vOptSolver,
a "get and run" solver of multiobjective linear optimization problems built on Julia and JuMP

MCDM'2017: 24th International Conference on Multiple Criteria Decision Making
July 10 – 14, 2017. Ottawa, Canada

Xavier GANDIBLEUX[1], Gauthier SOLEILHAC[1], Flavien LUCAS[1], Anthony PRZYBYLSKI[1], Stefan RUZIKA[2], Pascal HALFFMANN[2]

July 5, 2017

Université de Nantes, France[1] – University of Koblenz-Landau, Germany[2]

vOpt Research Project

Exact Efficient Solution of Mixed Integer Programming Problems with Multiple Objective Functions (ANR/DFG-14-CE35-0034-01)

Parts (algorithms, instances) of the outputs are integrated in:

- **vOptSolver**: a solver of multiobjective linear optimization problems (MOCO, MOIP, MOMILP, MOLP)
- **vOptLib**: a numerical instances library of multiobjective linear optimization problems (MOCO, MOIP, MOMILP, MOLP)
  History: MCDMlib (opened in 1998) — MOCOlib, the MOCO section of the MCDMlib (opened in 2007) — GUEPARDlib (opened in 2010)

Content

1. Introduction

2. vOptSolver

3. Now and tomorrow

1. Introduction

Context

Computing Y_N for Multiobjective linear Optimization Problems (MOP):

$$\begin{array}{rcl} \min z(x) & = & Cx \\ \text{s/t } Tx & \leqq & d \\ x & \in & \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2} \end{array} \qquad \begin{array}{rcl} \min z(x) & = & Cx \\ \text{s/t } Tx & \leqq & d \\ x & \in & \mathbb{R}^{n_1} \end{array} \qquad \begin{array}{rcl} \min z(x) & = & Cx \\ \text{s/t } Tx & \leqq & d \\ x & \in & \mathbb{Z}^{n_2} \end{array}$$

Multi Objective Mixed-Integer Linear Problem (MOMILP)	Multi Objective Linear Problem (MOLP)	Multi Objective Integer Problem (MOIP)

MO(M)ILP + structure
MultiObjective Combinatorial Optimization (MOCO)

where:

$$\begin{array}{rcl} T \in \mathbb{Z}^{m \times n} & \longrightarrow & m \text{ constraints, } i = 1, \ldots, m \\ C \in \mathbb{Z}^{n \times p} & \longrightarrow & \text{the objective matrix} \\ X = \{x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2} | Tx \leq d\} \subseteq \mathbb{R}^n & \longrightarrow & \text{the set of feasible solutions} \\ Y = z(X) \subseteq \mathbb{R}^p & \longrightarrow & \text{the set of images} \end{array}$$

Identified software for computing exact Y_N of MOP

- **ADBASE** (Ralph Steuer, 1975)
  - Problem class solved: MOLP
  - Algorithm(s): simplex algorithm
  - not available online

- **Bensolve** (Andreas Löhne, 2017); available online:
  - Problem class solved: vector linear programs (including MOLP)
  - Algorithm(s): Benson's algorithm (language C)
  - `http://bensolve.org`

- **Inner** (Laszlo Csirmaz, 2016); available online:
  - Problem class solved: MOLP
  - `https://github.com/lcsirmaz/inner`

- **PolySCIP** (Sebastian Schenker, 2016); available online:
  - Problem classes solved: p-LP, 2-IP, 3-IP (MOMILP experimentally)
  - `http://polyscip.zib.de`

Presentation of vOptSolver (1/2)

- Aims
  - Solver of multiobjective linear optimization problems for scientifics and practionners
  - Easy to formulate a problem, to provide data, to solve a problem, to collect the outputs, to analyze the solutions
  - Natural and intuitive use for mathematicians, informaticians, engineers

- Purposes
  - Solving needs:
    methods and algorithms for performing numerical experiments
  - Research needs:
    support and primitives for the development of new algorithms
  - Pedagogic needs:
    environment for practicing of theories and algorithms

Presentation of vOptSolver (2/2)

- Characteristics
  - Efficient, flexible, evolutive solver
  - Free, open source, multi-platform, reusing existing specifications
  - Easy installation, no need of being expert in computer science

- Background
  - Julia programming language, and JuMP algebraic language
  - Usual free (GLPK) and commercial (CPLEX, GUROBI) MILP solvers
  - homemade solvers implemented in C/C++ language

vOptSolver

$\sim$

a backbone,
based on Julia and JuMP,
embedding algorithms coded in C/C++/Julia

Why Julia and JuMP as cement of vOptSolver?

Julia programming language

- a high-level, high-performance programming language for scientific computing
- familiar for the practitioners of Matlab, Fortran, Python, C/C++, Pascal, etc.
- integrates natively a wrapper to external programs written in C or C++
- `http://julialang.org/`

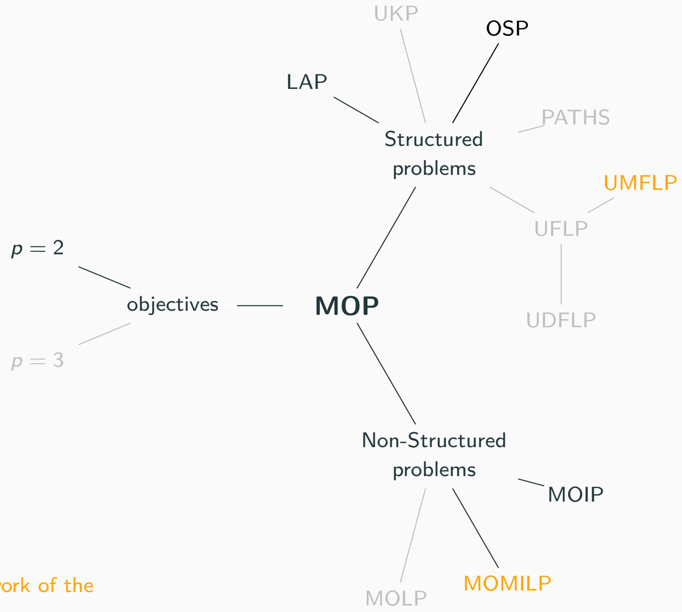
JuMP algebraic language (Julia for mathematical optimization)

- a domain-specific modeling language for mathematical optimization in Julia
- familiar for the practitioners of GMP, AMPL, MPL, GAMS, OPL, and etc.
- wrapped with several open-source/commercial solvers (e.g. GLPK and CPLEX).
- `http://jump.readthedocs.io/en/latest/`

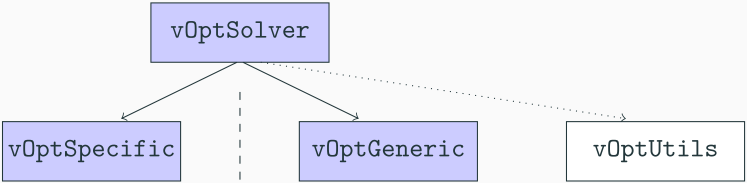
More:
Miles Lubin and Iain Dunning: Computing in Operations Research Using Julia. *INFORMS Journal on Computing* 27:2 , 238-248, 2015.

2. vOptSolver

Multi-objective linear optimization problems targeted



Framework of the
ANR/DFG research project



vOptSpecific

Multiobjective structured problems

Application Programming Interface (API)

Collection of specific algorithms

vOptGeneric

Multiobjective non-structured problems

Algebraic Language (JuMP)

Collection of generic algorithms

vOptUtils

Multiobjective primitives

Application Programming Interface (API)

Collection of useful algorithms

Using vOptSolver

vOptSolver has been tested with:

- Julia v0.6
- GLPK v4.60

Two modes:

1. a distant use
   - in the cloud with JuliaBox.

2. a local use
   - on macOS (tested on v10.12.5)
   - on linux (tested on ubuntu 14.04 LTS)
   - on windows (perhaps later...)

Example: 2-LAP

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

```
n  =  5

C1 = [  3   9   0   0   6 ;
       16   0   6  12  19 ;
        2   7  11  15   8 ;
        4  11   7  16   3 ;
        2   5   1   9   0   ]

C2 = [ 16   5   6  19  12 ;
       15   7  13   7   7 ;
        1   2  13   2   3 ;
       14   7   8   1   7 ;
       10  10   1   0   0   ]
```

Example: 2-LAP and vOptSpecific

Algorithm: Two phases

A. Przybylski, X. Gandibleux, and M. Ehrgott. Two phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research*, 185(2):509–533, 2008.

Implementation: algorithm provided in language C

Output: $X_E \subseteq \mathbb{N}^n$, $Y_N \subseteq \mathbb{Z}^2$

Program:

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

Example: 2-LAP and vOptGeneric

Algorithm: ϵ-constraint

Y.V. Haimes, L.S. Lasdon, D.A. Wismer: On a bicriterion formation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man and Cybernetics*. Volume SMC-1, Issue 3, 296-297, 1971.

MILP: GLPK

Output: $X_E \subseteq \mathbb{N}^n$, $Y_N \subseteq \mathbb{Z}^2$

Program:

```
using vOptGeneric
using GLPK
m = vModel( solver = GLPKSolverLP() )
@addVar( m , x[1:N,1:N]  >= 0 )
@addobjective( m , Min, sum{ C1[i,j]*x[i,j], i=1:N,j=1:N } )
@addobjective( m , Min, sum{ C2[i,j]*x[i,j], i=1:N,j=1:N } )
@constraint( m , cols[i=1:N], sum{x[i,j], j=1:N} == 1 )
@constraint( m , rows[j=1:N], sum{x[i,j], i=1:N} == 1 )
@time solve( m , method = :epsilon , step = 0.5 )
```

Performances

Measure of CPUt (s) for 2-LAP when the algorithm is embedded or not:

Computer characteristics: processor intel core i5 6300U 2.40GHz x 2 – RAM: 13Mb – OS: linux ubuntu 14.04 LTS 64 bits

instance id	$\mid Y_N \mid$	without vOptSolver	with vOptSpecific		with vOptGeneric		
					with GLPK		with CPLEX
		local	local	distant	local	distant	local
2AP50-1	163	0,548	0,72	0,76	18,74	18,17	19,51
2AP50-1A40	216	0,94	1,116	1,19	27,28	28,2	30,27
2AP50-1A60	304	0,9	1,05	15	39,86	37,85	35,89
2AP50-1A80	375	1,27	1,62	14,98	47,5	50,62	43,97
2AP50-1A100	301	0,84	1,08	14,57	44,91	45,27	39,72
sum		*4,79*	*6,02*	*46,95*	*192,74*	*204,25*	*185,37*
ratio			*1,25*	*7,79*		*1,05*	*0,96*
2AP100-1	223	10,68	10,14	11,07	N.A.	N.A.	89,72
2AP100-1A40	429	11,78	12,64	31,21	N.A.	N.A.	150,4
2AP100-1A60	585	18,96	19,07	39,12	N.A.	N.A.	N.A.
2AP100-1A80	845	28,39	29,26	69,92	N.A.	N.A.	N.A.
2AP100-1A100	947	26,23	32,37	72,36	N.A.	N.A.	N.A.
sum		*96,04*	*103,48*	*223,68*			
ratio			*1,07*	*2,16*			

N.A: not available (timeout of 180s)

- The overcost due to using vOptSpecific is negligible
- In distant mode, the displays on the console are penalizing (verbose to switch off)

3. Now and tomorrow

Conclusion

- Pros:
    - offert an open backbone to the user's community, blending high-performance computation and open-source principle, marrying user-friendly and easy-to-install, multi-plateform
    - going faster to the essential for structured problems, thanks to vOptSpecific; confort of modeling for non-structured problems, thanks to vOptGeneric
- Cons:
    - the use of Julia, a young and evolving programming language: another language (but it does not be learn for using vOptSolver) and it is evolving very fast (but it is not a problem for the users of vOptSolver)
    - few specific algorithms are currently integrated (but more are coming, from us, and from the community), some generic algorithms have some limitations (but future releases will drop off progresively the limitations)

On-going works

- in the very short term
    - deliver the functionalities awaited for testing theories and algorithms developed in the framework of the vOpt research project
    - integrate all the solving algorithms developed at Nantes by our research group
    - collaborate with all colleagues who are willing to contribute to the development of the solver
- in the mid term
    - introduce vOptSolver in OR/MCDM courses and seminars, as support for exercices and research projects
    - integrate feedbacks of users to enhance the use of vOptSolver, release versions incorporating new/improved problems/algorithms
    - consider to integrate approximation algorithms, such as multiobjective (meta)heuristics

Follow/join us here

Homepage of vOptSolver:
http://voptsolver.github.io/vOptSolver/

- Repository of vOptSolver:
  http://github.com/vOptSolver
- Repository of vOptSpecific:
  http://github.com/vOptSolver/vOptSpecific.jl
- Repository of vOptGeneric:
  http://github.com/vOptSolver/vOptGeneric.jl

Contact concerning vOptSolver:
vopt@univ-nantes.fr

Homepage of the ANR/DFG research project "vOpt":
http://vopt-anr-dfg.univ-nantes.fr