3.2 Lingo基础知识与实例

3.2.1Lingo基础知识讲解

1.Lingo基本概念

Lingo是Linear Interactive and General Optimizer的缩写即“交互式的线性和通用优化求解器”，由美国LINDO系统公司推出，用于求解非线性规划，及一些线性和非线性方程组的求解等，功能十分强大，是求解优化模型的最佳选择。

特色

其特色在于内置建模语言，提供有十几个内部函数，允许决策变量是整数（即整数规划，包括0-1整数规划），方便灵活，而且执行速度非常快。能方便与Excel、数据库等其他软件交换数据。

2.Lingo函数 ——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提供的与概率相关的函数。

2.Lingo函数 ——Lingo有9种类型的函数

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。

2.Lingo函数——（1）基本运算符

这些运算符是最基本的，或许可以认为它们不是一类函数。事实上，在Lingo中它们是非常重要的。

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

运算符	取反	乘方	乘	除	加	减
符号	-	∧	*	/	+	-

Lingo唯一的一元算术运算符是取反函数

运算符的优先级由高到低为：取反 → 减

运算的次序可以用圆括号“（）”来改变。

2.Lingo函数——（1）基本运算符

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，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flase
#ne#	若两个运算符不相等，则为true；否则为flase
#gt#	若左边的运算符严格大于右边的运算符，则为true；否则为flase
#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flase。

2.Lingo函数——（1）基本运算符

② 逻辑运算符

#lt#	若左边的运算符严格小于右边的运算符，则为true；否则为flase
#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flase
#and#	仅当两个参数都为true时，结果为true；否则为flase
#or#	仅当两个参数都为false时，结果为false；否则为true

运算符优先级

高：　#not#

　　　#eq#　#ne#　#gt#　#ge#　#lt#　#le#

低：　#and#　#or#

字母缩写辅助：

g:greater

e:equal

l:less

n:not

t:than

2.Lingo函数——（1）基本运算符

③关系运算符

在Lingo中，关系运算符主要用在模型中，用来指定一个表达式的左边是否等于、小于等于或者大于、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。

2.Lingo函数——（1）基本运算符

③关系运算符

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：

$$A+\varepsilon<=B$$

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。

2.Lingo函数——（1）基本运算符

以上三类操作符（算术、逻辑、关系）的优先级：

高 ^ #not# －（取反）

* /

\+ -

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

#and# #or#

低 <= = >=

2.Lingo函数——（2）数学函数

@abs(x)：返回x的绝对值

@exp(x)：返回常数e的x次方

@sin(x)：返回x的正弦值，x采用弧度制

@log(x)：返回x的自然对数

@cos(x)：返回x的余弦值

@lgm(x)：返回x的gamma函数的自然对数

2.Lingo函数——（2）数学函数

@tan(*x*)：返回*x*的正切值

@sign(*x*)：如果*x* <0返回-1；否则，返回1

@smax(*x*1,*x*2,…,*xn*)：返回*x*1，*x*2，…，*xn*中的最大值

@smin(*x*1,*x*2,…,*xn*)：返回*x*1，*x*2，…，*xn*中的最小值

@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

2.Lingo函数——（3）金融函数

金融函数主要用于计算净值。包括以下两个函数：

@FPA(I，N)

返回后面情形下的总净现值：单位时段利率为I，从下个时段开始连续N个时段支付，每个时段支付单位费用。计算公式为：

$$@FPA(I,N)=\frac{1}{I}\left(1-\left(\frac{1}{1+I}\right)^{N}\right)$$

@FPL(I，N)

返回后面情形下的净现值：单位时段利率为I，从下个时段开始的第N个时段支付的费用。计算公式为：

$$@FPL(I,N)=\left(\frac{1}{1+I}\right)^{N}$$

2.Lingo函数——（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x^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。

2.Lingo函数——（4）概率函数

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（$n\le$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。

2.Lingo函数——（4）概率函数

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。

2.Lingo函数——（4）概率函数

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。

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。

2.Lingo函数——（5）变量定界函数

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$L \le x \le U$

@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+∞。@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。

2.Lingo函数——（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。

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。

2.Lingo函数——（6）集操作函数

③ @wrap(index,limit)

该函数返回j=index-k*limit，其中 k是一个整数，取适当值保证 j 落在区间[1，limit]内。该函数相当于index模limit。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。

2.Lingo函数——（7）集合循环函数

集合循环函数是指对集合上的元素（下标）进行循环操作的函数。一般用法如下：

@function(set_operator (set_name|condition:expression))

其中set_operator部分是集合函数名（见下），set_name是数据集合名，expression部分是表达式，|condition部分是条件，用逻辑表达式描述（无条件时可省略）.逻辑表达式中可以三种逻辑算符（#AND#（与）,#OR#（或），#NOT#（非））和六种关系算符（#EQ#（等于），#NE#（不等于），#GT#（大于），#GE#（大于等于），#LT#（小于），#LE#（小于等于））。

2.Lingo函数——（7）集合循环函数

常见的集合函数如下：

@FOR (set_name: constraint_expressions)	对集合(set_name)的每个元素独立地生成约束，约束由约束表达式（constraint_expressions）描述.
@MAX(set_name: expression)	返回集合上的表达式(expression)的最大值.
@MIN(set_name: expression)	返回集合上的表达式(expression)的最小值
@SUM(set_name: expression)	返回集合上的表达式(expression)的和
@SIZE(set_name)	返回数据集set_name中包含元素的个数
@IN(set_name, set_element)	如果数据集set_name中包含元素set_element则返回1，否则返回0

2.Lingo函数——（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。

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：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，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，可能就会导致一些问题，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.Lingo函数——（8）输入和输出函数

② @text函数

该函数被用于把数据部分解输出至文本文件中，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）。

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。

2.Lingo函数——（8）输入和输出函数

③ @ole函数　是从Excel中引入或输出数据的接口函数，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。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、自上而下来读的。

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。

2.Lingo函数——（8）输入和输出函数

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	**GlobalOptimum（全局最优）**
1	Infeasible（不可行）
2	Unbounded（无界）
3	Undetermined（不确定）
4	Feasible（可行）
5	InfeasibleorUnbounded（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	LocalOptimum（局部最优）
7	LocallyInfeasible（局部不可行，尽管可行解可能存在，但是Lingo并没有找到）
8	Cutoff（目标函数的截断值被达到）
9	NumericError（求解器因在某约束中遇到无定义的算术运算而停止）

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

2.Lingo函数——（8）输入和输出函数

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（dualprices）。

2.Lingo函数——（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。

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。

3.Lingo注意事项

（1）Lingo中模型以“MODEL：”开始，以“END”结束，对于简单的模型，这两个语句都可以省略；

（2）Lingo中每行后面均增加了一个分号“;”；

（3）所有符号都需在英文状态下输入；

（4）min=函数、max=函数，表示求函数的最小、最大值；

（5）Lingo中变量不区分大小写，变量名可以超过8个，但不能超过32个，需以字母开头；

3.Lingo注意事项

（6）用Lingo解优化模型时已假定所有变量非负，如果想解除这个限制可以用函数@free(x)，这样x可以取到任意实数；

（7）变量可以放在约束条件右端，同时数字也可以放在约束条件左边；

（8）Lingo模型语句由一系列语句组成，每一个语句都必须以“;”结尾；

（9）Lingo中以“!”开始的是说明语句，说明语句也以“;”结束。

3.2.1Lingo实例

1.简单线性规划求解

让我们来看点简单的问题

目标函数最大值 $\max z = 4x_1 + 3x_2$

约束条件 $\begin{cases} 2x_1 + x_2 \le 10 \\ x_1 + x_2 \le 8 \\ x_2 \le 7 \\ x_1, x_2 \ge 0 \end{cases}$

1.简单线性规划求解

如何用lingo实现刚才的问题？

Lingo程序

```
model:
   max=4*x1+3*x2;
   2*x1+x2<10;
   x1+x2<8;
   x2<7;
end
```

注：Lingo中“<”代表“<=”，“>”代表“>=”。Lingo中默认的变量都是大于等于0的，不用显式给出。

求解结果：$z=26$，$x_1=2$，$x_2=6$

2.整数规划求解

$$\text{Max} \quad z = 40x_1 + 90x_2$$

$$\text{Z qzq} \quad \begin{cases} 9x_1 + 7x_2 \le 56 \\ 7x_1 + 20x_2 \le 70 \\ x_1, x_2 \ge 0 \end{cases}$$

再来看看这个问题有什么不一样！

Lingo程序

```
model:
max=40*x1+90*x2;
9*x1+7*x2<56;
7*x1+20*x2<70;
@gin(x1);@gin(x2);
end
```

求解结果：z=340，x_1=4，x_2=2

3.0-1规划求解

0-1规划又该如何做？

$$\text{Max}\quad f = x_1^2 + 0.4x_2 + 0.8x_3 + 1.5x_4$$

$$3x_1 + 2x_2 + 6x_3 + 10x_4 \le 10$$

$$x_1, x_2, x_3, x_4 = 0\text{或}1$$

Lingo程序

```
model:
  Max=x1^2+0.4*x2+0.8*x3+1.5*x4;
  3*x1+2*x2+6*x3+10*x4<10;
  @bin(x1);@bin(x2);
  @bin(x3);@bin(x4);
end
```

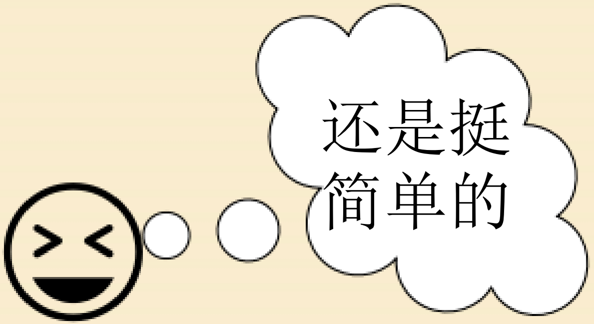
求解结果：f=1.8，x_1=1，x_2=0，x_3=1，x_4=0

4.非线性规划求解

$$\min \quad z = |x_1| - 2|x_2| - 3|x_3| + 4|x_4|$$

$$\begin{cases} x_1 - x_2 - x_3 + x_4 = 0 \\ x_1 - x_2 + x_3 - 3x_4 = 1 \\ x_1 - x_2 - 2x_3 + 3x_4 = -\dfrac{1}{2} \end{cases}$$

4.非线性规划求解



Lingo程序：

```
model:
max=@abs(x1)-2*@abs(x2)-3*@abs(x3)+4*@abs(x4);
x1-x2-x3+x4=0;
x1-x2+x3-3*x4=1;
x1-x2-2*x3+3*x4=-1/2;
@free(x1);@free(x2);
@free(x3);@free(x4);
end
```

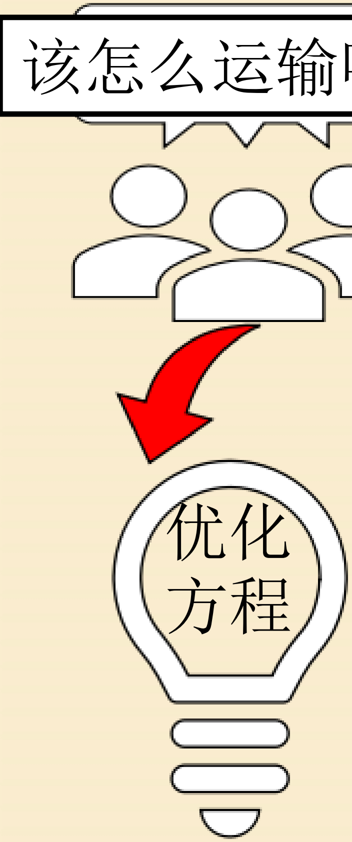
求解结果：$z=1.25$，$x_1=0.25$，$x_2=0$，$x_3=0$，$x_4=-0.25$

5.指派问题

某两个煤厂A1，A2每月进煤数量分别为60t和100t，联合供应3个居民区B1，B2，B3。3个居民区每月对煤的需求量依次分别为50t，70t，40t，煤厂A_1离3个居民区B_1，B_2，B_3的距离依次分别为10km，5km，6km，煤厂A_2离3个居民区B_1，B_2，B_3的距离分别为4km，8km，12km。问如何分配供煤量使得运输量（即t·km）达到最小？

	B1(距离)	B2(距离)	B3(距离)	进煤量(吨)
A1	10	5	6	60
A2	4	8	12	100
煤需求(吨)	50	70	40	160

5.指派问题



设：第 i 煤厂进煤量 s_i，第 j 个居民区需求量为 d_j，

煤厂 i 距居民区 j 的距离为 L_{ij}，煤厂 i 供给居民区 j 的煤量为 g_{ij} .

$$\min = \sum_{j=1}^{3}\sum_{i=1}^{2} g_{ij} \times L_{ij}$$

$$s.t.\begin{cases} \displaystyle\sum_{i=1}^{2} g_{ij} = d_j & (j=1,2,3) \\ \displaystyle\sum_{j=1}^{3} g_{ij} = s_i & (i=1,2) \end{cases}$$

5.指派问题

lingo程序

```
model:
sets:
supply/1..2/:s;
demand/1..3/:d;
link(supply,demand):road,g;
endsets
data:
road=10,5,6,
   4,8,12;
d=50,70,40;
s=60,100;
enddata
min=@sum(link(i,j):road(i,j)*g(i,j));
@for(demand(j):@sum(supply(i):g(i,j))=d(j));
@for(supply(i):@sum(demand(j):g(i,j))=s(i));
end
```

结果：

MIN=940
g(1,1)=0,
g(1,2)=20,
g(1,3)=40,
g(2,1)=50,
g(2,2)=50,
g(2,3)=0

6.图论问题

求出图3-6所示的最小费用和最大流量，以及在最小费用下的最大流量和最大流量下的最小费用。其中（x，y）中x表示流量，y表示费用。

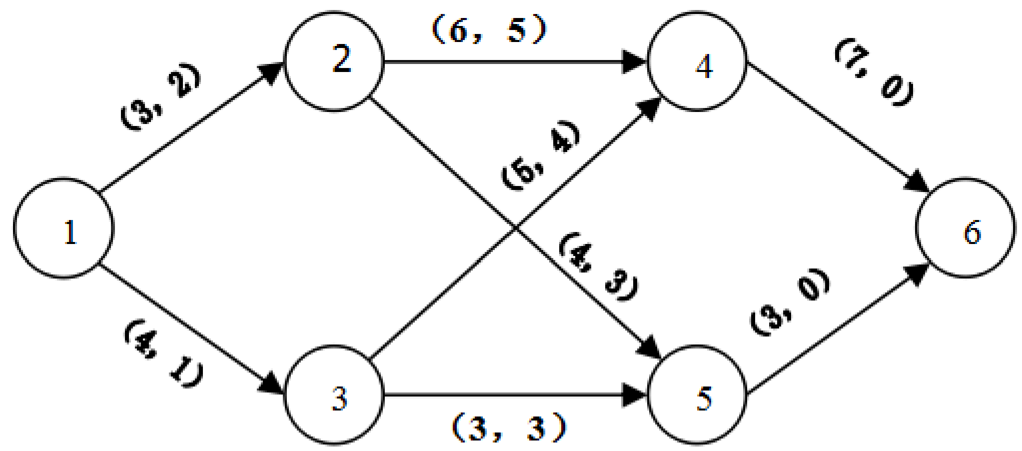


图 3-6

6.图论问题—(1)求最小费用

解法一：邻接矩阵0-1规划法

假设图中有n个原点，现需要求从定点1到n的最短路线线。设决策变量为 f_{ij} ，当 $f_{ij}=1$ 说明弧(i,j)位于定点1至定点n的路线上；否则 $f_{ij}=0$，其数学规划表达式为：

$$\min Z=\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}f_{ij}$$

约束条件，源点1只有一条路线出去，终点n只有一条路线进来，其余各点进去的和出去的路线相等，表达式见右式：

$$s.t.\begin{cases}\sum_{j=1}^{n}f_{1j}=1\\ \sum_{j=1}^{n}f_{jn}=1\\ \sum_{j=1}^{n}f_{ji}=\sum_{j=1}^{n}f_{ij} \qquad i\neq 1,n\end{cases}$$

6.图论问题—(1)求最小费用

解法一：邻接矩阵0-1规划法

lingo程序

```
model:
sets:
node/1..6/;
road(node,node)/1,2,1,3,2,4,2,5,3,4,3,5,4,6,5,6/:w,f;
endsets
data:
w=2,1,5,3,4,3,0,0;
enddata
n=@size(node);
[obj]min=@sum(road(i,j):w(i,j)*f(i,j));
@for(node(i)|i#ne#1#and#i#ne#n:@sum(road(j,i):f(j,i))=@sum(road(i,j):f(i,j)));
@sum(road(i,j)|i#eq#1:f(i,j))=1;
@sum(road(j,i)|i#eq#n:f(j,i))=1;
end
```

6.图论问题—(1)求最小费用

结果： Min=4，f(1,3)=1，f(3,5)=1，f(5,6)=1

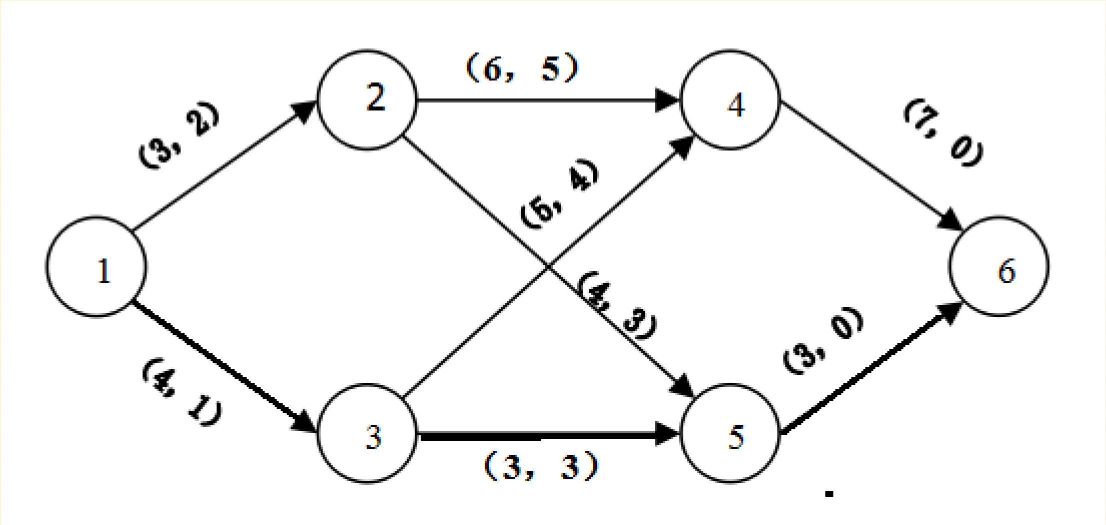


图3.7 结果路径图

6.图论问题—(1)求最小费用

解法二：求源点到任意点的最小费用，动态规划法

求1→6的最小费用，只要求1→4+4→6和1→5+5→6中的最小费用，以同样的方法向上推，求1→4的最小费用只要求出1→2+2→4和1→3+3→4中的最小费用即可。可以归纳出如下的表达式：

$$L(1)=0$$

$$L(i)=\min_{j\neq i}\left(L(j)+c(j,i)\right)\quad i\neq 1$$

6.图论问题—(1)求最小费用

lingo程序

解法二：求源点到任意点的最小费用，动态规划法

```
model:
sets:
node/1..6/:L;
road(node,node)/1,2,1,3,2,4,2,5,3,4,3,5,4,6,5,6/:c;
endsets
data:
c=2,1,5,3,4,3,0,0;
enddata
L(1)=0;
!求一点到任意点的最小费用;
@for(node(i)|i#gt#1:L(i)=@min(road(j,i)|j#ne#i:(L(j)+c(j,i))));
end
```

结果：L(1)=0,L(2)=2,L(3)=1,L(4)=5,L(5)=4,L(6)=4

6.图论问题—(1)求最小费用

解法三：邻接矩阵法

如果$(v_i,v_j)\in E$ 则称 v_i 与v_j邻接，具有n个顶点的图的邻接矩阵是一个$n\times n$阶矩阵其分量为$A=(a_{ij})_{n\times n}$ ，其分量为

$$a_{ij}=\begin{cases}1, & (v_i,v_j)\in E\\ 0, & \text{其他}\end{cases}$$

n个顶点的赋权图的赋权矩阵是一个$n\times n$阶矩阵$\boldsymbol{W}=(w_{ij})_{n\times n}$

其分量为

$$w_{ij}=\begin{cases}w(v_i,v_j), & (v_i,v_j)\in E\\ \infty, & \text{其他}\end{cases}$$

只需将动态规划的条件改一下即可。

$$L(1)=0$$

$$L(i)=\min_{j\neq i}\left[L(j)+a(j,i)\times w(j,i)\right],\quad a(i,j)\neq 0$$

6.图论问题—(1)求最小费用

解法三：邻接矩阵法

lingo程序

```
model:
sets:
node/1..6/:L;
road(node,node):a,w;
endsets
data:
a=0,1,1,0,0,0,
0,0,0,1,1,0,
0,0,0,1,1,0,
0,0,0,0,0,1,
0,0,0,0,0,1,
0,0,0,0,0,0;
w=9,2,1,9,9,9,
9,9,9,5,3,9,
9,9,9,4,3,9,
9,9,9,9,9,0,
9,9,9,9,9,0,
9,9,9,9,9,9;
enddata
L(1)=0; !求一点到任意点的最小费用;
@for(node(i)|i#gt#1:L(i)=@min(road(j,i)|j
#ne#i#and#a(j,i)#eq#1:(L(j)+w(j,i))));
end
```

结果：L(1)=0,L(2)=2,L(3)=1,L(4)=5,L(5)=4,L(6)=4

6.图论问题—(2)求最大流量

邻接矩阵法

同样也可以用三种方法求解，这里只给出邻接矩阵的解法，因为邻接矩阵最容易扩展到多个点，且邻接矩阵用其他的软件非常容易得到。w_{ij}表示最大流量。

$$\max v_f = \sum_{j=1}^{n} f_{1j}$$

$$s.t.\begin{cases} Vf = \sum_{j=1}^{n} f_{1j} \\ \sum_{j=1}^{n} f_{1j} = \sum_{j=1}^{n} f_{jn} \\ \sum_{j=1}^{n} f_{ji} = \sum_{j=1}^{n} f_{ij} \qquad i \neq 1, n \\ f_{ij} \leq w_{ij} \qquad i, j = 1, 2, \ldots, n \end{cases}$$

6.图论问题—(2)求最大流量

```
model:
sets:
node/1..6/;
road(node,node):w,a,f;
endsets
data:
a=0,1,1,0,0,0,0,0,0,1,1,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0;
w=0,3,4,0,0,0,0,0,0,6,4,0,0,0,0,5,3,0,0,0,0,0,0,7,0,0,0,0,0,3,0,0,0,0,0,0;
enddata
n=@size(node);
max=vf;
Vf=@sum(road(i,j)|i#eq#1:f(1,j));
Vf=@sum(road(j,i)|i#eq#n:f(j,n));
@for(node(i)|i#ne#1#and#i#ne#n:@sum(node(j):f(i,j))=@sum(node(j):f
(j,i)));
@for(road(i,j):f(i,j)<=w(i,j));
  end
```

6.图论问题—(2)求最大流量

结果1: Vf=7,F(1,2)=3,F(1,3)=4,F(2,5)=3,F(3,4)=4,F(4,6)=4,F(5,6)=3
结果2: Vf=7,F(1,2)=3,F(1,3)=4,F(2,4)=3,F(3,4)=4,F(4,6)=7

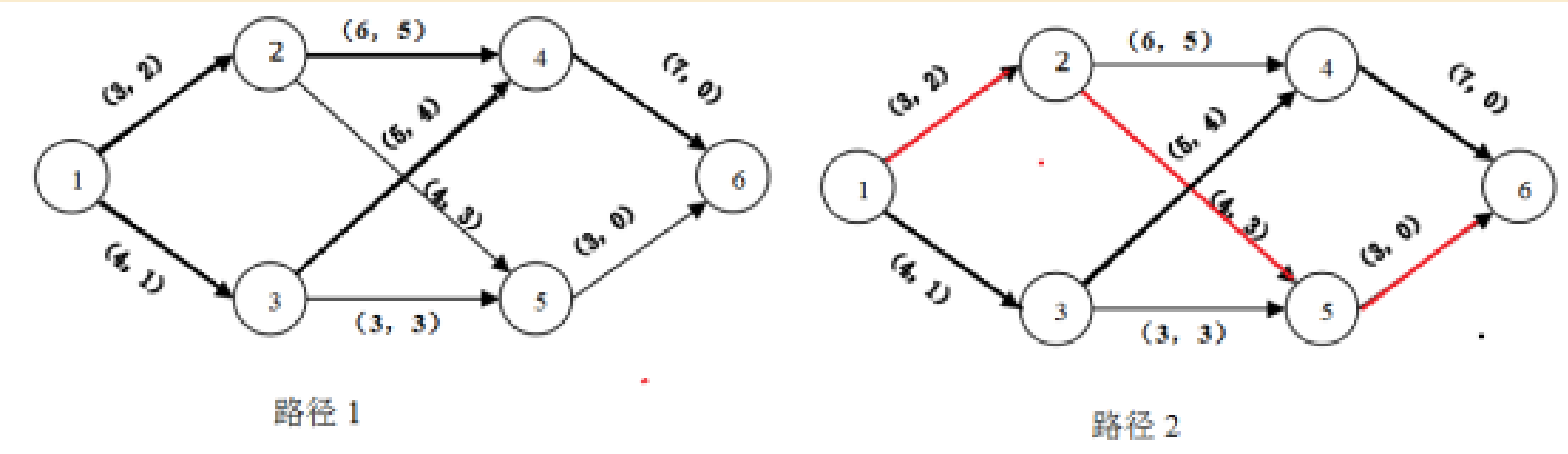


图3.8 结果路径图

结果解释：
路径1：1-->2-->4-->6(流量3)，1-->3-->4-->6(流量4)， 总流量7
路径2：1-->2-->5-->6(流量3)，1-->3-->4-->6(流量4)， 总流量7

6.图论问题—(3)最大流量的最小费用

用上面的方法得到的最大流量的算法只是其中的一种，而不是所有的走法，所以需要找出最优解，其中最小费用或者最短路线径是最常见的两类。

这里求最大流量下的最小费用，先要求出最大流量，然后设流量为已知条件，再求出最小费用就可以了。最大流量用前面的方法已经求出来了，约束条件和上面的一样，这里用 f_{ij} 表示现流量，x_{ij} 表示最大流量，即容量。

目标函数 $$\min Z = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}.sign(f_{ij})$$

6.图论问题—(3)最大流量的最小费用

约束条件，源点流出去的流量是最大流量，终点流进的也是最大流量，其余各点进去的和出去的路线相等，表达式如下：

$$
s.t.\begin{cases}
Vf = \sum_{j=1}^{n} f_{1j} \\
\sum_{j=1}^{n} f_{1j} = \sum_{j=1}^{n} f_{jn} \\
\sum_{j=1}^{n} f_{ji} = \sum_{j=1}^{n} f_{ij} \qquad i \neq 1, n \\
f_{ij} \leq x_{ij} \qquad i, j = 1, 2, \ldots., n
\end{cases}
$$

这里可以由上一问求出vf=7

6.图论问题—(3)最大流量的最小费用

```
model:
sets:
node/1..6/;
road(node,node)/1,2,1,3,2,4,2,5,3,4,3,5,4,6,5,6/:w,x,f;
endsets
data:
w=2,1,5,3,4,3,0,0;!费用;
x=3,4,6,4,5,3,7,3;!容量;
enddata
n=@size(node);
[obj]min=@sum(road(i,j):w(i,j)*@sign(f(i,j)));
@for(node(i)|i#ne#1#and#i#ne#n:@sum(road(j,i):f(j,i))=@sum(road(i,j):f(i,j)));
@sum(road(i,j)|i#eq#1:f(i,j))=7;
@sum(road(j,i)|i#eq#n:f(j,i))=7;
@for(road(i,j):f(i,j)<=x(i,j));
end
```

6.图论问题—(3)最大流量的最小费用

结果：Min=10

F(1,2)=3,F(1,3)=4,F(2,5)=3,F(3,4)=4,F(4,6)=4,F(5,6)=3

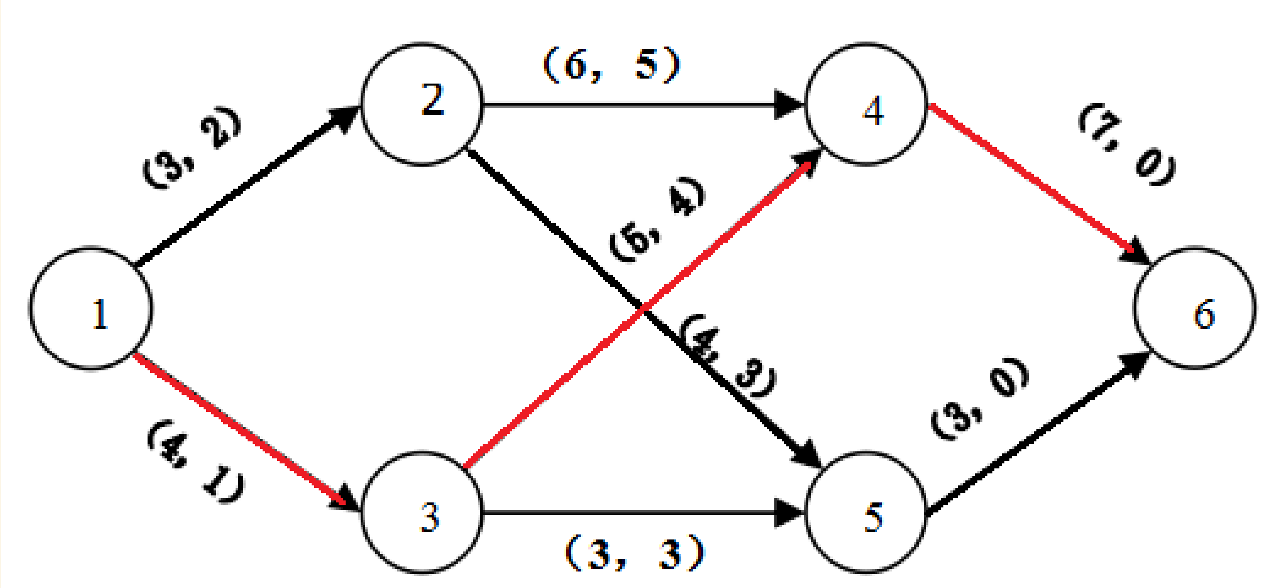


图3.9 结果路径图

结果解释：

路径1：1-->2-->4-->6(流量3)，1-->3-->4-->6(流量4)，总流量7,费用12

路径2：1-->2-->5-->6(流量3)，1-->3-->4-->6(流量4)，总流量7,费用10（最小）

6.图论问题—(4)最小费用下的最大流量

y_{ij} 表示 i 到 j 的费用，s_{ij} 表示 i 到 j 的现流量，x_{ij} 表示 i 到 j 的容量。

有关约束如下：

$s_{ij} \le x_{ij}$ 所有的现流量小于容量。

这里最小费用等于4，有如下约束条件：

$\sum_{i=1}^{n}\sum_{j=1}^{n} sign(s_{ij}).y_{ij} = Z$，sign 为符号函数，由前面得到最小费用 Z=4。

为使模型变为线性规划，引入 0-1 变量 u_{ij}，满足 $u_{ij} = \begin{cases} 1 & s_{ij} > 0 \\ 0 & s_{ij} = 0 \end{cases}$

则有：$u_{ij} \le s_{ij}$，$9u_{ij} \ge s_{ij}$

该约束变为 $\sum_{i=1}^{n}\sum_{j=1}^{n} u_{ij}.y_{ij} = Z$

6.图论问题—(4)最小费用下的最大流量

0-1线性规划模型为：

$$\max v_f = \sum_{j=1}^{n} s_{1j}$$

$$s.t.\begin{cases} \sum_{j=1}^{n} s_{1j} = \sum_{j=1}^{n} s_{jn} \\ \sum_{j=1}^{n} s_{ji} = \sum_{j=1}^{n} s_{ij} & i \neq 1,n \\ \sum_{i=1}^{n}\sum_{j=1}^{n} u_{ij} \cdot y_{ij} = Z \\ s_{ij} \le x_{ij} & i,j = 1,2,\ldots,n \\ u_{ij} \le s_{ij} & i,j = 1,2,\ldots,n \\ 9u_{ij} \ge s_{ij} \\ u_{ij} = 0\text{或}1,\ s_{ij}\text{取整} \end{cases}$$

6.图论问题—(4)最小费用下的最大流量

```
model:
sets:
node/1..6/;
road(node,node):x,y,s,u;
endsets
data:
x=0,3,4,0,0,0,
0,0,0,6,4,0,
0,0,0,5,3,0,
0,0,0,0,0,7,
0,0,0,0,0,3,
0,0,0,0,0,0;!容量矩阵;
y=9,2,1,9,9,9,
9,9,9,5,3,9,
9,9,9,4,3,9,
9,9,9,9,9,0,
9,9,9,9,9,0,
9,9,9,9,9,9;
enddata
!没有直接相连的费用采用充分大数9表示;
n=@size(node);
max=vf;
vf=@sum(node(j):s(1,j));
@for(node(i)|i#gt#1#and#i#ne#n:@sum(node(j):
s(i,j))=@sum(node(j):s(j,i)));
@sum(road(i,j):u(i,j)*y(i,j))=4;
@for(road(i,j):s(i,j)<=x(i,j));
@for(road(i,j):u(i,j)<=s(i,j));
@for(road(i,j):9*u(i,j)>=s(i,j));
@for(road:@gin(s));
@for(road:@bin(u));
  end
```

6.图论问题—(4)最小费用下的最大流量

结果：MAX=3

S(1,3)=3,S(3,5)=3,S(5,6)=3

路径：1-->3-->5-->6 （流量3）

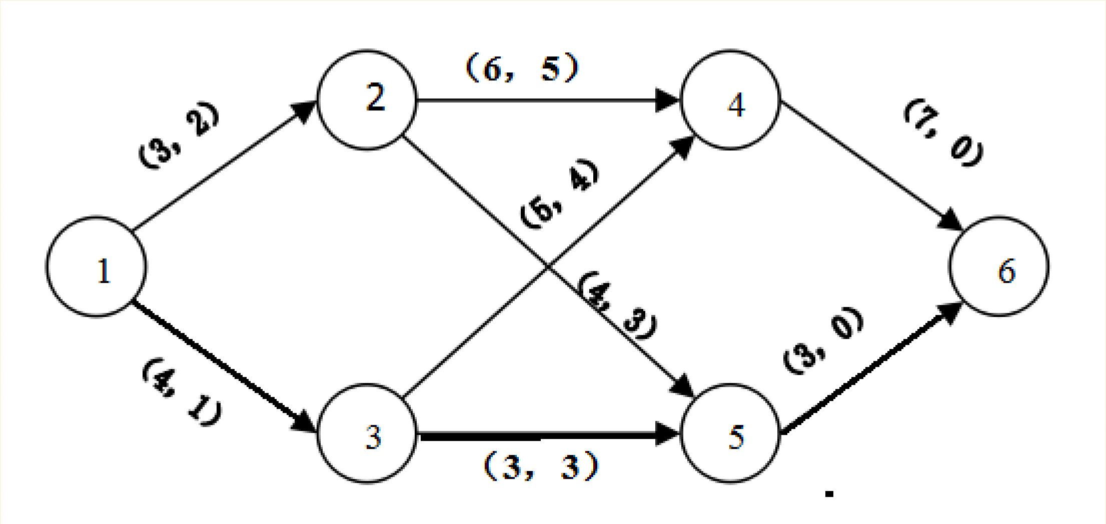


图3.10 结果路径图

谢 谢！