Julia tutorial (3)
MultiObjective Metaheuristic Implementation

Prof. Dr. Antonio J. Nebro[1] and Prof. Dr. Xavier Gandibleux[2]

1: Universidad de Málaga, Spain
2: Nantes Université, France

June 2024

Multiobjective Optimization Problem (MOP)

$$\begin{aligned} \min \quad & F(x) \\ s.t. \quad & x \in X \end{aligned}$$

where:

$$\begin{aligned} X \subseteq \mathbb{R}^n \quad &\longrightarrow \quad \text{the set of feasible solutions} \\ Y = F(X) \subseteq \mathbb{R}^p \quad &\longrightarrow \quad \text{the set of images} \end{aligned}$$

Computing sets Y_N and X_E for MOP:

$$y = F(x)$$

$y^* \in Y$ is nondominated, if $\nexists\, y \in Y$ such that $y_i \leqslant y_i^*,\ \forall i$ and $y \neq y^*$.

Y_N is the set of nondominated points.

$x^* \in X$ is efficient if y^* is nondominated.

X_E is a complete set of efficient solutions.

Multiobjective Optimization Problem (MOP)

$$\begin{aligned} \min \quad & F(x) \\ s.t. \quad & x \in X \end{aligned}$$

where:

$$\begin{aligned} X \subseteq \mathbb{R}^n \quad &\longrightarrow \quad \text{the set of feasible solutions} \\ Y = F(X) \subseteq \mathbb{R}^p \quad &\longrightarrow \quad \text{the set of images} \end{aligned}$$

Computing sets Y_N and X_E for MOP:

$$y = F(x)$$

$y^* \in Y$ is nondominated, if $\nexists\, y \in Y$ such that $y_i \leqslant y_i^*,\ \forall i$ and $y \neq y^*$.

Y_N is the set of nondominated points.

$x^* \in X$ is efficient if y^* is nondominated.

X_E is a complete set of efficient solutions.

Example 1

Compute X_E and Y_N for the following 2-IP:

$$
\begin{array}{lrrrrrr}
\max z_1 & = & x_1 & + & x_2 & & \\
\min z_2 & = & x_1 & + & 3x_2 & & \\
s.t. & & 2x_1 & + & 3x_2 & \leqslant & 30 \\
 & & 3x_1 & + & 2x_2 & \leqslant & 30 \\
 & & x_1 & - & x_2 & \leqslant & 5.5 \\
 & & x_1 & , & x_2 & \in & \mathbb{N}
\end{array}
$$

1. Exact resolution

Exact resolution

Easy in Julia with these packages:

- Model:
  `JuMP`
- Algorithm:
  `MultiObjectiveAlgorithms`
- Solver:
  `HiGHS`, `Gurobi`, etc.

From `vOptGeneric.jl` to `MultiObjectiveAlgorithms.jl`

vOptSolver (vOptGeneric and vOptSpecific)
`https://github.com/vOptSolver`

2015: Kick-off of vOpt ANR/DFG project
2017: Introduction of vOptSolver (vOptGeneric, vOptSpecific)

> Xavier Gandibleux, Gauthier Soleilhac, Anthony Przybylski, Stefan Ruzika. vOptSolver: an open source software environment for multiobjective mathematical optimization. *IFORS2017: 21st Conference of the International Federation of Operational Research Societies*. July 17-21, 2017. Quebec City (Canada).

2018: Julia 1.0
2019: End of vOpt project
2019: First discussions on discourse/new design of MOO in JuMP
2022: JuMP 1.0
2023: Introduction of MultiObjectiveAlgorithms
2023: End of vOptGeneric

JuMP
`https://jump.dev/`

> Miles Lubin. The state of JuMP. *JuliaCon 2023: The 10th Annual JuliaCon - Co-Located with JuMP-dev*, July 25th - 29th, 2023. MIT, Cambridge, (USA).

MultiObjectiveAlgorithms (MOA)
`https://github.com/jump-dev/MultiObjectiveAlgorithms.jl`

> Oscar Dowson, Xavier Gandibleux, Gokhan Kof. From vOptGeneric.jl to MultiObjectiveAlgorithms.jl *JuliaDays 2023*, 4-6 October 2023. Paris (France).

From `vOptGeneric.jl` to `MultiObjectiveAlgorithms.jl`

vOptSolver (vOptGeneric and vOptSpecific)
`https://github.com/vOptSolver`

2015: Kick-off of vOpt ANR/DFG project
2017: Introduction of vOptSolver (vOptGeneric, vOptSpecific)

> Xavier Gandibleux, Gauthier Soleilhac, Anthony Przybylski, Stefan Ruzika. vOptSolver: an open source software environment for multiobjective mathematical optimization. *IFORS2017: 21st Conference of the International Federation of Operational Research Societies*. July 17-21, 2017. Quebec City (Canada).

2018: Julia 1.0
2019: End of vOpt project
2019: First discussions on discourse/new design of MOO in JuMP
2022: JuMP 1.0
2023: Introduction of MultiObjectiveAlgorithms
2023: End of vOptGeneric

JuMP
`https://jump.dev/`

> Miles Lubin. The state of JuMP. *JuliaCon 2023: The 10th Annual JuliaCon - Co-Located with JuMP-dev*, July 25th - 29th, 2023. MIT, Cambridge, (USA).

MultiObjectiveAlgorithms (MOA)
`https://github.com/jump-dev/MultiObjectiveAlgorithms.jl`

> Oscar Dowson, Xavier Gandibleux, Gokhan Kof. From vOptGeneric.jl to MultiObjectiveAlgorithms.jl *JuliaDays 2023*, 4-6 October 2023. Paris (France).

Getting Started

Install:

```
julia> using Pkg
```

```
julia> Pkg.add("JuMP")
```

```
julia> Pkg.add("MultiObjectiveAlgorithms")
```

```
julia> Pkg.add("HiGHS")
```

Getting Started

Install:

```
julia> using Pkg
```

```
julia> Pkg.add("JuMP")
```

```
julia> Pkg.add("MultiObjectiveAlgorithms")
```

```
julia> Pkg.add("HiGHS")
```

Getting Started

Install:

```
julia> using Pkg
```

```
julia> Pkg.add("JuMP")
```

```
julia> Pkg.add("MultiObjectiveAlgorithms")
```

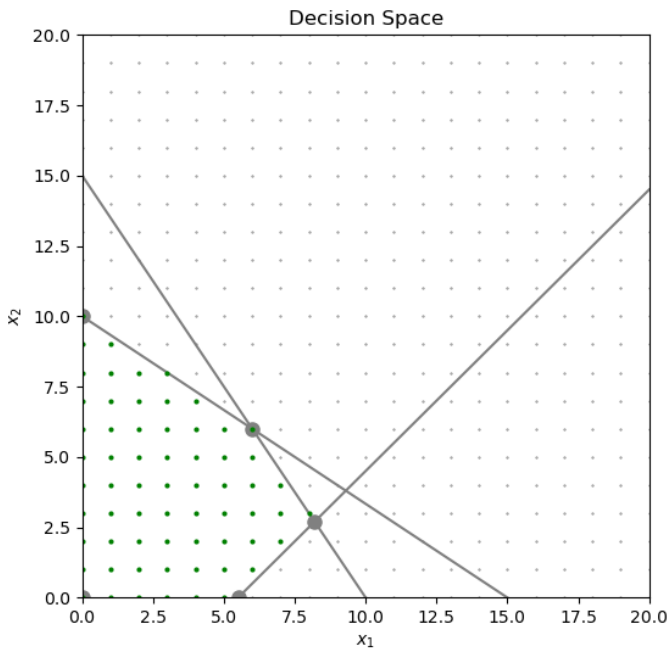
```
julia> Pkg.add("HiGHS")
```

Example 1 (Con't)

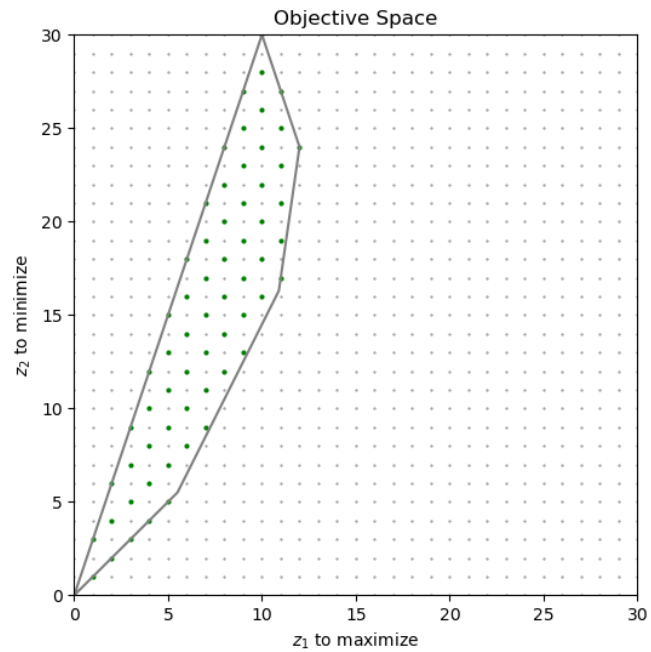
$$
\begin{array}{lcrcrcl}
\max z_1 & = & x_1 & + & x_2 & & \\
\min z_2 & = & x_1 & + & 3x_2 & & \\
s.t. & & 2x_1 & + & 3x_2 & \leqslant & 30 \\
 & & 3x_1 & + & 2x_2 & \leqslant & 30 \\
 & & x_1 & - & x_2 & \leqslant & 5.5 \\
 & & x_1 & , & x_2 & \in & \mathbb{N}
\end{array}
$$

```
julia> c1 = [1,1]
julia> c2 = [1,3]
julia> A = [2 3; 3 2; 1 -1]
julia> b = [30, 30, 5.5]
```

Decision space:



Objective space:



Example 1 (Con't)

Write the corresponding 2-IP model with JuMP

```
julia> using JuMP, HiGHS
julia> import MultiObjectiveAlgorithms as MOA
```

```
julia> model = Model( )
julia> @variable(model, x1⩾0, Int)
julia> @variable(model, x2⩾0, Int)
julia> @expression(model, fct1, x1 + x2)                    # to maximize
julia> @expression(model, fct2, x1 + 3 * x2)                # to minimize
julia> @objective(model, Max, [fct1, (-1) * fct2]))
julia> @constraint(model, 2*x1 + 3*x2 ⩽ 30))
julia> @constraint(model, 3*x1 + 2*x2 ⩽ 30))
julia> @constraint(model, x1 - x2 ⩽ 5.5))
```

```
julia> print(model)
```


Example 1 (Con't)

Setup the MIP solver, e.g. `HiGHS`

```
julia> set_optimizer(model,()->MOA.Optimizer(HiGHS.Optimizer))
```

Setup the algorithm: ϵ-constraint; step=1 (default value)

```
julia> set_attribute(model, MOA.Algorithm(), MOA.EpsilonConstraint())
```

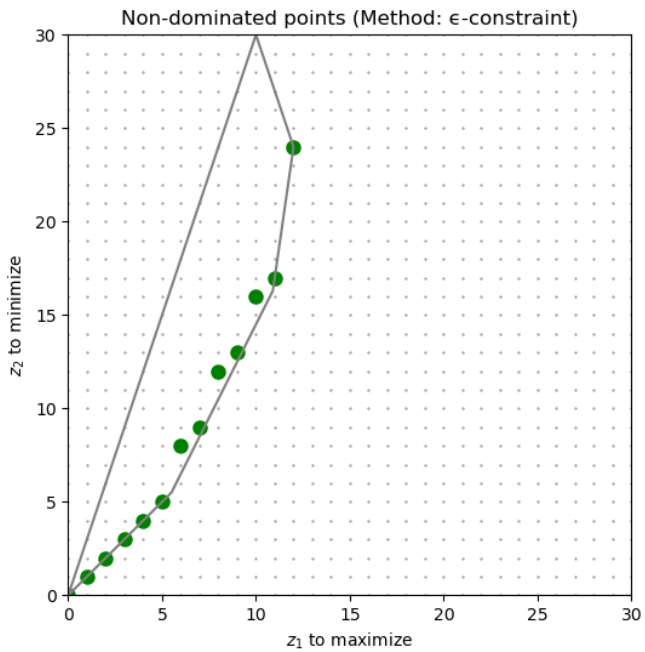
Optimize the MOO problem

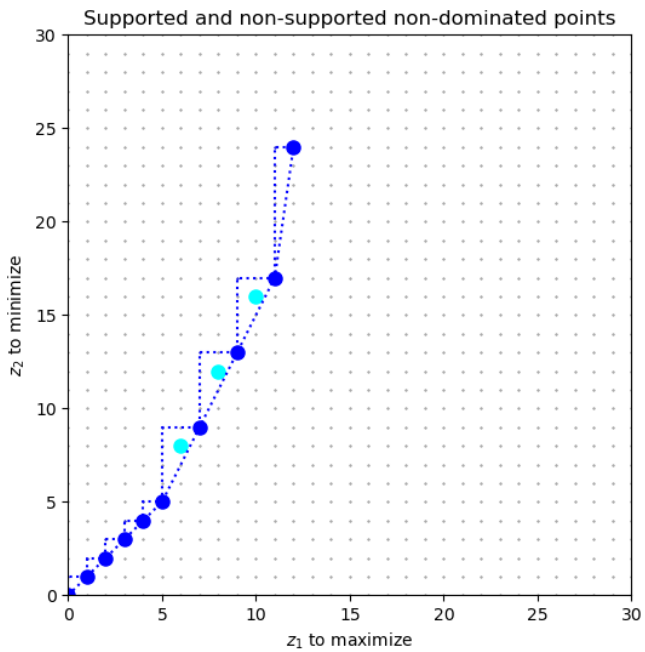
```
julia> optimize!(model)
```

Example 1 (Con't)

Get X_E and Y_N

```
julia> for i in 1:result_count(model)
julia>     z1_opt = objective_value(model; result = i)[1]
julia>     z2_opt = -1 * objective_value(model; result = i)[2]
julia>     x1_opt = value(x1; result = i)
julia>     x2_opt = value(x2; result = i)
julia> end
```





1. Heuristic resolution

You enter on the scene Antonio...

During your talk, could you illustrate the resolution of the example 1 with MetaJul?

References

- Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B. Shah. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59: 65–98. 2017.
- Miles Lubin, Oscar Dowson, Joaquim Dias Garcia, Joey Huchette, Benoît Legat, Juan Pablo Vielma. JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*. 2023.
- JuMP
  `https://jump.dev/`
- MultiObjectiveAlgorithms (MOA)
  `https://github.com/jump-dev/MultiObjectiveAlgorithms.jl`