vOptSolver – Version 0.2

Software developed with the support of the ANR/DFG-14-CE35-0034-01

Université de Nantes

https://github.com/vOptSolver
https://voptsolver.github.io/vOptSpecific/
https://voptsolver.github.io/vOptGeneric/

July 4, 2017

Table of contents

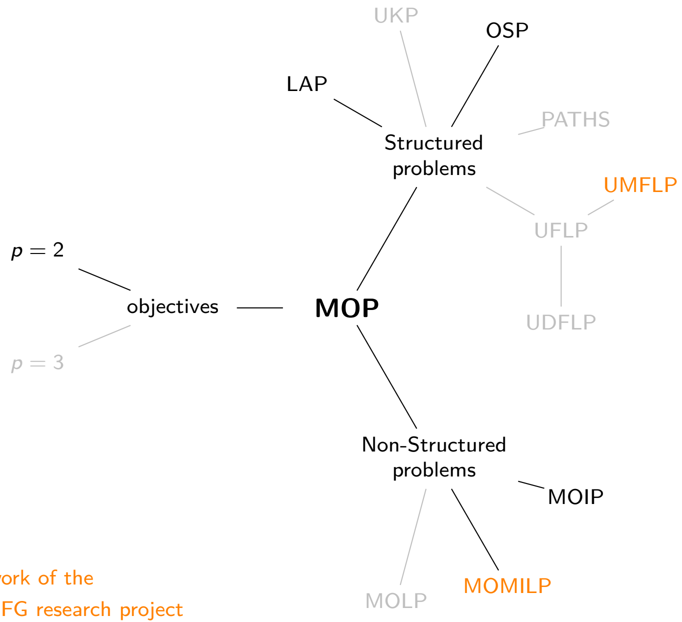
Convention: text in grey $\Rightarrow$ functionality integrated in future releases

Introduction to vOptSolver

Multi-objective linear optimization problems targeted

- LP: Linear Program
- MILP: Mixed Integer Linear Program
- IP: Integer Linear program
- CO: Combinatorial Optimization
- LAP: Linear Assignment Problem
- OSP: One machine Scheduling Problem
- UKP: Unidimensional 01 Knapsack Problem
- MKP: Multidimensional 01 Knapsack Problem
- UFLP: Uncapacitated Facility Location Problem
- UDFLP: Discrete Uncapacitated Facility Location Problem
- UMFLP: Mixed Uncapacitated Facility Location Problem
- SSCFLP: Single Source Capacitated Facility Location Problem
- CFLP: Capacitated Facility Location Problem
- PATHS: shortest paths problem

Multi-objective linear optimization problems targeted



Framework of the
ANR/DFG research project

Solutions reached

For a given problem, the aim is to compute

Y_N, the set of nondominated "points"

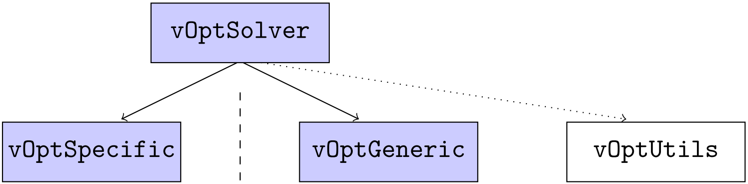
corresponding to

X_E, a complete set of efficient solutions

More on definitions and notations, refer to this book:

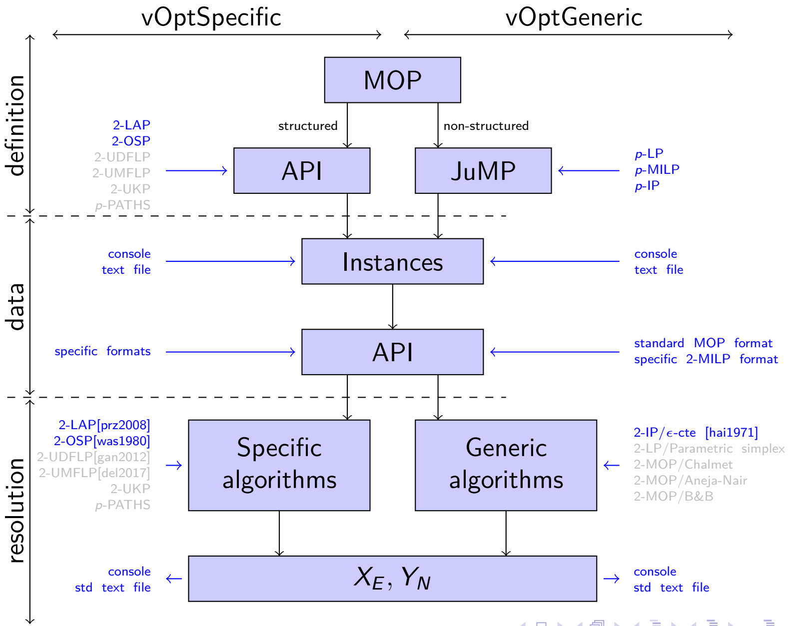
Matthias Ehrgott.
Multicriteria Optimization.
Springer-Verlag New York, 2005.

Design of vOptSolver



vOptSpecific	vOptGeneric	vOptUtils
Multiobjective structured problems	Multiobjective non-structured problems	Multiobjective primitives
Application Programming Interface (API)	Algebraic Language (JuMP)	Application Programming Interface (API)
Collection of specific algorithms	Collection of generic algorithms	Collection of useful algorithms

Design of vOptSolver in details



Design of vOptSolver in details

`vOptUtils`:

a collection of algorithms for managing and analyzing outcome set

- multidimensional datastructure for filtering and storing Y_N
- primitives for ploting Y_N
- primitives for analyzing Y_N

Integrated specific algorithms (1/2)

- 2-LAP [prz2008]; in C:

  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.

  output: $X_E \subseteq \mathbb{N}^n$, $Y_N \subseteq \mathbb{Z}^p$

- 2-OSP with here $1 \mid . \mid (\Sigma C_i, T_{max})$ [was1980]; in Julia:

  L.N. Van Wassenhove and L.F. Gelders. Solving a bicriterion scheduling problem. *European Journal of Operational Research*, 4(1):42–48, 1980.

  output: $X_E \subseteq \mathbb{N}^n$, $Y_N \subseteq \mathbb{Z}^p$

- 2-UKP [jor2010]; in Julia:

  J. Jorge. *Nouvelles propositions pour la résolution exacte du sac à dos multi-objectif unidimensionnel en variables binaires*. Thèse de doctorat, Université de Nantes - France. 2010.

  output: $X_E \subseteq \{0,1\}^n$, $Y_N \subseteq \mathbb{Z}^p$

Integrated specific algorithms (2/2)

- 2-UDFLP [gan2012]; in C++:

  X. Gandibleux, A. Przybylski, S. Bourougaa, A. Derrien, A. Grimault. Computing the efficient frontier for the 0/1 biobjective uncapacitated facility location problem. *10th International Conference on Multiple Objective Programming and Goal Programming*. June 11-13 2012, Niagara Falls, Canada.

  output: $X_E \subseteq \{0,1\}^n$, $Y_N \subseteq \mathbb{Z}^p$

- 2-UMFLP [del2017]; in C++:

  Q. Delmée, X. Gandibleux, A. Przybylski. Résolution exacte du problème de localisation de services bi-objectif sans contrainte de capacité en variables mixtes. *ROADEF2017 : 18ème congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision*, Feb 2017, Metz, France. 2017

  output: $X_E \subseteq \{0,1\}^{n_1} \times \mathbb{R}^{n_2}$, Y_N

- PATHS [gan2004]; in C:

  X. Gandibleux, Fr. Beugnies and S. Randriamasy: Martins' algorithm revisited for multi-objective shortest path problems with a MaxMin cost function. *4OR: A Quarterly Journal of Operations Research*. Volume 4, Number 1, pp. 47-59, 2006.

  output: X_E, Y_N

Integrated generic algorithms (1/2)

- 2-IP/ϵ-constraint method [hai1971]; in Julia:

  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, Pages 296-297, July 1971.

  output: $X_E \subseteq \mathbb{Z}^n$, $Y_N \subseteq \mathbb{Z}^p$

- 2-IP/dichotomic method:

  Aneja, Y. and K. Nair: Bicriteria transportation problem. *Management Science* 25 (1), 73?78. 1979.

  output: $X_{SE} \subseteq \mathbb{Z}^n$, $Y_{SN} \subseteq \mathbb{Z}^p$

- 2-IP/Chalmet & al., 1986:

  L.G. Chalmet, L. Lemonidis, and D.J. Elzinga. An algorithm for the bi-criterion integer programming problem. *European Journal of Operational Research*, 25:292-300, 1986.

  output: $X_E \subseteq \mathbb{Z}^n$, $Y_N \subseteq \mathbb{Z}^p$

Integrated generic algorithms (2/2)

- 2-MILP/Vincent & al., 2014:

  Th. Vincent, F. Seipp, S. Ruzika, A. Przybylski, X. Gandibleux. Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case. *Computers & Operations Research*, Volume 40, Issue 1, pp. 498–509, 2013.

  Fl. Lucas. *Multiobjective branch & cut*. Master Thesis, University of Nantes, France. June 2017.

  output: $X_E \subseteq \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}$, Y_N

- 2-LP/parametric simplex:

  Description available in: Matthias Ehrgott. *Multicriteria Optimization*. Springer-Verlag New York, 2005.

  output: $X_{SE} \subseteq \mathbb{R}^n$, $Y_{SN} \subseteq \mathbb{R}^p$

Selectable (MI)LP Engines

- open source:
  - GLPK (GNU Linear Programming Kit)
- commercial:
  - CPLEX
  - GUROBI

NB: CPLEX and GUROBI are currently not available on JuliaBox

Instructions for installing and running vOptSolver

Instructions for installing vOptSolver

vOptSolver has been tested with:

- Julia v0.6
- GLPK v4.60

Choose between:

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

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

Instructions for a distant use in the cloud

- Local installation
  1. nothing to do
- Run Julia
  1. go to `https://juliabox.com` and sign in to open a session
  2. click on the icon "console"
  3. when the prompt is ready, type in the console
     ```
     julia
     ```
- Before the first use of `vOptSolver`, add the following packages:
  1. when the prompt *julia* is ready, type in the terminal
     ```
     Pkg.add("GLPK")
     Pkg.clone("http://github.com/vOptSolver/vOptGeneric.jl")
     Pkg.clone("http://github.com/vOptSolver/vOptSpecific.jl")
     Pkg.build("vOptSpecific")
     ```

  At this point, vOptSolver is properly installed

Instructions for a local use on your own computer

- Local installation
  1. install Julia on your computer, instructions here:
     `http://julialang.org/downloads/`
  2. install (e.g.) GLPK on your computer, instructions here:
     `http://jump.readthedocs.io/en/latest/installation.html`

  NB: a standard C/C++ compiler must be installed (GCC is suggested)

  At this point, Julia and GLPK are properly installed

- Run Julia
  1. on linux: open a console on your computer; when the prompt is ready, type in the console `julia`
  2. on macOS: locate the application julia and click on the icon; julia console comes on the screen

- Before the first use of `vOptSolver`, add both following packages:
  1. when the prompt julia is ready, type in the console

```
Pkg.add("GLPK")
Pkg.clone("http://github.com/vOptSolver/vOptGeneric.jl")
Pkg.clone("http://github.com/vOptSolver/vOptSpecific.jl")
Pkg.build("vOptSpecific")
```

  At this point, vOptSolver is properly installed

Instructions for running vOptSolver

When vOptSolver is properly installed, vOptSpecific and vOptGeneric are ready locally or in the cloud.

- Run Julia
  1. open a console on your computer or in the cloud
  2. when the prompt is ready, type in the console
     `julia`
- when the prompt *julia* is ready, type in the terminal
  1. `using vOptSpecific`
  2. `using vOptGeneric`
  3. `using GLPK`

  Remark 1: you may invoke only `using vOptSpecific` if you are only working with `vOptSpecific`. Same remark for `vOptGeneric`.
  Remark 2: you may invoke CPLEX or GUROBI in the place of GLPK (ps: in local mode, the MILP solver selected must be properly installed)
- vOptSpecific and vOptGeneric are ready. See examples for further informations

vOptSpecific problem by problem

definition (problem, inputs)
data (console, text file)
resolution (API, outputs, text file)

$$
\left[\begin{array}{lcll}
\min z^k & = & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}^{k} x_{ij} & k = 1, \ldots, p \\
s/c & & \sum_{i=1}^{n} x_{ij} = 1 & j = 1, \ldots, n \\
 & & \sum_{j=1}^{n} x_{ij} = 1 & i = 1, \ldots, n \\
 & & x_{ij} = (0, 1) & \begin{array}{l} i = 1, \ldots, n \\ j = 1, \ldots, n \end{array}
\end{array}\right] \qquad \text{(p-LAP)}
$$

LAP | Definition | Inputs

Valid for 2-LAP.

- `n` (integer):
  number n of assignments task-resource
- `C1` (matrix of $n \times n$ of integers):
  coefficients c_{ij}^1 of the objective 1
- `C2` (matrix of $n \times n$ of integers):
  coefficients c_{ij}^2 of the objective 2

LAP | Data | Example (console)

```
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   ]
```

LAP | Data | Example (text file)

```
5
 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
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
```

n

C1

C2

- load2LAP
  Load an instance of a 2-LAP from the file `fname`
  `MOCO = load2LAP( fname )`

LAP | Resolution | API

- set2LAP
  Create a new instance of a 2-LAP and set up all required values
  `MOCO = set2LAP( n , C1, C2 )`

- LAP_Przybylski2008
  Set up the solver to use for the 2-LAP
  `solver = LAP_Przybylski2008( )`

- vSolve
  Solve the instance provided with the mentioned solver and return the results
  `z1, z2, σ = vSolve( MOCO , solver )`

LAP | Resolution | Outputs (specification)

Valid for 2-LAP.

vSolve returns:

- z1: vector of $(1, \ldots, \mid Y_N \mid)$ of integers
- z2: vector of $(1, \ldots, \mid Y_N \mid)$ of integers
- σ: matrix of $(1, \ldots, \mid Y_N \mid ; \sigma_1, \ldots, \sigma_n)$ of integers where
  - σ_i: a permutation coding $(x_{ij} = 1 \Leftrightarrow \sigma_i = j)$

OSP | Definition | One machine scheduling problem

The general one machine scheduling problem considered here is defined as:

$$1 \mid r_i \mid (f_1, f_2) \tag{2-OSP}$$

and the specific OSP currently considered is:

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

OSP | Definition | Inputs

Valid for 2-OSP.

- `n` (integer):
  number n of jobs, $i = 1 \ldots n$
- `r` (vector of n integers):
  r_i, the release date for job i
- `p` (vector of n integers):
  p_i, the processing time for job i
- `d` (vector of n integers):
  d_i, the due date for job i
- `w` (vector of n integers):
  w_i, the weight associated to job i

OSP | Data | Example (console)

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

OSP | Data | Example (text file)

4				n
2	4	3	1	p
1	2	4	6	d
0	0	0	0	r
1	1	1	1	w

- load2OSP
  Load an instance of a 2-OSP from the file `fname`
  `id = load2OSP( fname )`

OSP | Resolution | API

- set2OSP
  Create a new instance of a 2-OSP and set up all required values
  `id = set2OSP( n , P, D, R, W )`
- OSP_vanwassenhove1980
  Set up the solver to use for the 2-OSP
  `solver = OSP_vanwassenhove1980( )`
- vSolve
  Solve an instance of a 2OSP with the mentioned solver and return the results
  `status = vSolve( id , solver )`

OSP | Resolution | Outputs (specification)

Valid for 2-OSP.

vSolve returns:

- z1: vector of $(1, \ldots, | Y_N |)$ of integers
- z2: vector of $(1, \ldots, | Y_N |)$ of integers
- σ: matrix of $(1, \ldots, | Y_N | ; \sigma_1, \ldots, \sigma_n)$ of integers where
  - σ_i: a permutation coding ($\sigma_i = j \Leftrightarrow$ job j in position i)

vOptGeneric problem by problem

definition (problem, inputs, JuMP)
data (console, text file)
resolution (solve, outputs, text file)

MOMILP-MOLP-MOIP | Definition

Three Multi Objective Linear Optimization Problems:

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

Multi Objective
Mixed-Integer Linear
Problem (MOMILP)

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

Multi Objective
Linear
Problem (MOLP)

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

Multi Objective
Integer
Problem (MOIP)

where:

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

MOMILP-MOLP-MOIP | Formulation

Following specifications of JuMP for formulating a model, see:

- `http://jump.readthedocs.io/en/latest/quickstart.html#creating-a-model`

Plus:

- `@addObjective( <model>, <opt>, <function>)`

  where

  - model: id of the model concerned
  - opt: function to `Min` imize or `Max` imize
  - function: definition of the function

MOMILP-MOLP-MOIP | Example of formulation

$$\begin{array}{ll} \min & c_1^1 x_1 + c_2^1 x_2 \\ \min & c_1^2 x_1 + c_2^2 x_2 \\ s/t & t_{11} x_1 + t_{12} x_2 \leq d_1 \\ & t_{21} x_1 + t_{22} x_2 \leq d_2 \\ & x_1 \in \mathbb{R}, x_2 \in \mathbb{Z} \end{array}$$

```
# --- create a MOMILP and set the model ---
  MOMILP = vModel(solver=<working on it>)
  @variable(MOMILP, x1 >= 0)
  @variable(MOMILP, x2 >= 0, Int)
  @addObjective(MOMILP, Min, c11*x1 + c12*x1)
  @addObjective(MOMILP, Min, c21*x1 + c22*x1)
  @constraint(MOMILP, t11*x1 + t12*x2 <= d1)
  @constraint(MOMILP, t21*x1 + t22*x2 <= d2)
```

$$\begin{array}{ll} \min & c_1^1 x_1 + c_2^1 x_2 \\ \min & c_1^2 x_1 + c_2^2 x_2 \\ s/t & t_{11} x_1 + t_{12} x_2 \leq d_1 \\ & t_{21} x_1 + t_{22} x_2 \leq d_2 \\ & x_1, x_2 \in \mathbb{R} \end{array}$$

```
# --- create a MOLP and set the model ---
  MOLP = vModel(solver=<working on it>)
  @variable(MOMILP, x1 >= 0)
  @variable(MOMILP, x2 >= 0)
  @addObjective(MOLP, Min, c11*x1 + c12*x1)
  @addObjective(MOLP, Min, c21*x1 + c22*x1)
  @constraint(MOLP, t11*x1 + t12*x2 <= d1)
  @constraint(MOLP, t21*x1 + t22*x2 <= d2)
```

$$\begin{array}{ll} \min & c_1^1 x_1 + c_2^1 x_2 \\ \min & c_1^2 x_1 + c_2^2 x_2 \\ s/t & t_{11} x_1 + t_{12} x_2 \leq d_1 \\ & t_{21} x_1 + t_{22} x_2 \leq d_2 \\ & x_1, x_2 \in \mathbb{Z} \end{array}$$

```
# --- create a MOIP and set the model ---
  MOIP = vModel(solver=GLPKSolverMIP())
  @variable(MOIP, x1 >= 0, Int)
  @variable(MOIP, x2 >= 0, Int)
  @addObjective(MOIP, Min, c11*x1 + c12*x1)
  @addObjective(MOIP, Min, c21*x1 + c22*x1)
  @constraint(MOIP, t11*x1 + t12*x2 <= d1)
  @constraint(MOIP, t21*x1 + t22*x2 <= d2)
```

MOMILP-MOLP-MOIP | Data | Example (console)

```
c11 =  3   ;  c12 =  1
c21 = -1   ;  c22 = -2

t11 =  0   ;  t12 =  1   ;  d1 = 3
t21 =  3   ;  t22 = -1   ;  d2 = 6
```

MOMILP-MOLP-MOIP | Data | Example (format MOP)

parseMOP

Load an instance (MOP format) from the file `fname`

`jumpModel = parseMOP( fname , solver=<solver to invoke> )`

writeMOP

Write a model according to the MOP format to a file `fname`

`writeMOP( vModel , fname )`

NB: the MOP format is an extension of

- the MPS format
  (`https://en.wikipedia.org/wiki/MPS_(format)`)
- to multiple objectives
  (`http://moplib.zib.de/format_desc/mop_format.pdf`)

MOMILP-MOLP-MOIP | Example of resolution

MOMILP
Algorithm: branch&bound
Solver: working on it

```
# --- solve the model ---
  status = solve( MOMILP )

# --- Print the results ---
  print_X_E( MOMILP )
```

MOLP
Algorithm: parametric simplex
Solver: working on it

```
# --- solve the model ---
  status = solve( MOLP )

# --- Print the results ---
  print_X_E( MOLP )
```

MOIP
Algorithm: ϵ-constraint
Solver: GLPK or CPLEX

```
# --- solve the model ---
  status = solve( MOIP , method =:epsilon , step = 0.5 )

# --- Print the results ---
  print_X_E( MOIP )

# --- Get the results ---
  getY_N( MOIP )
```

MOMILP-MOLP-MOIP | Resolution | Outputs (specification)

Y_N of the 2-MOP is defined by

- 2-MILP: a mixed nondominated set (composed of edges that are either closed, half-open, open or reduced to a point)
- 2-LP: a continuous nondominated set (composed of edges)
- 2-IP: a discrete nondominated set (composed of points)

Technical overview

Technical overview ....

User application written in Julia

Declare

No-structured MOP

IP/MIP/LP

Structured MOP

LAP

$1 \mid r_i \mid (f_1, f_2)$

Solve

Specific

2-LAP $\rightarrow$ Y_N

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

Generic

2-ILP $\rightarrow$ Y_{SN}
dicho

2-ILP $\rightarrow$ Y_N
Chalmet

2-MILP $\rightarrow$ Y_N
B&C

2-ILP $\rightarrow$ Y_N
ϵ-cte

2-LP $\rightarrow$ Y_N
simplex

Utils

Input/Output

read/write format MOP
plot Y_N

Misc

maintain Y_N
analyze Y_N

Interface:

Library solvers
C/C++ specifics

Modules C

LAP

Interface:

Library solvers
MIP generics

Solvers MIP

open source
GLPK
commercial
CPLEX, GUROBI

History

History

- v1.0: July 2016; first prototype; not opened to the public
- v2.0: June 23, 2017; first stable version; released to the public with
  - a depot on github, webpages, and a tutorial
  - JuMP extended to multiple objectives (vOptGeneric)
  - primitives for loading/printing/writing a non-structured problem in std MOP format
  - ϵ-constraint algorithm for bi-objective integer programming
  - backbone in Julia (vOptSpecific)
  - API and solver for bi-objective linear assignment problem
  - API and solver for bi-objective one machine scheduling problem
  - primitives for I/O of a 2LAP and 2OSP on files

Taskforce

Taskforce

Involved in the development of vOptSolver:

Currently:

- GANDIBLEUX Xavier (coordinator)
- SOLEILHAC Gauthier
- PRZYBYLSKI Anthony

Previously:

- CHATELLIER Pauline
- DUMEZ Dorian
- LUCAS Flavien

Links and contact

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