

附录三 运筹学的 LINGO 软件

§ 1 简介

LINGO 软件有多种版本, 如 LINDO, GINO 和 LINGO(包括 LINGO NL)软件。

LINDO 是一种专门用于求解数学规划问题的优化计算软件包, 版权现在由美国 LINDO 系统公司 (Lindo System Inc.) 所拥有。LINDO 软件包的特点是程序执行速度快, 易于方便地输入、修改、求解和分析一个数学规划 (优化问题), 因此 LINDO 在教学, 科研和工业界得到广泛应用。有关该软件的发行版本, 发行价格和其它最新信息都可以从 LINDO 系统公司的 INTERNET 网络站点 <http://www.lindo.com> 获取, 该站点还提供部分 LINDO 软件的演示版本或测试版本。

LINDO 由美国芝加哥大学的 Linus Schrage 教授首先开发, 随后又推出了 GINO, LINGO, LINGO NL (又称 LINGO2) 和“what's best!”等优化软件, 现在一般仍用 LINDO 作为这些软件的统称。各组件的功能各有侧重, 分别简要介绍如下:

(i) LINDO 是 Linear Interactive and Discrete Optimizer 字首的缩写形式, 可以用来求解线性规划 (LP—Linear Programming), 整数规划 (IP—Integer Programming) 和二次规划 (QP—Quadratic Programming) 问题。

(ii) GINO 是 General Interactive Optimizer 字首的缩写形式, 可以用来求解非线性规划 (NLP—Non-Linear Programming) 问题, 也可用于求解一些线性和非线性方程 (组) 以及代数方程求根等。GINO 中包含了各种一般的数学函数 (包括大量的概率函数), 可供使用者建立问题模型时调用。

(iii) LINGO 可以用来求解线性, 非线性和整数规划问题。

(iv) LINGO NL (LINGO2) 可以用来求解线性, 非线性和整数规划问题。

与 LINDO 和 GINO 不同的是, LINGO 和 LINGO NL (LINGO2) 包含了内置的建模语言, 允许以简练, 直观的方式描述较大规模的优化问题, 模型中所需的数据可以以一定格式保存在独立的文件中。

(v) “what's best!” 组件主要用于数据文件是由电子表格软件 (如 LUTOS1-2-3 和 MS OFFICE 等) 生成的情形。

LINDO 软件包有多种版本, 但其软件内核和使用方法基本上是类似的。下面介绍 LINGO 组件的基本使用方法。

§ 2 LINGO 快速入门

当你在 windows 下开始运行 LINGO 系统时, 会得到一个窗口:

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, 建立的模型都要在该窗口内编码实现。下面举两个例子。

例 2.1 如何在 LINGO 中求解如下的 LP 问题:

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

由于 LINGO 中已假设所有的变量是非负的, 所以非负约束不必再输入到计算机中, LINGO 也不区分变量中的大小写字符 (任何小写字符将被转换为大写字符); 约束条件


中的“<=”及“>=”可用“<”及“>”代替。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 2.2 使用 LINGO 软件计算 6 个发点 8 个收点的最小费用运输问题。产销单位运价如表 1。

表 1 运输费用表

单位运价 产地 \ 销地	销地								产量
	B ₁	B ₂	B ₃	B ₄	B ₅	B ₆	B ₇	B ₈	
A ₁	6	2	6	7	4	2	5	9	60
A ₂	4	9	5	3	8	5	8	2	55
A ₃	5	2	1	9	7	4	3	3	51
A ₄	7	6	7	3	9	2	7	1	43
A ₅	2	3	9	5	7	2	6	5	41
A ₆	5	5	2	2	8	1	4	3	52
销量	35	37	22	32	41	32	43	38	

解 设 x_{ij} ($i=1,2,\dots,6; j=1,2,\dots,8$) 表示 i 产地运到 j 销地的量, c_{ij} 表示 i 产地到 j 销地的单位运价, d_j 表示 j 销地的需求量, e_i 表示 i 产地的产量, 建立如下线性规划模型

$$\begin{aligned} \min \quad & \sum_{i=1}^6 \sum_{j=1}^8 c_{ij} x_{ij} \\ \text{s. t.} \quad & \begin{cases} \sum_{i=1}^6 x_{ij} = d_j, & j=1,2,\dots,8 \\ \sum_{j=1}^8 x_{ij} \leq e_i, & i=1,2,\dots,6 \\ x_{ij} \geq 0, & i=1,2,\dots,6; j=1,2,\dots,8 \end{cases} \end{aligned}$$

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;
```

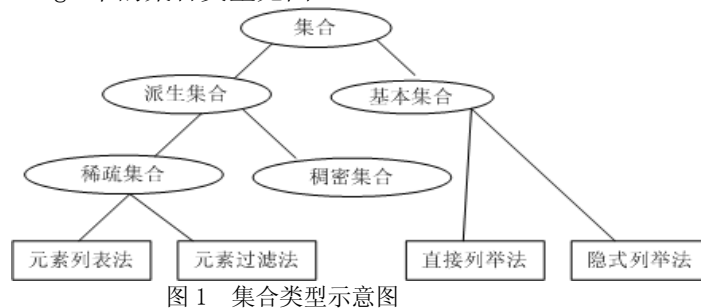
```

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));
!下面是数据;
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。

Lingo 模型由 4 个部分构成：目标与约束，集合，数据，初始。
Lingo 中的集合类型见图 1。



Lingo 中有三类运算符：算术运算符，逻辑运算符和关系运算符。运算符的优先级见表 2。

表 2 运算符优先级表

优先级	运算符
最高	#NOT# , - (负号)
	^
	*, /
	+, - (减号)
	#EQ#, #NE#, #GE#, #GT#, #LE#, #LT#
	#AND#, #OR#
最低	<, =, >

注意使用 Lingo 中的四个集合循环函数：FOR, SUM, MAX, MIN。

在 Lingo 中可以使用外部数据文件，有如下几种方法：

- (1) 复制—粘贴方法 (Copy—Paste);
- (2) @FILE 输入数据, @TEXT 输出数据;
- (3) @OLE 函数与电子表格软件 (如 EXCEL) 连接;

(4) @ODBC 函数与数据库连接。

§ 3 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集成集 (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。

3.1 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。

3.2 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。

3.2.1 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，中间用空格或逗号搁开，允许混合使用。

例 3.1 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。列表如表 3。

表 3 隐式罗列成员示例

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 3.2

!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。如下所示：

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。

3.2.2 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 3.3

```
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	(A, M, 2)
3	(A, N, 1)
4	(A, N, 2)
5	(B, M, 1)
6	(B, M, 2)
7	(B, N, 1)
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 3.4

```
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) #gt# 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来定义。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。

§4 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）。

4.1 模型的数据部分

4.1.1 数据部分入门

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 4.1

```

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 4.2

```
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。

4.1.2 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 4.3

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

也可以同时指定多个参数。

例 4.4

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

4.1.3 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。LINGO 可方便地做到这件事。

在本该放数的地方输入一个问号（?）。

例 4.5

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

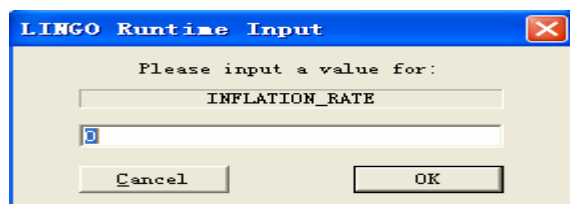


图 2 交互式输入对话框

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图 2 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。

4.1.4 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 4.6

```
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 4.7

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

4.1.5 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 4.8

```
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。

4.2 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 4.9

```

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。

§ 5 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. 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。

5.1 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。

5.1.1 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：

^ 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）

^

* /

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 5.1 算术运算符示例。

$2 - 5 / 3$, $(2 + 4) / 5$ 等等。

5.1.2 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，来控制在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 5.2 逻辑运算符示例

2 #gt# 3 #and# 4 #gt# 2，其结果为假（0）。

5.1.3 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 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 <= = >=

5.2 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。

@mod(x, y): 返回 x 除以 y 的余数。

@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 $x \geq 0$ 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 5.3 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 3 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, 函数@bnd 定义变量的下界和上界。

5.3 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 5.4 (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)

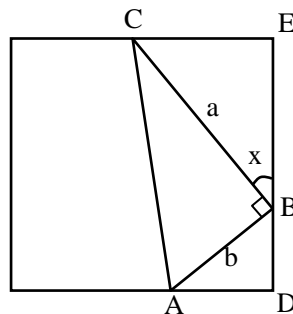


图 3 三角形示意图

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：

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

5.4 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 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 5.5

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

例 5.6 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

```

5.5 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。

5.6 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 5.7 全集为 I，B 是 I 的一个子集，C 是 B 的补集。

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

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 5.8 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 5.9

```
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, limit] 内。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。

5.7 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；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 5.10 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 5.11 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 5.12 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 5.13 （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-5,7))) >= required(J));
end

```

计算的部分结果为

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。

5.8 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。

例 5.14 以例 2.2 来讲解@file 函数的用法。

注意到在例 2.2 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读到模型中来。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。如果最后一条记录没有记录结束标记，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 5.15 借用例 5.13,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 5.16

```
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。下面图 4 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					

图 4 Excel 图表

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。

5.9 结果报告函数

1. @WRITE(obj1[, ..., objn])

这个函数只能在数据段中使用，用于输出一系列结果 (obj1, ..., objn)，其中 obj1, ..., objn 等可以是变量（但不能只是属性），也可以是字符串（放在单引号中的为字符串）或换行(@NEWLINE(1))等。结果可以输出到一个文件，或电子表格（如 Excel），或数据库，这取决于@WRITE 所在的输出语句中左边的定位函数。例如：

DATA:

```
@TEXT()=@WRITE(' A is ',A,' ,B is ',B,' ,A/B is' ,A/B);
```

ENDDATA

其中 A, B 是模型中的变量，则上面语句的作用是在屏幕上输出 A, B 以及 A/B 的值（注意上面语句中还增加了一些字符串，使结果读起来更方便）。假设计算结束时 A=10, B=5，则输出为

A is 10, B is 5, A/B is 2

2. @WRITEFOR(setname[(set_index_list)[|condition]]:obj1[, objn])

这个函数可以看作是函数@WRITE 在循环情况下的推广，它输出集合上定义的属性对应的多个变量的取值（因此它实际上也是一个集合循环函数）。

3. @ITERS()

这个函数只能在程序的数据段使用，调用时不需要任何参数，总是返回 LINGO 求解器计算所使用的总迭代次数。例如：

```
@TEXT()=@WRITE(' Iterations= ',@ITERS());
```

将迭代次数显示在屏幕上。

4. @NEWLINE(n)

这个函数在输出设备上输出 n 个新行（n 为一个正整数）。

5. @STRLEN(string)

这个函数返回字符串“string”的长度，如@STRLEN(123)返回值为 3。

6. @NAME(var_or_row_reference)

这个函数返回变量名或行名。

例 5.17

model:

sets:

```
warehouses/wh1..wh3/: capacity;
```

```
vendors/v1..v4/: 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));
```

data:

```
capacity=60 55 51;
```

```
demand=35 37 22 32;
```

```
cost=6 2 6 7 4 9 5 3 5 2 1 9;
```

```
@text()=@writefor(links(i,j)|volume(i,j)#gt#0:
```

```
  @name(volume),' ',volume,@newline(1));
```

```
@text()=@write(@newline(1));
```

```
@text()=@writefor(links(i,j)|volume(i,j)#gt#0:
```

-840-

```

    '从仓库',warehouses(i),'到顾客',vendors(j),'供货',volume(i,j),'件',@newline(1));
enddata
end

```

输出结果如下：

```

VOLUME( WH1, V2) 20
VOLUME( WH2, V1) 23
VOLUME( WH2, V4) 32
VOLUME( WH3, V1) 12
VOLUME( WH3, V2) 17
VOLUME( WH3, V3) 22

```

从仓库WH1到顾客V2供货20件
 从仓库WH2到顾客V1供货23件
 从仓库WH2到顾客V4供货32件
 从仓库WH3到顾客V1供货12件
 从仓库WH3到顾客V2供货17件
 从仓库WH3到顾客V3供货22件

注意：从上面的例子还可以看出，“变量”是指“数组元素” volume(wh1,v1)、volume(wh2,v4)等，即属性加上相应的下标（集合元素）。同理，可以想像，约束名也是指模型展开后的约束名（用 LINGO|Generate 命令可以清楚地看到约束展开后的情况），即也应该是带有相应的下标（集合元素）的。

7. 符号 “*”

在@write 和@writefor 函数中，可以使用符号 “*” 表示将一个字符串重复多次，用法是将 “*” 放在一个正整数 n 和这个字符串之间，表示将这个字符串重复 n 次。

例 5.18

```

model:
sets:
    warehouses/wh1..wh3/: capacity;
    vendors/v1..v4/: demand;
    links(warehouses,vendors): cost, volume;
endsets
min=@sum(links: cost*volume);
@for(vendors(J):@sum(warehouses(I): volume(I,J))=demand(J));
@for(warehouses(I):@sum(vendors(J): volume(I,J))<=capacity(I));
data:
    n=3;
    capacity=60 55 51;
    demand=35 37 22 32;
    cost=6 2 6 7 4 9 5 3 5 2 1 9;
    @text()=@write('运输货物数量示意图:',@newline(1));
    @text()=@writefor(links(i,j)|volume(i,j)#gt#0:
        @name(volume),n*' ',volume,volume*'+',@newline(1));
enddata
end

```

程序执行的效果如下：

运输货物数量示意图：

```

VOLUME( WH1, V2) 20+++++++++

```

```

VOLUME( WH2, V1) 23+++++
VOLUME( WH2, V4) 32+++++
VOLUME( WH3, V1) 12+++++
VOLUME( WH3, V2) 17+++++
VOLUME( WH3, V3) 22+++++

```

8. @format(value, format_descriptor)

在@write和@writefor函数中，可以使用@format函数对数值设定输出格式。其中value表示要输出的数值，而format_descriptor（格式描述符）表示输出格式。格式描述符的含义与C语言中的格式描述是类似的，如“12.2f”表示输出一个十进制数，总共占12，其中有2位小数。

例 5.19

```

model:
sets:
    warehouses/wh1..wh3/: capacity;
    vendors/v1..v4/: demand;
    links(warehouses,vendors): cost, volume;
endsets
min=@sum(links: cost*volume);
@for(vendors(J):@sum(warehouses(I): volume(I,J))=demand(J));
@for(warehouses(I):@sum(vendors(J): volume(I,J))<=capacity(I));
data:
    capacity=60 55 51;
    demand=35 37 22 32;
    cost=6 2 6 7 4 9 5 3 5 2 1 9;
    @text('solution.txt')=@writefor(warehouses(i):
    @writefor(vendors(j):@format(volume(i,j),'5.0f')),@newline(1));
enddata
end

```

9. @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。

10. @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。

11. @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 5.20

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

计算结果为:

```
Global optimal solution found at iteration:      0
Objective value:                               -1.000000
0
```

Variable	Value	Reduced Cost
X	4.712388	0.000000
Row	Slack or Surplus	Dual Price
1	-1.000000	-1.000000

结果中的 0 就是@status() 返回的结果，表明最终解是全局最优的。

12. @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）。

5.10 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 5.21 求解最优化问题

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

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,-y)+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 5.22

```
model:
x=1;
@warn('x 是正数',x #gt# 0);
```


end

§ 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。

从窗口菜单中选用“Command Window”命令或直接按 Ctrl+l 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. 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. 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. 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. 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
smpls 将当前模型保存为 MPS 文件

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. 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. 退出系统(Quit)

quit 退出 LINGO 系统

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。

表 4 LINGO 相关参数表

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	OBJECTS	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	整型绝对误差限
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	ORROUTE	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	SOLVEL	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

$$\begin{cases} x^2 + y^2 = 1 \\ 2x^2 + x + y^2 + y = 4 \end{cases}$$

其 LINGO 代码如下:

```
model:
INIT:
x=1;y=1;
ENDINIT
x^2+y^2=2;
2*x^2+x+y^2+y=4;
end
```

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花费时间中的最大值。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。

表 5 任务时间表

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

假如有 11 件任务（A—K）分配到 4 个工作站（1—4），任务的优先次序如图 5。每件任务所花费的时间如表 5。

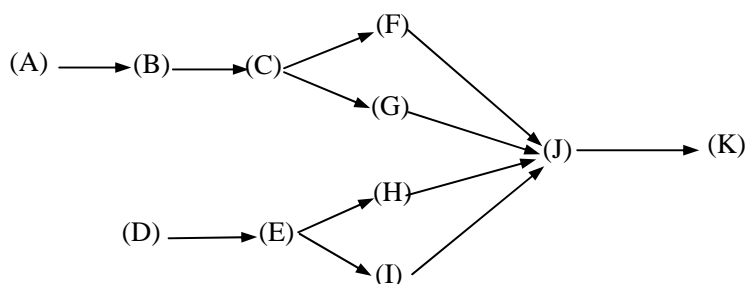


图 5 任务流程图

编写 LINGO 程序如下：

```

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;
    @FOR( TASK(I): @SUM( STATION(K): X(I, K)) = 1);
    !对于每一个存在优先关系的作业对来说，前者对应的工作站 I 必须小于后
    者对应的工作站 J;
    @FOR( PRED(I, J): @SUM( STATION(K): X(I, K) - X(J, 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
  
```

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 $\sum_{i \neq j} c_{ij} x_{ij}$ 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。例如如图 6 的情形，以上两个条件都满足，但它显然不是 TSP 的解，它存在两个子巡回。

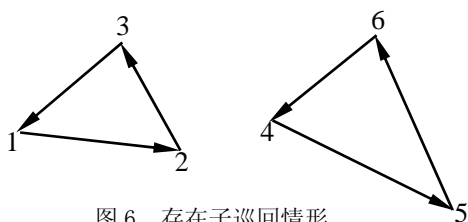


图 6 存在子巡回情形

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（虽然这些变量在最优解中取普通的整数值）。现在附加下面形式的约束条件

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

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 $i_1 i_2 \dots i_k i_1$ ，则必有

$$u_{i_1} - u_{i_2} + n \leq n - 1$$

$$u_{i_2} - u_{i_3} + n \leq n - 1$$

...

$$u_{i_k} - u_{i_1} + n \leq n - 1$$

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 $1i_1 \cdots i_{n-1}1$, 可取

$$u_i = \text{访问城市 } i \text{ 的顺序数, 取值范围为 } \{0, 1, \cdots, n-2\}。$$

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

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

(ii) 非总巡回上的边

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

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。

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

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:

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 $\{0, 1, 2, \cdots, n\}$ 的一个置换 π 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
```

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

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 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图 7，各铲位和各卸点之间的距离（公里）如表 6，各铲位矿石、岩石数量(万吨)和矿石的平均铁含量如表 7。

表 6 铲位和卸点之间的距离表

	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

表 7 铲位矿石、岩石数量和矿石的平均铁含量表

	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%

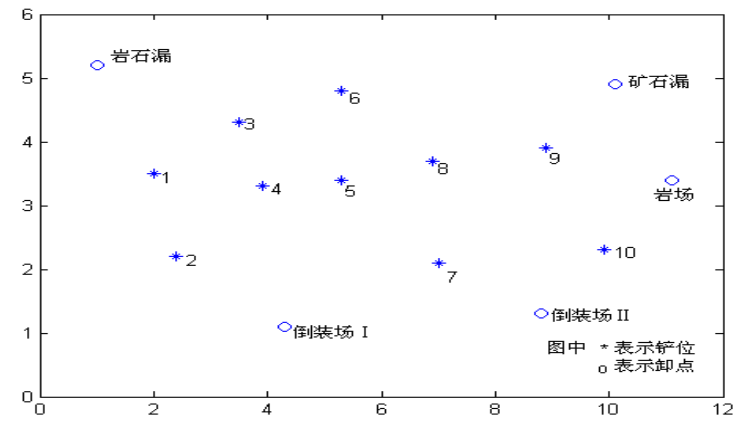


图 7 各个铲位和卸点位置的示意图

各种符号及单位说明如下：

x_{ij} : 从 i 号铲位到 j 号卸点的石料运量, 单位: 车·次;

c_{ij} : 从 i 号铲位到 j 号卸点的距离, 单位: 公里;

T_{ij} : 在 i 号铲位到 j 号卸点路线上运行一个周期平均所需时间, 单位: 分;

A_{ij} : 从 i 号铲位到 j 号卸点最多能同时运行的卡车数, 单位: 辆;

B_{ij} : 从 i 号铲位到 j 号卸点, 一辆车一个班次中最多可以运行次数, 单位: 次;

p_i : i 号铲位的矿石铁含量乘以 100,

$P = (p_1, \dots, p_{10}) = (30, 28, 29, 32, 31, 33, 32, 31, 33, 31)$;

q_j : j 号卸点任务需求,

$Q = (q_1, \dots, q_5) = (1.2, 1.3, 1.3, 1.9, 1.3) \times 10000 / 154$, 单位: 车·次;

ck_i : i 号铲位的铁矿石储量, 单位: 万吨;

cy_i : i 号铲位的岩石储量, 单位: 万吨;

f_i : 描述第 i 号铲位是否使用的 0-1 变量, $f_i = \begin{cases} 1, & \text{使用第 } i \text{ 号铲位} \\ 0, & \text{不使用第 } i \text{ 号铲位} \end{cases}$ 。

建立如下模型

$$\min \sum_{i=1}^{10} \sum_{j=1}^5 154 c_{ij} \cdot x_{ij}$$

$$\text{s. t. } x_{ij} \leq A_{ij} \cdot B_{ij}, \quad i = 1, \dots, 10, \quad j = 1, \dots, 5$$

$$\sum_{j=1}^5 x_{ij} \leq 96 f_i, \quad i = 1, \dots, 10$$

$$\sum_{i=1}^{10} x_{ij} \leq 160, \quad j = 1, \dots, 5$$

$$\left. \begin{aligned} x_{i1} + x_{i2} + x_{i5} &\leq ck_i \times 10000 / 154 \\ x_{i3} + x_{i4} &\leq cy_i \times 10000 / 154 \end{aligned} \right\}, \quad i = 1, \dots, 10$$

$$\sum_{i=1}^{10} x_{ij} \geq q_j, \quad j = 1, \dots, 5$$

$$\left. \begin{aligned} \sum_{i=1}^{10} x_{ij} \times (p_i - 30.5) &\leq 0 \\ \sum_{i=1}^{10} x_{ij} (p_i - 28.5) &\geq 0 \end{aligned} \right\}, \quad j = 1, 2, 5$$

$$\sum_{i=1}^{10} \sum_{j=1}^5 \frac{x_{ij}}{B_{ij}} \leq 20$$

$$\sum_{i=1}^{10} f_i \leq 7$$

x_{ij} 为整数, $i=1,\dots,10$, $j=1,\dots,5$

f_i 为 0-1 变量, $i=1,\dots,10$

编写 LINGO 程序如下:

```
model:
title CUMCM-2003B;
sets:
cai / 1..10 /:p,cy,ck,f;
xie / 1 .. 5 /:q;
link(cai,xie):a,b,c,t,x;
endsets
data:
v=28;
p=30 28 29 32 31 33 32 31 33 31;
q= 1.2 1.3 1.3 1.9 1.3 ;
c=5.2600    1.9000    5.8900    0.6400    4.4200
    5.1900    0.9900    5.6100    1.7600    3.8600
    4.2100    1.9000    5.6100    1.2700    3.7200
    4.0000    1.1300    4.5600    1.8300    3.1600
    2.9500    1.2700    3.5100    2.7400    2.2500
    2.7400    2.2500    3.6500    2.6000    2.8100
    2.4600    1.4800    2.4600    4.2100    0.7800
    1.9000    2.0400    2.4600    3.7200    1.6200
    0.6400    3.0900    1.0600    5.0500    1.2700
    1.2700    3.5100    0.5700    6.1000    0.5000;
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
@for(link:t=120*c/v+8;a=@floor(t/5);b=@floor((485-5*a)/t));
min=@sum( link:x*154*c); !目标函数;
@for (link: x<=a*b); !道路能力约束;
@for (cai(i): @sum(xie(j):x(i,j))<=f(i)*96); !电铲能力约束;
@for (xie(j):@sum(cai(i):x(i,j))<=160); !卸点能力约束;
!以下是铲位储量约束;
@for (cai(i): x(i,1)+x(i,2)+x(i,5)<=ck(i)*10000/154);
@for (cai(i): x(i,3)+x(i,4)<=cy(i)*10000/154);
!产量任务约束;
@for (xie(j) : @sum(cai(i):x(i,j)) >= q(j)*10000/154);
@sum(cai(i): x(i,1)*(p(i)-30.5) )<=0; !铁含量约束;
@sum(cai(i): x(i,2)*(p(i)-30.5) )<=0;
@sum(cai(i): x(i,5)*(p(i)-30.5) )<=0;
@sum(cai(i): x(i,1)*(p(i)-28.5) )>=0;
@sum(cai(i): x(i,2)*(p(i)-28.5) )>=0;
@sum(cai(i): x(i,5)*(p(i)-28.5) )>=0;
@sum(link:x/b)<=20; !车辆能力约束;
@sum(cai: f)<=7; !电铲数量约束;
@for(link : @gin(x)); !整数约束;
@for(cai: @bin(f)); !0-1变量约束;
end
```

例 7.6 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：

$$\begin{aligned} \min \quad & \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^n x_{ij} = 1, \quad j=1,2,\dots,n \\ & \sum_{j=1}^n x_{ij} = 1, \quad i=1,2,\dots,n \\ & x_{ij} = 0,1, \quad i=1,2,\dots,n, \quad j=1,2,\dots,n \end{aligned}$$

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。

编写 LINGO 程序如下：

```
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
min=@sum(links: cost*volume); !目标函数;
!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
```

习题

1. 求解指派问题，其系数矩阵为

$$C = \begin{bmatrix} 16 & 15 & 19 & 22 \\ 17 & 21 & 19 & 18 \\ 24 & 22 & 18 & 17 \\ 17 & 19 & 22 & 16 \end{bmatrix}$$

2. 求解线性规划问题

$$\max Z_1 = 100x_1 + 90x_2 + 80x_3 + 70x_4$$

$$\text{s.t.} \begin{cases} x_1 + x_2 \geq 30 \\ x_3 + x_4 \geq 30 \\ 3x_1 + 2x_3 \leq 120 \\ 3x_2 + 2x_4 \leq 48 \\ x_i \geq 0, i = 1, \dots, 4 \end{cases}$$

3. 一家超市公司准备在某市建立两个分店，向 7 个区的居民经营，每个区的居民人数（单位：千人）已经表示在图 8 上。每个分店只能向本区和一个相邻区的居民经营，这两个分店应该建在何处，才能使所能供应的居民的数量最大？

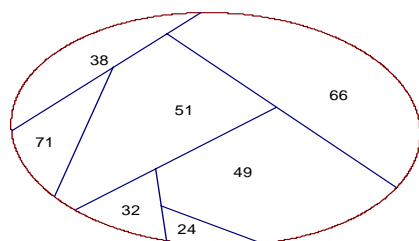


图 8 居民区位置示意图