Examples of using vOptSpecific, vOptGeneric, and vOptTools

Xavier Gandibleux

July 21, 2017

Abstract

This document provides examples of using vOptSpecific, vOptGeneric, and vOptTools.

Contents

1 Examples with the bi-objective linear assignment problem

The bi-objective linear assignment problem (2LAP)

$$\left[\begin{array}{cclll} \min z^k & = & \displaystyle\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^k x_{ij} & k = 1, \dots, 2 \\ s/c & & \displaystyle\sum_{i=1}^{n} x_{ij} = 1 & j = 1, \dots, n \\ & & \displaystyle\sum_{j=1}^{n} x_{ij} = 1 & i = 1, \dots, n \\ & & x_{ij} = (0,1) & i = 1, \dots, n \\ & & & j = 1, \dots, n \end{array}\right]$$

A numerical instance of 2LAP

$$C^1 = \begin{pmatrix} 3 & 9 & 7 \\ 16 & 10 & 6 \\ 2 & 7 & 11 \end{pmatrix}$$

$$C^2 = \begin{pmatrix} 16 & 15 & 6 \\ 5 & 7 & 13 \\ 1 & 2 & 13 \end{pmatrix}$$

1.1 vOptSpecific and 2LAP: data are explicitely provided

The program

```
n = 3

C1 = [ 3 9 7 ;
       16 10 6 ;
        2 7 11 ]

C2 = [ 16 15 6 ;
         5 7 13 ;
         1 2 13 ]

using vOptSpecific

m = set2LAP( n , C1 , C2 )
z1, z2, S = vSolve( m ) # solver by default: LAP_Przybylski2008( )
```

The result

```
julia> z1
4-element Array{Int64,1}:
 16
 19
 30
 17

julia> z2
4-element Array{Int64,1}:
 31
 14
 13
 29

julia> S
4x3 Array{Int64,2}:
 0  2  1
 2  1  0
 1  2  0
 2  0  1
```

1.2 vOptSpecific and 2LAP: data and solver are explicitely provided

The program

```
n = 3

C1 = [ 3 9 7 ;
      16 10 6 ;
       2 7 11 ]

C2 = [ 16 15 6 ;
        5 7 13 ;
        1 2 13 ]

using vOptSpecific

m = set2LAP( n , C1 , C2 )
solver = LAP_Przybylski2008( )
z1, z2, S = vSolve( m , solver )
```

The result

```
julia> z1
4-element Array{Int64,1}:
 16
 19
 30
 17

julia> z2
4-element Array{Int64,1}:
 31
 14
 13
 29

julia> S
4x3 Array{Int64,2}:
 0  2  1
 2  1  0
 1  2  0
 2  0  1
```

1.3 vOptSpecific and 2LAP: read data on a file according to the LAP format

The program

```
using vOptSpecific

m = load2LAP("2ap03.dat")
z1, z2, S = vSolve( m ) # solver by default: LAP_Przybylski2008( )
```

Example of a datafile (2ap03.dat)

```
3
 3   9   7
16  10   6
 2   7  11
16  15   6
 5   7  13
 1   2  13
```

The result

```
julia> z1
4-element Array{Int64,1}:
 16
 19
 30
 17

julia> z2
4-element Array{Int64,1}:
 31
 14
 13
 29

julia> S
4x3 Array{Int64,2}:
 0  2  1
 2  1  0
 1  2  0
 2  0  1
```

1.4 vOptGeneric and 2LAP (2IP): data are explicitely provided

The program

```
n = 3

C1 = [ 3 9 7 ;
      16 10 6 ;
       2 7 11 ]

C2 = [ 16 15 6 ;
        5 7 13 ;
        1 2 13 ]

using vOptGeneric
using GLPK ; using GLPKMathProgInterface

m = vModel( solver = GLPKSolverMIP() )
@variable( m , x[1:n,1:n] , Bin )
@addobjective( m , Min, sum( C1[i,j]*x[i,j] for i=1:n,j=1:n ) )
@addobjective( m , Min, sum( C2[i,j]*x[i,j] for i=1:n,j=1:n ) )
@constraint( m , cols[i=1:n], sum(x[i,j] for j=1:n) == 1 )
@constraint( m , rows[j=1:n], sum(x[i,j] for i=1:n) == 1 )
solve( m , method = :epsilon , step = 0.5 )
```

The result

```
julia> print_X_E(m)
(16.0, 31.0) : x[1,1]=1 x[2,3]=1 x[3,2]=1
(17.0, 29.0) : x[1,2]=1 x[2,3]=1 x[3,1]=1
(19.0, 14.0) : x[1,3]=1 x[2,2]=1 x[3,1]=1
(30.0, 13.0) : x[1,3]=1 x[2,1]=1 x[3,2]=1

julia> getY_N(m)
4-element Array{Tuple{Vararg{Float64,N}} where N,1}:
 (16.0, 31.0)
 (17.0, 29.0)
 (19.0, 14.0)
 (30.0, 13.0)
```

1.5 vOptGeneric and 2LAP (2LP): data are explicitely provided

The program

```
n = 3

C1 = [ 3 9 7 ;
      16 10 6 ;
       2 7 11 ]

C2 = [ 16 15 6 ;
        5 7 13 ;
        1 2 13 ]

using vOptGeneric
using GLPK ; using GLPKMathProgInterface

m = vModel( solver = GLPKSolverLP() )
@variable( m , x[1:n,1:n] >= 0 )
@addobjective( m , Min, sum( C1[i,j]*x[i,j] for i=1:n,j=1:n ) )
@addobjective( m , Min, sum( C2[i,j]*x[i,j] for i=1:n,j=1:n ) )
@constraint( m , cols[i=1:n], sum(x[i,j] for j=1:n) == 1 )
@constraint( m , rows[j=1:n], sum(x[i,j] for i=1:n) == 1 )
solve( m , method = :epsilon , step = 0.5 )
```

The result

```
julia> getY_N(m)
37-element Array{Tuple{Vararg{Float64,N}} where N,1}:
 (16.0, 31.0)
 (16.0882, 30.5)
 (16.1765, 30.0)
 (16.2647, 29.5)
 (16.3529, 29.0)
 (16.4412, 28.5)
 (16.5294, 28.0)
 (16.6176, 27.5)
 (16.7059, 27.0)
 (16.7941, 26.5)
 ...
 (18.4706, 17.0)
 (18.5588, 16.5)
 (18.6471, 16.0)
 (18.7353, 15.5)
 (18.8235, 15.0)
 (18.9118, 14.5)
 (19.0, 14.0)
 (24.5, 13.5)
 (30.0, 13.0)
```

1.6 vOptGeneric and 2LAP (2IP): write on a file according to the MOP format

The program

```
n = 3

C1 = [ 3 9 7 ;
      16 10 6 ;
       2 7 11 ]

C2 = [ 16 15 6 ;
        5 7 13 ;
        1 2 13 ]

using vOptGeneric
using GLPK ; using GLPKMathProgInterface

m = vModel( solver = GLPKSolverMIP() )
@variable( m , x[1:n,1:n] , Bin )
@addobjective( m , Min, sum( C1[i,j]*x[i,j] for i=1:n,j=1:n ) )
@addobjective( m , Min, sum( C2[i,j]*x[i,j] for i=1:n,j=1:n ) )
@constraint( m , cols[i=1:n], sum(x[i,j] for j=1:n) == 1 )
@constraint( m , rows[j=1:n], sum(x[i,j] for i=1:n) == 1 )
writeMOP( m , "2ap03.mop" )
```

Example of a datafile (`2ap03.mop`)

```
NAME   vOptModel
OBJSENSE
 MIN
ROWS
 N  OBJ1
 N  OBJ2
 E  CON1
 E  CON2
 E  CON3
 E  CON4
 E  CON5
 E  CON6
COLUMNS
    MARKER    'MARKER'                 'INTORG'
    x_1_1  CON1  1
    x_1_1  CON4  1
    x_1_1  OBJ1  3
    x_1_1  OBJ2  16
    x_1_2  CON1  1
    x_1_2  CON5  1
    x_1_2  OBJ1  9
    x_1_2  OBJ2  15
    x_1_3  CON1  1
    x_1_3  CON6  1
    x_1_3  OBJ1  7
    x_1_3  OBJ2  6
```

```
    x_2_1  CON2  1
    x_2_1  CON4  1
    x_2_1  OBJ1  16
    x_2_1  OBJ2  5
    x_2_2  CON2  1
    x_2_2  CON5  1
    x_2_2  OBJ1  10
    x_2_2  OBJ2  7
    x_2_3  CON2  1
    x_2_3  CON6  1
    x_2_3  OBJ1  6
    x_2_3  OBJ2  13
    x_3_1  CON3  1
    x_3_1  CON4  1
    x_3_1  OBJ1  2
    x_3_1  OBJ2  1
    x_3_2  CON3  1
    x_3_2  CON5  1
    x_3_2  OBJ1  7
    x_3_2  OBJ2  2
    x_3_3  CON3  1
    x_3_3  CON6  1
    x_3_3  OBJ1  11
    x_3_3  OBJ2  13
    MARKER    'MARKER'                 'INTEND'
RHS
    RHS    CON1    1
    RHS    CON2    1
    RHS    CON3    1
    RHS    CON4    1
    RHS    CON5    1
    RHS    CON6    1
BOUNDS
  UP BOUND x_1_1 1
  UP BOUND x_1_2 1
  UP BOUND x_1_3 1
  UP BOUND x_2_1 1
  UP BOUND x_2_2 1
  UP BOUND x_2_3 1
  UP BOUND x_3_1 1
  UP BOUND x_3_2 1
  UP BOUND x_3_3 1
ENDATA
```

1.7 vOptGeneric and 2LAP (2IP): read data on a file according to the MOP format

The program

```julia
using vOptGeneric
using GLPK ; using GLPKMathProgInterface

m = parseMOP("2ap03.mop" , solver=GLPKSolverMIP() )
solve( m , method = :epsilon , step = 0.5 )
```

Example of a datafile (`2ap03.mop`)

```
NAME    vOptModel
OBJSENSE
 MIN
ROWS
 N  OBJ1
 N  OBJ2
 E  CON1
 E  CON2
 E  CON3
 E  CON4
 E  CON5
 E  CON6
COLUMNS
    MARKER     'MARKER'                 'INTORG'
    x_1_1   CON1   1
    x_1_1   CON4   1
    x_1_1   OBJ1   3
    x_1_1   OBJ2   16
    x_1_2   CON1   1
    x_1_2   CON5   1
    x_1_2   OBJ1   9
    x_1_2   OBJ2   15
    x_1_3   CON1   1
    x_1_3   CON6   1
    x_1_3   OBJ1   7
    x_1_3   OBJ2   6
    x_2_1   CON2   1
    x_2_1   CON4   1
    x_2_1   OBJ1   16
    x_2_1   OBJ2   5
    x_2_2   CON2   1
    x_2_2   CON5   1
    x_2_2   OBJ1   10
    x_2_2   OBJ2   7
    x_2_3   CON2   1
    x_2_3   CON6   1
    x_2_3   OBJ1   6
    x_2_3   OBJ2   13
    x_3_1   CON3   1
```

```
    x_3_1  CON4  1
    x_3_1  OBJ1  2
    x_3_1  OBJ2  1
    x_3_2  CON3  1
    x_3_2  CON5  1
    x_3_2  OBJ1  7
    x_3_2  OBJ2  2
    x_3_3  CON3  1
    x_3_3  CON6  1
    x_3_3  OBJ1  11
    x_3_3  OBJ2  13
    MARKER    'MARKER'                 'INTEND'
RHS
    RHS    CON1    1
    RHS    CON2    1
    RHS    CON3    1
    RHS    CON4    1
    RHS    CON5    1
    RHS    CON6    1
BOUNDS
  UP BOUND x_1_1 1
  UP BOUND x_1_2 1
  UP BOUND x_1_3 1
  UP BOUND x_2_1 1
  UP BOUND x_2_2 1
  UP BOUND x_2_3 1
  UP BOUND x_3_1 1
  UP BOUND x_3_2 1
  UP BOUND x_3_3 1
ENDATA
```

The result

```
julia> print_X_E(m)
(16.0, 31.0) : x_1_1=1 x_2_3=1 x_3_2=1
(17.0, 29.0) : x_1_2=1 x_2_3=1 x_3_1=1
(19.0, 14.0) : x_1_3=1 x_2_2=1 x_3_1=1
(30.0, 13.0) : x_1_3=1 x_2_1=1 x_3_2=1


julia> getY_N(m)
4-element Array{Tuple{Float64,Float64},1}:
 (16.0, 31.0)
 (17.0, 29.0)
 (19.0, 14.0)
 (30.0, 13.0)
```

2 Examples with a bi-objective one machine scheduling problem

The specific bi-objective one machine scheduling problem (2OSP) considered is

$$1 \mid . \mid (\Sigma C_i, T_{max})$$

A numerical instance of this 2OSP

i	1	2	3	4
p_i	2	4	3	1
d_i	1	2	4	6
r_i	not required			
w_i	not required			

2.1 vOptSpecific and 2OSP: data are explicitely provided

The program

```julia
n = 4
p = [ 2 , 4 , 3 , 1 ]
d = [ 1 , 2 , 4 , 6 ]

using vOptSpecific

id = set2OSP( n , p , d )
z1, z2 , S = vSolve( id )
```

The result

```
julia> z1
3-element Array{Int64,1}:
 20
 21
 27

julia> z2
3-element Array{Int64,1}:
 8
 6
 5

julia> S
3-element Array{Array{Int64,1},1}:
 [4, 1, 3, 2]
 [4, 1, 2, 3]
 [1, 2, 3, 4]
```

2.2 vOptSpecific and 2OSP: all data and the solver are explicitely provided

The program

```
n = 4
p = [ 2 , 4 , 3 , 1 ]
d = [ 1 , 2 , 4 , 6 ]
r = [ 0 , 0 , 0 , 0 ]
w = [ 1 , 1 , 1 , 1 ]

using vOptSpecific

id = set2OSP( n , p , d , r , w )
solver = OSP_VanWassenhove1980( )
z1, z2 , S = vSolve( id , solver )
```

The result

```
julia> z1
3-element Array{Int64,1}:
 20
 21
 27

julia> z2
3-element Array{Int64,1}:
 8
 6
 5

julia> S
3-element Array{Array{Int64,1},1}:
 [4, 1, 3, 2]
 [4, 1, 2, 3]
 [1, 2, 3, 4]
```

3 vOptTools

3.1 Script for plotting Y_N

The program

```

z1 = [ 16 ; 19 ; 30 ; 17 ]
z2 = [ 31 ; 14 ; 13 ; 29 ]

using PyPlot

title(L"$Y_N$")
xlabel(L"$z_1$")
ylabel(L"$z_2$")
axis([0, 40, 0, 40])

plot( z1, z2, color="green", linestyle="", marker="o", label="YN" )
```

The result