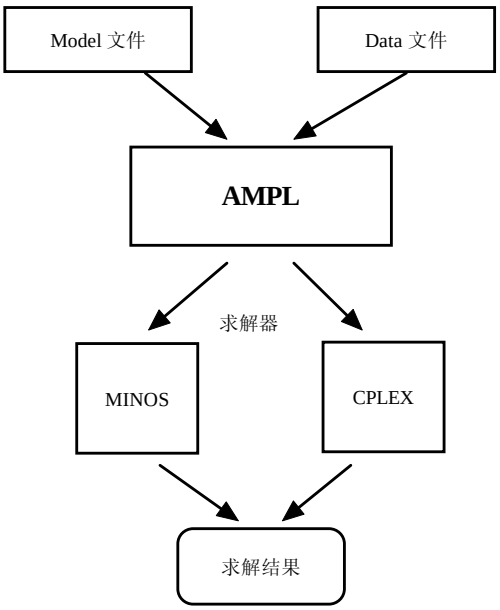
AMPL简介

March 4, 2008

1 什么是AMPL

AMPL——-"A Mathematical Programming Language", 是一种强大灵活的综合性代数模型语言，可以解决优化过程中经常遇到的线性、非线性和整型数学规划问题, 由朗讯公司(Lucent Technologies) 的研发部门贝尔实验室(Bell Laboratories)开发. AMPL提供直观简明的代数符号用以描述复杂的模型. AMPL软件是付费的, 不过可以使用免费的学生版. http://www.ampl.com/NEW/TABLES/amplcml.zip 其中包括了学生版的CPLEX 等求解器. 本节简要介绍AMPL的基本概念和用法,主要针对线性规划问题. 详细完整资料见参考书[]: A Modeling Language for math Programming, 作者Robert Fourer, David M.Gay和Brian W.Kernighan.

AMPL解决数学规划问题的框架见下图.



2 开始运行AMPL

运行AMPL之前需要两个文件.

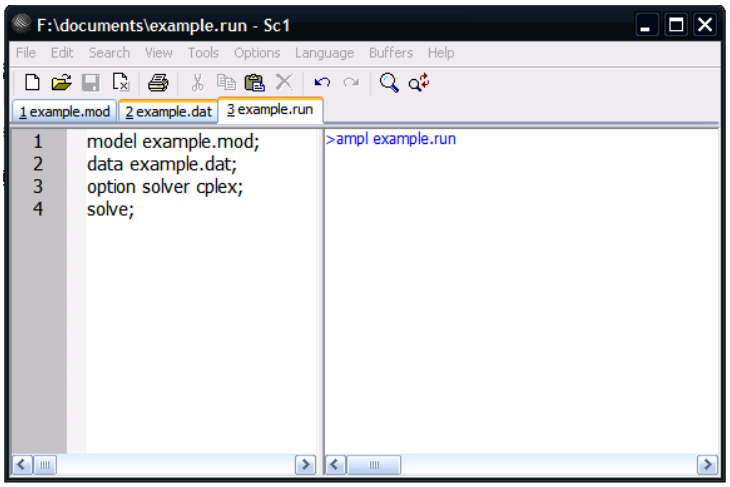
- 模型文件: 用AMPL语言编写问题的模型(model)文件, 模型文件以".mod"为扩展名.模型文件中一般包括: 集合(set), 参数(parameter),变量(variable),目标函数(objective function), 约束(constraint).
- 数据文件: 模型文件中的集合和参数的具体取值数据可以放在模型文件中, 也可以放在单独的数据文件, 这样同一个模型可以适用不同的数据. 数据文件文件以".dat"为扩展名.

也可以编写一个包含求解模型的一系列顺序执行的AMPL命令的批处理文件, 以".run"为扩展名. 下面是一个批处理文件例子:

```
model example.mod;
data example.dat;
option solver cplex;
solve;
```

AMPL自身并不是一个求解程序, 其作用只是类似编译器,它将模型.mod 转换成专门的.nl 文件. 在读入模型文件和数据文件后然后调用其他能够求解各种数学规划问题的求解器(solver), 语句"model example.mod"使AMPL读入模型文件example.mod. 语句"data example.dat"使AMPL读入数据文件example.dat. 语句"option solver cplex" 指定求解器为CPLEX, 目前AMPL支持世界上大多数流行的求解器,如MINOS, LP_SOLVE等.

模型文件,数据文件和批处理文件可以用任何文本编辑器编辑. 推荐使用下图显示的Scite编辑器(开源), 编写和调试比较方便. 在Scite左边输入区编辑, 在右边输出区运行AMPL.



也可以在AMPL命令提示符下输入

ampl:eample.run

如果没有批处理文件, 也可以一个一个命令输入:

```
ampl: model example.mod;
ampl: data example.dat;
ampl: option solver cplex;
ampl: solve;
```

如果要改变求解器CPLEX的默认设置,重新设置CPLEX的某些选项, 比如求解时间限定为1秒, 用命令:

```
ampl: option cplex_option 'time=1';
```

3 AMPL基本语法

AMPL的一些语法规则:

- 区分大小写
- 忽略空格
- 每个语句以分号";"结尾
- 以#开始的一行或以 之间为注释部分
- :=表示赋值, =表示约束, ==表示判断.

下面介绍AMPL主要的关键词及其用法.

3.1 模型文件

以线性规划为例说明. 线性规划的一般形式为

$$\begin{aligned} \min \ & \sum_{i\in I} c_i x_i \\ s.t \ & \sum_{i\in I} a_{ij} x_i \le b_j \quad \forall j \in J \end{aligned}$$

这里, c_i, a_{ij}, b_j称为**参数(parameters)**, x_i是**变量(variables)**, I和J称为**集合(sets)**, $\sum_{i\in I} c_i x_i$称为**目标函数(objective)**, $\sum_{i\in I} a_{ij} x_i \le b_j$称为**约束(constraints)**.

AMPL 用接近数学描述的语句把数学规划问题写为一个模型文件, 以.mod为扩展名,如example.mod.下面是相应的关键词声明.

- 集合,参数,变量
  set
  param
  var
- 目标函数
  maximize
  minimize
- 约束
  subject to

3.1.1 集合

集合是与变量和约束相联系的对象的全体. 如变量的下标集合. 集合在数据文件中赋值. 较简单的集合也可以在模型文件中赋值. 声明集合的语法为

set [*set name*][*set expression*]

其中[set name]是必须的.下表中的可选关键字可以用来描述集合

关键字	含义
dimen n	定义集合的维数为n
within (*setexp*)	(*setexp*)的子集
:= (*setexp*)	定义集合元素
default (*setexp*)	定义集合的默认值
ordered	有序集合

其中*setexp*为集合表示式, 外面用大括号. 例如:

```
set A;
set B :=1..5;          #B={1,2,3,4,5}
set C := {1,2,3,4,5,6};
set D :=i..j by k;
set E :={i in C:x[i] in A};
set F within interval [i, j);
set G := D diff C;
set H ordered;
```

二维或多维集合的声明, 例如:
set S:=P cross Q; # cross 为集合的笛卡尔积.
set S:= {i in P, j in Q:i<j}

对于类似"for all $i \in \mathcal{P}$ 或"for t=1,..., T"的语句AMPL表示为其指标集{i in P}, {t in 1..T},下标表达式外面用花括号. 也可以表示多个下标,以及下标满足的条件.如{i in P, j in Q:i<>j}.

3.1.2 参数

参数是需要在模型中被赋值的数据. 其赋值在数据文件中给出. 参数可以是标量,向量,或矩阵.其语法为

param [*name*]{*index1, index2,...*}{attributes};

其中[name]是必有的,{index1}..是可选的,说明参数数据的大小或范围, {attributes}说明参数具有的属性, 如可能的取值等等.下表列出了参数的属性及其含义.

关键字	含义
binary	参数取值0或1
integer	参数取值为整数
symbolic	参数为符号形式
< <= = != > >= (exp.)	参数必须满足某表达式
default (*exp.*)	参数的默认值
in (*setexp*)	参数属于集合*setexp*

例如:
param $T > 0$ integer;
param c{I,J}; #c_{ij}
param $Y >= 1$ default 5; #Y默认为5

3.1.3 变量

变量是模型求解需要求出的量, 即决策变量. 变量的声明类似与参数. 其语法为

var [*name*]{*index1, index2,...*}{attributes};

下表列出了参数的属性及其含义

关键字	含义
binary	变量取值为0或1
integer	变量取值为整数
:=	变量初始估计值
<=>==	变量满足的表示式

例如:
var x{p, 1..T}>= 0, <=M;

3.1.4 目标函数

线性规划的目标函数的语法为

maximize [*objective name*]: **sum** {[index] in [index set]}[parameter]*[variable]+...;

如果是最小化目标函数用关键字**minimize**

3.1.5 约束

一般的线性约束的语法为

subject to[*constraints name*] *index in index set*:
sum {[**index**] **in** [*index set*]}
[*parameter*]*[*variable*]+...+{<=, >=, =}[*parameter*]

3.2 数据文件

模型文件里出现的集合和参数如果没有赋值, 则需要在数据文件里赋值. 数据文件也是文本文件,以.dat作为扩展名,如example.dat. 数据文件的一般形式为

data;
set [*set name*] **:=** [*item* 1], [*item* 2],...;
param [*parameter name*] **:=** [*value*1]...;
end;

3.2.1 set的赋值

简单的集合直接列出所有元素,如
set P:=a,b;

若P是有序集合, 则
set P[1]=a;
set P[2]=b;

或
set P:=(1,a)(2,b);
或列表

3.2.2 param的赋值

param的值可以是标量,向量,矩阵.

1. **标量:**如参数a的值为100, 赋值格式为:

   param a:=100;

2. **向量:**如向量**v**的值为$\mathbf{v}_1 = 10$,$\mathbf{v}_2 = 100$, 赋值格式为:

   param v:=1 10 2 100;

3. **矩阵:**矩阵的赋值采用列表的方式, 例如: 矩阵

$$A = \begin{bmatrix} 10 & 11 & 12 \\ 13 & 14 & 15 \end{bmatrix}$$

   赋值格式为:

```
param A: 1    2    3 :=
1      10  11  12
2      13  14  15;
```

   或使用转置符号"tr", 适用于较宽的矩阵

```
param A (tr): 1 2 :=
1    10  13
2    11  14
3    12  15
```

3.3 显示,输入和输出

如果需要显示有关解的信息,比如决策变量Var的值, 可以用命令:

```
ampl: display Var;
```

如果要保存结果为sol.txt,用命令:

```
ampl: display Var > sol.txt;
```

对于线性规划, 变量Var的简约费用(reduced cost)和对偶价格(dual price)的显示分别为

ampl: display Var.rc;

ampl: display Var.dual;

从外部读入一个数据文件input.txt, 并把其值赋值给参数par, 可以用命令:

ampl: read par < input.txt;

如果需要通过键盘输入给参数赋值, 用命令:

ampl: read par< −;

另外, printf语句输出字符串到屏幕.例如

ampl: let year:=2008; printf `"The year is %i.\n",year`;
显示结果为: The year is 2008

ampl: let mypi=3,14; printf `"I like mypi= %g.\n",my_pi`;
显示结果为: I like mypi=3.14
expand constr展开名为constr的所有约束.

3.4 数学运算与函数

运算符: `+,-,*,/,^`分别为加,减,乘,除,乘方. less为非负减(若差小于0,则返回0),mod为同余.
迭代运算符: sum求和, prod求积, exits, forall.
abs(x),acos(x),acosh(x),asinh(x),atanh(x),ceil(x),cos(x),exp(x), floor(x), log(x),log10(x),sin(x),sqrt(x),tan(x),tanh(x). 都是通常意义下的数学函数.
Uniform01()或Uniform(0,1)返回一个0,1之间的(伪)随机数.
ctime()返回字符串表示的当前时间.
与集合有关的函数:
card(set):返回集合set的元素个数.
cross(A,B): A,B的笛卡尔积
diff: 集合差运算. 如"let S := P diff {Q,R}"表示$S = P\backslash(Q \cup R)$
first(set):返回集合set的第一个元素.
in: 属于$\in$.
inter: 集合交运算."A inter B" 表示$A \cap B$.

last(set): 返回集合set的最后一个元素.
member(j,set): 返回集合set的第j个元素.
max(set): 返回set中的最大者.
min(set): 返回set中的最小者.
next(t,S,n): 返回位于集合S中的元素t之后的第n个元素.
ord(t,S): 返回元素t在集合S中的位置序号.
ord0(t,S): 返回元素t在集合S中的位置序号,若$t \notin S$则返回0.
prev(t,S,n): 返回位于集合S中的元素t之前的第n个元素.
union: 集合的并
within: 集合的包含于.

3.5 编程语句

reset:清除已有的数据和模型.
reset data: 清除已有的数据.
let: 赋值给参数或变量.如let{i in I} c[i]:=d[i].
for,repeat,while,until,break: 用于循环语句. 例:
for {i in I}{执行语句},
repeat {执行语句} while {条件语句},
repeat {执行语句} until {条件语句},
if ...then.. else: 条件语句.
and, or: 逻辑与, 逻辑或.