Pesquisa Operacional
Aula 05/02/2020
Xpress Mosel

Xpress Mosel

- Ao abrir o software, clique em File → New
- Aparecerá então uma janela pedindo que você dê nome a este novo arquivo que você vai criar. Escolha um nome e clique em Salvar.
- Abrirá uma outra janela com o conteúdo a seguir.
- Apague o conteúdo e escreva o programa do slide do Primeiro exemplo. Execute o programa clicando na setinha verde, na barra de ferramentas.

```
!@encoding CP1252
model ModelName
uses "mmxprs"; ! Biblioteca para resolver problemas lineares

!optional parameters section observe que o comentário é com exclamação
parameters
!  SAMPLEPARAM1='c:\test\'
!  SAMPLEPARAM2=false
   PROJECTDIR='' ! for when file is added to project
end-parameters

!sample declarations section
declarations
! ...
  Objective:linctr
end-declarations

if PROJECTDIR <> '' then
  setparam('workdir', PROJECTDIR)
  writeln("Project directory: " + PROJECTDIR)
end-if

writeln("Begin running model") observe que tudo que está entre aspas é texto
!...
writeln("End running model")

end-model
```

Xpress Mosel

- Vamos iniciar a apresentação da sintaxe do Xpress Mosel com o exemplo a seguir
- Minimizar $-5x_1 - 2x_2$

  Sujeito a: $x_1 + x_2 \leq 4$

  $x_1 \leq 3$

  $x_2 \leq 2$

  $x_1 , x_2 \geq 0$

Primeiro exemplo

```
model exemplo_livro
  uses "mmxprs"
  declarations
    ! ...
    x1,x2: mpvar
  end-declarations
  x1 + x2 <=4
  x1<= 3
  x2 <= 2
  minimize(-5*x1-2*x2)

  writeln("Begin running model")
  !...
  writeln("x1: ", getsol(x1))
  writeln("x2: ", getsol(x2))
  writeln("FO: ", getobjval)
  writeln("End running model")
end-model
```

Bibliotecas Xpress

- mmive : possui rotinas para desenhar grafos;
- mmquad : possui rotinas para resolver problemas quadráticos;
- mmodbc : possui rotinas para conectar o Xpress a arquivos tipo planilhas e bancos de dados
- mmnl : possui rotinas para resolver problemas não lineares

Primeiro exemplo

```
model exemplo_livro
  uses "mmxprs"
  declarations
    ! ...
    x1,x2: mpvar
  end-declarations
  x1 + x2 <=4
  x1<= 3
  x2 <= 2
  minimize(-5*x1-2*x2)

  writeln("Begin running model")
  !...
  writeln("x1: ", getsol(x1))
  writeln("x2: ", getsol(x2))
  writeln("FO: ", getobjval)
  writeln("End running model")
end-model
```

```
Begin running model
x1: 3
x2: 1
FO: -17
End running model
```

Xpress Mosel

- Mosel User Guide pág 7
- variáveis de decisão são declaradas no bloco declarations/end-declarations;
- mpvar variáveis são restritas a serem não negativas;
- Ao declarar uma variável x como mpvar, você pode obter o seu custo reduzido utilizando a função getrcost(x) e pode obter o seu valor ótimo através da função getsol(x);

Xpress Mosel

- Pode-se definir as restrições como linctr para se obter acesso, por exemplo, ao seu valor dual e também ao valor da folga através das funções getdual(*) e getslack(*)

Primeiro exemplo – segunda forma de modelar no Xpress

```
model exemplo_livro
  uses "mmxprs"
  declarations
    ! ...
    x1,x2: mpvar
    restr1,restr2,restr3, objective: linctr
  end-declarations
  restr1:= x1 + x2 <=4
  restr2:= x1<= 3
  restr3:= x2 <= 2
  objective:=-5*x1-2*x2
  minimize(objective)
  writeln("Begin running model")
  !...
  writeln("x1: ", getsol(x1))
  writeln("x2: ", getsol(x2))
  writeln("FO: ", getobjval)
  writeln("folga prim restr= ", getslack(restr1))
  writeln("folga seg restr= ", getslack(restr2))
  writeln("folga ter restr= ", getslack(restr3))
  writeln("End running model")
end-model
```

```
Begin running model
x1: 3
x2: 1
FO: -17
folga prim restr= 0
folga seg restr= 0
folga ter restr= 1
End running model
```

```
model exemplo_livro
  uses "mmxprs"
  declarations
    ! ...
    x1,x2: mpvar
    restr1,restr2,restr3,objective: linctr
  end-declarations
  restr1:= x1 + x2 <=4
  restr2:= x1<= 3
  restr3:= x2 <= 2
  objective:=-5*x1-2*x2
  minimize(objective)

  writeln("Begin running model")
  !...
  writeln("x1: ", getsol(x1))
  writeln("x2: ", getsol(x2))
  writeln("FO: ", getobjval)
  writeln("folga prim restr= ", getslack(restr1))
  writeln("folga seg restr= ", getslack(restr2))
  writeln("folga ter restr= ", getslack(restr3))
  writeln("valor dual prim restr= ", getdual(restr1))
  writeln("valor dual seg restr= ", getdual(restr2))
  writeln("valor dual ter restr= ", getdual(restr3))
  writeln("End running model")
end-model
```

```
Begin running model
x1: 3
x2: 1
FO: -17
folga prim restr= 0
folga seg restr= 0
folga ter restr= 1
valor dual prim restr= -2
valor dual seg restr= -3
valor dual ter restr= 0
End running model
```

Segundo exemplo

Minimizar $4\,x1 + x2 + x3 + 3\,x4$

Sujeito a: $3\,x_1 + 4\,x_2 + x_3 + 2\,x_4 \geq 13$

$2\,x_1 + 3\,x_2 + 5\,x_3 + 5\,x_4 \geq 16$

$x_1, x_2, x_3, x_4 \geq 0$

Segundo exemplo

```
model exemplo2
uses "mmxprs";
declarations
! ...
  x1,x2,x3,x4 : mpvar
end-declarations
2*x1 + 3*x2 + 5*x3 + 5*x4 >= 16
3*x1 + 4*x2 + x3 + 2*x4 >= 13
minimize(4*x1 + x2 + x3 + 3*x4)
writeln("Begin running model")
!...
 writeln("x1: ", getsol(x1))
 writeln("x2: ", getsol(x2))
 writeln("x3: ", getsol(x3))
 writeln("x4: ", getsol(x4))
 writeln("FO: ", getobjval)
writeln("End running model")

end-model
```

```
Begin running model
x1: 0
x2: 2.88235
x3: 1.47059
x4: 0
FO: 4.35294
End running model
```

Xpress Mosel

- Podemos declarar vetores para simplificar, por exemplo

  x1, x2, x3, x4

  fazendo:

  quant_prod = 1..4

  x : array(quant_prod) of mpvar

```
model exemplo2
uses "mmxprs";
declarations
! ...
  quant_prod = 1..4
 x : array(quant_prod) of mpvar
 A1 : array(quant_prod) of real
 A2 : array(quant_prod) of real
 c: array(quant_prod) of real
end-declarations
A1::[2, 3, 5, 5]
A2 :: [3, 4, 1, 2]
c::[5,  1,  1,  3]
sum(i in quant_prod) A1(i)*x(i) >= 16
sum(i in quant_prod) A2(i)*x(i) >= 13
objective:= sum(i in quant_prod) c(i)*x(i)

minimize(objective)
writeln("Begin running model")
!...
forall(i in quant_prod) do
 writeln("x(",i,")= ", getsol(x(i)))
end-do
 writeln("FO: ", getobjval)
writeln("End running model")
end-model
```

Observe que a impressão do valor das variáveis mudou neste exemplo porque agora temos um vetor de variáveis, então precisamos de uma estrutura de repetição FOR, que no Xpress Mosel tem a sintaxe
forall(….) do
para imprimir a solução das variáveis

```
Begin running model
x(1)= 0
x(2)= 2.88235
x(3)= 1.47059
x(4)= 0
FO: 4.35294
End running model
```

Xpress Mosel

- Podemos também definir A como a matriz de coeficientes como fazemos normalmente ao resolver o problema utilizando o método Simplex.
- Da mesma forma, continuamos podendo obter as folgas e valores duais das restrições, declarando as restrições também como um array de linctr.

```
model exemplo2
uses "mmxprs";
declarations
! ...
  quant_prod = 1..4
  quant_restr = 1..2
 x : array(quant_prod) of mpvar
 A : array(quant_restr,quant_prod) of real
 b : array(quant_restr) of real
 c: array(quant_prod) of real
end-declarations
A::[2, 3, 5, 5, 3, 4, 1, 2]
b::[16, 13]
c::[5,  1,  1,  3]
forall(i in quant_restr) do
   sum(j in quant_prod) A(i,j)*x(j) >= b(i)
end-do
objective:= sum(i in quant_prod) c(i)*x(i)
 minimize(objective)
writeln("Begin running model")
!...
forall(i in quant_prod) do
   writeln("x(",i,")= ", getsol(x(i)))
end-do
 writeln("FO: ", getobjval)
writeln("End running model")
end-model
```

```
Begin running model
x(1)= 0
x(2)= 2.88235
x(3)= 1.47059
x(4)= 0
FO: 4.35294
End running model
```

```
model exemplo2
uses "mmxprs";
declarations
 quant_prod = 1..4
 quant_restr = 1..2
  x : array(quant_prod) of mpvar
 A : array(quant_restr,quant_prod) of real
 b : array(quant_restr) of real
 c: array(quant_prod) of real
 restr : array(quant_restr) of linctr
end-declarations
A::[2, 3, 5, 5, 3, 4, 1, 2]
b::[16, 13]
c::[5,  1,  1,  3]
forall(i in quant_restr) do
  restr(i):= sum(j in quant_prod) A(i,j)*x(j) >= b(i)
end-do
objective:= sum(i in quant_prod) c(i)*x(i)
minimize(objective)
writeln("Begin running model")
forall(i in quant_prod) do
   writeln("x(",i,")= ", getsol(x(i)))
end-do
forall(i in quant_restr) do
   writeln("folga restr(",i,")= ", getslack(restr(i)))
end-do
 writeln("FO: ", getobjval)
writeln("End running model")
end-model
```

```
Begin running model
x(1)= 0
x(2)= 2.88235
x(3)= 1.47059
x(4)= 0
folga restr(1)= 0
folga restr(2)= 0
FO: 4.35294
End running model
```