LINGO优化

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。

我们关注近几年全国赛赛题的同学们都会发现，优化问题始终是数学建模的热点，近几年整数规划、二次规划的问题多次出现。优化问题往往有建模简单，求解困难的特点，如何找到我们所需要的全局最优解或者局部最优解是非常重要的，Lingo是我们完成优化建模求解的有效工具，它的学习直接关系到了我们建模的最终成败。

我其实也是一个Lingo的初学者，还只是对Lingo做了初步的了解，因为我感觉到它其实还是非常博大的，因为Lingo让我体会到了解决实际问题的兴奋，体会到了面向对象编程思想对数学的意义。

甚至我用Lingo赚到了钱，呵呵，大家不要小看它呀！

首先我先说说我学习Lingo的三个最大的体会：

1、Lingo中最重要的概念是“集”。可以说真正能用“集”的思想去建模，你才真正把计算机和数学融为一体了，因为“集”是计算机中的面向对象编程思想的体现。

2、一定要会用@for和@sum两个函数。因为在优化模型中，通常都会有很多的决策变量和约束条件，这两个函数不会用，那你的模型几乎很难放在Lingo中。

3、一定要能看得懂求解结果。复杂的优化问题并不能保证得到全局最优解，Lingo有的时候也无能为力，我们不能完全依赖它，有的时候还要帮它绕过一些困难。另外灵敏性分析的结果也非常重要，这些在Lingo的结果报告中可以给出。

好的，我们先来谈谈“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）。一旦把对象聚合成集，就可以利用集来最大限度的发挥LINGO建模语言的优势。

其实，我理解Lingo中的“集”和软件工程中的“类”的概念是一致的。

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。

我认为集的引入让Lingo可以从一个优化软件升级为一门面向对象的建模语言。

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。

如果我们把“集”看成是“类”的话，“属性”就是类的一个“实例”。

集和类一样也有继承的特点，在Lingo中我们叫做派生。

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。

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。

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。

我们可以使用Mon..Fri来表示Mon,Tue,Wed,Thu,Fri，使用Oct2001..Jan2002来表示Oct2001,Nov2001,Dec2001,Jan2002。是不是非常方便呢？

成员列表是集的一个取值的空间，就像对象的域一样。

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。集属性的值一旦在模型中被确定，就不可能再更改。

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。

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。

我们来看一个例子：

例2

```
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
```

这个例子能让我们学到很多有关Lingo的知识，首先我们必须知道用！和；括起来的部分是Lingo的注释部分，我们都知道一个好的程序员首先要会写注释，在lingo中也是一样，我在网上看到很多同学求助lingo程序，但是他们的程序都没有注释，根本无法看懂，别人也就无法帮你了！

#eq#，#ge#，#and#都是Lingo的逻辑运算符，详细地大家可以参见下表：

#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

#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

这些运算符的优先级由高到低为：

高　#not#

　　#eq#　#ne#　#gt#　#ge#　#lt#　#le#

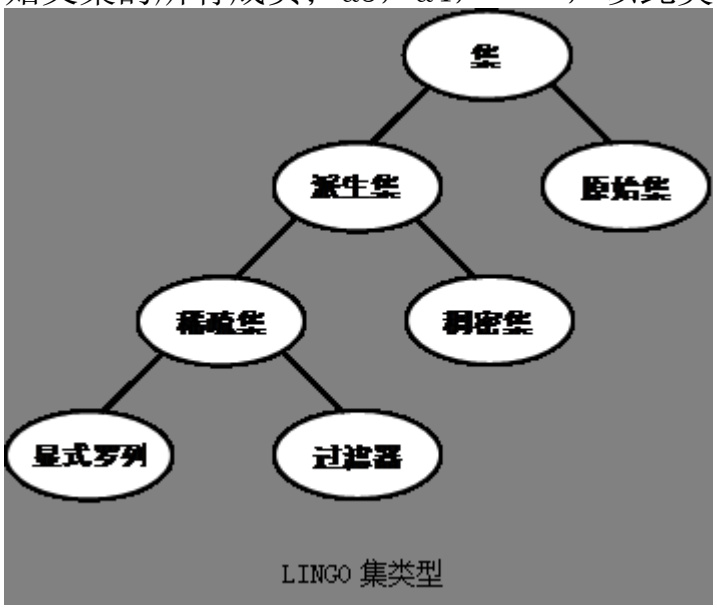
低　#and#　#or#

这里linkmf就是由students派生出来的集。linkmf2又是派生于linkmf的集。

linkmf是由两个students集派生的，呵呵，这里面linkmf升级为了一个二维的东西，矩阵终于出现了，是不是还有很多同学在困惑矩阵如何在lingo中表示呢？其实就是一个简单的派生而已。

用竖线（|）来标记一个成员资格过滤器的开始，它可以用逻辑表达式来限制成员，这样我们就可以方便地使用逻辑表达式来构造一个稀疏集。

&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。



上面是有关集的一些内容，虽然有一些粗略，但是毕竟这种建模思想的提高不是一节课可以解决的问题，好在在数学中国论坛里面有很多网友发了一些实际问题的Lingo原代码，建议网友下载一些仔细研究这些例子，你就会有建模感觉的。

接下来我们来接触两个Lingo的函数，虽然Lingo的函数并不是很多，但是这两个的使用频率非常高，有人说会了这两个函数的使用，你基本上就可以用Lingo编程了，我觉得说的没错，我的Lingo学习也是从这两个函数开始的。

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。

注意@for的用法分成两个部分，：号前面是一个集，也可以用竖线（|）来标记一个成员资格过滤器来用逻辑表达式来限制成员。

：号后面是一个约束表达式，对：号前面集中的每个成员约束表

达式都要成立。

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 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
```

注意@sum的用法也分成两个部分，：号前面是一个集，也可以用竖线（|）来标记一个成员资格过滤器来用逻辑表达式来限制成员。

：号后面是集的一个属性的表达式。

$$
\begin{cases}
\min \quad z=\sum_{\substack{i,j=1\\ i\neq j}}^{n} c_{ij}x_{ij} \\
s.t. \quad \sum_{i=1}^{n} x_{ij}=1, \qquad j=1,2,\cdots,n \\
\qquad \sum_{j=1}^{n} x_{ij}=1, \qquad i=1,2,\cdots,n \\
\qquad u_i-u_j+n\,x_{ij}\le n-1, \quad 2\le i\ne j\le n \\
\qquad x_{ij}=0,1, \qquad i,j=1,2,\cdots,n \\
\qquad u_i\ge 0, \qquad i=2,3,\cdots,n
\end{cases}
$$

我们用@for和@sum的组合就可以在Lingo中描述这个模型，代码如下：

```
 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);
      );
```

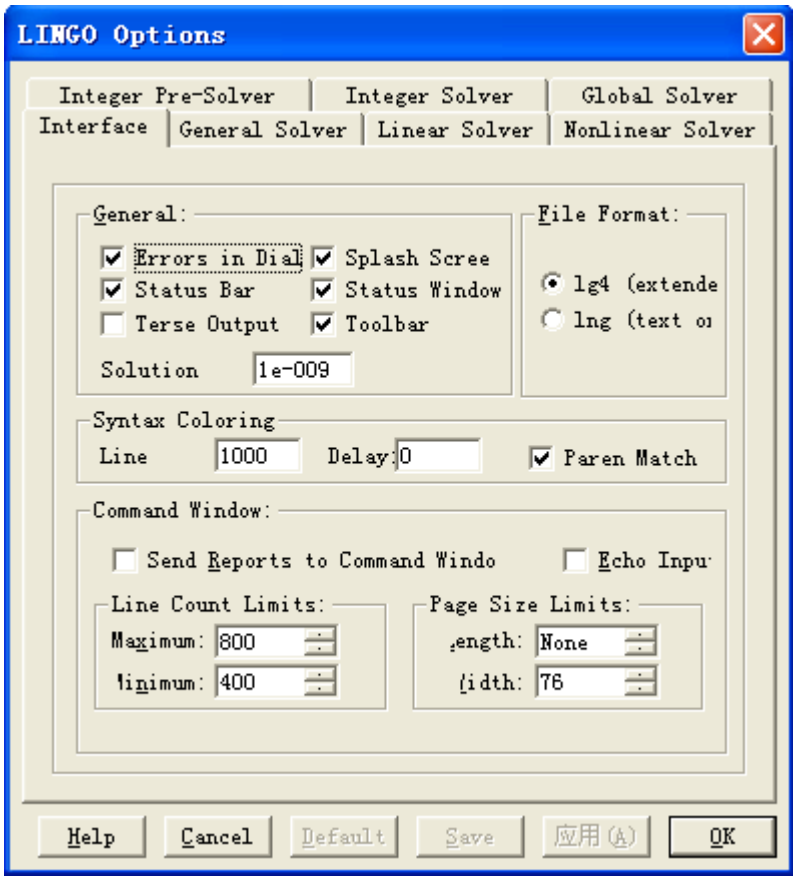
!限制u的范围以加速模型的求解，保证所加限制并不排除掉TSP问题的最优解;

```
      @for(city(I) | I #gt# 1: u(I)<=n-2 );
      !定义X为0\1变量;
      @for( link: @bin( x));
    End
```

上面定义所有变量为0-1变量的@for语句很有用，有一些网友在网上提过这个问题。

最后我们了解一下Lingo的必要设置

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。如下图：



在Global Solver（全局最优求解器）选项卡中你可以设置Use Global Solver 来使用全局最优求解程序，好多网友是不是还在困惑为什么我的全局求解器总是不工作呢？

另外你可以设置一些求解器的求解方法和求解迭代步数，这对一些较难求解的问题是非常有用的。

最后我们关注一下求解器状态窗口（LINGO Solver Status）

这里面我们要了解State的含义："Global Optimum"（全局最优解），"Local Optimum"（局部最优解），"Feasible"（可行解），"Infeasible"（不可行），"Unbounded"（无界），"Interrupted"（中断），"Undetermined"（未确定）

Solver Type：B-and-B （分枝定界法），Global （全局最优求解），Multistart（用多个初始点求解）

Steps：特殊求解程序当前运行步数：

分枝数（对B-and-B程序）；

子问题数（对Global程序）；

初始点数（对Multistart程序）

Active：有效步数。

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。

好了，由于很多网友晚上要断电，所以我们的课先讲到这，大家可以在数学中国社区论坛上和我继续交流，谢谢大家了！