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 9 – May-June 2022

Optimisation

vOptSolver

Overview of vOptSolver

An ecosystem for modeling and solving multiobjective linear optimization problems (MOCO, MOIP, MOMIP, MOLP):

- it deals with structured and non-structured optimization problems with at least two objectives
- it integrates several specific and generic exact algorithms for computing efficient solutions
- Natural and intuitive use for mathematicians, informaticians, engineers
- Efficient, flexible, evolutive solver
- Aims to be easy to formulate a problem, to provide data, to solve a problem, to collect the outputs, to analyze the solutions
- Free, open source (MIT licence), multi-platform, reusing existing specifications
- Using usual free (GLPK, Clp/Cbc) and commercial (GUROBI, CPLEX) MILP solvers

Overview of vOptSolver

An ecosystem for modeling and solving multiobjective linear optimization problems (MOCO, MOIP, MOMIP, MOLP):

- it deals with structured and non-structured optimization problems with at least two objectives
- it integrates several specific and generic exact algorithms for computing efficient solutions
- Natural and intuitive use for mathematicians, informaticians, engineers
- Efficient, flexible, evolutive solver
- Aims to be easy to formulate a problem, to provide data, to solve a problem, to collect the outputs, to analyze the solutions
- Free, open source (MIT licence), multi-platform, reusing existing specifications
- Using usual free (GLPK, Clp/Cbc) and commercial (GUROBI, CPLEX) MILP solvers

Getting started with vOptGeneric

Install:

```
using Pkg
Pkg.add("vOptGeneric")
```

```
Pkg.add("GLPK")
```

Setup:

```
using vOptGeneric
```

```
using GLPK
```

Getting started with vOptGeneric

Install:

```
using Pkg
Pkg.add("vOptGeneric")
```

```
Pkg.add("GLPK")
```

Setup:

```
using vOptGeneric
```

```
using GLPK
```

Example with vOptGeneric

For the bi-objective unidimensional 01 knapsack problem,

$$\max \left\{ (p^1 x,\ p^2 x) \mid wx \leqslant c,\ x \in \{0,1\}^n \right\}$$

with[1]

$$n = 5$$
$$p^1 = (10,\ 3,\ 6,\ 8,\ 2)$$
$$p^2 = (12,\ 9,\ 11,\ 5,\ 6)$$
$$w = (4,\ 5,\ 2,\ 5,\ 6)$$
$$c = 17$$

compute Y_N, the set of non-dominated points using the ϵ-constraint method.

[1] exercise 10.2, page 290 of *Multicriteria Optimization* (2nd edt), M. Ehrgott, Springer 2005

Setup the data

```
julia> p1 = [10,3,6,8,2]          # coef vct of the obj 1
julia> p2 = [12,9,11,5,6]         # coef vct of the obj 2
julia> w = [4,5,2,5,6]              # coef vct of weights
julia> c = 17                          # nominal capacity
julia> n = length(p1)                   # number of items
```

Setup the model

```
julia> kp = vModel( GLPK.Optimizer )
julia> @variable(kp, x[1:n],Bin)
julia> @addobjective(kp,Max,sum(p1[j]*x[j] for j=1:n))
julia> @addobjective(kp,Max,sum(p2[j]*x[j] for j=1:n))
julia> @constraint(kp,sum(w[j]*x[j] for j=1:n) ≤ c)
```

Solve and display results

Invoking the solver (dichotomic method):

```
julia> vSolve(kp,method=:epsilon,step=0.5,verbose=false)
```

Querying the results:

```
julia> Y_N = getY_N(kp)
```

Displaying the results (X_SE and Y_SN):

```
julia> for i = 1:length(Y_N)
julia>     X = value.(x, i)
julia>     print("X = ", findall(elt -> elt ≈ 1, X))
julia>     println(" | Z = ",Y_N[i])
julia> end
```

```
julia> printX_E(kp)
```

Plot results

```
julia> using PyPlot
julia> z1, z2 = map(x -> x[1],Y_N), map(x -> x[2],Y_N)
julia> PyPlot.title("Knapsack")
julia> PyPlot.xlabel("$z_1$ to maximize")
julia> PyPlot.ylabel("$z_2$ to maximize")
julia> grid()
julia> plot(z1,z2,"bx",markersize="8",label="$Y_N$")
julia> legend(loc=1,fontsize="small")
julia> show()
```

Review and exercises

(notebook)