Programming

optimisation and operations research algorithms with Julia

for Business Tasks

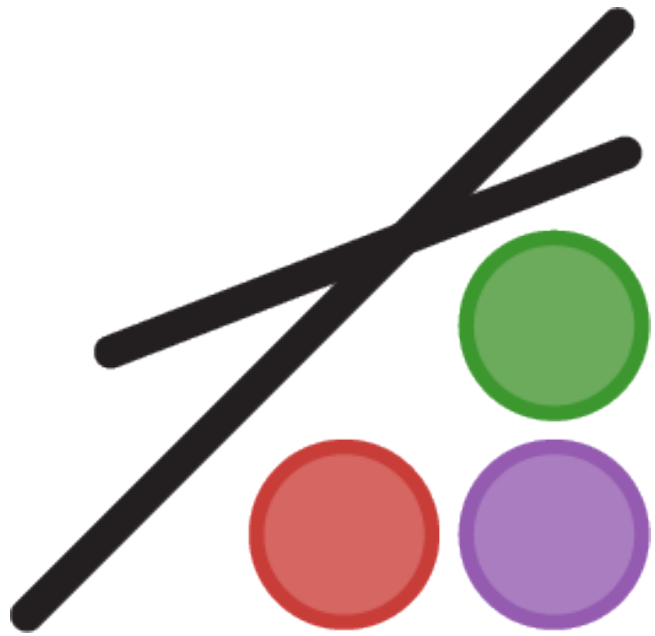
Prof. Dr. Xavier Gandibleux

Université de Nantes
Département Informatique – UFR Sciences et Techniques
France

Lesson 6 – May-June 2022

Optimisation

JuMP (part 2)



version 1.0.0 and later

Writing an implicit model (1/5)

Example (cont'd):

$$\left[\begin{array}{rrrcrl} \max z(x) = & x_1 & + & 3x_2 & & (0) \\ s.t & x_1 & + & x_2 & \leqslant 14 & (1) \\ & -2x_1 & + & 3x_2 & \leqslant 12 & (2) \\ & 2x_1 & - & x_2 & \leqslant 12 & (3) \\ & x_1 & , & x_2 & \geqslant 0 & (4) \end{array}\right]$$

$$\downarrow$$

$$c = (1,3),\ d = (14,12,12),\ T = \begin{pmatrix} 1 & 1 \\ -2 & 3 \\ 2 & -1 \end{pmatrix}$$

$$\left[\begin{array}{lll} \max z(x) = & \sum_{j=1}^{2} c_j x_j & (0) \\ s.t & \sum_{j=1}^{2} t_{ij} x_j \leqslant d_i \quad i = 1,3 & (1-3) \\ & x_j \geqslant 0 \qquad\qquad j = 1,2 & (4) \end{array}\right]$$

Writing an implicit model (2/5)

Example (cont'd):

$$c = (1, 3),\ d = (14, 12, 12),\ T = \begin{pmatrix} 1 & 1 \\ -2 & 3 \\ 2 & -1 \end{pmatrix}$$

```
c = [1, 3]
d = [14, 12, 12]
T = [1 1 ; -2 3; 2 -1]
n,m = size(T)
```

Writing an implicit model (2/5)

Example (cont'd):

$$\left[\begin{array}{lll} \max z(x) = & \sum_{j=1}^{2} c_j x_j & (0) \\ s.t & \sum_{j=1}^{2} t_{ij} x_j \leqslant d_i \quad i = 1,3 & (1-3) \\ & x_j \geqslant 0 \quad j = 1,2 & (4) \end{array}\right]$$

```
julia> md = Model(GLPK.Optimizer)
julia> @variable(md, x[1:m]⩾0)
julia> @objective(md, Max,sum(c[j]*x[j] for j=1:m))
julia> @constraint(md, cst[i=1:n],sum(T[i,j]*x[j]
                                      for j=1:m)⩽d[i])
```

Structured variables in JuMP

Scalar variables:

```
julia> @variable(md, x≥0)
```

Structured variables:

- Arrays (one-based integer ranges)

```
julia> @variable(md, x[1:4]≥0)
julia> @variable(md, x[1:2, 1:2]≥0, Int)
```

- DenseAxisArrays (indices are not one-based integer ranges)

```
julia> @variable(md, x[-4:4]≥0)
julia> @variable(md, y[[:tram,:train]], Bin)
```

- SparseAxisArrays (indices do not form a rectangular set)

```
julia> @variable(model, x[i=1:2, j=i:2])
```

Structured variables in JuMP

Scalar variables:

```
julia> @variable(md, x≥0)
```

Structured variables:

- Arrays (one-based integer ranges)

```
julia> @variable(md, x[1:4]≥0)
julia> @variable(md, x[1:2, 1:2]≥0, Int)
```

- DenseAxisArrays (indices are not one-based integer ranges)

```
julia> @variable(md, x[-4:4]≥0)
julia> @variable(md, y[[:tram,:train]], Bin)
```

- SparseAxisArrays (indices do not form a rectangular set)

```
julia> @variable(model, x[i=1:2, j=i:2])
```

More on variables

Providing a primal starting solution (a MIP-start; a warmstart)

```
julia> @variable(model, x, start = 44)
```

```
julia> set_start_value(x, 44)
```

Deleting variables:

```
julia> delete(model, x)
```

More on variables

Providing a primal starting solution (a MIP-start; a warmstart)

```
julia> @variable(model, x, start = 44)
```

```
julia> set_start_value(x, 44)
```

Deleting variables:

```
julia> delete(model, x)
```

More on the objective

Modify an objective:

call `@objective` with the new objective function:

```
julia> @objective(model, Min, 2x)
```

Modify an objective coefficient:

use `set_objective_coefficient`:

```
set_objective_coefficient(model, variable, coef)
```

```
julia> set_objective_coefficient(model, x2, 5)
```

More on the objective

Modify an objective:

call `@objective` with the new objective function:

```
julia> @objective(model, Min, 2x)
```

Modify an objective coefficient:

use `set_objective_coefficient`:

```
set_objective_coefficient(model, variable, coef)
```

```
julia> set_objective_coefficient(model, x2, 5)
```

More on constraints

Modify the coefficient of a variable for a given constraint:

```
set_normalized_coefficient(cstID, var, coef)
```

```
julia> set_normalized_coefficient(model, x2, 5)
```

Set the right-hand side term of a constraint to a value:

```
set_normalized_rhs(constraintID, value)
```

```
julia> set_normalized_rhs(cst2, 10)
```

Delete a constraint from a model:

```
delete(model, constraintID)
```

```
julia> delete(model, cst2)
```

More on constraints

Modify the coefficient of a variable for a given constraint:

```
set_normalized_coefficient(cstID, var, coef)
```

```
julia> set_normalized_coefficient(model, x2, 5)
```

Set the right-hand side term of a constraint to a value:

```
set_normalized_rhs(constraintID, value)
```

```
julia> set_normalized_rhs(cst2, 10)
```

Delete a constraint from a model:

```
delete(model, constraintID)
```

```
julia> delete(model, cst2)
```

More on constraints

Modify the coefficient of a variable for a given constraint:

```
set_normalized_coefficient(cstID, var, coef)
```

```
julia> set_normalized_coefficient(model, x2, 5)
```

Set the right-hand side term of a constraint to a value:

```
set_normalized_rhs(constraintID, value)
```

```
julia> set_normalized_rhs(cst2, 10)
```

Delete a constraint from a model:

```
delete(model, constraintID)
```

```
julia> delete(model, cst2)
```

More on solutions

Primal solution values for a scalar variable:

```
value(variable)
```

```
julia> value(x1)
```

Primal solution values for a structured variable:

```
value(variable[index])
```

```
julia> value(x[3])
```

```
value.(variable)
```

```
julia> value.(x)
```

More on solutions

Primal solution values for a scalar variable:

```
value(variable)
```

```
julia> value(x1)
```

Primal solution values for a structured variable:

```
value(variable[index])
```

```
julia> value(x[3])
```

```
value.(variable)
```

```
julia> value.(x)
```

Review and exercises

(notebook)