

Julia Tutorial (3)

MultiObjective Metaheuristic Implementation

Part II: non-exact methods

<https://github.com/jMetal/Julia/resources>

Antonio J. Nebro
Xavier Gandibleux

University of Málaga – SPAIN
Nantes Université – FRANCE



MultiObjective Metaheuristics

- When a multi-objective problem cannot be solved with exact techniques the alternative is to use non-exact methods, such as metaheuristics

Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison

CHRISTIAN BLUM

Université Libre de Bruxelles

AND

ANDREA ROLI

Università degli Studi di Bologna



"Metaheuristics are high level strategies for exploring search spaces by using different methods."

ACM Computing Surveys, Vol. 35, No. 3, September 2003.

MultiObjective MOEAs

- The question is how to implement a multi-objective metaheuristic
 - Multi-Objective Evolutionary Algorithms (MOEAs)
- The adopted scheme will influence in
 - Code easy to understand
 - Efficiency
 - Reusability and extensibility

MultiObjective MOEAs

- Our focus:
 - Implementation of NSGA-II in Julia
 - Solving the example problem

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

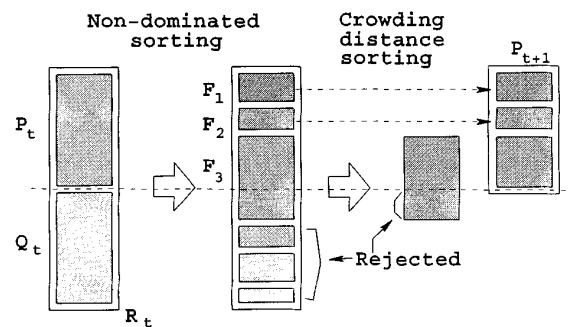
MultiObjective MOEAs

182

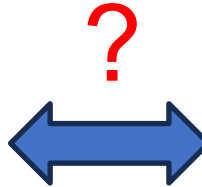
IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 6, NO. 2, APRIL 2002

A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II

Kalyanmoy Deb, Associate Member, IEEE, Amrit Pratap, Sameer Agarwal, and T. Meyarivan



$R_t = P_t \cup Q_t$	combine parent and offspring population
$\mathcal{F} = \text{fast-nondominated-sort}(R_t)$	$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \dots)$, all nondominated fronts of R_t
$P_{t+1} = \emptyset$ and $i = 1$	
until $ P_{t+1} + \mathcal{F}_i \leq N$	until the parent population is filled
$\text{crowding-distance-assignment}(\mathcal{F}_i)$	calculate crowding-distance in \mathcal{F}_i
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i$	include i th nondominated front in the parent pop
$i = i + 1$	check the next front for inclusion
Sort(\mathcal{F}_i, \prec_n)	sort in descending order using \prec_n
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - P_{t+1})]$	choose the first $(N - P_{t+1})$ elements of \mathcal{F}_i
$Q_{t+1} = \text{make-new-pop}(P_{t+1})$	use selection, crossover and mutation to create a new population Q_{t+1}
$t = t + 1$	increment the generation counter

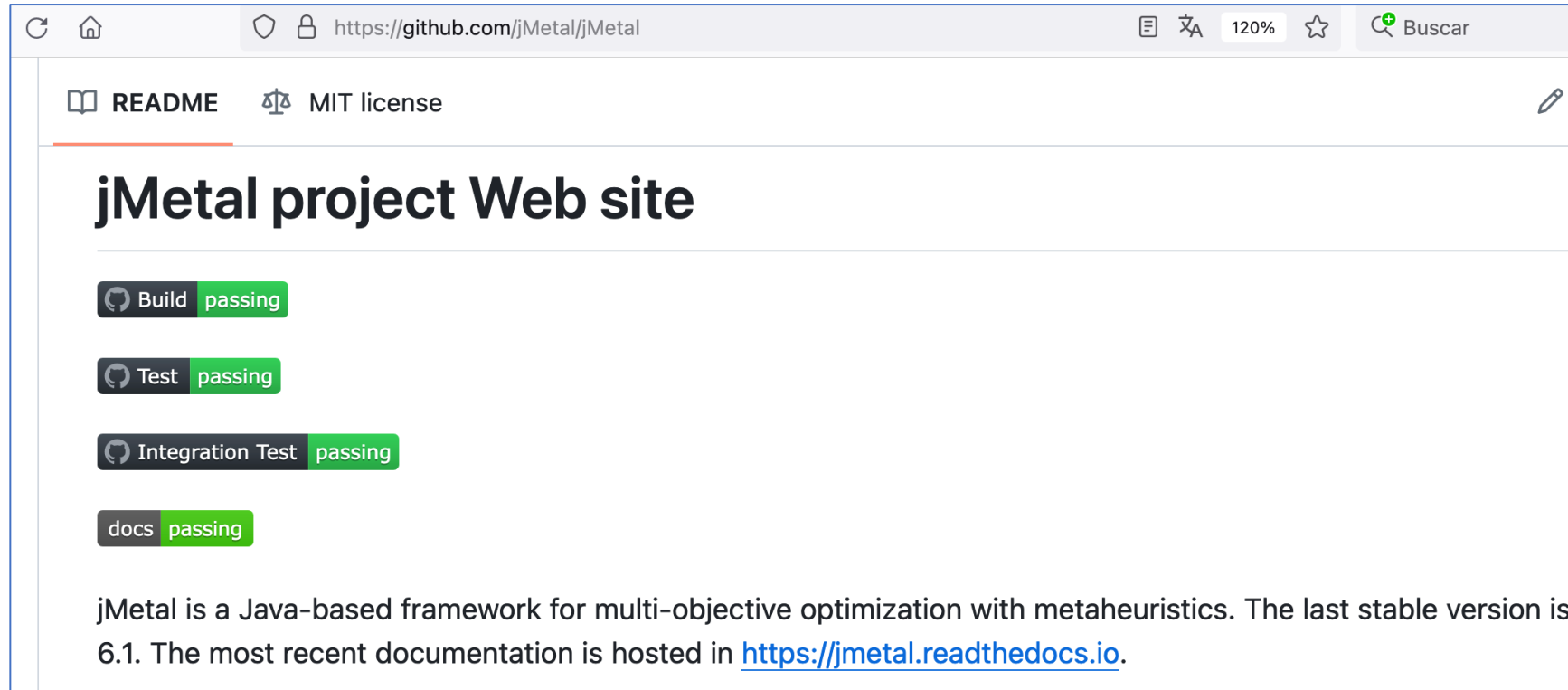


Algorithm 1 Pseudo-code of an evolutionary algorithm

```

1:  $P(0) \leftarrow \text{GenerateInitialSolutions}()$ 
2:  $t \leftarrow 0$ 
3: Evaluate( $P(0)$ )
4: while not StoppingCriterion() do
5:    $M(t) \leftarrow \text{Select}(P(t))$ 
6:    $Q(t) \leftarrow \text{Variate}(M(t))$ 
7:   Evaluate( $Q(t)$ )
8:    $P(t+1) \leftarrow \text{Update}(P(t), Q(t))$ 
9:    $t \leftarrow t + 1$ 
10: end while
  
```

Implementing NSGA-II in jMetal

A screenshot of the GitHub repository page for jMetal. The browser address bar shows "https://github.com/jMetal/jMetal". The page has tabs for "README" and "MIT license". The main heading is "jMetal project Web site". Below this, there are four status bars: "Build passing", "Test passing", "Integration Test passing", and "docs passing". At the bottom, a paragraph states: "jMetal is a Java-based framework for multi-objective optimization with metaheuristics. The last stable version is 6.1. The most recent documentation is hosted in <https://jmetal.readthedocs.io>."

<https://github.com/jMetal>

Implementing NSGA-II in jMetal

- Three alternatives
 - Monolithic
 - jMetal 4.3 (2012): <https://github.com/jMetal/jMetal/blob/v4.3/jmetal/metaheuristics/nsgaii/NSGAI.java>
 - Object-oriented
 - jMetal 5.0 (2015):
 - <https://github.com/jMetal/jMetal/blob/main/jmetal-core/src/main/java/org/uma/jmetal/algorithm/impl/AbstractEvolutionaryAlgorithm.java>
 - <https://github.com/jMetal/jMetal/blob/main/jmetal-algorithm/src/main/java/org/uma/jmetal/algorithm/multiobjective/nsgaii/NSGAI.java>
 - Component-based
 - jMetal 6.0 (2022)
 - <https://github.com/jMetal/jMetal/blob/main/jmetal-component/src/main/java/org/uma/jmetal/component/algorithm/EvolutionaryAlgorithm.java>
 - <https://github.com/jMetal/jMetal/blob/main/jmetal-component/src/main/java/org/uma/jmetal/component/algorithm/multiobjective/NSGAIBuilder.java>

Implementing NSGA-II in jMetal

Algorithm 1 Pseudo-code of an evolutionary algorithm

```

1:  $P(0) \leftarrow \text{GenerateInitialSolutions}()$ 
2:  $t \leftarrow 0$ 
3: Evaluate( $P(0)$ )
4: while not StoppingCriterion() do
5:    $M(t) \leftarrow \text{Select}(P(t))$ 
6:    $Q(t) \leftarrow \text{Variate}(M(t))$ 
7:   Evaluate( $Q(t)$ )
8:    $P(t+1) \leftarrow \text{Update}(P(t), Q(t))$ 
9:    $t \leftarrow t+1$ 
10: end while

```

```

public class EvolutionaryAlgorithm<S>
{
    public void run() {
        population = createInitialPopulation.create();
        population = evaluation.evaluate(population);
        initProgress();
        while (!termination.isMet(attributes)) {
            List<S> matingPopulation = selection.select(population);
            List<S> offspringPopulation = variation.variate(population, matingPopulation);
            offspringPopulation = evaluation.evaluate(offspringPopulation);

            population = replacement.replace(population, offspringPopulation);
            updateProgress();
        }
    }
}

```

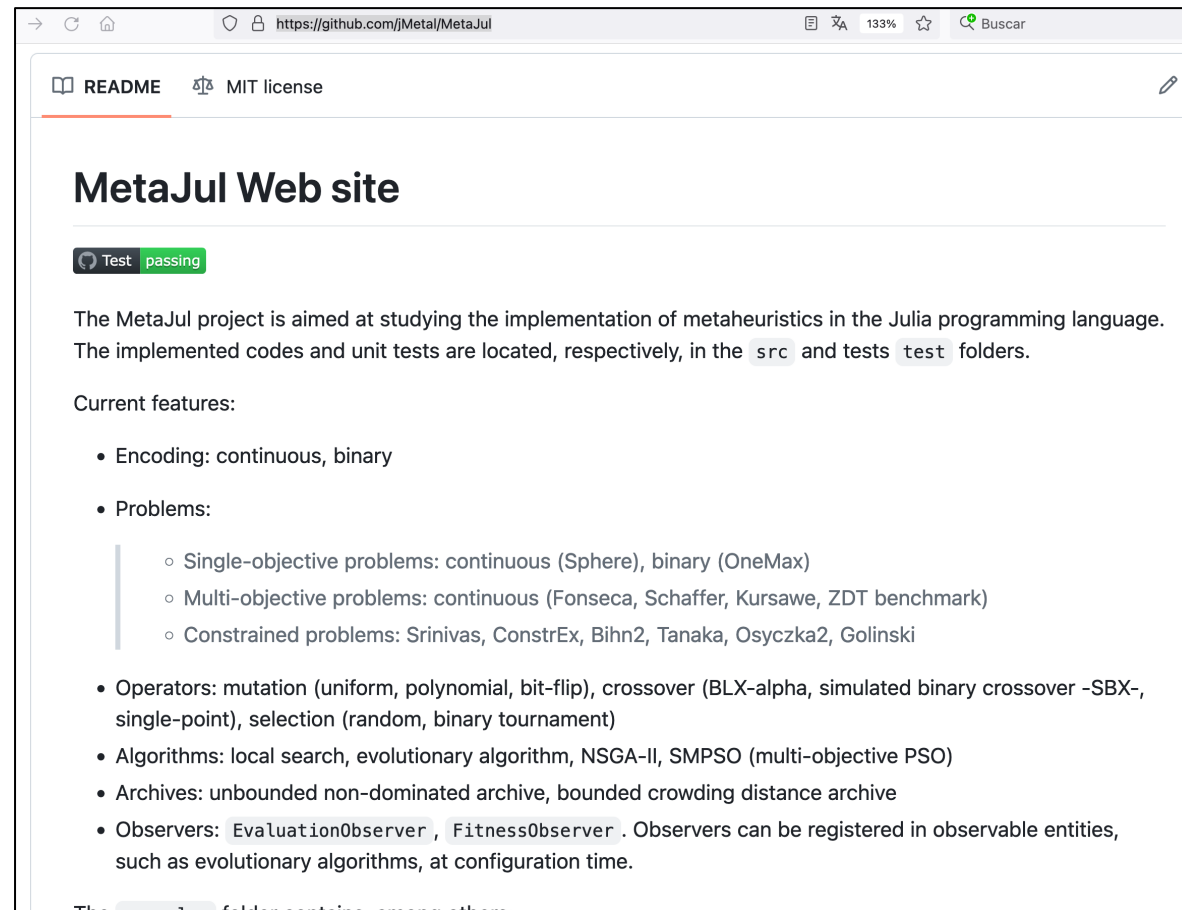
Component-based architecture

Component	Implementations
Solutions creation	Latin hypercube sampling Random Scatter Search
Evaluation	Sequential Multithreaded Sequential with external archive
Termination	Computing time Evaluations Keyboard Quality indicator
Selection	N-ary tournament Random Differential evolution Neighborhood Population and neighborhood
Variation	Crossover and mutation Differential evolution
Replacement	(μ, λ) $(\mu + \lambda)$ Ranking and density estimator Pairwise MOEA/D replacement strategy SMS-EMOA replacement strategy

Component catalog

Implementing NSGA-II in Julia

- MetaJul project



Implementing NSGA-II in Julia

MetaJul

```
public class EvolutionaryAlgorithm<S>
{
    public void run() {
        population = createInitialPopulation.create();
        population = evaluation.evaluate(population);
        initProgress();
        while (!termination.isMet(attributes)) {
            List<S> matingPopulation = selection.select(population);
            List<S> offspringPopulation = variation.variate(population, matingPopulation);
            offspringPopulation = evaluation.evaluate(offspringPopulation);

            population = replacement.replace(population, offspringPopulation);
            updateProgress();
        }
    }
}
```

jMetal



Algorithm 1 Pseudo-code of an evolutionary algorithm

```
1:  $P(0) \leftarrow \text{GenerateInitialSolutions}()$ 
2:  $t \leftarrow 0$ 
3: Evaluate( $P(0)$ )
4: while not StoppingCriterion() do
5:    $M(t) \leftarrow \text{Select}(P(t))$ 
6:    $Q(t) \leftarrow \text{Variate}(M(t))$ 
7:   Evaluate( $Q(t)$ )
8:    $P(t+1) \leftarrow \text{Update}(P(t), Q(t))$ 
9:    $t \leftarrow t+1$ 
10: end while
```



```
mutable struct EvolutionaryAlgorithm <: Algorithm
    name::String

    foundSolutions::Vector

    solutionsCreation::SolutionsCreation
    evaluation::Evaluation
    termination::Termination
    selection::Selection
    variation::Variation
    replacement::Replacement
end

function evolutionaryAlgorithm(ea::EvolutionaryAlgorithm)
    population = create(ea.solutionsCreation)
    population = evaluate(ea.evaluation, population)

    evaluations = length(population)

    while !isMet(ea.termination, ea.status)
        matingPool = select(ea.selection, population)
        offspringPopulation = variate(ea.variation, population, matingPool)
        offspringPopulation = evaluate(ea.evaluation, offspringPopulation)
        population = replace_(ea.replacement, population, offspringPopulation)

        evaluations += length(offspringPopulation)
    end
    foundSolutions = population
    return foundSolutions
end
```

Implementing the problem in MetaJul

```
problem = ContinuousProblem{Int64}("integerProblem")

addVariable(problem, Bounds{Int64}(0, 20))
addVariable(problem, Bounds{Int64}(0, 20))

f1 = x -> -1.0 * (x[1] + x[2])
f2 = x -> x[1] + 3 * x[2]

addObjective(problem, f1)
addObjective(problem, f2)

c1 = x -> -2 * x[1] - 3 * x[2] + 30.0
c2 = x -> -3 * x[1] - 2 * x[2] + 30.0
c3 = x -> -x[1] + x[2] + 5.5

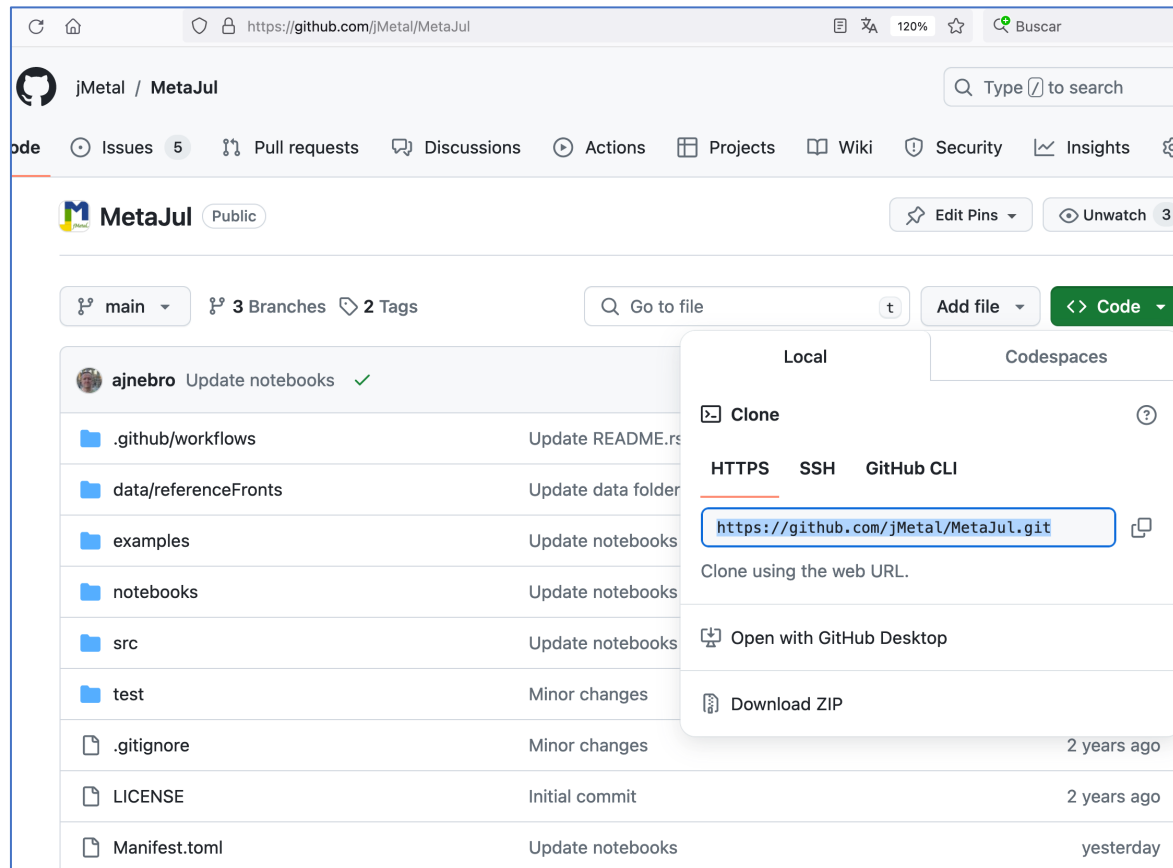
addConstraint(problem, c1)
addConstraint(problem, c2)
addConstraint(problem, c3)
```

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

```
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)
```

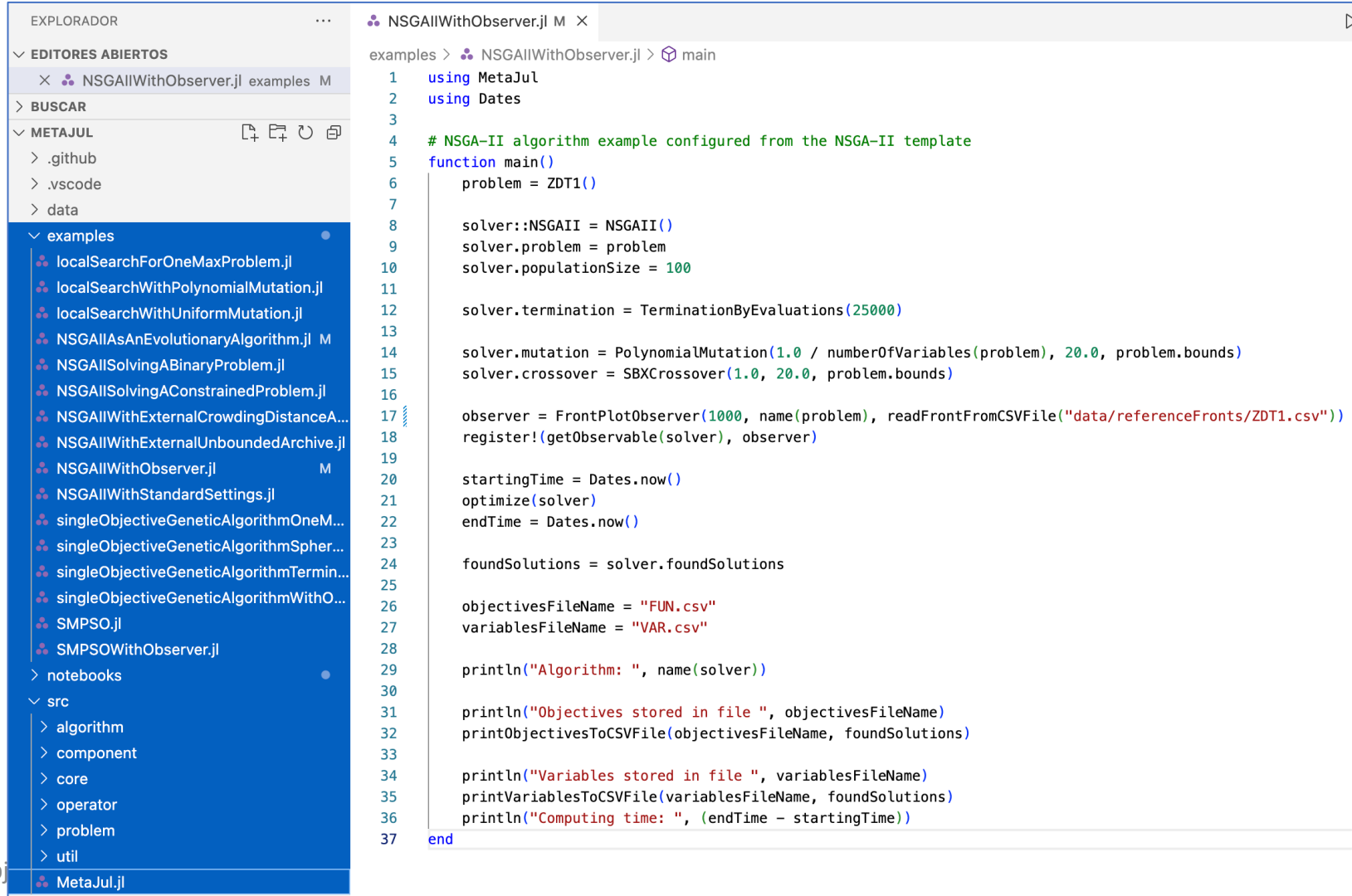
Getting starting with MetaJul

- Download the project from <https://github.com/jMetal/MetaJul>



```
MetaJul — -zsh — 105x25
(base) ajnebro@MacBook-Pro-115 mic2024 % git clone https://github.com/jMetal/MetaJul.git
Clonando en 'MetaJul'...
remote: Enumerating objects: 2704, done.
remote: Counting objects: 100% (1496/1496), done.
remote: Compressing objects: 100% (867/867), done.
remote: Total 2704 (delta 889), reused 1208 (delta 609), pack-reused 1208
Recibiendo objetos: 100% (2704/2704), 17.00 MiB | 1.84 MiB/s, listo.
Resolviendo deltas: 100% (1769/1769), listo.
(base) ajnebro@MacBook-Pro-115 mic2024 % cd MetaJul
(base) ajnebro@MacBook-Pro-115 MetaJul % ls
LICENSE          Project.toml     data             notebooks        test
Manifest.toml    README.rst      examples         src
(base) ajnebro@MacBook-Pro-115 MetaJul %
```

Project structure



The screenshot shows an IDE with a file explorer on the left and a code editor on the right. The file explorer displays the project structure, including a folder named 'examples' which contains several files, including 'NSGAIWithObserver.jl'. The code editor shows the content of 'NSGAIWithObserver.jl', which is a Julia script for the NSGA-II algorithm. The code includes imports for 'MetaJul' and 'Dates', a comment indicating it's an example configured from the NSGA-II template, and a 'main' function that sets up the solver, problem, and observer, and prints the results.

```

1  using MetaJul
2  using Dates
3
4  # NSGA-II algorithm example configured from the NSGA-II template
5  function main()
6      problem = ZDT1()
7
8      solver::NSGAI = NSGAI()
9      solver.problem = problem
10     solver.populationSize = 100
11
12     solver.termination = TerminationByEvaluations(25000)
13
14     solver.mutation = PolynomialMutation(1.0 / numberOfVariables(problem), 20.0, problem.bounds)
15     solver.crossover = SBXCrossover(1.0, 20.0, problem.bounds)
16
17     observer = FrontPlotObserver(1000, name(problem), readFrontFromCSVFile("data/referenceFronts/ZDT1.csv"))
18     register!(getObservable(solver), observer)
19
20     startingTime = Dates.now()
21     optimize(solver)
22     endTime = Dates.now()
23
24     foundSolutions = solver.foundSolutions
25
26     objectivesFileName = "FUN.csv"
27     variablesFileName = "VAR.csv"
28
29     println("Algorithm: ", name(solver))
30
31     println("Objectives stored in file ", objectivesFileName)
32     printObjectivesToCSVFile(objectivesFileName, foundSolutions)
33
34     println("Variables stored in file ", variablesFileName)
35     printVariablesToCSVFile(variablesFileName, foundSolutions)
36     println("Computing time: ", (endTime - startingTime))
37 end

```

Running NSGA-II from the command line



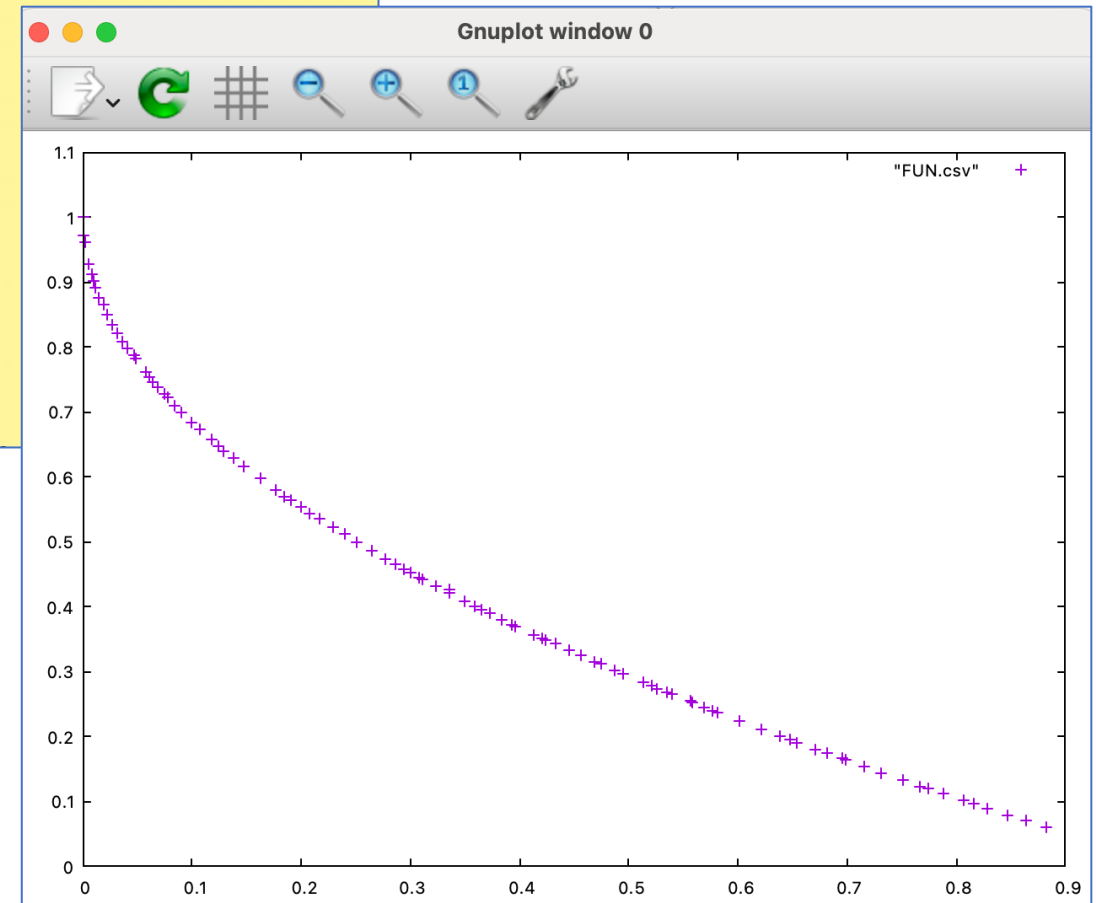
```
(base) ajnebro@MacBook-Pro-115 MetaJul % julia --project=. -L "examples/NSGAIAsAnEvolutionaryAlgorithm.jl" -e "main()"
Algorithm: NSGA-II
Computing time: 1701 milliseconds
Objectives stored in file FUN.csv
Variables stored in file VAR.csv
(base) ajnebro@MacBook-Pro-115 MetaJul % gnuplot

G N U P L O T
Version 6.0 patchlevel 0    last modified 2023-12-09

Copyright (C) 1986-1993, 1998, 2004, 2007-2023
Thomas Williams, Colin Kelley and many others

gnuplot home:      http://www.gnuplot.info
faq, bugs, etc:    type "help FAQ"
immediate help:    type "help" (plot window: hit 'h')

Terminal type is now qt
gnuplot> set datafile separator ','
gnuplot> plot "FUN.csv"
```



Running NSGA-II from the REPL

```

((base) ajnebro@MacBook-Pro-115 metajul % julia --project=.
Documentation: https://docs.julialang.org
Type "?" for help, "j?" for Pkg help.
Version 1.10.3 (2024-04-30)
Official https://julialang.org/ release

[julia> include("examples/NSGAIAsAnEvolutionaryAlgorithm.jl")
main (generic function with 1 method)

[julia> main()
Algorithm: NSGA-II
Computing time: 1370 milliseconds
Objectives stored in file FUN.csv
Variables stored in file VAR.csv

[julia> main()
Algorithm: NSGA-II
Computing time: 821 milliseconds
Objectives stored in file FUN.csv
Variables stored in file VAR.csv

[julia>
((base) ajnebro@MacBook-Pro-115 metajul % gnuplot

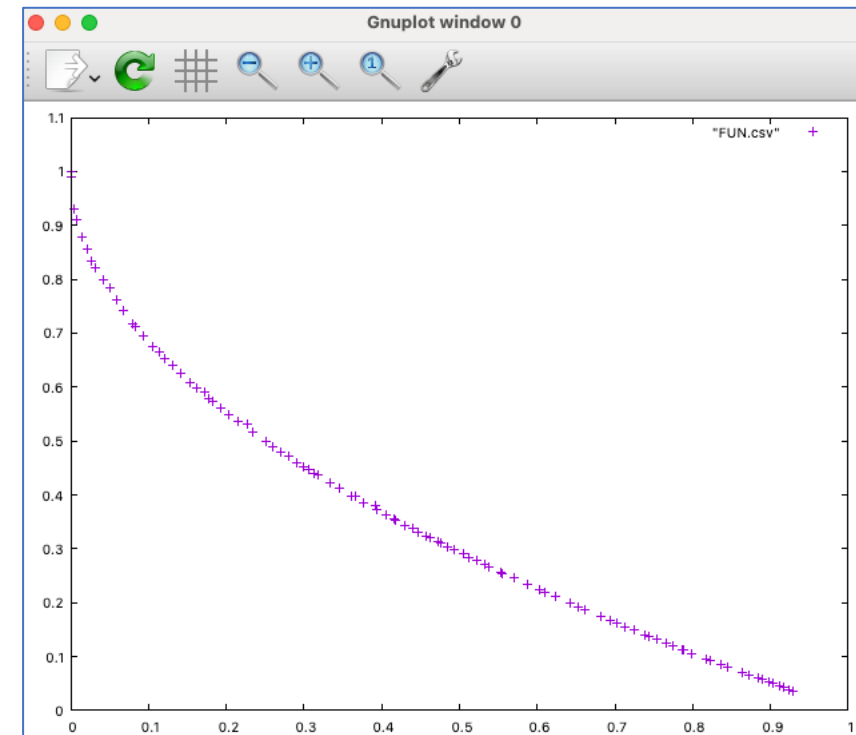
      G N U P L O T
      Version 6.0 patchlevel 0    last modified 2023-12-09

      Copyright (C) 1986-1993, 1998, 2004, 2007-2023
      Thomas Williams, Colin Kelley and many others

      gnuplot home:      http://www.gnuplot.info
      faq, bugs, etc:    type "help FAQ"
      immediate help:    type "help" (plot window: hit 'h')

      Terminal type is now qt
[gnuplot> set datafile separator ','
[gnuplot> plot "FUN.csv"
qt.qpa.fonts: Populating font family aliases took 57 ms. Replace uses of missing font family
_Sans" with one that exists to avoid this cost.
gnuplot>

```



Notebooks

EDITORES ABIERTOS

BUSCAR

METAJUL

.fleet

.github

.idea

.vscode

data

deps

examples

notebooks

.ipynb_checkpoints

NSGA-II notebook_files

Dealing with constrained problems.ipynb

Defining problems.ipynb

NSGA-II notebook.ipynb

NSGA-II with external archive.ipynb

src

test

.gitignore

FUN.csv

LICENSE

Manifest.toml

notebooks > Dealing with constrained problems.ipynb > M+ Dealing with problems having side-constraints

+ Código

+ Markdown

Ejecutar todo

Restart

Stop

Borrar todas las salidas

Esquema

...

Julia relea

Dealing with problems having side-constraints

In this notebook, we illustrate how to define a constrained problem and how it can be solved using NSGA-II. We assume that the included in the [Defining problems](#) notebook have been assimilated.

Defining a constrained problem

We use an example the problem known as Srinivas, which has the following formulation (see [DPA02](#)):):

EXPLORADOR

EDITORES ABIERTOS

BUSCAR

METAJUL

.fleet

.github

.idea

.vscode

data

deps

examples

notebooks

.ipynb_checkpoints

NSGA-II notebook_files

Dealing with constrained problems.ipynb

Defining problems.ipynb

NSGA-II notebook.ipynb

NSGA-II with external archive.ipynb

src

test

.gitignore

FUN.csv

LICENSE

Manifest.toml

p.jl

Project.toml

NSGA-II notebook.ipynb x Defining problems.ipynb

notebooks > NSGA-II notebook.ipynb > ...

+ Código

+ Markdown

Ejecutar todo

Restart

Stop

Borrar todas las salidas

Esquema

...

Julia relea

Examples of configuring and running NSGA-II

This notebook includes two examples where NSGA-II is used to solve a continuous and a binary multi-objective problems.

```

using MetaJul
using Dates

```

Configure the algorithm to solve a continuous problem (Kursawe)

Indicate the problem to solve

```

problem = kursawe()

```

```

ContinuousProblem{Float64}(Bounds{Float64}[Bounds{Float64}(-5.0, 5.0), Bounds{Float64}(-5.0, 5.0), Bounds{Float64}(-5.0, 5.0), Bounds{Float64}(-5.0, 5.0), Bounds{Float64}(-5.0, 5.0), Bounds{Float64}(-5.0, 5.0), Bounds{Float64}(-5.0, 5.0), Bounds{Float64}(-5.0, 5.0), Bounds{Float64}(-5.0, 5.0), Bounds{Float64}(-5.0, 5.0)])

```




Thanks for your attention