 0 \\ 2y, & y \leq 0 \end{cases} \\
& x + y \geq 30 \\
& x, y \geq 0
\end{aligned}$$

其 LINGO 代码如下：

```

model:
    min=fx+fy;
    fx=@if(x #gt# 0, 100, 0)+2*x;
    fy=@if(y #gt# 0, 60, 0)+3*y;
    x+y>=30;
end

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。
    例 4.19 示例。
model:
    x=1;
    @warn(' x 是正数', x #gt# 0);
end

```

## § 5 LINGO WINDOWS 命令

### 5.1 文件菜单 (File Menu)

#### 1. 1. 新建 (New)

从文件菜单中选用“新建”命令、单击“新建”按钮或直接按 F2 键可以创建一个新的“Model”窗口。在这个新的“Model”窗口中能够输入所要求解的模型。

#### 2. 2. 打开 (Open)

从文件菜单中选用“打开”命令、单击“打开”按钮或直接按 F3 键可以打开一个已经存在的文本文件。这个文件可能是一个 Model 文件。

#### 3. 3. 保存 (Save)

从文件菜单中选用“保存”命令、单击“保存”按钮或直接按 F4 键用来保存当前活动窗口（最前台的窗口）中的模型结果、命令序列等保存为文件。

#### 4. 4. 另存为... (Save As... )

从文件菜单中选用“另存为...”命令或按 F5 键可以将当前活动窗口中的内容保存为文本文件，其文件名为你在“另存为...”对话框中输入的文件名。利用这种方法你可以将任何窗口的内容如模型、求解结果或命令保存为文件。

#### 5. 5. 关闭 (Close)

在文件菜单中选用“关闭” (Close) 命令或按 F6 键将关闭当前活动窗口。如果这个窗口是新建窗口或已经改变了当前文件的内容，LINGO 系统将会提示是否想要保存改变后的内容。

#### 6. 6. 打印 (Print)

在文件菜单中选用“打印” (Print) 命令、单击“打印”按钮或直接按 F7 键可以将当前活动窗口中的内容发送到打印机。

#### 7. 7. 打印设置 (Print Setup... )

在文件菜单中选用“打印设置...”命令或直接按 F8 键可以将文件输出到指定的打印机。

#### 8. 8. 打印预览(Print Preview)

在文件菜单中选用“打印预览...”命令或直接按 Shift+F8 键可以进行打印预览。

#### 9. 9. 输出到日志文件(Log Output...)

从文件菜单中选用“Log Output...”命令或按 F9 键打开一个对话框，用于生成一个日志文件，它存储接下来在“命令窗口”中输入的所有命令。

#### 10. 提交 LINGO 命令脚本文件(Take Commands...)

从文件菜单中选用“Take Commands...”命令或直接按 F11 键就可以将 LINGO 命令脚本(command script)文件提交给系统进程来运行。

#### 11. 引入 LINGO 文件(Import Lingo File...)

从文件菜单中选用“Import Lingo File...”命令或直接按 F12 键可以打开一个 LINGO 格式模型的文件，然后 LINGO 系统会尽可能把模型转化为 LINGO 语法允许的程序。

#### 12. 退出(Exit)

从文件菜单中选用“Exit”命令或直接按 F10 键可以退出 LINGO 系统。

### 5.2 编辑菜单(Edit Menu)

#### 1. 1. 恢复(Undo)

从编辑菜单中选用“恢复”(Undo)命令或按 Ctrl+Z 组合键，将撤销上次操作、恢复至其前的状态。

#### 2. 2. 剪切(Cut)

从编辑菜单中选用“剪切”(Cut)命令或按 Ctrl+X 组合键可以将当前选中的内容剪切至剪贴板中。

#### 3. 3. 复制(Copy)

从编辑菜单中选用“复制”(Copy)命令、单击“复制”按钮或按 Ctrl+C 组合键可以将当前选中的内容复制到剪贴板中。

#### 4. 4. 粘贴(Paste)

从编辑菜单中选用“粘贴”(Paste)命令、单击“粘贴”按钮或按 Ctrl+V 组合键可以将剪贴板中的当前内容复制到当前插入点的位置。

#### 5. 5. 粘贴特定..(Paste Special..)

与上面的命令不同，它可以用于剪贴板中的内容不是文本的情形。

#### 6. 6. 全选(Select All)

从编辑菜单中选用“Select All”命令或按 Ctrl+A 组合键可选定当前窗口中的所有内容。

#### 7. 7. 匹配小括号(Match Parenthesis)

从编辑菜单中选用“Match Parenthesis”命令、单击“Match Parenthesis”按钮或按 Ctrl+P 组合键可以为当前选中的开括号查找匹配的闭括号。

#### 8. 8. 粘贴函数(Paste Function)

从编辑菜单中选用“Paste Function”命令可以将 LINGO 的内部函数粘贴到当前插入点。

### 5.3 LINGO 菜单

#### 1. 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. 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. 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. 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。

下面我们看一个简单的具体例子。

**例 5.1** 某家具公司制造书桌、餐桌和椅子，所用的资源有三种：木料、木工和漆工。生产数据如下表所示：

|      | 每个书桌  | 每个餐桌   | 每个椅子   | 现有资源总数 |
|------|-------|--------|--------|--------|
| 木料   | 8 单位  | 6 单位   | 1 单位   | 48 单位  |
| 漆工   | 4 单位  | 2 单位   | 1.5 单位 | 20 单位  |
| 木工   | 2 单位  | 1.5 单位 | 0.5 单位 | 8 单位   |
| 成品单价 | 60 单位 | 30 单位  | 20 单位  |        |

若要求桌子的生产量不超过 5 件，如何安排三种产品的生产可使利润最大？

用 DESKS、TABLES 和 CHAIRS 分别表示三种产品的生产量，建立 LP 模型。

$\max = 60 \times \text{desks} + 30 \times \text{tables} + 20 \times \text{chairs};$

$8 \times \text{desks} + 6 \times \text{tables} + \text{chairs} \leq 48;$

$4 \times \text{desks} + 2 \times \text{tables} + 1.5 \times \text{chairs} \leq 20;$

$2 \times \text{desks} + 1.5 \times \text{tables} + 0.5 \times \text{chairs} \leq 8;$

$\text{tables} \leq 5;$

求解这个模型，并激活灵敏性分析。这时，查看报告窗口 (Reports Window)，可以看到如下结果。

Global optimal solution found at iteration: 3  
Objective value: 280.0000

| Variable | Value    | Reduced Cost |
|----------|----------|--------------|
| DESKS    | 2.000000 | 0.000000     |
| TABLES   | 0.000000 | 5.000000     |
| CHAIRS   | 8.000000 | 0.000000     |

| Row | Slack or Surplus | Dual Price |
|-----|------------------|------------|
| 1   | 280.0000         | 1.000000   |
| 2   | 24.00000         | 0.000000   |
| 3   | 0.000000         | 10.00000   |
| 4   | 0.000000         | 10.00000   |
| 5   | 5.000000         | 0.000000   |

“Global optimal solution found at iteration: 3”表示 3 次迭代后得到全局最优解。“Objective value:280.0000”表示最优目标值为 280。“Value”给出最优解中各变量的值：造 2 个书桌 (desks)，0 个餐桌 (tables)，8 个椅子 (chairs)。所以 desks、chairs 是基变量（非 0），tables 是非基变量（0）。

“Slack or Surplus”给出松弛变量的值：

第 1 行松弛变量 =280（模型第一行表示目标函数，所以第二行对应第一个约束）

第 2 行松弛变量 =24

第 3 行松弛变量 =0

第 4 行松弛变量 =0

第 5 行松弛变量 =5



“Reduced Cost” 列出最优单纯形表中判别数所在行的变量的系数，表示当变量有微小变动时，目标函数的变化率。其中基变量的 reduced cost 值应为 0，对于非基变量  $X_j$ ，相应的 reduced cost 值表示当某个变量  $X_j$  增加一个单位时目标函数减少的量（max 型问题）。本例中：变量 tables 对应的 reduced cost 值为 5，表示当非基变量 tables 的值从 0 变为 1 时（此时假定其他非基变量保持不变，但为了满足约束条件，基变量显然会发生变化），最优的目标函数值  $= 280 - 5 = 275$ 。

“DUAL PRICE”（对偶价格）表示当对应约束有微小变动时，目标函数的变化率。输出结果中对应于每一个约束有一个对偶价格。若其数值为  $p$ ，表示对应约束中不等式右端项若增加 1 个单位，目标函数将增加  $p$  个单位（max 型问题）。显然，如果在最优解处约束正好取等号（也就是“紧约束”，也称为有效约束或起作用约束），对偶价格值才可能不是 0。本例中：第 3、4 行是紧约束，对应的对偶价格值为 10，表示当紧约束

3)  $4 \text{ DESKS} + 2 \text{ TABLES} + 1.5 \text{ CHAIRS} \leq 20$

变为 3)  $4 \text{ DESKS} + 2 \text{ TABLES} + 1.5 \text{ CHAIRS} \leq 21$

时，目标函数值  $= 280 + 10 = 290$ 。对第 4 行也类似。

对于非紧约束（如本例中第 2、5 行是非紧约束），DUAL PRICE 的值为 0，表示对应约束中不等式右端项的微小扰动不影响目标函数。有时，通过分析 DUAL PRICE，也可对产生不可行问题的原因有所了解。

灵敏度分析的结果是

Ranges in which the basis is unchanged:

| Variable | Objective Coefficient Ranges |           |           |
|----------|------------------------------|-----------|-----------|
|          | Current                      | Allowable | Allowable |
|          | Coefficient                  | Increase  | Decrease  |
| DESKS    | 60.00000                     | 0.0       | 0.0       |
| TABLES   | 30.00000                     | 0.0       | 0.0       |
| CHAIRS   | 20.00000                     | 0.0       | 0.0       |

| Row | Righthand Side Ranges |           |           |
|-----|-----------------------|-----------|-----------|
|     | Current               | Allowable | Allowable |
|     | RHS                   | Increase  | Decrease  |
| 2   | 48.00000              | 0.0       | 0.0       |
| 3   | 20.00000              | 0.0       | 0.0       |
| 4   | 8.000000              | 0.0       | 0.0       |
| 5   | 5.000000              | 0.0       | 0.0       |

目标函数中 DESKS 变量原来的费用系数为 60，允许增加（Allowable Increase）=4、允许减少（Allowable Decrease）=2，说明当它在  $[60-4, 60+20] = [56, 80]$  范围变化时，最优基保持不变。对 TABLES、CHAIRS 变量，可以类似解释。由于此时约束没有变化（只是目标函数中某个费用系数发生变化），所以最优基保持不变的意思也就是最优解不变（当然，由于目标函数中费用系数发生了变化，所以最优值会变化）。

第 2 行约束中右端项（Right Hand Side，简称为 RHS）原来为 48，当它在  $[48-24, 48+\infty] = [24, \infty]$  范围变化时，最优基保持不变。第 3、4、5 行可以类似解释。不过由于此时约束发生变化，最优基即使不变，最优解、最优值也会发生变化。

灵敏性分析结果表示的是最优基保持不变的系数范围。由此，也可以进一步确定当目标函数的费用系数和约束右端项发生小的变化时，最优基和最优解、最优值如何变化。下面我们通过求解一个实际问题来进行说明。

**例 5.2** 一奶制品加工厂用牛奶生产  $A_1, A_2$  两种奶制品，1 桶牛奶可以在甲车间用 12 小时加工成 3 公斤  $A_1$ ，或者在乙车间用 8 小时加工成 4 公斤  $A_2$ 。根据市场需求，生产的  $A_1, A_2$  全部能售出，且每公斤  $A_1$  获利 24 元，每公斤  $A_2$  获利 16 元。现在加工厂每天能得到 50 桶牛奶

的供应，每天正式工人总的劳动时间 480 小时，并且甲车间每天至多能加工 100 公斤  $A_1$ ，乙车间的加工能力没有限制。试为该厂制订一个生产计划，使每天获利最大，并进一步讨论以下 3 个附加问题：

- 1) 若用 35 元可以买到 1 桶牛奶，应否作这项投资？若投资，每天最多购买多少桶牛奶？
- 2) 若可以聘用临时工人以增加劳动时间，付给临时工人的工资最多是每小时几元？
- 3) 由于市场需求变化，每公斤  $A_1$  的获利增加到 30 元，应否改变生产计划？

模型代码如下：

```
max=72*x1+64*x2;
x1+x2<=50;
12*x1+8*x2<=480;
3*x1<=100;
```

求解这个模型并做灵敏性分析，结果如下。

Global optimal solution found at iteration: 0  
Objective value: 3360.000

| Variable | Value            | Reduced Cost |
|----------|------------------|--------------|
| X1       | 20.00000         | 0.000000     |
| X2       | 30.00000         | 0.000000     |
| Row      | Slack or Surplus | Dual Price   |
| 1        | 3360.000         | 1.000000     |
| 2        | 0.000000         | 48.00000     |
| 3        | 0.000000         | 2.000000     |
| 4        | 40.00000         | 0.000000     |

Ranges in which the basis is unchanged:

| Objective Coefficient Ranges |                     |                    |                    |
|------------------------------|---------------------|--------------------|--------------------|
| Variable                     | Current Coefficient | Allowable Increase | Allowable Decrease |
| X1                           | 72.00000            | 24.00000           | 8.000000           |
| X2                           | 64.00000            | 8.000000           | 16.00000           |
| Righthand Side Ranges        |                     |                    |                    |
| Row                          | Current RHS         | Allowable Increase | Allowable Decrease |
| 2                            | 50.00000            | 10.00000           | 6.666667           |
| 3                            | 480.0000            | 53.33333           | 80.00000           |
| 4                            | 100.0000            | INFINITY           | 40.00000           |

结果告诉我们：这个线性规划的最优解为  $x_1=20$ ， $x_2=30$ ，最优值为  $z=3360$ ，即用 20 桶牛奶生产  $A_1$ ，30 桶牛奶生产  $A_2$ ，可获最大利润 3360 元。输出中除了告诉我们问题的最优解和最优值以外，还有许多对分析结果有用的信息，下面结合题目中提出的 3 个附加问题给予说明。3 个约束条件的右端不妨看作 3 种“资源”：原料、劳动时间、车间甲的加工能力。输出中 Slack or Surplus 给出这 3 种资源在最优解下是否有剩余：原料、劳动时间的剩余均为零，车间甲尚余 40（公斤）加工能力。

目标函数可以看作“效益”，成为紧约束的“资源”一旦增加，“效益”必然跟着增长。输出中 DUAL PRICES 给出这 3 种资源在最优解下“资源”增加 1 个单位时“效益”的增量：原料增加 1 个单位（1 桶牛奶）时利润增长 48（元），劳动时间增加 1 个单位（1 小时）时利润增长 2（元），而增加非紧约束车间甲的能力显然不会使利润增长。这里，“效益”的增量可以看作“资源”的潜在价值，经济学上称为影子价格，即 1 桶牛奶的影子价格为 48 元，1 小时劳动的影子价格为 2 元，车间甲的影子价格为零。读者可以用直接求解的

办法验证上面的结论，即将输入文件中原料约束 milk) 右端的 50 改为 51，看看得到的最优值（利润）是否恰好增长 48（元）。用影子价格的概念很容易回答附加问题 1)：用 35 元可以买到 1 桶牛奶，低于 1 桶牛奶的影子价格 48，当然应该作这项投资。回答附加问题 2)：聘用临时工人以增加劳动时间，付给的工资低于劳动时间的影子价格才可以增加利润，所以工资最多是每小时 2 元。

目标函数的系数发生变化时（假定约束条件不变），最优解和最优值会改变吗？这个问题不能简单地回答。上面输出给出了最优基不变条件下目标函数系数的允许变化范围： $x_1$  的系数为  $(72-8, 72+24) = (64, 96)$ ； $x_2$  的系数为  $(64-16, 64+8) = (48, 72)$ 。注意： $x_1$  系数的允许范围需要  $x_2$  系数 64 不变，反之亦然。由于目标函数的费用系数变化并不影响约束条件，因此此时最优基不变可以保证最优解也不变，但最优值变化。用这个结果很容易回答附加问题 3)：若每公斤  $A_1$  的获利增加到 30 元，则  $x_1$  系数变为  $30 \times 3 = 90$ ，在允许范围内，所以不应改变生产计划，但最优值变为  $90 \times 20 + 64 \times 30 = 3720$ 。

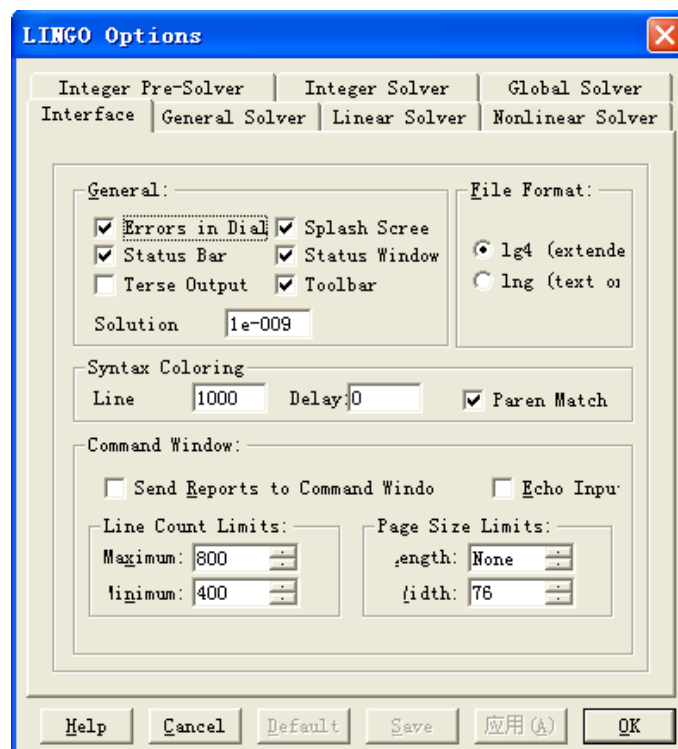
下面对“资源”的影子价格作进一步的分析。影子价格的作用（即在最优解下“资源”增加 1 个单位时“效益”的增量）是有限制的。每增加 1 桶牛奶利润增长 48 元（影子价格），但是，上 9

面输出的 CURRENT RHS 的 ALLOWABLE INCREASE 和 ALLOWABLE DECREASE 给出了影子价格有意义条件下约束右端的限制范围：milk) 原料最多增加 10（桶牛奶），time) 劳动时间最多增加 53（小时）。现在可以回答附加问题 1) 的第 2 问：虽然应该批准用 35 元买 1 桶牛奶的投资，但每天最多购买 10 桶牛奶。顺便地说，可以用低于每小时 2 元的工资聘用临时工人以增加劳动时间，但最多增加 53.3333 小时。

需要注意的是：灵敏性分析给出的只是最优基保持不变的充分条件，而不一定是必要条件。比如对于上面的问题，“原料最多增加 10（桶牛奶）”的含义只能是“原料增加 10（桶牛奶）”时最优基保持不变，所以影子价格有意义，即利润的增加大于牛奶的投资。反过来，原料增加超过 10（桶牛奶），影子价格是否一定没有意义？最优基是否一定改变？一般来说，这是不能从灵敏性分析报告中直接得到的。此时，应该重新用新数据求解规划模型，才能做出判断。所以，从正常理解的角度来看，我们上面回答“原料最多增加 10（桶牛奶）”并不是完全科学的。

### 5. 5. 模型通常形式... (Generate...)

从 LINGO 菜单中选用“Generate...”命令或直接按 Ctrl+G 组合键可以创建当前模型的代数形式、LINGO 模型或 MPS 格式文本。



## 6. 6. 选项 .. (Options...)

从 LINGO 菜单中选用 “Options...” 命令、单击 “Options...” 按钮或直接按 Ctrl+I 组合键可以改变一些影响 LINGO 模型求解时的参数。该命令将打开一个含有 7 个选项卡的窗口，你可以通过它修改 LINGO 系统的各种参数和选项。如上图。

修改完以后，你如果单击 “Apply (应用)” 按钮，则新的设置马上生效；如果单击 “OK (确定)” 按钮，则新的设置马上生效，并且同时关闭该窗口。如果单击 “Save (保存)” 按钮，则将当前设置变为默认设置，下次启动 LINGO 时这些设置仍然有效。单击 “Default (缺省值)” 按钮，则恢复 LINGO 系统定义的原始默认设置 (缺省设置)。

### (1) Interface (界面) 选项卡

| 选项组                       | 选项                           | 含义   |
|---------------------------|------------------------------|--|
| General<br>(一般选项)         | Errors In Dialogs (错误对话框)    | 如果选择该选项，求解程序遇到错误时将打开一个对话框显示错误，你关闭该对话框后程序才会继续执行；否则，错误信息将在报告窗口显示，程序仍会继续执行                                  |
|                           | Splash Screen (弹出屏幕)         | 如果选择该选项，则 LINGO 每次启动时会在屏幕上弹出一个对话框，显示 LINGO 的版本和版权信息；否则不弹出  |
|                           | Status Bar (状态栏)             | 如果选择该选项，则 LINGO 系统在主窗口最下面一行显示状态栏；否则不显示   |
|                           | Status Window (状态窗口)         | 如果选择该选项，则 LINGO 系统每次运行 LINGO Solve 命令时会在屏幕上弹出状态窗口；否则不弹出  |
|                           | Terse Output (简洁输出)          | 如果选择该选项，则 LINGO 系统对求解结果报告等将以简洁形式输出；否则以详细形式输出   |
|                           | Toolbar (工具栏)                | 如果选择该选项，则显示工具栏；否则不显示   |
|                           | Solution Cutoff (解的截断)       | 小于等于这个值的解将报告为 “0” (缺省值是 $10^{-9}$ )  |
| File Format<br>(文件格式)     | lg4 (extended) (lg4, 扩展格式)   | 模型文件的缺省保存格式是 lg4 格式 (这是一种二进制文件，只有 LINGO 能读出)   |
|                           | lng (text only) (lng, 纯文本格式) | 模型文件的缺省保存格式是 lng 格式 (纯文本)  |
| Syntax Coloring<br>(语法配色) | Line limit (行数限制)            | 语法配色的行数限制 (缺省为 1000)。LINGO 模型窗口中将 LINGO 关键此显示为蓝色，注释为绿色，其他为黑色，超过该行数限制后则不再区分颜色。特别地，设置行数限制为 0 时，整个文件不再区分颜色。 |
|                           | Delay (延迟)                   | 设置语法配色的延迟时间 (秒，缺省为 0，从最后一次击键算起)。   |
|                           | Paren Match (括号匹配)           | 如果选择该选项，则模型中当前光标所在处的括号及其相匹配的括号将以红色显示；否则不使用该功能  |

|                          |   |  |
|--------------------------|---|--|
| Command Window<br>(命令窗口) | Send Reports to Command Window<br>(报告发送到命令窗口) | 如果选择该选项, 则输出信息会发送到命令窗口; 否则不使用该功能   |
|                          | Echo Input<br>(输入信息反馈)                        | 如果选择该选项, 则用 File Take Command 命令执行命令脚本文件时, 处理信息会发送到命令窗口; 否则不使用该功能  |
|                          | Line Count Limits<br>(行数限制)                   | 命令窗口能显示的行数的最大值为 Maximum (缺省为 800); 如果要显示的内容超过这个值, 每次从命令窗口滚动删除的最小行数为 Minimum (缺省为 400)                      |
|                          | Page Size Limit<br>(页面大小限制)                   | 命令窗口每次显示的行数的最大值为 Length (缺省为没有限制), 显示这么多行后会暂停, 等待用户响应; 每行最大字符数为 Width (缺省为 74, 可以设定为 64-200 之间), 多余的字符将被截断 |

## (2) General Solver (通用求解器) 选项卡

| 选项组   | 选项                     | 含义  |
|---|------------------------|---|
| Generator Memory Limit (MB)<br>矩阵生成器的内存限制 (兆) |                        | 缺省值为 32M, 矩阵生成器使用的内存超过该限制, LINGO 将报告 "The model generator ran out of memory"  |
| Runtime Limits<br>运行限制                        | Iterations<br>迭代次数     | 求解一个模型时, 允许的最大迭代次数 (缺省值为无限)   |
|   | Time (sec)<br>运行时间 (秒) | 求解一个模型时, 允许的最大运行时间 (缺省值为无限)   |
| Dual Computations<br>(对偶计算)                   |                        | 求解时控制对偶计算的级别, 有三种可能的设置: <ul style="list-style-type: none"> <li>• None: 不计算任何对偶信息;</li> <li>• Prices: 计算对偶价格 (缺省设置);</li> <li>• Prices and Ranges: 计算对偶价格并分析敏感性。</li> </ul>  |
| Model Regeneration<br>(模型的重新生成)               |                        | 控制重新生成模型的频率, 有三种可能的设置: <ul style="list-style-type: none"> <li>• Only when text changes: 只有当模型的文本修改后才再生成模型;</li> <li>• When text changes or with external references: 当模型的文本修改或模型含有外部引用时 (缺省设置);</li> <li>• Always: 每当有需要时。</li> </ul> |

|  |                    |  |
|--|--------------------|--|
| Linearization<br>(线性化)   | Degree<br>(线性化程度)  | <p>决定求解模型时线性化的程度，有四种可能的设置：</p> <p>Solver Decides: 若变量数小于等于 12 个，则尽可能全部线性化；否则不做任何线性化（缺省设置）</p> <ul style="list-style-type: none"> <li>• None: 不做任何线性化</li> <li>• Low: 对函数@ABS(), @MAX(), @MIN(), @SMAX(), @SMIN(), 以及二进制变量与连续变量的乘积项做线性化</li> <li>• High: 同上，此外对逻辑运算符#LE#, #EQ#, #GE#, #NE#做线性化</li> </ul> |
|  | Big M (线性化的大 M 系数) | 设置线性化的大 M 系数（缺省值为 $10^6$ ）。  |
|  | Delta (线性化的误差限)    | 设置线性化的误差限（缺省值为 $10^{-6}$ ）。  |
| Allow Unrestricted Use of Primitive Set Member Names<br>(允许无限制地使用基本集合的成员名) |                    | 选择该选项可以保持与 LINGO4.0 以前的版本兼容：即允许使用基本集合的成员名称直接作为该成员在该集合的索引值（LINGO4.0 以后的版本要求使用@INDEX 函数）。  |
| Check for Duplicate Names in Data and Model (检查数据和模型中的名称是否重复使用)            |                    | 选择该选项, LINGO 将检查数据和模型中的名称是否重复使用，如基本集合的成员名是否与决策变量名重复。   |
| Use R/C format names for MPS I/O (在 MPS 文件格式的输入输出中使用 R/C 格式的名称)            |                    | 在 MPS 文件格式的输入输出中，将变量和行名转换为 R/C 格式  |

### (3) Linear Solver (线性求解器) 选项卡

| 选项组                             | 选项         | 含义   |
|---------------------------------|------------|--|
| Method<br>求解方法                  |            | <p>求解时的算法，有四种可能的设置：</p> <ul style="list-style-type: none"> <li>• Solver Decides: LINGO 自动选择算法（缺省设置）</li> <li>• Primal Simplex: 原始单纯形法</li> <li>• Dual Simplex: 对偶单纯形法</li> <li>• Barrier: 障碍法（即内点法）</li> </ul> |
| Initial Linear Feasibility Tol. | 初始线性可行性误差限 | 控制线性模型中约束满足的初始误差限（缺省值为 $3 \times 10^{-6}$ ）  |
| Final Linear Feasibility Tol.   | 最后线性可行性误差限 | 控制线性模型中约束满足的最后误差限（缺省值为 $10^{-7}$ ）   |
| Model Reduction<br>模型降维         |            | <p>控制是否检查模型中的无关变量，从而降低模型的规模：</p> <ul style="list-style-type: none"> <li>• Off: 不检查</li> <li>• On: 检查</li> </ul>  |

|  |                         |  |
|--|-------------------------|--|
|  |                         | • Solver Decides: LINGO 自动决定 (缺省设置)  |
| Pricing Strategies<br>价格策略 (决定出基变量的策略) | Primal Solver<br>原始单纯形法 | 有三种可能的设置:<br>• Solver Decides: LINGO 自动决定 (缺省设置)<br>• Partial: LINGO 对一部分可能的出基变量进行尝试<br>• Devex: 用 Steepest-Edge (最陡边) 近似算法对所有可能的变量进行尝试, 找到使目标值下降最多的出基变量 |
|  | Dual Solver<br>对偶单纯形法   | 有三种可能的设置:<br>• Solver Decides: LINGO 自动决定 (缺省设置)<br>• Dantzig: 按最大下降比例法确定出基变量<br>• Steepest-Edge: 最陡边策略, 对所有可能的变量进行尝试, 找到使目标值下降最多的出基变量                   |
| Matrix Decomposition<br>矩阵分解           |                         | 选择该选项, LINGO 将尝试将一个大模型分解为几个小模型求解; 否则不尝试  |
| Scale Model<br>模型尺度的改变                 |                         | 选择该选项, LINGO 检查模型中的数据是否平衡 (数量级是否相差太大) 并尝试改变尺度使模型平衡; 否则不尝试  |

#### (4) Nonlinear Solver (非线性求解器) 选项卡

| 选项组   | 选项                              | 含义   |
|---|---------------------------------|--|
| Initial Feasibility Tol.<br>初始非线性可行性误差限         | Nonlinear                       | 控制模型中约束满足的初始误差限 (缺省值为 $10^{-3}$ )                                  |
| Final Nonlinear Feasibility Tol.<br>最后非线性可行性误差限 |                                 | 控制模型中约束满足的最后误差限 (缺省值为 $10^{-6}$ )                                  |
| Nonlinear Optimality Tol.<br>非线性规划的最优性误差限       |                                 | 当目标函数在当前解的梯度小于等于这个值以后, 停止迭代 (缺省值为 $2 \times 10^{-7}$ )             |
| Slow Progress Iteration Limit<br>缓慢改进的迭代次数的上限   |                                 | 当目标函数在连续这么多次迭代没有显著改进以后, 停止迭代 (缺省值为 5)                              |
| Derivatives<br>导数                               | Numerical<br>数值法                | 用有限差分法计算数值导数 (缺省值)   |
|   | Analytical<br>解析法               | 用解析法计算导数 (仅对只含有算术运算符的函数使用)   |
| Strategies<br>策略                                | Crash Initial Solution<br>生成初始解 | 选择该选项, LINGO 将用启发式方法生成初始解; 否则不生成 (缺省值)                             |
|   | Quadratic Recognition<br>识别二次规划 | 选择该选项, LINGO 将判别模型是否为二次规划, 若是则采用二次规划算法 (包含在线性规划的内点法中); 否则不判别 (缺省值) |



|  |                                       |   |
|--|---------------------------------------|---|
|  | Selective Constraint Eval<br>有选择地检查约束 | 选择该选项，LINGO 在每次迭代时只检查必须检查的约束（如果有些约束函数在某些区域没有定义，这样做会出现错误）；否则，检查所有约束（缺省值） |
|  | SLP Directions<br>SLP 方向              | 选择该选项，LINGO 在每次迭代时用 SLP (Successive LP, 逐次线性规划) 方法寻找搜索方向（缺省值）           |
|  | Steepest Edge<br>最陡边策略                | 选择该选项，LINGO 在每次迭代时将对所有可能的变量进行尝试，找到使目标值下降最多的变量进行迭代；缺省值为不使用最陡边策略          |

#### (5) Integer Pre-Solver (整数预处理求解器) 选项卡

| 选项组                         | 选项                      | 含义  |
|-----------------------------|-------------------------|---|
| Heuristics<br>启发式方法         | Level                   | 控制采用启发式搜索的次数（缺省值为 3，可能的值为 0-100）。启发式方法的目的是从分枝节点的连续解出发，搜索一个好的整数解。  |
|                             | Min Seconds             | 每个分枝节点使用启发式搜索的最小时间（秒）   |
| Probing Level<br>探测水平（级别）   |                         | 控制采用探测（Probing）技术的级别（探测能够用于混合整数线性规划模型，收紧变量的上下界和约束的右端项的值）。可能的取值为： <ul style="list-style-type: none"> <li>• Solver Decides: LINGO 自动决定（缺省设置）</li> <li>• 1-7: 探测级别逐步升高。</li> </ul>           |
| Constraint Cuts<br>约束的割（平面） | Application<br>应用节点     | 控制在分枝定界树中，哪些节点需要增加割（平面），可能的取值为： <ul style="list-style-type: none"> <li>• Root Only: 仅根节点增加割（平面）</li> <li>• All Nodes: 所有节点均增加割（平面）</li> <li>• Solver Decides: LINGO 自动决定（缺省设置）</li> </ul> |
|                             | Relative Limit<br>相对上限  | 控制生成的割（平面）的个数相对于原问题的约束个数的上限（比值），缺省值为 0.75   |
|                             | Max Passes<br>最大迭代检查的次数 | 为了寻找合适的割，最大迭代检查的次数。有两个参数： <ul style="list-style-type: none"> <li>• Root: 对根节点的次数（缺省值为 200）</li> <li>• Tree: 对其他节点的次数（缺省值为 2）</li> </ul>   |
|                             | Types<br>类型             | 控制生成的割（平面）的策略，共有 12 种策略可供选择。（如想了解细节，请参阅整数规划方面的专著）   |

#### (6) Integer Solver (整数求解器) 选项卡

整数预处理程序只用于整数线性规划模型（ILP 模型），对连续规划和非线性模型无效。

| 选项组                  | 选项                                  | 含义   |
|----------------------|-------------------------------------|--|
| Branching<br>分枝      | Direction                           | 控制分枝策略中优先对变量取整的方向，有三种选择： <ul style="list-style-type: none"> <li>• Both: LINGO 自动决定（缺省设置）</li> <li>• Up: 向上取整优先</li> <li>• Down: 向下取整优先</li> </ul>  |
|                      | Priority                            | 控制分枝策略中优先对哪些变量进行分枝，有两种选择： <ul style="list-style-type: none"> <li>• LINGO Decides: LINGO 自动决定（缺省设置）</li> <li>• Binary: 二进制（0-1）变量优先</li> </ul>  |
| Integrality<br>整性    | Absolute<br>绝对误差限                   | 当变量与整数的绝对误差小于这个值时，该变量被认为是整数。缺省值为 $10^{-6}$   |
|                      | Relative<br>相对误差限                   | 当变量与整数的相对误差小于这个值时，该变量被认为是整数。缺省值为 $8 \times 10^{-6}$  |
| LP Solver<br>LP 求解程序 | Warm Start<br>热启动                   | 当以前面的求解结果为基础，热启动求解程序时采用的算法，有四种可能的设置： <ul style="list-style-type: none"> <li>• LINGO Decides: LINGO 自动选择算法（缺省设置）</li> <li>• Primal Simplex: 原始单纯形法</li> <li>• Dual Simplex: 对偶单纯形法</li> <li>• Barrier: 障碍法（即内点法）</li> </ul> |
|                      | Cold Start<br>冷启动                   | 当不以前面的求解结果为基础，冷启动求解程序时采用的算法，有四种可能的设置：（同上，略）  |
| Optimality<br>最优性    | Absolute<br>目标函数的绝对误差限              | 当当前目标函数值与最优值的绝对误差小于这个值时，当前解被认为是最优解（也就是说：只需要搜索比当前解至少改进这么多个单位的解）。缺省值为 $8 \times 10^{-8}$   |
|                      | Relative<br>目标函数的相对误差限              | 当当前目标函数值与最优值的相对误差小于这个值时，当前解被认为是最优解（也就是说：只需要搜索比当前解至少改进这么多百分比的解）。缺省值为 $5 \times 10^{-8}$   |
|                      | Time To Relative<br>开始采用相对误差限的时间（秒） | 在程序开始运行后这么多秒内，不采用相对误差限策略；此后才使用相对误差限策略。缺省值为 100 秒。  |
| Tolerances<br>误差限    | Hurdle<br>篱笆值                       | 同上一章 LINDO 部分的介绍   |
|                      | Node Selection<br>节点选择              | 控制如何选择节点的分枝求解，有以下选项： <ul style="list-style-type: none"> <li>• LINGO Decides: LINGO 自动选择（缺省设置）</li> <li>• Depth First: 按深度优先</li> <li>• Worst Bound: 选择具有最坏界的节点</li> <li>• Best Bound: 选择具有最好的界的节点</li> </ul>               |

|  |                         |   |
|--|-------------------------|---|
|  | Strong Branch<br>强分枝的层数 | 控制采用强分枝的层数。也就是说，对前这么多层的分枝，采用强分枝策略。所谓强分枝，就是在一个节点对多个变量分别尝试进行预分枝，找出其中最好的解（变量）进行实际分枝。 |
|--|-------------------------|---|

(7) Global Solver（全局最优求解器）选项卡

| 选项组                       | 选项                              | 含义   |
|---------------------------|---------------------------------|--|
| Global Solver<br>全局最优求解程序 | Use Global Solver<br>使用全局最优求解程序 | 选择该选项, LINGO 将用全局最优求解程序求解模型，尽可能得到全局最优解（求解花费的时间可能很长）; 否则不使用全局最优求解程序，通常只得到局部最优解  |
|                           | Variable Upper Bound<br>变量上界    | 有两个域可以控制变量上界（按绝对值）：<br>1、 Value：设定变量的上界，缺省值为 $10^{10}$ ；<br>2、 Application 列表框设置这个界的三种应用范围：<br>• None：所有变量都不使用这个上界；<br>• All：所有变量都使用这个上界；<br>• Selected：先找到第 1 个局部最优解，然后对满足这个上界的变量使用这个上界（缺省设置） |
|                           | Tolerances<br>误差限               | 有两个域可以控制变量上界（按绝对值）：<br>1、 Optimality：只搜索比当前解至少改进这么多个单位的解（缺省值为 $10^{-6}$ ）；<br>2、 Delta：全局最优求解程序在凸化过程中增加的约束的误差限（缺省值为 $10^{-7}$ ）。   |

|                               |                  |   |
|-------------------------------|------------------|---|
|                               | Strategies<br>策略 | <p>可以控制全局最优求解程序的三类策略：</p> <p>1、Branching：第 1 次对变量分枝时使用的分枝策略：</p> <ul style="list-style-type: none"> <li>• Absolute Width（绝对宽度）</li> <li>• Local Width（局部宽度）</li> <li>• Global Width（全局宽度）</li> <li>• Global Distance（全局距离）</li> <li>• Abs (Absolute) Violation（绝对冲突）</li> <li>• Rel (Relative) Violation（相对冲突，缺省设置）</li> </ul> <p>2、Box Selection：选择活跃分枝节点的方法：</p> <ul style="list-style-type: none"> <li>• Depth First（深度优先）</li> <li>• Worst Bound（具有最坏界的分枝优先，缺省设置）</li> </ul> <p>3、Reformulation：模型重整的级别：</p> <ul style="list-style-type: none"> <li>• None（不进行重整）</li> <li>• Low（低）</li> <li>• Medium（中）</li> <li>• High（高，缺省设置）</li> </ul> |
| Multistart Solver<br>多初始点求解程序 | Attempts<br>尝试次数 | <p>尝试多少个初始点求解，有以下几种可能的设置：</p> <ul style="list-style-type: none"> <li>• Solver Decides：由 LINGO 决定（缺省设置，对小规模 NLP 问题为 5 次，对大规模问题不使用多点求解）</li> <li>• Off：不使用多点求解</li> <li>• N (&gt;1 的正整数)：N 点求解</li> <li>• Barrier：障碍法（即内点法）</li> </ul>  |

#### 5.4 窗口菜单 (Windows Menu)

##### 1. 1. 命令行窗口 (Open Command Window)

从窗口菜单中选用“Open Command Window”命令或直接按 Ctrl+1 可以打开 LINGO 的命令行窗口。在命令行窗口中可以获得命令行界面，在“:”提示符后可以输入 LINGO 的命令行命令。

##### 2. 2. 状态窗口 (Status Window)

从窗口菜单中选用“Status Window”命令或直接按 Ctrl+2 可以打开 LINGO 的求解状态窗口。

如果在编译期间没有表达错误，那么 LINGO 将调用适当的求解器来求解模型。当求解器开始运行时，它就会显示如下的求解器状态窗口 (LINGO Solver Status)。



求解器状态窗口  
中断求解器按钮  
大多数情况下  
的解是无  
少需要中  
解、可能

教程  
提供了一个中  
求解。在绝  
模型，返回  
度很快，很  
本就不最优

在中断求解器按钮的右边的是关闭按钮 (Close)。点击它可以关闭求解器状态窗口，不过可在任何时间通过选择 Windows|Status Window 再重新打开。

在中断求解器按钮的右边的是标记为更新时间间隔 (Update Interval) 的域。LINGO 将根据该域指示的时间 (以秒为单位) 为周期更新求解器状态窗口。可以随意设置该域，不过若设置为 0 将导致更长的求解时间——LINGO 花费在更新的时间会超过求解模型的时间。

变量框 (Variables)

Total 显示当前模型的全部变量数，Nonlinear 显示其中的非线性变量数，Integers 显示其中的整数变量数。非线性变量是指它至少处于某一个约束中的非线性关系中。例如，对约束

$$X+Y=100;$$

X 和 Y 都是线性变量。对约束

$$X*Y=100;$$

X 和 Y 的关系是二次的，所以 X 和 Y 都是非线性变量。对约束

$$X*X+Y=100;$$

X 是二次方是非线性的，Y 虽与 X 构成二次关系，但与 X\*X 这个整体是一次的，因此 Y 是线性变量。被计数变量不包括 LINGO 确定为定值的变量。例如：

$$X=1;$$

$$X+Y=3;$$

这里 X 是 1，由此可得 Y 是 2，所以 X 和 Y 都是定值，模型中的 X 和 Y 都用 1 和 2 代换掉。

约束 (Constraints) 框

Total 显示当前模型扩展后的全部约束数，Nonlinear 显示其中的非线性约束数。非线性约束是该约束中至少有一个非线性变量。如果一个约束中的所有变量都是定值，那么该约束就被剔除出模型 (该约束为真)，不计入约束总数中。

非零 (Nonzeroes) 框

Total 显示当前模型中全部非零系数的数目，Nonlinear 显示其中的非线性变量系数的数目。

内存使用 (Generator Memory Used, 单位: K) 框

显示当前模型在内存中使用的内存量。可以通过使用 LINGO|Options 命令修改模型的最大内存使用量。

已运行时间 (Elapsed Runtime) 框

显示求解模型到目前所用的时间，它可能受到系统中别的应用程序的影响。

求解器状态 (Solver Status) 框

显示当前模型求解器的运行状态。域的含义如下。

| 域名 | 含义 | 可能的显示 |
|----|----|-------|
|----|----|-------|

|               |                         |   |
|---------------|-------------------------|---|
| Model Class   | 当前模型的类型（请参阅本书第 1 章）     | LP, QP, ILP, IQP, PILP, PIQP, NLP, INLP, PINLP（以 I 开头表示 IP, 以 PI 开头表示 PIP）  |
| State         | 当前解的状态                  | "Global Optimum", "Local Optimum", "Feasible", "Infeasible"（不可行）, "Unbounded"（无界）, "Interrupted"（中断）, "Undetermined"（未确定） |
| Objective     | 当前解的目标函数值               | 实数  |
| Infeasibility | 当前约束不满足的总量（不是不满足的约束的个数） | 实数（即使该值=0, 当前解也可能不可行, 因为这个量中没有考虑用上下界形式给出的约束）  |
| Iterations    | 目前为止的迭代次数               | 非负整数  |

#### 扩展求解器状态 (Extended Solver Status) 框

显示 LINGO 中几个特殊求解器的运行状态。包括分枝定界求解器 (Branch-and-Bound Solver)、全局求解器 (Global Solver) 和多初始点求解器 (Multistart Solver)。该框中的域仅当这些求解器运行时才会更新。域的含义如下。

| 域名          | 含义   | 可能的显示  |
|-------------|--|--|
| Solver Type | 使用的特殊求解程序  | B-and-B（分枝定界法）<br>Global（全局最优求解）<br>Multistart（用多个初始点求解） |
| Best Obj    | 目前为止找到的可行解的最佳目标函数值   | 实数   |
| Obj Bound   | 目标函数值的界  | 实数   |
| Steps       | 特殊求解程序当前运行步数：<br>分枝数（对 B-and-B 程序）；<br>子问题数（对 Global 程序）；<br>初始点数（对 Multistart 程序） | 非负整数   |
| Active      | 有效步数   | 非负整数   |

其余几个命令都是对窗口的排列，这里不作介绍，试一试便知。

### 5.5 帮助菜单 (Help Menu)

#### 1. 1. 帮助主题 (Help Menu)

从帮助菜单中选用“Help Menu”可以打开 LINGO 的帮助文件。

#### 2. 2. 关于 LINGO (About Lingo)

关于当前 LINGO 的版本信息等。

## § 6 LINGO 的命令行命令

以下将按类型列出在 LINGO 命令行窗口中使用的命令，每条命令后都附有简要的描述说明。

在平台中，从的窗口菜单中选用“Command Window”命令或直接按 Ctrl+1 可以打开 LINGO 的命令行窗口，便可以在命令提示符“:”后输入以下命令。

如果需要以下命令的详细描述说明，可以查阅 LINGO 的帮助。

**1. 1. LINGO 信息**

Cat        显示所有命令类型  
 Com       按类型显示所用 LINGO 命令  
 Help      显示所需命令的简要帮助信息  
 Mem      显示内存变量的信息

**2. 2. 输入(Input)**

model     以命令行方式输入一个模型  
 take      执行一个文件的命令正本或从磁盘中读取某个模型文件

**3. 3. 显示(Display)**

look      显示当前模型的内容  
 genl      产生 LINGO 兼容的模型  
 gen       生成并显示整个模型  
 hide      为模型设置密码保护  
 pause     暂停屏幕输出直至再次使用此命令

**4. 4. 文件输出(File Output)**

div       将模型结果输出到文件  
 svrt      将模型结果输出到屏幕  
 save      将当前模型保存到文件  
 smps      将当前模型保存为 MPS 文件

**5. 5. 求解模型(Solution)**

go        求解当前模型  
 solu      显示当前模型的求解结果

**6. 6. 编辑模型(Problem Editing)**

del       从当前模型中删除指定的某一行或某两行之间(包括这两行)的所有行  
 ext       在当前模型中添加几行  
 alt       用新字符串替换掉某一行中、或某两行之间的所有行中的旧字符串

**7. 7. 退出系统(Quit)**

quit      退出 LINGO 系统

**8. 8. 系统参数(System Parameters)**

page      以“行”为单位设置每页长度  
 ter       以简略方式输出结果  
 ver       以详细方式输出结果  
 wid       以“字符”为单位设置显示和输出宽度  
 set       重新设置默认参数  
 freeze    保存当前参数设置, 以备下一次重新启动 LINGO 系统时还是这样的设置  
 time      显示本次系统的运行时间

这里详细说明 SET 指令。凡是用户能够控制的 LINGO 系统参数, SET 命令都能够对它进行设置。SET 命令的使用格式为:

SET parameter\_name | parameter\_index [ parameter\_value ],

其中 parameter\_name 是参数名, parameter\_index 是参数索引(编号), parameter\_value 是参数值。当不写出参数值时, 则 SET 命令的功能是显示该参数当前的值。此外, “setdefault”命令用于将所有参数恢复为系统的默认值(缺省值)。这些设置如果不用“freeze”命令保存到配置文件 lingo.cnf 中, 则退出 LINGO 系统后这些设置就无效了。

| 索引 | 参数名    | 缺省值    | 简要说明       |
|----|--------|--------|------------|
| 1  | ILFTOL | 0.3e-5 | 初始线性可行误差限  |
| 2  | FLFTOL | 0.1e-6 | 最终线性可行误差限  |
| 3  | INFTOL | 0.1e-2 | 初始非线性可行误差限 |
| 4  | FNFTOL | 0.1e-5 | 最终非线性可行误差限 |



|    |        |         |  |
|----|--------|---------|--|
| 5  | RELINT | 0.8e-5  | 相对整性误差限  |
| 6  | NOPTOL | 0.2e-6  | 非线性规划 (NLP) 的最优性误差限  |
| 7  | ITRSLW | 5       | 缓慢改进的迭代次数的上限   |
| 8  | DERCMP | 0       | 导数 (0:数值导数, 1:解析导数)  |
| 9  | ITRLTM | 0       | 迭代次数上限 (0:无限制)   |
| 10 | TIMLIM | 0       | 求解时间的上限 (秒) (0:无限制)  |
| 11 | OBJCTS | 1       | 是否采用目标割平面法 (1:是, 0:否)  |
| 12 | MXMEMB | 32      | 模型生成器的内存上限 (兆字节) (对某些机器, 可能无意义)                              |
| 13 | CUTAPP | 2       | 割平面法的应用范围 (0:根节点, 1:所有节点, 2:LINGO 自动决定)                      |
| 14 | ABSINT | .000001 | 整性绝对误差限  |
| 15 | HEURIS | 3       | 整数规划 (IP) 启发式求解次数 (0:无, 可设定为 0~100)                          |
| 16 | HURDLE | none    | 整数规划 (IP) 的“篱笆”值 (none:无, 可设定为任意实数值)                         |
| 17 | IPTOLA | .8e-7   | 整数规划 (IP) 的绝对最优性误差限  |
| 18 | IPTOLR | .5e-7   | 整数规划 (IP) 的相对最优性误差限  |
| 19 | TIM2RL | 100     | 采用 IPTOLR 作为判断标准之前, 程序必须求解的时间 (秒)                            |
| 20 | NODESL | 0       | 分枝节点的选择策略 (0: LINGO 自动选择; 1: 深度优先; 2: 最坏界的节点优先; 3: 最好界的节点优先) |
| 21 | LENPAG | 0       | 终端的页长限制 (0:没有限制; 可设定任意非负整数)                                  |
| 22 | LINLEN | 76      | 终端的行宽限制 (0:没有限制; 可设定为 64-200)                                |
| 23 | TERSEO | 0       | 输出级别 (0:详细型, 1:简洁型)  |
| 24 | STAWIN | 1       | 是否显示状态窗口 (1:是, 0:否, Windows 系统才能使用)                          |
| 25 | SPLASH | 1       | 弹出版本和版权信息 (1:是, 0:否, Windows 系统才能使用)                         |
| 26 | OROUTE | 0       | 将输出定向到命令窗口 (1:是, 0:否, Windows 系统才能使用)                        |
| 27 | WNLINE | 800     | 命令窗口的最大显示行数 (Windows 系统才能使用)                                 |
| 28 | WNTRIM | 400     | 每次从命令窗口滚动删除的最小行数 (Windows 系统才能使用)                            |
| 29 | STABAR | 1       | 显示状态栏 (1:是, 0:否, Windows 系统才能使用)                             |
| 30 | FILFMT | 1       | 文件格式 (0:lng 格式, 1:lg4 格式, Windows 系统才能使用)                    |
| 31 | TOOLBR | 1       | 显示工具栏 (1:是, 0:否, Windows 系统才能使用)                             |
| 32 | CHKDUP | 0       | 检查数据与模型中变量是否重名 (1:是, 0:否)                                    |
| 33 | ECHOIN | 0       | 脚本命令反馈到命令窗口 (1:是, 0:否)                                       |
| 34 | ERRDLG | 1       | 错误信息以对话框显示 (1:是, 0:否, Windows 系统才能使用)                        |

|    |        |       |   |
|----|--------|-------|---|
| 35 | USEPNM | 0     | 允许无限制地使用基本集合的成员名(1:是, 0:否)  |
| 36 | NSTEEP | 0     | 在非线性求解程序中使用最陡边策略选择变量(1:是, 0:否)  |
| 37 | NCRASH | 0     | 在非线性求解程序中使用启发式方法生成初始解(1:是, 0:否)   |
| 38 | NSLPDR | 1     | 在非线性求解程序中用 SLP 法寻找搜索方向 (1:是, 0:否)   |
| 39 | SELCON | 0     | 在非线性求解程序中有选择地检查约束(1:是, 0:否)   |
| 40 | PRBLVL | 0     | 对混合整数线性规划 (MILP) 模型, 采用探测 (Probing) 技术的级别(0:LINGO 自动决定; 1:无; 2-7: 探测级别逐步升高) |
| 41 | SOLVE  | 0     | 线性求解程序(0: LINGO 自动选择, 1: 原始单纯形法, 2: 对偶单纯形法, 3: 障碍法 (即内点法))                  |
| 42 | REDUCE | 2     | 模型降维(2:LINGO 决定, 1:是, 0:否)  |
| 43 | SCALEM | 1     | 变换模型中的数据的尺度 (1:是, 0:否)  |
| 44 | PRIMPR | 0     | 原始单纯形法决定出基变量的策略(0: LINGO 自动决定, 1: 对部分出基变量尝试, 2: 用最陡边法对所有变量进行尝试)             |
| 45 | DUALPR | 0     | 对偶单纯形法决定出基变量的策略(0: LINGO 自动决定, 1:按最大下降比例法确定, 2: 用最陡边法对所有变量进行尝试)             |
| 46 | DUALCO | 1     | 指定对偶计算的级别 (0: 不计算任何对偶信息; 1: 计算对偶价格; 2: 计算对偶价格并分析敏感性)                        |
| 47 | RCMPSN | 0     | Use RC format names for MPS I/O (1:yes, 0:no)                               |
| 48 | MREGEN | 1     | 重新生成模型的频率 (0: 当模型的文本修改后;1: 当模型的文本修改或模型含有外部引用时; 3:每当有需要时)                    |
| 49 | BRANDR | 0     | 分枝时对变量取整的优先方向(0: LINGO 自动决定;1: 向上取整优先;2: 向下取整优先)                            |
| 50 | BRANPR | 0     | 分枝时变量的优先级 (0:LINGO 自动决定, 1:二进制 (0-1) 变量)                                    |
| 51 | CUTOFF | .1e-8 | 解的截断误差限   |
| 52 | STRONG | 10    | 指定强分枝的层次级别  |
| 53 | REOPTB | 0     | IP 热启动时的 LP 算法 (0: LINGO 自动选择; 1: 障碍法 (即内点法); 2: 原始单纯形法; 3: 对偶单纯形法)         |
| 54 | REOPTX | 0     | IP 冷启动时的 LP 算法 (选项同上)   |
| 55 | MAXCTP | 200   | 分枝中根节点增加割平面时, 最大迭代检查的次数   |
| 56 | RCTLIM | .75   | 割(平面)的个数相对于原问题的约束个数的上限(比值)  |
| 57 | GUBCTS | 1     | 是否使用广义上界 (GUB) 割 (1:是, 0:否)   |
| 58 | FLWCTS | 1     | 是否使用流 (Flow) 割 (1:是, 0:否)   |
| 59 | LFTCTS | 1     | 是否使用 Lift 割 (1:是, 0:否)  |

|    |        |         |  |
|----|--------|---------|--|
| 60 | PLOCTS | 1       | 是否使用选址问题的割 (1:是, 0:否)  |
| 61 | DISCTS | 1       | 是否使用分解割 (1:是, 0:否)   |
| 62 | KNPCTS | 1       | 是否使用背包覆盖割 (1:是, 0:否)   |
| 63 | LATCTS | 1       | 是否使用格 (Lattice) 割 (1:是, 0:否)   |
| 64 | GOMCTS | 1       | 是否使用 Gomory 割 (1:是, 0:否)   |
| 65 | COFCTS | 1       | 是否使用系数归约割 (1:是, 0:否)   |
| 66 | GCDCTS | 1       | 是否使用最大公因子割 (1:是, 0:否)  |
| 67 | SCLRLM | 1000    | 语法配色的最大行数 (仅 Windows 系统使用)   |
| 68 | SCLRDL | 0       | 语法配色的延时 (秒) (仅 Windows 系统使用)   |
| 69 | PRNCLR | 1       | 括号匹配配色 (1:是, 0:否, 仅 Windows 系统使用)  |
| 70 | MULTIS | 0       | NLP 多点求解的次数 (0:无, 可设为任意非负整数)   |
| 71 | USEQPR | 0       | 是否识别二次规划 (1:是, 0:否)  |
| 72 | GLOBAL | 0       | 是否对 NLP 采用全局最优求解程序 (1:是, 0:否)  |
| 73 | LNRISE | 0       | 线性化级别 (0:LINGO 自动决定, 1:无, 2:低, 3:高)  |
| 74 | LNBIGM | 100,000 | 线性化的大 M 系数   |
| 75 | LNDLTA | .1e-5   | 线性化的 Delta 误差系数  |
| 76 | BASCTS | 0       | 是否使用基本 (Basis) 割 (1:是, 0:否)  |
| 77 | MAXCTR | 2       | 分枝中非根节点增加割平面时, 最大迭代检查的次数   |
| 78 | HUMNTM | 0       | 分枝中每个节点使用启发式搜索的最小时间 (秒)  |
| 79 | DECOMP | 0       | 是否使用矩阵分解技术 (1:是, 0:否)  |
| 80 | GLBOPT | .1e-5   | 全局最优求解程序的最优性误差限  |
| 81 | GLBDLT | .1e-6   | 全局最优求解程序在凸化过程中增加的约束的误差限  |
| 82 | GLBVBD | .1e+11  | 全局最优求解程序中变量的上界   |
| 83 | GLBUBD | 2       | 全局最优求解程序中变量的上界的应用范围 (0: 所有变量都不使用上界; 1: 所有变量都使用上界; 2: 部分使用)                         |
| 84 | GLBBRN | 5       | 全局最优求解程序中第 1 次对变量分枝时使用的分枝策略 (0: 绝对宽度; 1: 局部宽度; 2: 全局宽度; 3: 全局距离; 4: 绝对冲突; 5: 相对冲突) |
| 85 | GLBBXS | 1       | 全局最优求解程序选择活跃分枝节点的方法 (0: 深度优先; 1: 具有最坏界的分枝优先)                                       |
| 86 | GLBREF | 3       | 全局最优求解程序中模型重整的级别: (0: 不进行重整; 1: 低; 2: 中; 3: 高)                                     |

## § 7 综合举例

### 例 7.1 求解非线性方程组

$$\begin{cases} x^2 + y^2 = 2 \\ 2x^2 + x + y^2 + y = 4 \end{cases}$$

其 LINGO 代码如下:

```

model:
  x^2+y^2=2;
  2*x^2+x+y^2+y=4;
end

```

计算的部分结果为

```

Feasible solution found at iteration:      0

Variable      Value
X              0.4543360
Y              1.339247

```

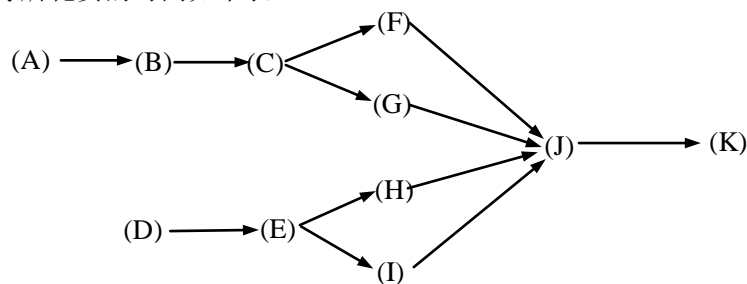
**例 7.2 装配线平衡模型** 一条装配线含有一系列的工作站，在最终产品的加工过程中每个工作站执行一种或几种特定的任务。装配线周期是指所有工作站完成分配给它们各自的任務所化费时间中的最大值。平衡装配线的目标是为每个工作站分配加工任务，尽可能使每个工作站执行相同数量的任务，其最终标准是装配线周期最短。不适当的平衡装配线将会产生瓶颈——有较少任务的工作站将被迫等待其前面分配了较多任务的工作站。

问题会因为众多任务间存在优先关系而变得更复杂，任务的分配必须服从这种优先关系。

这个模型的目标是最小化装配线周期。有 2 类约束：

- ① 要保证每件任务只能也必须分配至一个工作站来加工；
- ② 要保证满足任务间的所有优先关系。

例 有 11 件任务 (A—K) 分配到 4 个工作站 (1—4)，任务的优先次序如下图。每件任务所花费的时间如下表。



| 任务 | A  | B  | C | D  | E  | F  | G  | H  | I  | J | K |
|----|----|----|---|----|----|----|----|----|----|---|---|
| 时间 | 45 | 11 | 9 | 50 | 15 | 12 | 12 | 12 | 12 | 8 | 9 |

MODEL:

!装配线平衡模型;

SETS:

!任务集合，有一个完成时间属性 T;

TASK/ A B C D E F G H I J K/: T;

!任务之间的优先关系集合 (A 必须完成才能开始 B, 等等);

PRED( TASK, TASK)/ A,B B,C C,F C,G F,J G,J

J,K D,E E,H E,I H,J I,J /;

! 工作站集合;

STATION/1..4/;

TXS( TASK, STATION): X;

! X 是派生集合 TXS 的一个属性。如果 X (I, K) =1, 则表示第 I 个任务指派给第 K 个工作站完成;

ENDSETS

DATA:

!任务 A B C D E F G H I J K 的完成时间估计如下;

T = 45 11 9 50 15 12 12 12 12 8 9;

ENDDATA

! 当任务超过 15 个时, 模型的求解将变得很慢;  
 ! 每一个作业必须指派到一个工作站, 即满足约束①;  
 @FOR( TASK( I): @SUM( STATION( K): X( I, K)) = 1);  
 ! 对于每一个存在优先关系的作业来说, 前者对应的工作站 I 必须小于后者对应的工作站 J, 即满足约束②;  
 @FOR( PRED( I, J): @SUM( STATION( K): K \* X( J, K) - K \* X( I, K)) >= 0);  
 ! 对于每一个工作站来说, 其花费时间必须不大于装配线周期;  
 @FOR( STATION( K):  
 @SUM( TXS( I, K): T( I) \* X( I, K)) <= CYCTIME);  
 ! 目标函数是最小化转配线周期;  
 MIN = CYCTIME;  
 ! 指定 X(I, J) 为 0/1 变量;  
 @FOR( TXS: @BIN( X));

END

计算的部分结果为

Global optimal solution found at iteration: 1255  
 Objective value: 50.00000

| Variable | Value    | Reduced Cost |
|----------|----------|--------------|
| CYCTIME  | 50.00000 | 0.000000     |
| X( A, 1) | 1.000000 | 0.000000     |
| X( A, 2) | 0.000000 | 0.000000     |
| X( A, 3) | 0.000000 | 45.00000     |
| X( A, 4) | 0.000000 | 0.000000     |
| X( B, 1) | 0.000000 | 0.000000     |
| X( B, 2) | 0.000000 | 0.000000     |
| X( B, 3) | 1.000000 | 11.00000     |
| X( B, 4) | 0.000000 | 0.000000     |
| X( C, 1) | 0.000000 | 0.000000     |
| X( C, 2) | 0.000000 | 0.000000     |
| X( C, 3) | 0.000000 | 9.000000     |
| X( C, 4) | 1.000000 | 0.000000     |
| X( D, 1) | 0.000000 | 0.000000     |
| X( D, 2) | 1.000000 | 0.000000     |
| X( D, 3) | 0.000000 | 50.00000     |
| X( D, 4) | 0.000000 | 0.000000     |
| X( E, 1) | 0.000000 | 0.000000     |
| X( E, 2) | 0.000000 | 0.000000     |
| X( E, 3) | 1.000000 | 15.00000     |
| X( E, 4) | 0.000000 | 0.000000     |
| X( F, 1) | 0.000000 | 0.000000     |
| X( F, 2) | 0.000000 | 0.000000     |
| X( F, 3) | 0.000000 | 12.00000     |
| X( F, 4) | 1.000000 | 0.000000     |
| X( G, 1) | 0.000000 | 0.000000     |
| X( G, 2) | 0.000000 | 0.000000     |
| X( G, 3) | 0.000000 | 12.00000     |
| X( G, 4) | 1.000000 | 0.000000     |
| X( H, 1) | 0.000000 | 0.000000     |
| X( H, 2) | 0.000000 | 0.000000     |
| X( H, 3) | 1.000000 | 12.00000     |

|          |          |          |
|----------|----------|----------|
| X( H, 4) | 0.000000 | 0.000000 |
| X( I, 1) | 0.000000 | 0.000000 |
| X( I, 2) | 0.000000 | 0.000000 |
| X( I, 3) | 1.000000 | 12.00000 |
| X( I, 4) | 0.000000 | 0.000000 |
| X( J, 1) | 0.000000 | 0.000000 |
| X( J, 2) | 0.000000 | 0.000000 |
| X( J, 3) | 0.000000 | 8.000000 |
| X( J, 4) | 1.000000 | 0.000000 |
| X( K, 1) | 0.000000 | 0.000000 |
| X( K, 2) | 0.000000 | 0.000000 |
| X( K, 3) | 0.000000 | 9.000000 |
| X( K, 4) | 1.000000 | 0.000000 |

### 例 7.3 旅行售货员问题（又称货郎担问题，Traveling Salesman Problem）

有一个推销员，从城市 1 出发，要遍访城市 2, 3, ..., n 各一次，最后返回城市 1。已知从城市 i 到 j 的旅费为  $C_{ij}$ ，问他应按怎样的次序访问这些城市，使得总旅费最少？

可以用多种方法把 TSP 表示成整数规划模型。这里介绍的一种建立模型的方法，是把该问题的每个解（不一定是最优的）看作是一次“巡回”。

在下述意义下，引入一些 0-1 整数变量：

$$x_{ij} = \begin{cases} 1, & \text{巡回路线是从 } i \text{ 到 } j, \text{ 且 } i \neq j \\ 0, & \text{其它情况} \end{cases}$$

其目标只是使  $\sum_{i,j=1}^n C_{ij} x_{ij}$  为最小。

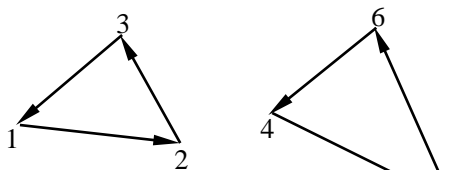
这里有两个明显的必须满足的条件：

访问城市 i 后必须要有一个即将访问的确切城市；访问城市 j 前必须要有一个刚刚访问过的确切城市。用下面的两组约束分别实现上面的两个条件。

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

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

到此我们得到了一个模型，它是一个指派问题的整数规划模型。但以上两个条件对于 TSP 来说并不充分，仅仅是必要条件。例如：



以上两个条件都满足，但它显然不是 5 的解，它存在两个子巡回。

这里，我们将叙述一种在原模型上附加充分的约束条件以避免产生子巡回的方法。把额外变量  $u_i$  ( $i = 2, 3, \dots, n$ ) 附加到问题中。可把这些变量看作是连续的（最然这些变量在最优解中取普通的整数）。现在附加下面形式的约束条件

$$u_i - u_j + n x_{ij} \leq n - 1, \quad 2 \leq i \neq j \leq n。$$

为了证明该约束条件有预期的效果，必须证明：（1）任何含子巡回的路线都不满足该约束条件；（2）全部巡回都满足该约束条件。

首先证明（1），用反证法。假设还存在子巡回，也就是说至少有两个子巡回。那么至少存在一个子巡回中不含城市 1。把该子巡回记为  $i_1 i_2 \cdots i_k i_1$ ，则必有

$$\begin{aligned}
 u_{i_1} - u_{i_2} + n &\leq n-1 \\
 u_{i_2} - u_{i_3} + n &\leq n-1 \\
 &\dots \\
 u_{i_k} - u_{i_1} + n &\leq n-1
 \end{aligned}$$

把这  $k$  个式子相加，有

$$n \leq n-1, \text{ 矛盾!}$$

故假设不正确，结论 (1) 得证。

下面证明 (2)，采用构造法。对于任意的总巡回  $1 i_1 \cdots i_{n-1} 1$ ，可取  $u_i =$  访问城市  $i$  的顺序数，取值范围为  $\{0, 1, \dots, n-2\}$ 。因此， $u_i - u_j \leq n-2$ ， $2 \leq i \neq j \leq n$ 。下面来证明总巡回满足该约束条件。

(i) 总巡回上的边

$$\begin{cases}
 u_{i_1} - u_{i_2} + n = n-1 \leq n-1 \\
 u_{i_2} - u_{i_3} + n = n-1 \leq n-1 \\
 \dots \\
 u_{i_{n-2}} - u_{i_{n-1}} + n = n-1 \leq n-1
 \end{cases}$$

(ii) 非总巡回上的边

$$\begin{cases}
 u_{i_r} - u_j \leq n-2 \leq n-1, & r=1, 2, \dots, n-2, \quad j \in \{2, 3, \dots, n\} - \{i_r, i_{r+1}\} \\
 u_{i_{n-1}} - u_j \leq n-2 \leq n-1, & j \in \{2, 3, \dots, n\} - \{i_r\}
 \end{cases}$$

从而结论 (2) 得证。

这样我们把 TSP 转化成了一个混合整数线性规划问题。

$$\begin{cases}
 \min \quad z = \sum_{\substack{i,j=1 \\ i \neq j}}^n c_{ij} x_{ij} \\
 s.t. \quad \sum_{i=1}^n x_{ij} = 1, & j=1, 2, \dots, n \\
 \sum_{j=1}^n x_{ij} = 1, & i=1, 2, \dots, n \\
 u_i - u_j + n x_{ij} \leq n-1, & 2 \leq i \neq j \leq n \\
 x_{ij} = 0, 1, & i, j=1, 2, \dots, n \\
 u_i \geq 0, & i=2, 3, \dots, n
 \end{cases}$$

显然，当城市个数较大（大于 30）时，该混合整数线性规划问题的规模会很大，从而给求解带来很大问题。TSP 已被证明是 NP 难问题，目前还没有发现多项式时间的算法。对于小规模问题，我们求解这个混合整数线性规划问题的方式还是有效的。

TSP 是一个重要的组合优化问题，除了有直观的应用外，许多其它看似无联系的优化问题也可转化为 TSP。例如：

**问题 1** 现需在一台机器上加工  $n$  个零件（如烧瓷器），这些零件可按任意先后顺序在机器上加工。我们希望加工完成所有零件的总时间尽可能少。由于加工工艺的要求，加工零件  $j$  时机器必须处于相应状态  $s_j$ （如炉温）。设起始未加工任何零件时机器处于状态  $s_0$ ，且当所有零件加工完成后需恢复到  $s_0$  状态。已知从状态  $s_i$  调整到状态  $s_j$  ( $j \neq i$ ) 需要时间  $c_{ij}$ 。零件  $j$  本身加工时间为  $p_j$ 。为方便起见，引入一个虚零件 0，其加工时间为 0，要求状态为  $s_0$ ，则  $\{0, 1, 2, \dots, n\}$  的一个圈置换  $\pi$  就表示对所有零件的一个加工顺序，在此置换下，

完成所有加工所需要的总时间为

$$\sum_{i=0}^n (c_{i\pi(i)} + p_{\pi(i)}) = \sum_{i=0}^n c_{i\pi(i)} + \sum_{j=0}^n p_j.$$

由于  $\sum_{j=0}^n p_j$  是一个常数, 故该零件的加工顺序问题变成 TSP。

!旅行售货员问题;

model:

sets:

```
city / 1.. 5/: u;
link( city, city):
    dist, ! 距离矩阵;
    x;
```

endsets

```
n = @size( city);
```

data: !距离矩阵, 它并不需要是对称的;

```
dist = @qrand(1); !随机产生, 这里可改为你要解决的问题的数据;
```

enddata

!目标函数;

```
min = @sum( link: dist * x);
```

```
@FOR( city( K):
```

!进入城市 K;

```
@sum( city( I) | I #ne# K: x( I, K)) = 1;
```

!离开城市 K;

```
@sum( city( J) | J #ne# K: x( K, J)) = 1;
```

```
);
```

!保证不出现子圈;

```
@for(city(I) | I #gt# 1:
```

```
@for( city( J) | J#gt#1 #and# I #ne# J:
```

```
u(I)-u(J)+n*x(I, J)<=n-1);
```

```
);
```

!限制 u 的范围以加速模型的求解, 保证所加限制并不排除掉 TSP 问题的最优解;

```
@for(city(I) | I #gt# 1: u(I)<=n-2 );
```

!定义 x 为 0\1 变量;

```
@for( link: @bin( x));
```

end

计算的部分结果为:

```
Global optimal solution found at iteration:      77
Objective value:                                1.692489
```

| Variable    | Value     | Reduced Cost |
|-------------|-----------|--------------|
| N           | 5.000000  | 0.000000     |
| U( 1)       | 0.000000  | 0.000000     |
| U( 2)       | 1.000000  | 0.000000     |
| U( 3)       | 3.000000  | 0.000000     |
| U( 4)       | 2.000000  | 0.000000     |
| U( 5)       | 0.000000  | 0.000000     |
| DIST( 1, 1) | 0.4491774 | 0.000000     |
| DIST( 1, 2) | 0.2724506 | 0.000000     |



|             |               |               |
|-------------|---------------|---------------|
| DIST( 1, 3) | 0.1240430     | 0.000000      |
| DIST( 1, 4) | 0.9246848     | 0.000000      |
| DIST( 1, 5) | 0.4021706     | 0.000000      |
| DIST( 2, 1) | 0.7091469     | 0.000000      |
| DIST( 2, 2) | 0.1685199     | 0.000000      |
| DIST( 2, 3) | 0.8989646     | 0.000000      |
| DIST( 2, 4) | 0.2502747     | 0.000000      |
| DIST( 2, 5) | 0.8947571     | 0.000000      |
| DIST( 3, 1) | 0.8648940E-01 | 0.000000      |
| DIST( 3, 2) | 0.6020591     | 0.000000      |
| DIST( 3, 3) | 0.3380884     | 0.000000      |
| DIST( 3, 4) | 0.6813164     | 0.000000      |
| DIST( 3, 5) | 0.2236271     | 0.000000      |
| DIST( 4, 1) | 0.9762987     | 0.000000      |
| DIST( 4, 2) | 0.8866343     | 0.000000      |
| DIST( 4, 3) | 0.7139008     | 0.000000      |
| DIST( 4, 4) | 0.2288770     | 0.000000      |
| DIST( 4, 5) | 0.7134250     | 0.000000      |
| DIST( 5, 1) | 0.8524679     | 0.000000      |
| DIST( 5, 2) | 0.2396538     | 0.000000      |
| DIST( 5, 3) | 0.5735525     | 0.000000      |
| DIST( 5, 4) | 0.1403314     | 0.000000      |
| DIST( 5, 5) | 0.6919708     | 0.000000      |
| X( 1, 1)    | 0.000000      | 0.4491774     |
| X( 1, 2)    | 0.000000      | 0.2724506     |
| X( 1, 3)    | 0.000000      | 0.1240430     |
| X( 1, 4)    | 0.000000      | 0.9246848     |
| X( 1, 5)    | 1.000000      | 0.4021706     |
| X( 2, 1)    | 0.000000      | 0.7091469     |
| X( 2, 2)    | 0.000000      | 0.1685199     |
| X( 2, 3)    | 0.000000      | 0.8989646     |
| X( 2, 4)    | 1.000000      | 0.2502747     |
| X( 2, 5)    | 0.000000      | 0.8947571     |
| X( 3, 1)    | 1.000000      | 0.8648940E-01 |
| X( 3, 2)    | 0.000000      | 0.6020591     |
| X( 3, 3)    | 0.000000      | 0.3380884     |
| X( 3, 4)    | 0.000000      | 0.6813164     |
| X( 3, 5)    | 0.000000      | 0.2236271     |
| X( 4, 1)    | 0.000000      | 0.9762987     |
| X( 4, 2)    | 0.000000      | 0.8866343     |
| X( 4, 3)    | 1.000000      | 0.7139008     |
| X( 4, 4)    | 0.000000      | 0.2288770     |
| X( 4, 5)    | 0.000000      | 0.7134250     |
| X( 5, 1)    | 0.000000      | 0.8524679     |
| X( 5, 2)    | 1.000000      | 0.2396538     |
| X( 5, 3)    | 0.000000      | 0.5735525     |
| X( 5, 4)    | 0.000000      | 0.1403314     |
| X( 5, 5)    | 0.000000      | 0.6919708     |

**例 7.4 最短路问题** 给定  $N$  个点  $p_i (i=1, 2, \dots, N)$  组成集合  $\{p_i\}$ , 由集合中任一点  $p_i$  到另一点  $p_j$  的距离用  $c_{ij}$  表示, 如果  $p_i$  到  $p_j$  没有弧联结, 则规定  $c_{ij} = +\infty$ , 又规定  $c_{ii} = 0 (1 \leq i \leq N)$ , 指定一个终点  $p_N$ , 要求从  $p_i$  点出发到  $p_N$  的最短路线。这里我们用动

态规划方法来做。用所在的点  $P_i$  表示状态，决策集合就是除  $P_i$  以外的点，选定一个点  $P_j$  以后，得到效益  $c_{ij}$  并转入新状态  $P_j$ ，当状态是  $P_N$  时，过程停止。显然这是一个不定期多阶段决策过程。

定义  $f(i)$  是由  $P_i$  点出发至终点  $P_N$  的最短路程，由最优化原理可得

$$\begin{cases} f(i) = \min_j \{c_{ij} + f(j)\}, & i = 1, 2, \dots, N-1 \\ f(N) = 0 \end{cases}$$

这是一个函数方程，用 LINGO 可以方便的解决。

!最短路问题;

model:

data:

n=10;

enddata

sets:

cities/1..n/: F; !10个城市;

roads(cities,cities)/

1,2 1,3

2,4 2,5 2,6

3,4 3,5 3,6

4,7 4,8

5,7 5,8 5,9

6,8 6,9

7,10

8,10

9,10

/: D, P;

endsets

data:

D=

6 5

3 6 9

7 5 11

9 1

8 7 5

4 10

5

7

9;

enddata

F(n)=0;

@for(cities(i) | i #lt# n:

F(i)=@min(roads(i,j): D(i,j)+F(j));

);

!显然，如果  $P(i, j)=1$ ，则点  $i$  到点  $n$  的最短路径的第一步是  $i \rightarrow j$ ，否则就不是。

由此，我们就可方便的确定出最短路径;

@for(roads(i,j):

P(i,j)=@if(F(i) #eq# D(i,j)+F(j), 1, 0)

);

end

计算的部分结果为:

Feasible solution found at iteration:

0

| Variable  | Value    |
|-----------|----------|
| N         | 10.00000 |
| F( 1)     | 17.00000 |
| F( 2)     | 11.00000 |
| F( 3)     | 15.00000 |
| F( 4)     | 8.000000 |
| F( 5)     | 13.00000 |
| F( 6)     | 11.00000 |
| F( 7)     | 5.000000 |
| F( 8)     | 7.000000 |
| F( 9)     | 9.000000 |
| F( 10)    | 0.000000 |
| P( 1, 2)  | 1.000000 |
| P( 1, 3)  | 0.000000 |
| P( 2, 4)  | 1.000000 |
| P( 2, 5)  | 0.000000 |
| P( 2, 6)  | 0.000000 |
| P( 3, 4)  | 1.000000 |
| P( 3, 5)  | 0.000000 |
| P( 3, 6)  | 0.000000 |
| P( 4, 7)  | 0.000000 |
| P( 4, 8)  | 1.000000 |
| P( 5, 7)  | 1.000000 |
| P( 5, 8)  | 0.000000 |
| P( 5, 9)  | 0.000000 |
| P( 6, 8)  | 1.000000 |
| P( 6, 9)  | 0.000000 |
| P( 7, 10) | 1.000000 |
| P( 8, 10) | 1.000000 |
| P( 9, 10) | 1.000000 |

### 例 7.5 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  $28 \text{ km/h}$ 。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点点火一次。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  $60 \text{ m}$  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。

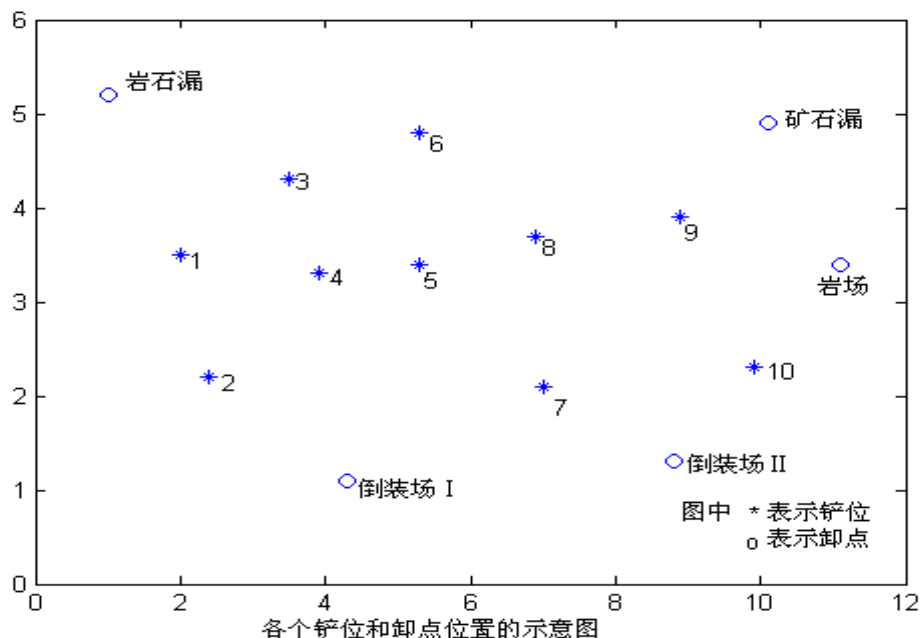
某露天矿有铲位 10 个，卸点 5 个，现有铲车 7 台，卡车 20 辆。各卸点一个班次的产量要求：矿石漏 1.2 万吨、倒装场 I 1.3 万吨、倒装场 II 1.3 万吨、岩石漏 1.9 万吨、岩场 1.3 万吨。

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%   |



```

model:
title CUMCM-2003B-01;
sets:
cai / 1..10 /:crate,cnum,cy,ck,flag;
xie / 1..5 /:xsubject,xnum;

```

```

link( xie, cai ):distance, lsubject, number, che, b;
endsets
data:
crate=30 28 29 32 31 33 32 31 33 31;
xsubject= 1.2 1.3 1.3 1.9 1.3 ;
distance= 5.26 5.19 4.21 4.00 2.95 2.74 2.46 1.90 0.64 1.27
          1.90 0.99 1.90 1.13 1.27 2.25 1.48 2.04 3.09 3.51
          5.89 5.61 5.61 4.56 3.51 3.65 2.46 2.46 1.06 0.57
          0.64 1.76 1.27 1.83 2.74 2.60 4.21 3.72 5.05 6.10
          4.42 3.86 3.72 3.16 2.25 2.81 0.78 1.62 1.27 0.50;
cy = 1.25 1.10 1.35 1.05 1.15 1.35 1.05 1.15 1.35 1.25;
ck = 0.95 1.05 1.00 1.05 1.10 1.25 1.05 1.30 1.35 1.25;
enddata
!目标函数;
min=@sum( cai (i):
        @sum ( xie (j):
            number (j,i)*154*distance (j,i)));

!max =@sum(link(i,j):number(i,j));
!max=xnum (3)+xnum (4)+xnum (1)+xnum (2)+xnum(5);
!min=@sum( cai (i):
!        @sum ( xie (j):
!            number (j,i)*154*distance (j,i)));
!xnum (1)+xnum (2)+xnum(5)=340;
!xnum (1)+xnum (2)+xnum(5)=341;
!xnum (3)=160;
!xnum (4)=160;
!卡车每一条路线上最多可以运行的次数;
@for (link (i,j):
b(i,j)=@floor((8*60-(@floor((distance(i,j)/28*60*2+3+5)/5)-1)*5)/(distance(i,j)/28*
60*2+3+5)));
!b(i,j)=@floor(8*60/(distance(i,j)/28*60*2+3+5)));

!t(i,j)=@floor((distance(i,j)/28*60*2+3+5)/5);
!b(i,j)=@floor((8*60-(@floor((distance(i,j)/28*60*2+3+5)/5))*5)/(distance(i,j)/28*6
0*2+3+5)));
!每一条路线上的最大总车次的计算;
@for( link (i,j):
lsubject(i,j)=(@floor((distance(i,j)/28*60*2+3+5)/5))*b(i,j));
!计算各个铲位的总产量;
@for (cai (j):
        cnum(j)=@sum(xie(i):number(i,j)));
!计算各个卸点的总产量;
@for (xie(i):
        xnum(i)=@sum(cai (j):number(i,j)));
!道路能力约束;
@for (link (i,j):
        number(i,j)<=lsubject(i,j));
!电铲能力约束;
@for (cai (j) :
        cnum(j) <= flag(j)*8*60/5 );
!电铲数量约束 ---- added by Xie Jinxing, 2003-09-07;
@sum(cai (j): flag(j) ) <=7;

```

```

!卸点能力约束;
@for (xie (i):
    xnum (i)<=8*20);
!铲位产量约束;
@for (cai (i):    number(1,i)+number(2,i)+number(5,i)<=ck(i)*10000/154);
@for (cai (i):    number(3,i)+number(4,i)<=cy(i)*10000/154);
!产量任务约束;
@for (xie (i):
    xnum (i)>= xsubject (i)*10000/154);
!铁含量约束;
@sum(cai (j):
    number(1,j)*(crate(j)-30.5) )<=0;
@sum(cai (j):
    number(2,j)*(crate(j)-30.5) )<=0;
@sum(cai (j):
    number(5,j)*(crate(j)-30.5) )<=0;
@sum(cai (j):
    number(1,j)*(crate(j)-28.5) )>=0;
@sum(cai (j):
    number(2,j)*(crate(j)-28.5) )>=0;
@sum(cai (j):
    number(5,j)*(crate(j)-28.5) )>=0;
!关于车辆的具体分配;
@for (link (i,j):
    che (i,j)=number (i,j)/b(i,j));
!各个路线所需卡车数简单加和;
hehe=@sum (link (i,j): che (i,j));
!整数约束;
@for (link (i,j): @gin(number (i,j)));
@for (cai (j): @bin(flag (j)));
!车辆能力约束;
hehe<=20;
ccnum=@sum(cai (j): cnum(j) );
end

```

### 例 7.6 最小生成树 (Minimal Spanning Tree, MST) 问题

求解最小生成树的方法虽然很多,但是利用 LINGO 建立相应的整数规划模型是一种新的尝试。这对于处理非标准的 MST 问题非常方便。我们主要参考了文[7]。

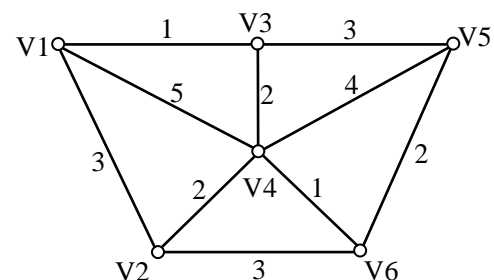
在图论中,称无圈的连通图为树。在一个连通图  $G$  中,称包含图  $G$  全部顶点的树为图  $G$  的生成树。生成树上各边的权之和称为该生成树的权。连通图  $G$  的权最小的生成树称为图  $G$  的最小生成树。

许多实际问题都可以归结为最小生成树。例如,如何修筑一些公路把若干个城镇连接起来;如何架设通讯网络将若干个地区连接起来;如何修筑水渠将水源和若干块待灌溉的土地连接起来等等。为了说明问题,以下面的问题作为范例。

范例:假设某电话公司计划在六个村庄架设电话线,各村庄之间的距离如图所示。试求出使电话线总长度最小的架线方案。

为了便于计算机求解,特作如下规定:(1)节点  $V_1$  表示树根;  
(2)当两个节点之间没有线路时,规定两个节点之间的距离为  $M$  (较大的值)。

MST 的整数规划模型如下:



### 例 7.7 分配问题（指派问题，Assignment Problem）

这是个给  $n$  个人分配  $n$  项工作以获得某个最高总效果的问题。第  $i$  个人完成第  $j$  项工作需要平均时间  $c_{ij}$ 。要求给每个人分配一项工作，并要求分配完这些工作，以使完成全部任务的总时间为最小。该问题可表示如下：

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

显然，此问题可看作是运输问题的特殊情况。可将此问题看作具有  $n$  个源和  $n$  个汇的问题，每个源有 1 单位的可获量，而每个汇有 1 单位的需要量。从表面看，这问题要求用整数规划以保证  $x_{ij}$  能取 0 或 1。然而，幸运的是，此问题是运输问题的特例，因此即使不限制  $x_{ij}$  取 0 或 1，最优解也将取 0 或 1。如果把婚姻看作分配问题，丹茨证明，整数性质证明一夫一妻会带来最美满幸福的生活！显然，分配问题可以作为线性规划问题来求解，尽管模型可能很大。例如，给 100 人分配 100 项工作将使所得的模型具有 10000 个变量。这时，如采用专门算法效果会更好。时间复杂度为  $O(n^3)$  的匈牙利算法便是好选择，这是由 Kuhn（1955）提出的。

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

```

计算的部分结果为:

Global optimal solution found at iteration:

14

Objective value:

18.00000

| Variable         | Value    | Reduced Cost |
|------------------|----------|--------------|
| VOLUME ( W1, J1) | 0.000000 | 4.000000     |
| VOLUME ( W1, J2) | 0.000000 | 0.000000     |
| VOLUME ( W1, J3) | 0.000000 | 3.000000     |
| VOLUME ( W1, J4) | 0.000000 | 4.000000     |
| VOLUME ( W1, J5) | 1.000000 | 0.000000     |
| VOLUME ( W1, J6) | 0.000000 | 0.000000     |
| VOLUME ( W1, J7) | 0.000000 | 0.000000     |
| VOLUME ( W2, J1) | 0.000000 | 2.000000     |
| VOLUME ( W2, J2) | 0.000000 | 7.000000     |
| VOLUME ( W2, J3) | 0.000000 | 2.000000     |
| VOLUME ( W2, J4) | 1.000000 | 0.000000     |
| VOLUME ( W2, J5) | 0.000000 | 4.000000     |
| VOLUME ( W2, J6) | 0.000000 | 3.000000     |
| VOLUME ( W2, J7) | 0.000000 | 3.000000     |
| VOLUME ( W3, J1) | 0.000000 | 5.000000     |
| VOLUME ( W3, J2) | 0.000000 | 2.000000     |
| VOLUME ( W3, J3) | 0.000000 | 0.000000     |
| VOLUME ( W3, J4) | 0.000000 | 8.000000     |
| VOLUME ( W3, J5) | 0.000000 | 5.000000     |
| VOLUME ( W3, J6) | 0.000000 | 4.000000     |
| VOLUME ( W3, J7) | 1.000000 | 0.000000     |
| VOLUME ( W4, J1) | 0.000000 | 5.000000     |
| VOLUME ( W4, J2) | 0.000000 | 4.000000     |
| VOLUME ( W4, J3) | 0.000000 | 4.000000     |
| VOLUME ( W4, J4) | 0.000000 | 0.000000     |
| VOLUME ( W4, J5) | 0.000000 | 5.000000     |
| VOLUME ( W4, J6) | 1.000000 | 0.000000     |
| VOLUME ( W4, J7) | 0.000000 | 2.000000     |
| VOLUME ( W5, J1) | 1.000000 | 0.000000     |
| VOLUME ( W5, J2) | 0.000000 | 1.000000     |
| VOLUME ( W5, J3) | 0.000000 | 6.000000     |
| VOLUME ( W5, J4) | 0.000000 | 2.000000     |
| VOLUME ( W5, J5) | 0.000000 | 3.000000     |
| VOLUME ( W5, J6) | 0.000000 | 0.000000     |
| VOLUME ( W5, J7) | 0.000000 | 1.000000     |
| VOLUME ( W6, J1) | 0.000000 | 4.000000     |
| VOLUME ( W6, J2) | 0.000000 | 4.000000     |
| VOLUME ( W6, J3) | 1.000000 | 0.000000     |
| VOLUME ( W6, J4) | 0.000000 | 0.000000     |
| VOLUME ( W6, J5) | 0.000000 | 5.000000     |
| VOLUME ( W6, J6) | 0.000000 | 10.00000     |
| VOLUME ( W6, J7) | 0.000000 | 0.000000     |
| VOLUME ( W7, J1) | 0.000000 | 7.000000     |
| VOLUME ( W7, J2) | 1.000000 | 0.000000     |



|                  |          |          |
|------------------|----------|----------|
| VOLUME ( W7, J3) | 0.000000 | 0.000000 |
| VOLUME ( W7, J4) | 0.000000 | 9.000000 |
| VOLUME ( W7, J5) | 0.000000 | 0.000000 |
| VOLUME ( W7, J6) | 0.000000 | 3.000000 |
| VOLUME ( W7, J7) | 0.000000 | 5.000000 |

### 例 7.8 二次分配问题 (Quadratic Assignment Problem)

这个问题是指派问题的一种推广。可以把指派问题看作线性规划问题，故较易求解，而二次分配问题是纯整数规划问题，往往很难求解。

与分配问题一样，二次分配问题也与两个目标集合 S、T 有关。S 和 T 含有相同数目的元素，以便达到某一目标。这里两种必须满足的条件：必须把 S 的每个元素确切地分配给 T 的一个元素；T 的每个元素只能接受 S 的一个元素。可引入 0-1 变量：

$$x_{ij} = \begin{cases} 1, & \text{把 } i \text{ (S 的一个元素) 分配给 } j \text{ (T 的一个元素)} \\ 0, & \text{其它} \end{cases}$$

用和分配问题相同的约束条件给出以上两个条件：

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

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

但是本问题的目标比分配问题的更加复杂。我们得到的价格系数  $c_{ijkl}$ ，其解释是：在  $i$  (S 的一个元素) 分配给  $j$  (T 的一个元素) 的同时把  $k$  (S 的一个元素) 分配给  $l$  (T 的一个元素) 所应承担的费用。显然，只有当  $x_{ij} = 1$  且  $x_{kl} = 1$ ，即其乘积  $x_{ij}x_{kl} = 1$  时，才承担这种费用。于是本目标变成一个 0-1 变量的二次表达式：

$$\sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \sum_{l=1}^n c_{ijkl} x_{ij} x_{kl}$$

最常见的是系数  $c_{ijkl}$  从其它系数  $t_{ik}$  和  $d_{jl}$  的乘积推出来的情况： $c_{ijkl} = t_{ik}d_{jl}$ 。为了弄清这个相当复杂的模型，研究下面两个应用是有好处的。

首先认为 S 是一个  $n$  个工厂的集合，T 是一个  $n$  个城市的集合。本问题就是要在每一城市中设置一个工厂，并要使工厂之间总的通讯费用最小。通讯费用取决于 (1) 每对工厂之间通讯的次数；(2) 每对工厂所在两个城市之间的距离。

显然，有些工厂很少与别的工厂通讯，虽相距甚远而费用却不大。另一方面，有些工厂可能需要大量通讯。通讯费取决于距离的远近。在这个应用中， $t_{ik}$  表示工厂  $i$  和工厂  $k$  之间的通讯次数 (以适当的单位计量)； $d_{jl}$  为城市  $j$  和城市  $l$  之间每单位的通讯费用 (显然这与  $j$  和  $l$  之间的距离有关)。如果工厂  $i$  和  $k$  分别设在城市  $j$  和  $l$ ，显然这两家间的通讯费由  $c_{ijkl} = t_{ik}d_{jl}$  来确定。因而总费用可用上述目标函数来表示。

**例 7.9** 有 4 名同学到一家公司参加三个阶段的面试：公司要求每个同学都必须首先找公司秘书初试，然后到部门主管处复试，最后到经理处参加面试，并且不允许插队（即在任何一个阶段 4 名同学的顺序是一样的）。由于 4 名同学的专业背景不同，所以每人在三个阶段的面试时间也不同，如下表所示（单位：分钟）：

|     | 秘书初试 | 主管复试 | 经理面试 |
|-----|------|------|------|
| 同学甲 | 13   | 15   | 20   |
| 同学乙 | 10   | 20   | 18   |
| 同学丙 | 20   | 16   | 10   |
| 同学丁 | 8    | 10   | 15   |

这 4 名同学约定以后一起离开公司。假定现在是早晨 8:00，问他们最早何时能离开公司？（建立规划模型求解）

本问题是一个排列排序问题。对于阶段数不小于 3 的问题没有有效算法，也就是说对于学生数稍多一点儿（比如 20）的情况是无法精确求解的。为此人们找到了很多近似算法。这里我们建立的规划模型可以实现该问题的精确求解，但你会看到它的变量和约束是学生数的平方。因此，当学生数稍多一点儿规划模型的规模经很大，求解会花费很长时间。

记

!三阶段面试模型;

model:

sets:

students; !学生集三阶段面试模型;

phases; !阶段集;

sp(students, phases):t, x;

ss(students, students) | &1 #LT# &2:y;

endsets

data:

students = s1..s4;

phases = p1..p3;

t=

13 15 20

10 20 18

20 16 10

8 10 15;

enddata

ns=@size(students); !学生数;

np=@size(phases); !阶段数;

!单个学生面试时间先后次序的约束;

@for(sp(I, J) | J #LT# np:

x(I, J)+t(I, J)<=x(I, J+1)

);

!学生间的面试先后次序保持不变的约束;

@for(ss(I, K):

@for(phases(J):

x(I, J)+t(I, J)-x(K, J)<=200\*y(I, K);

```

        x(K, J) + t(K, J) - x(I, J) <= 200 * (1 - y(I, K));
    )
);
! 目标函数;
min = TMAX;
@for(students(I):
    x(I, 3) + t(I, 3) <= TMAX
);
! 把 Y 定义 0-1 变量;
@for(ss: @bin(y));
end

```

计算的部分结果为:

Global optimal solution found at iteration: 898  
 Objective value: 84.00000

| Variable   | Value    | Reduced Cost |
|------------|----------|--------------|
| NS         | 4.000000 | 0.000000     |
| NP         | 3.000000 | 0.000000     |
| TMAX       | 84.00000 | 0.000000     |
| X( S1, P1) | 8.000000 | 0.000000     |
| X( S1, P2) | 21.00000 | 0.000000     |
| X( S1, P3) | 36.00000 | 0.000000     |
| X( S2, P1) | 21.00000 | 0.000000     |
| X( S2, P2) | 36.00000 | 0.000000     |
| X( S2, P3) | 56.00000 | 0.000000     |
| X( S3, P1) | 31.00000 | 0.000000     |
| X( S3, P2) | 56.00000 | 0.000000     |
| X( S3, P3) | 74.00000 | 0.000000     |
| X( S4, P1) | 0.000000 | 1.000000     |
| X( S4, P2) | 8.000000 | 0.000000     |
| X( S4, P3) | 18.00000 | 0.000000     |
| Y( S1, S2) | 0.000000 | -200.0000    |
| Y( S1, S3) | 0.000000 | 0.000000     |
| Y( S1, S4) | 1.000000 | 200.0000     |
| Y( S2, S3) | 0.000000 | -200.0000    |
| Y( S2, S4) | 1.000000 | 0.000000     |
| Y( S3, S4) | 1.000000 | 0.000000     |