SSBSE'17

Paderborn, 9-11 September 2017

TUTORIAL

Multiobjective Optimization with the jMetal Framework. Applications to SBSE

Antonio J. Nebro
PhD in Computer Science

ajnebro@uma.es

UNIVERSIDAD DE MÁLAGA

Khaos INVESTIGACIÓN

About this tutorial

- jMetal is
  - A Java-based framework for multi-objective optimization with metaheuristics
  - Developed by computer science engineers to make research on multi-objective optimization
  - Open source project (MIT license)
- Web sites
  - Web page: http://jmetal.github.io/jMetal/
  - GitHub: https://github.com/jMetal/jMetal
  - Documentation: https://github.com/jMetal/jMetalDocumentation

About this tutorial

- jMetal and SBSE
  - jMetal is a tool used in SBSE
  - Reference:
    - "Pareto-optimal search-based software engineering (POSBSE): A literature survey. A.S. Sayyad, H. Ammar. 2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE). 2013"
    - jMetal was applied in 13 papers
      - Requirements, testing, debugging, software project management

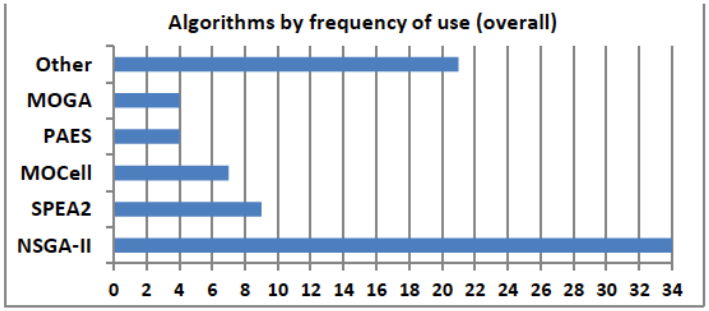


Figure 3: Algorithms by frequency of use

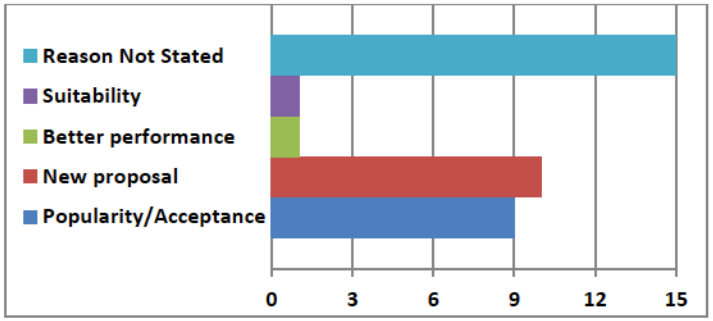


Figure 5: Reasons for adopting an algorithm

About this tutorial

- Purpose of this tutorial
  - Offer a practical view of the jMetal framework
    - Give some hints about what algorithm to use to solve a multi-objective problem from an experimental point of view
  - Show how you can use jMetal to:
    - Configure and run multi-objective metaheuristics
    - Develop experimental studies to assess the performance of algorithms
    - Apply jMetal to a SBSE case study: the Next Release Problem (NRP)

About this tutorial

- Requirements to run the examples
  - Java JDK 8
  - Maven
  - Gnuplot, R, … to plot data
  - R and Latex

- The stuff of this tutorial is available in GitHub:
  - https://github.com/jMetal/SSBSE2017Tutorial

Table of contents

- **Who, why, what, when**
- Background
- Starting with jMetal
- Experimental studies
- Case study: the next release problem
- Further developments and related projects

Who, why, what, when

- Antonio J. Nebro
  - PhD in Computer Science (1999)
  - Associate professor (University of Málaga – Spain)
  - Khaos research group (http://khaos.uma.es/en)

- Research interests
  - Multi-objective optimization
  - Parallel metaheuristics
  - Applications to real-world problems (bioinformatics, civil engineering, Big Data)
  - Software tools (jMetal, jMetalSP, jMetalPy)

Who, why, what, when

- Khaos research group

Who, why, what, when

- Research Lines:
  - Data Management and Integration (Relational Data Bases, NoSQL, Linked Data, Open Data)
  - Data Analysis (Data Mining, Test Mining, Information Recovery, Optimization Algorithms, Big Data Analytics)
  - Semantic Annotation. Integrated Data Analysis
  - Semantics (Scalable Reasoning, Semantic Relations Discovery, Semantic Assignation to Mobile Objects, Semantic Driven Contents Recommendation, Web Semantics for E-Sciences, Smart Data for Big Data interpretation)
  - **Metaheuristics. Multi-objective Optimization. Big Data Optimization**
- Applications (multi-disciplinary domains):
  - Life Sciences and Biomedicine
  - Cultural Patrimony and Tourism
  - E-commerce
  - Smart Cities

Who, why, what, when

- Juan J. Durillo
  - PhD in Computer Science (2011)
  - Assistant professor (University of Innsbruck– Austria)

- Research interests
  - Multi-objective optimization
  - Software automatic tuning
  - Cloud computing
  - Software tools (jMetal)

Who, why, what, when

- Matthieu Vergne
  - PhD in ICT (2016)
  - Java developer in France
  - jMetal contributor since 2014

- Research interests
  - Artificial Intelligence
  - Conceptual Modelling
  - Generic Programming

Who, why, what, when

- We started to work on multi-objective optimization in 2003
  - We wanted to :
    - Develop a multi-objective Scatter Search algorithm
    - Apply our background in parallism to multi-objective metaheuristics
- Software tools in 2004
  - Implementation of NSGA-II in lenguage C
  - The PISA Framework (also in C)
- But we were not comfortable with the available source codes
    - Lack of object-oriented design
    - It was difficult to us to read and understand the codes

Who, why, what, when

- What we wanted
  - A framework easy to use, flexible, extensible, portable for our own research
  - Written in Java
- Our approach:
  - To write our own framework from scratch for our own research: jMetal
  - Later we decided to put jMetal in public domain
  - We provided (limited) support (email, Facebook, LinkedIn)

Who, why, what, when

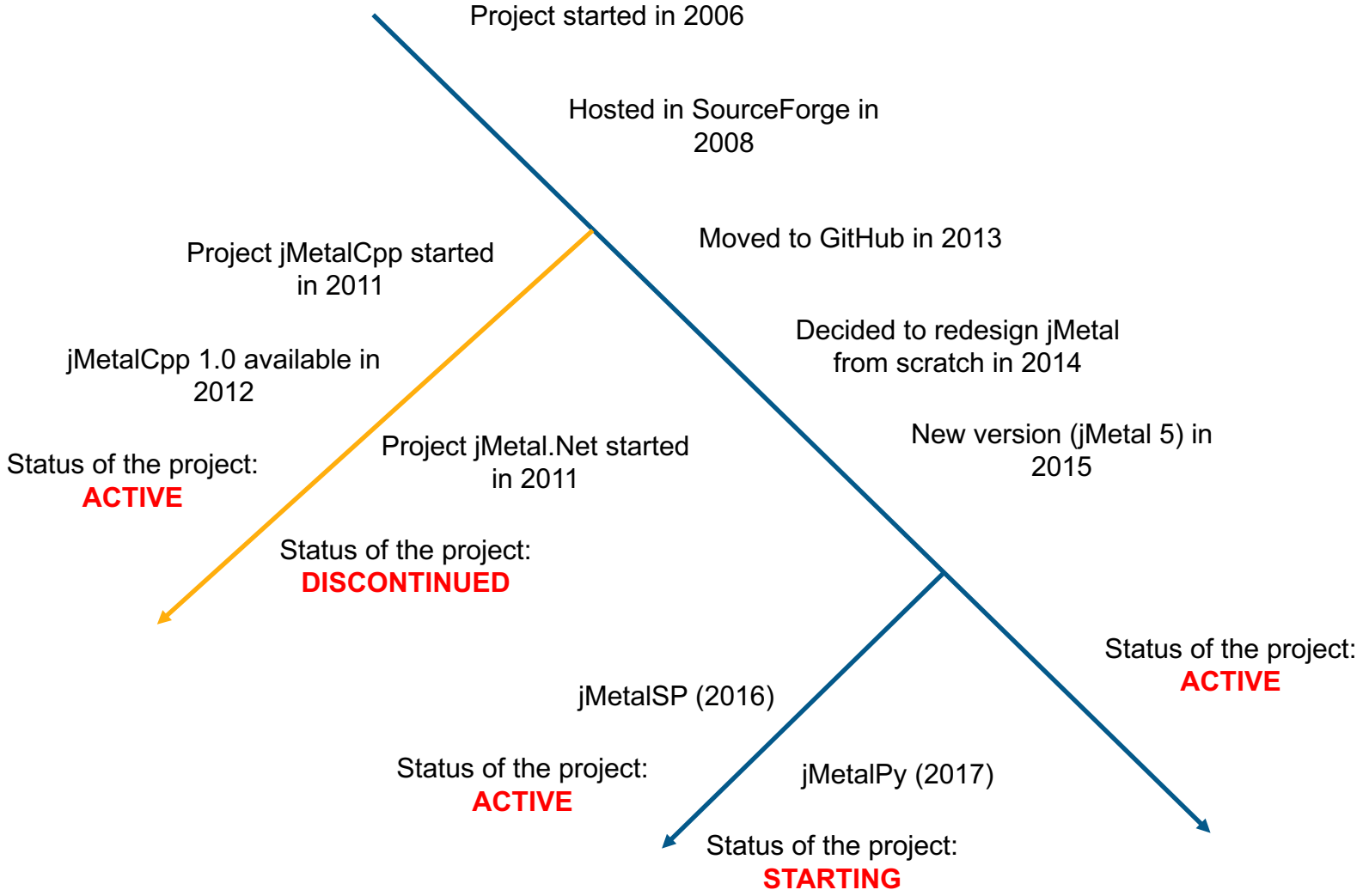


Table of contents

- Who, why, what, when
- **Background**
- jMetal features
- Configuring and running metaheuristics
- Experimental studies
- Case study: the next release problem
- Further developments and related projects

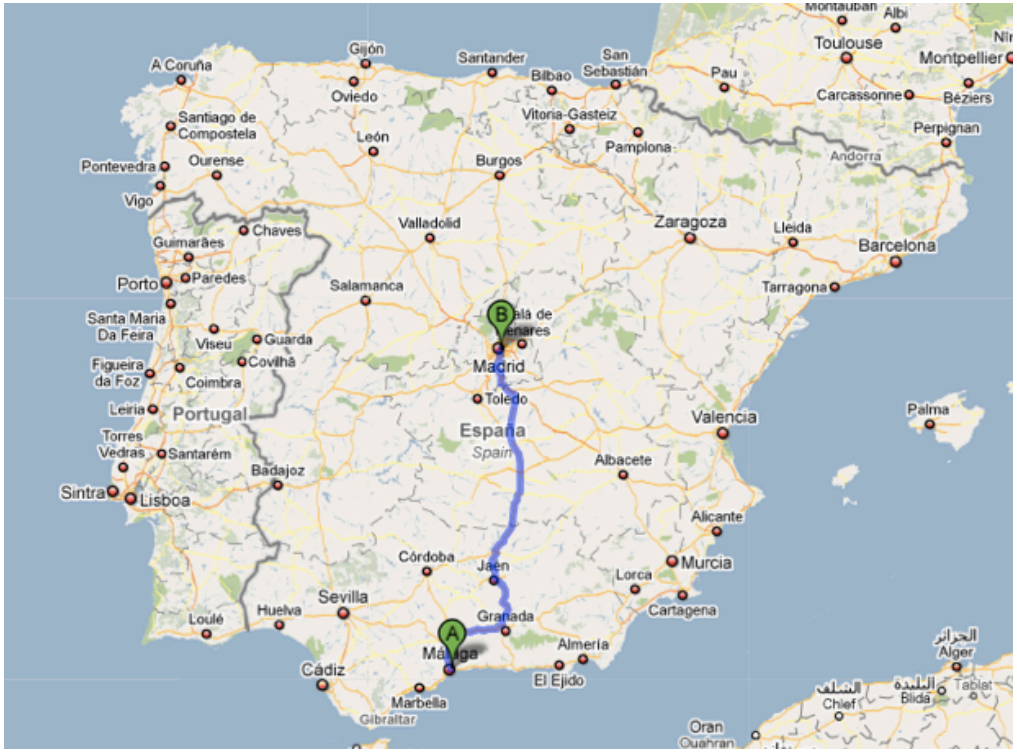
Background

- Many real-world optimization problems require to optimize more than one objective or function at the same time
  - These objectives are usually in conflict among them
  - Improving one means worsening the others

- Multi-objective (or multi-criteria) optimization
  - Discipline focused in solving multiobjective optimization problems (MOPs)

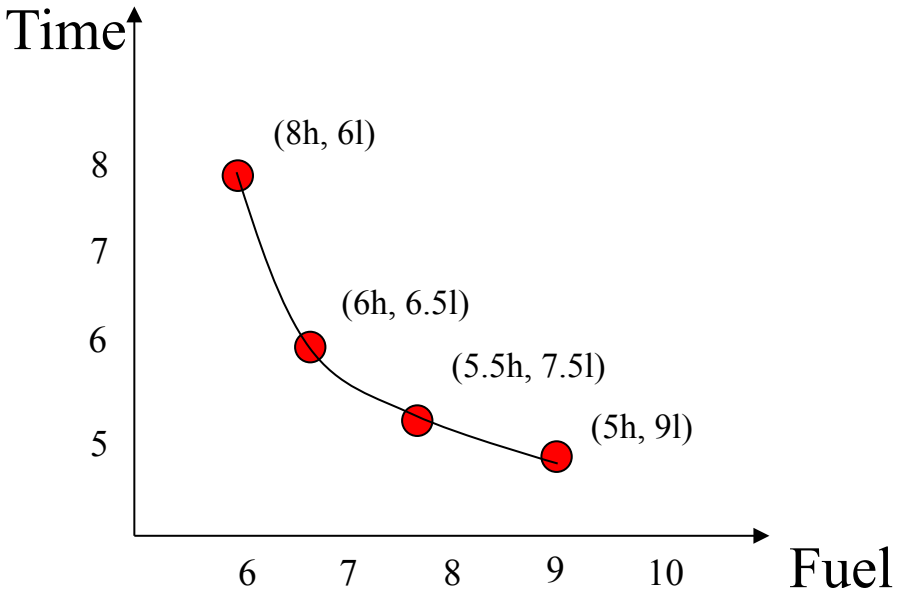
An example

- Example: travelling by car from Málaga to Madrid (535 km)
  - Objective 1:
    - Minimizing time
  - Objective 2:
    - Minimizing fuel
  - Constraints:
    - Max. speed: 120 km/h
    - Min. speed: 60 km/h
  - Decision variable:
    - mean car speed



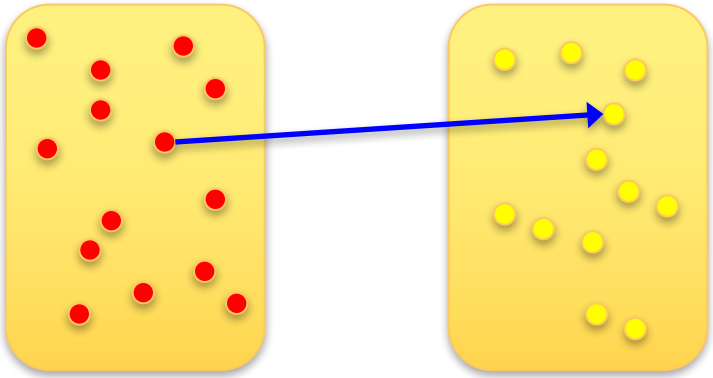
An example

- Example: travelling by car from Málaga to Madrid (535 km)
  - Extreme solutions
    - Time: 5 hours, fuel: 9.0 litres
    - Time: 8 hours, fuel: 6.0 litres
  - Other solutions
    - Time: 5.5 hours, fuel: 7.5 litres
    - Time: 6 hours, fuel: 6.5 litres



Consequences

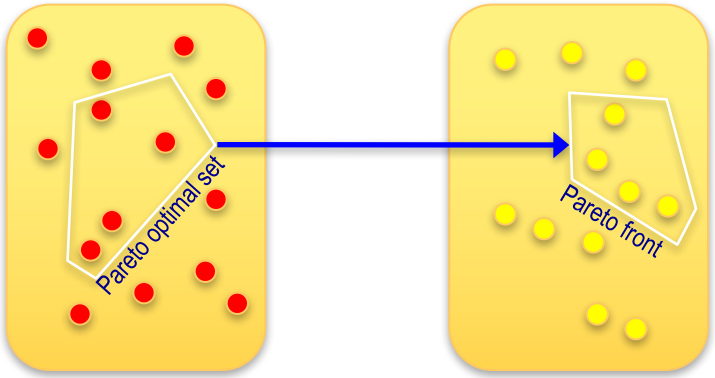
- In single-objective optimization (SO)
  - The optimum is a solution



X
(Solution space)

F(X)
(Objective space)

- In multi-objective optimization (MO)
  - The optimum (Pareto optimal set) is a set of (non-dominated) solutions



X
(Solution space)

F(X), G(X), …
Objective space

Dominance and non-dominance

- In single-objective optimization
  - We look for a single solution
  - The concept of "A better than B" is trivial
- In multi-objective optimization
  - We are not restricted to find a unique optimal solution
  - the concept of "A better than B" is not trivial

A	2	3	4	5
B	4	6	5	7

A is better than B

A	3	7	4	8
B	2	1	2	5

B is better than A

A	1	9	4	5
B	3	6	5	7

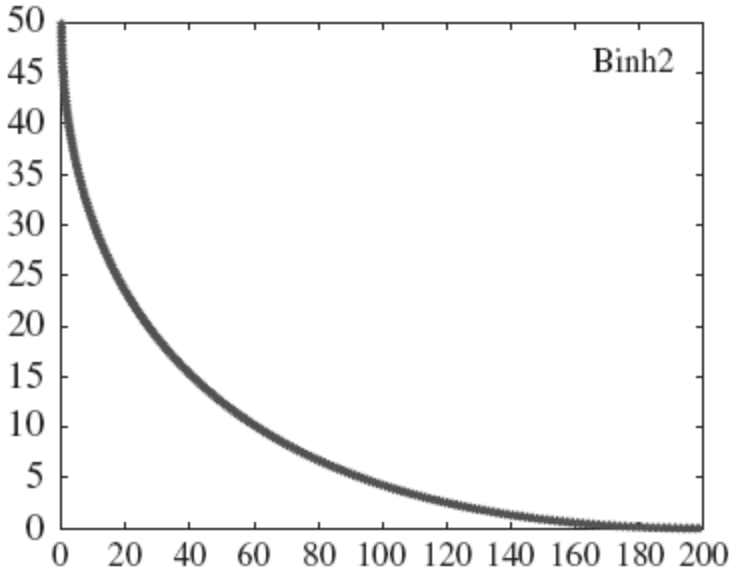
None is better

A and B are NON-DOMINATED

Pareto optimal set, Pareto optimal front

- The solution of a MOP is a set composed of those non-dominated solutions that cannot be dominated by any other solution outside that set: Pareto Optimal Set
- The correspondence of the Pareto optimal set in the objective space is known as Pareto Optimal Front (or just Pareto Front)

$$
\begin{aligned}
&\text{Min } F = (f_1(\vec{x}), f_2(\vec{x}))\\
&f_1(\vec{x}) = 4x_1^2 + 4x_2\\
&f_2(\vec{x}) = (x_1 - 5)^2 + (x_2 - 5)^2\\
&\text{Subject to:}\\
&g_1(\vec{x}) = (x_1 - 5)^2 + x_2^2 - 25 \le 0\\
&g_2(\vec{x}) = -(x_1 - 8)^2 - (x_2 + 3)^2 + 7.7 \le 0\\
&0 \le x_1 \le 5\\
&0 \ge x_2 \le 3
\end{aligned}
$$

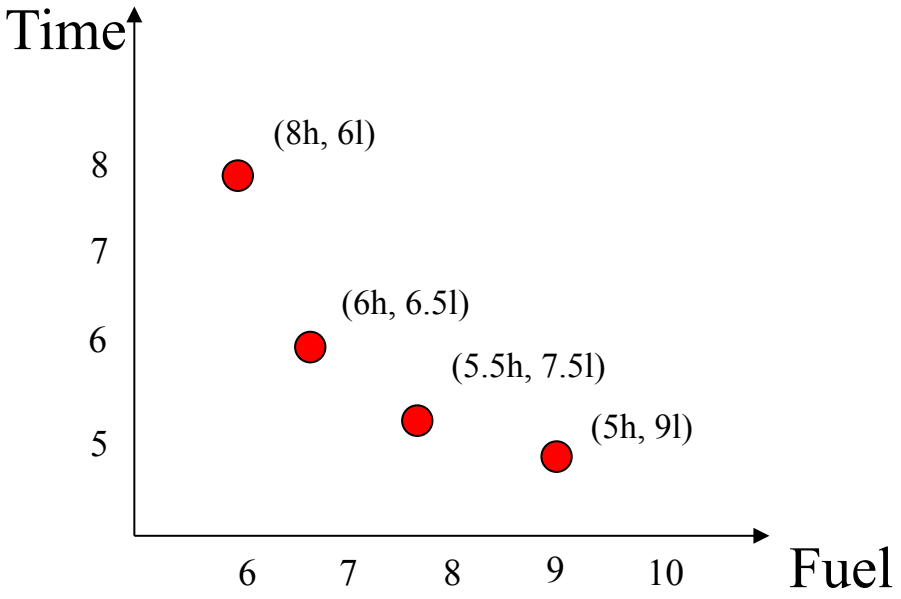


Multi-Objective Optimization and Decision Making

- Finding the Pareto front of a problem is not the last step in multi-objective optimization
- In practice, an expert in the domain (the decision maker) has to choose the best trade-off solution

Multi-Objective Optimization and Decision Making

- In the example of traveling from Málaga to Madrid
  - If time is important
    - Choose (5h, 9l)
  - If consumption is important:
    - Choose (8h, 6l)
  - Compromise solution:
    - (6h, 6.5l)

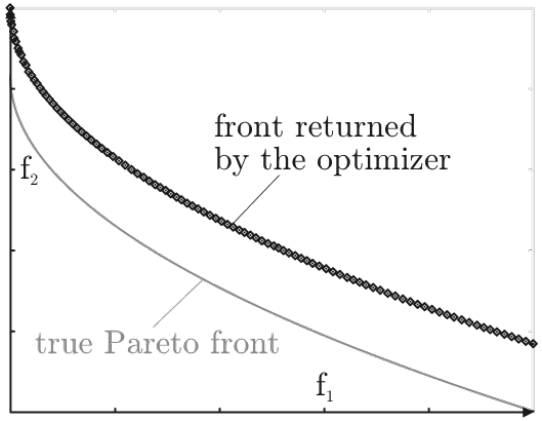


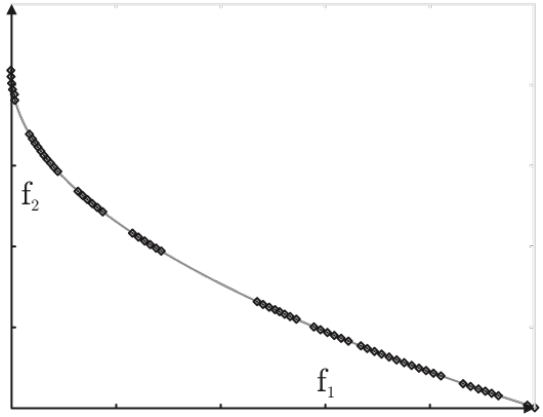
Goals of Multi-Objective Optimization

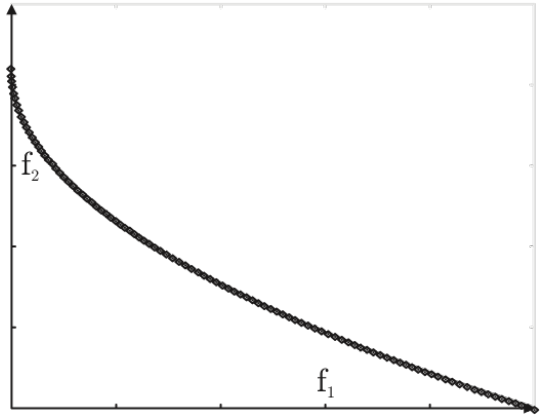
- The ideal goal is to obtain the Pareto front
- Unfortunately, this is unpractical in real-world problems
  - NP-hard complexity, non-linearity, epistasis, …
  - Frequently, exact techniques are not useful
- Alternative: Use non-exact algorithms
  - E.g. *Metaheuristics*
  - These techniques provide an *approximation* to the Pareto front

Goals of Multi-Objective Optimization

- Main goal when using non-exact tecniques:
  - Obtaining an approximation to the Pareto from with two properties
    - Convergence
    - Diversity







Other Important Goals in Practice

- Fast convergence to the Pareto front
  - The decision maker may need the results quickly

- This can be achieved
  - Designing more efficient algorithms
    - Requiring less function evaluations to converge
  - Using parallelism
    - To speed-up the execution time

Table of contents

- Who, why, what, when
- Background
- **Starting with jMetal**
- Experimental studies
- Case study: the next release problem
- Further developments and related projects

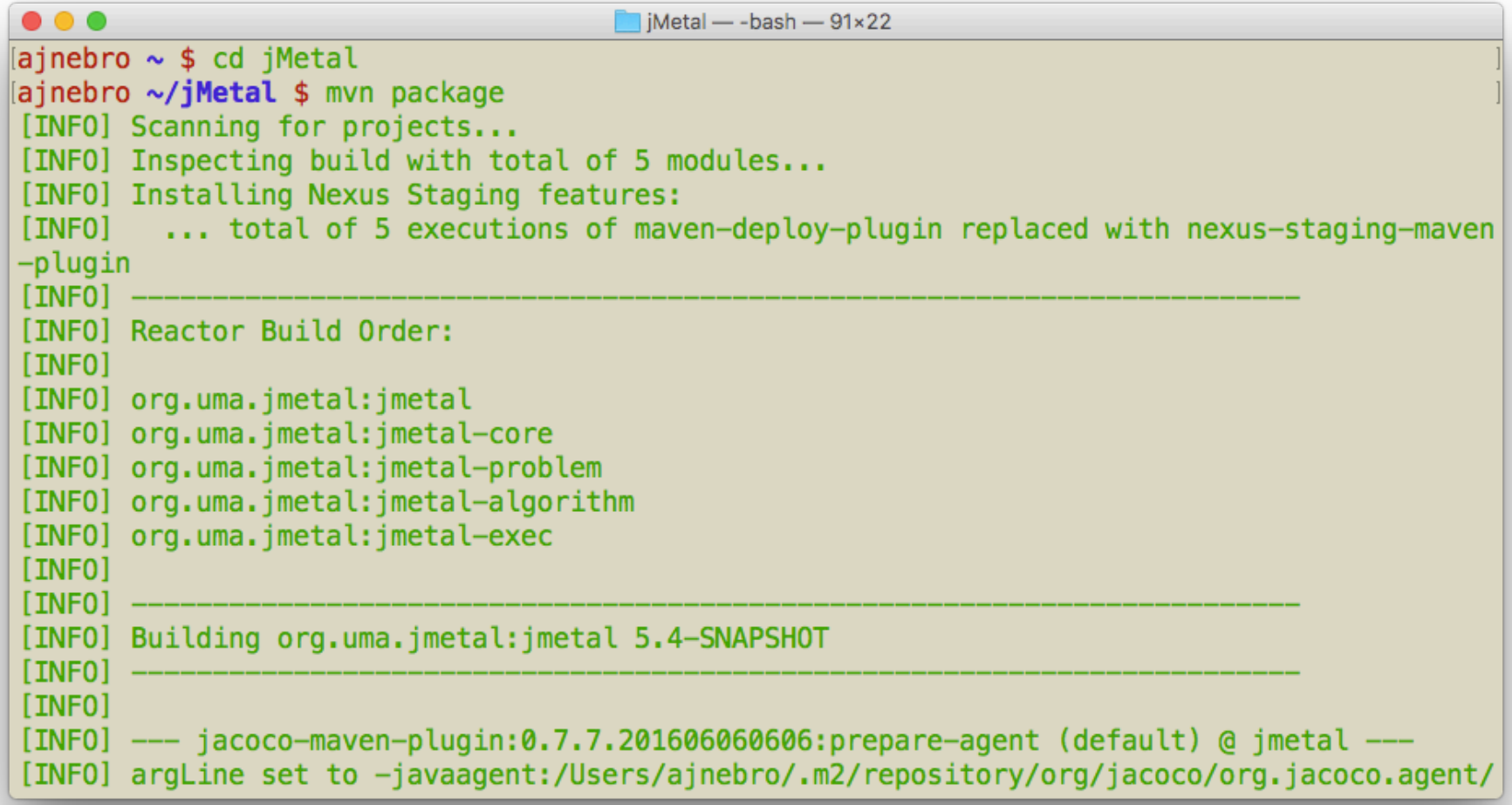
jMetal: a Maven Project Hosted in GitHub

- Getting the full source code of the project

```
ajnebro — -bash — 91×16
ajnebro ~ $ git clone https://github.com/jMetal/jMetal.git
Cloning into 'jMetal'...
remote: Counting objects: 53691, done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 53691 (delta 0), reused 11 (delta 0), pack-reused 53677
Receiving objects: 100% (53691/53691), 109.34 MiB | 5.82 MiB/s, done.
Resolving deltas: 100% (27036/27036), done.
ajnebro ~ $ ls jMetal/
LICENSE.txt                     jmetal-exec
README.md                       jmetal-problem
jmetal-algorithm                pom.xml
jmetal-core                     sonar-project.properties
ajnebro ~ $
```

jMetal: a Maven Project Hosted in GitHub

- The project can be built from the command line using Maven



jMetal: a Maven Project Hosted in GitHub

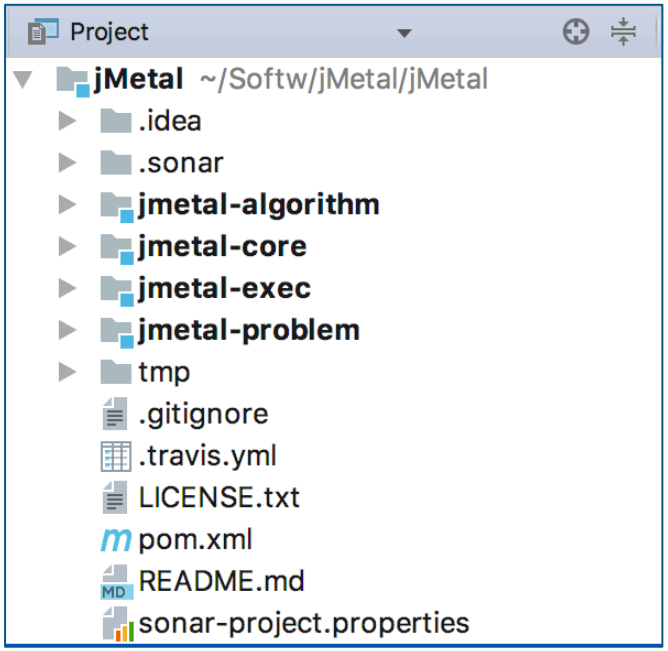
- You can use jMetal as a dependency in your Maven projects

pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-inst
         xsi:schemaLocation="http://maven.apache.org/POM/
    <modelVersion>4.0.0</modelVersion>

    <groupId>ssbse2017</groupId>
    <artifactId>nrp</artifactId>
    <version>1.0-SNAPSHOT</version>

    <build...>

    <dependencies>
        <dependency>
            <groupId>org.uma.jmetal</groupId>
            <artifactId>jmetal-core</artifactId>
            <version>5.3</version>
        </dependency>
        <dependency>
            <groupId>org.uma.jmetal</groupId>
            <artifactId>jmetal-algorithm</artifactId>
            <version>5.3</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

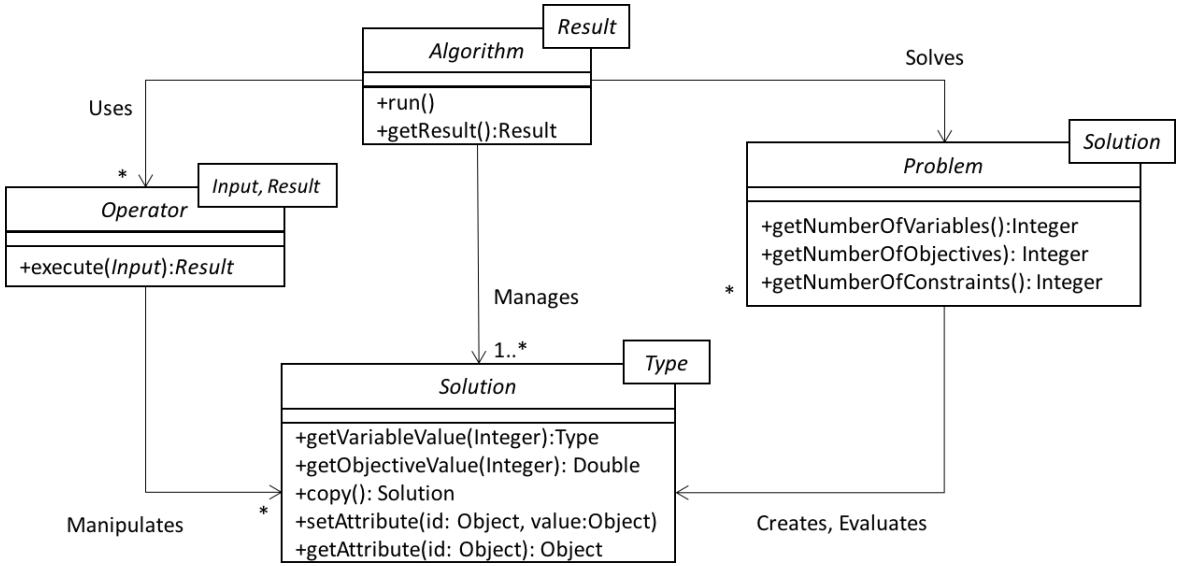
jMetal: a Maven Project Hosted in GitHub

- jMetal 5 is composed of four Maven sub-packages
- In the current version (jMetal 5.3):
  - Multi-Objective algorithms (20+)
  - Benchmark problems (70+)
  - Solution representations (6)
  - Quality indicators (10)



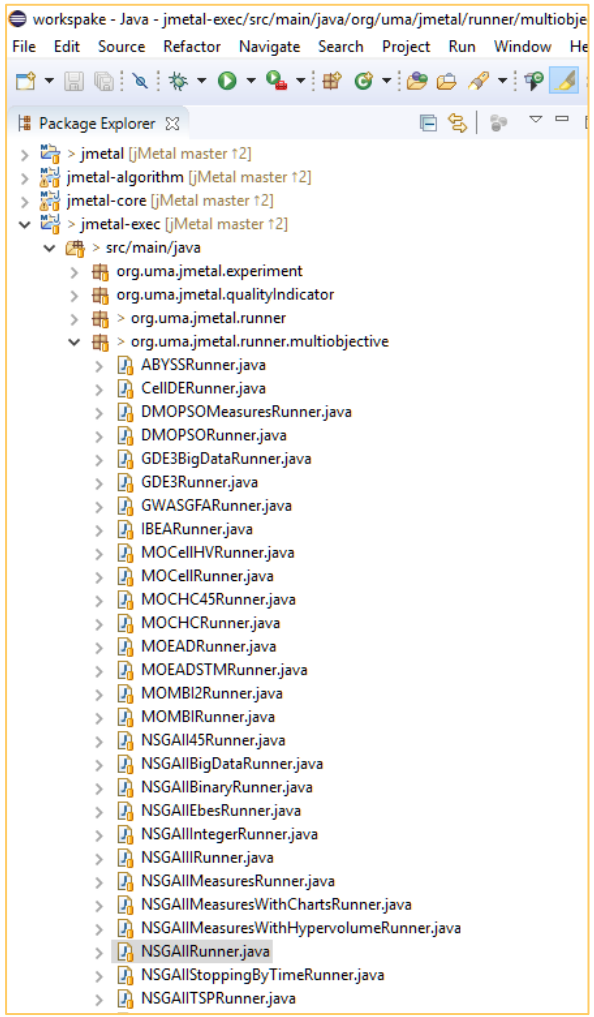
jMetal Architecture

- Basic idea:
  - An algorithm (metaheuristic) solves a problem by manipulating solutions through a set of operators
  - A problem creates and evaluates solutions



Architecture of jMetal 5

Example1: configuring and running NSGA-II




```java
public class NSGAIIRunner extends AbstractAlgorithmRunner {
  /**
   * @param args Command line arguments.
   */
  public static void main(String[] args) throws JMetalException, FileNotFoundException {
    Problem<DoubleSolution> problem;
    Algorithm<List<DoubleSolution>> algorithm;
    CrossoverOperator<DoubleSolution> crossover;
    MutationOperator<DoubleSolution> mutation;
    SelectionOperator<List<DoubleSolution>, DoubleSolution> selection;
    String referenceParetoFront = "" ;

    String problemName ;
    if (args.length == 1) {
      problemName = args[0];
    } else if (args.length == 2) {
      problemName = args[0] ;
      referenceParetoFront = args[1] ;
    } else {
      problemName = "org.uma.jmetal.problem.multiobjective.zdt.ZDT1";
      referenceParetoFront = "jmetal-problem/src/test/resources/pareto_fronts/ZDT1.pf" ;
    }

    problem = ProblemUtils.<DoubleSolution> loadProblem(problemName);

    double crossoverProbability = 0.9 ;
    double crossoverDistributionIndex = 20.0 ;
    crossover = new SBXCrossover(crossoverProbability, crossoverDistributionIndex) ;

    double mutationProbability = 1.0 / problem.getNumberOfVariables() ;
    double mutationDistributionIndex = 20.0 ;
    mutation = new PolynomialMutation(mutationProbability, mutationDistributionIndex) ;

    selection = new BinaryTournamentSelection<DoubleSolution>(
        new RankingAndCrowdingDistanceComparator<DoubleSolution>());

    algorithm = new NSGAIIBuilder<DoubleSolution>(problem, crossover, mutation)
        .setSelectionOperator(selection)
        .setMaxEvaluations(25000)
        .setPopulationSize(100)
        .build() ;

    AlgorithmRunner algorithmRunner = new AlgorithmRunner.Executor(algorithm)
        .execute() ;

    List<DoubleSolution> population = algorithm.getResult() ;
    long computingTime = algorithmRunner.getComputingTime() ;

    JMetalLogger.logger.info("Total execution time: " + computingTime + "ms");

    printFinalSolutionSet(population);
    if (!referenceParetoFront.equals("")) {
      printQualityIndicators(population, referenceParetoFront) ;
    }
  }
}
```

Example1: configuring and running NSGA-II

```
Problems  Javadoc  Declaration  Search  Console
<terminated> NSGAIIRunner [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (5 sept. 2017 10:41:35)
2017-09-05 10:41:35.682 INFORMACIÓN: Loggers configured with null [org.uma.jmetal.util.JMetalLogger configureLoggers]
2017-09-05 10:41:35.709 INFORMACIÓN: Total execution time: 410ms [org.uma.jmetal.runner.multiobjective.NSGAIIRunner main]
2017-09-05 10:41:35.746 INFORMACIÓN: Random seed: 1504600895258 [org.uma.jmetal.runner.AbstractAlgorithmRunner printFinal
2017-09-05 10:41:35.747 INFORMACIÓN: Objectives values have been written to file FUN.tsv [org.uma.jmetal.runner.AbstractA
2017-09-05 10:41:35.747 INFORMACIÓN: Variables values have been written to file VAR.tsv [org.uma.jmetal.runner.AbstractAl
2017-09-05 10:41:35.818 INFORMACIÓN:
Hypervolume (N) : 0.6597068785327224
Hypervolume     : 0.6597068785327224
Epsilon (N)     : 0.012834091463729713
Epsilon         : 0.012834091463729713
GD (N)          : 2.0135698109623943E-4
GD              : 2.0135698109623943E-4
IGD (N)         : 1.8094689755722708E-4
IGD             : 1.8094689755722708E-4
IGD+ (N)        : 0.003580506492809246
IGD+            : 0.003580506492809246
Spread (N)      : 0.33127271727483215
Spread          : 0.33127271727483215
Error ratio     : 1.0
 [org.uma.jmetal.runner.AbstractAlgorithmRunner printQualityIndicators]
```

```
Windows PowerShell
PS C:\Users\ajnebro\Softw\jMetal\jMetal> dir

    Directorio: C:\Users\ajnebro\Softw\jMetal\jMetal

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        05/09/2017     10:13                .git
d-----        05/09/2017     10:13                .idea
d-----        02/06/2017     11:36                .settings
d-----        05/09/2017      9:10                jmetal-algorithm
d-----        25/07/2017     16:58                jmetal-core
d-----        05/09/2017     11:19                jmetal-exec
d-----        25/07/2017     16:58                jmetal-problem
d-----        19/06/2017     13:30                target
-a----        25/07/2017     16:58           2152 .gitignore
-a----        02/06/2017     11:36            386 .project
-a----        05/09/2017     10:51           4231 .RData
-a----        05/09/2017     10:51            102 .Rhistory
-a----        25/07/2017     16:58            119 .travis.yml
-a----        05/09/2017     10:41           4086 FUN.tsv
-a----        12/05/2017     11:51            574 jmetal.iml
-a----        05/09/2017     10:41           2742 jMetal.log
-a----        25/07/2017     16:58           1101 LICENSE.txt
-a----        25/07/2017     16:58          10703 pom.xml
-a----        25/07/2017     16:58           1697 README.md
-a----        25/07/2017     16:58           1596 sonar-project.properties
-a----        05/09/2017     10:41          64146 VAR.tsv
```

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Addins ▾ Project: (None) ▾

Environment History

Import Dataset ▾ List ▾

Global Environment ▾

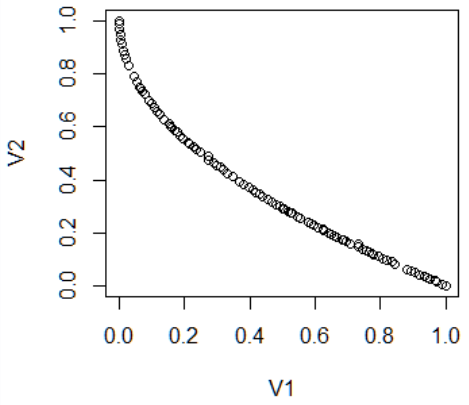
Data

a 100 obs. of 2 variables

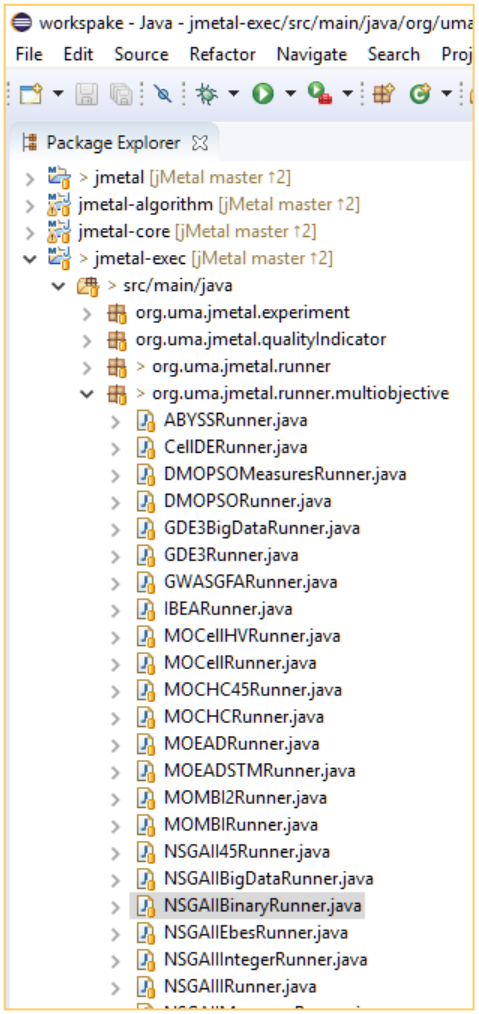
Files Plots Packages Help Viewer

Zoom Export ▾ Publish ▾

```
Darkness"
istical Comp

NO WARRANTY

tain conditi

tion details

ibutors.
nd
in publicati

on-line help,
 or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> setwd("C:/Users/ajnebro/Softw/jMetal/jMetal")
> a <- read.table("FUN.tsv")
> plot(a)
>
```



Example2: solving binary problems with NSGA-II



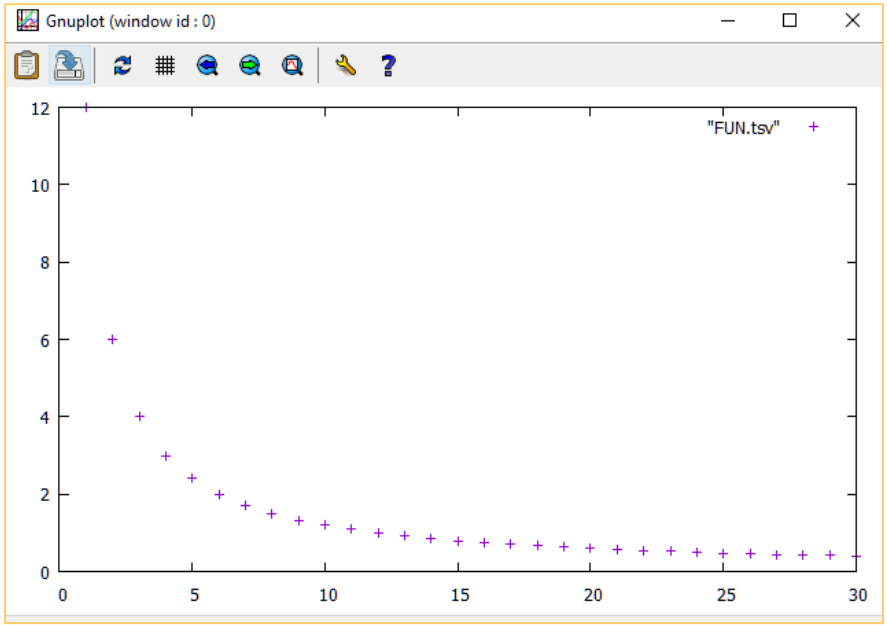

```java
public class NSGAIIBinaryRunner extends AbstractAlgorithmRunner {
  /**
   * @param args Command line arguments.
   */
  public static void main(String[] args) throws Exception {
    BinaryProblem problem;
    Algorithm<List<BinarySolution>> algorithm;
    CrossoverOperator<BinarySolution> crossover;
    MutationOperator<BinarySolution> mutation;
    SelectionOperator<List<BinarySolution>, BinarySolution> selection;

    String problemName ;
    String referenceParetoFront = "" ;
    if (args.length == 1) {
      problemName = args[0];
    } else if (args.length == 2) {
      problemName = args[0] ;
      referenceParetoFront = args[1] ;
    } else {
      problemName = "org.uma.jmetal.problem.multiobjective.zdt.ZDT5";
      referenceParetoFront = "" ;
    }

    problem = (BinaryProblem) ProblemUtils.<BinarySolution> loadProblem(problemName);

    double crossoverProbability = 0.9 ;
    crossover = new SinglePointCrossover(crossoverProbability) ;

    double mutationProbability = 1.0 / problem.getNumberOfBits(0) ;
    mutation = new BitFlipMutation(mutationProbability) ;

    selection = new BinaryTournamentSelection<BinarySolution>() ;

    algorithm = new NSGAIIBuilder<BinarySolution>(problem, crossover, mutation)
            .setSelectionOperator(selection)
            .setMaxEvaluations(25000)
            .setPopulationSize(100)
            .build() ;

    AlgorithmRunner algorithmRunner = new AlgorithmRunner.Executor(algorithm)
            .execute() ;

    List<BinarySolution> population = algorithm.getResult() ;
    long computingTime = algorithmRunner.getComputingTime() ;

    JMetalLogger.logger.info("Total execution time: " + computingTime + "ms");

    printFinalSolutionSet(population);
    if (!referenceParetoFront.equals("")) {
      printQualityIndicators(population, referenceParetoFront) ;
    }
  }
```

Example2: solving binary problems with NSGA-II

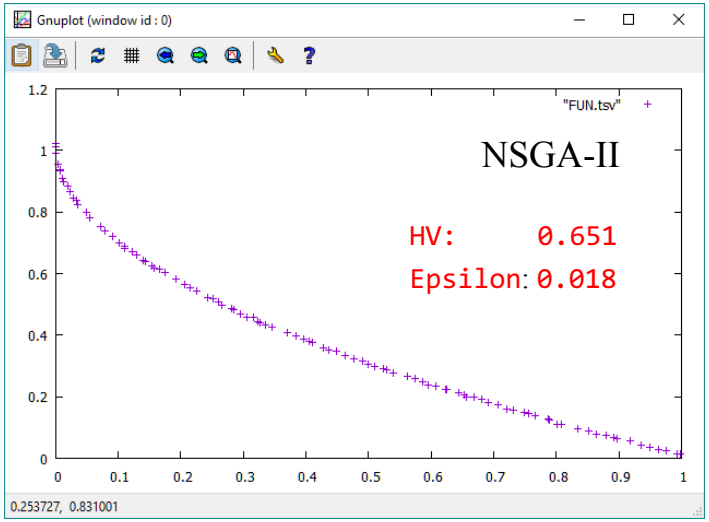
Problems | Javadoc | Declaration | Search | Console

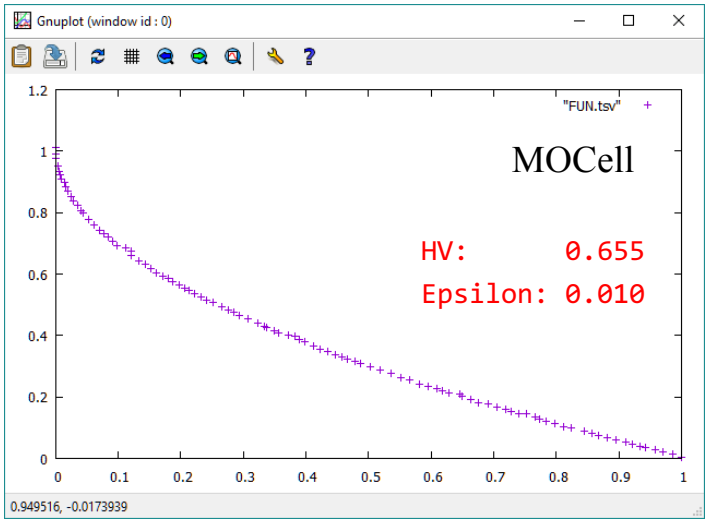
```
<terminated> NSGAIIBinaryRunner [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (5 sept. 2017 11:19:16)
2017-09-05 11:19:17.671 INFORMACIÓN: Loggers configured with null [org.uma.jmetal.util.JMetalLogger configureLoggers]
2017-09-05 11:19:17.700 INFORMACIÓN: Total execution time: 631ms [org.uma.jmetal.runner.multiobjective.NSGAIIBinaryRunner main]
2017-09-05 11:19:17.716 INFORMACIÓN: Random seed: 1504603157026 [org.uma.jmetal.runner.AbstractAlgorithmRunner printFinalSolutionSet]
2017-09-05 11:19:17.717 INFORMACIÓN: Objectives values have been written to file FUN.tsv [org.uma.jmetal.runner.AbstractAlgorithmRunner printFinalSolutionSet]
2017-09-05 11:19:17.717 INFORMACIÓN: Variables values have been written to file VAR.tsv [org.uma.jmetal.runner.AbstractAlgorithmRunner printFinalSolutionSet]
```

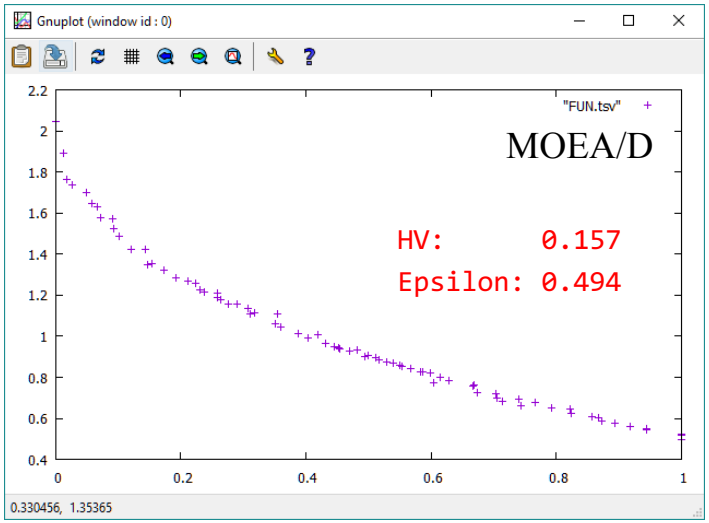
```
gnuplot 5.0 patchlevel 7 - console version
Terminal type set to 'wxt'
gnuplot> pwd
C:\Users\ajnebro\Documents
gnuplot> cd ..
         ^
         invalid expression

gnuplot> cd "../Softw/jMetal/jMetal"
gnuplot> plot "FUN.tsv"
gnuplot>
```

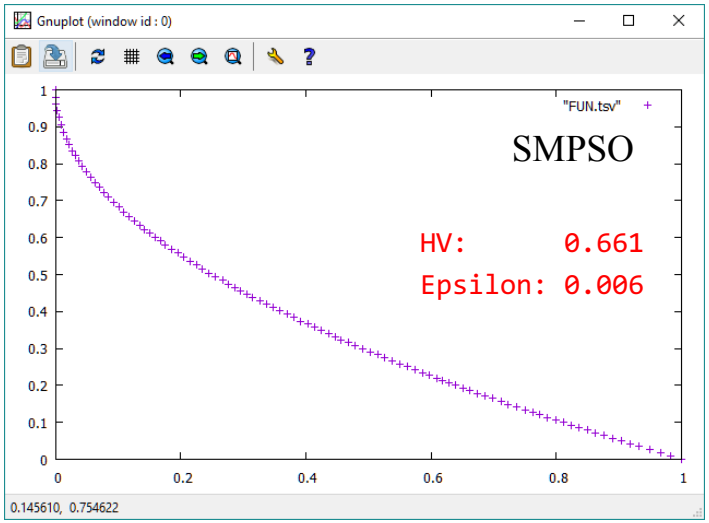


Example3: solving the ZDT4 problem







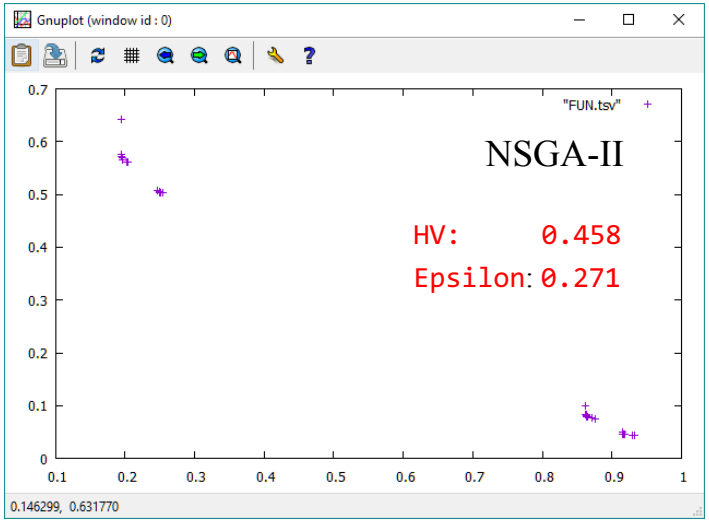


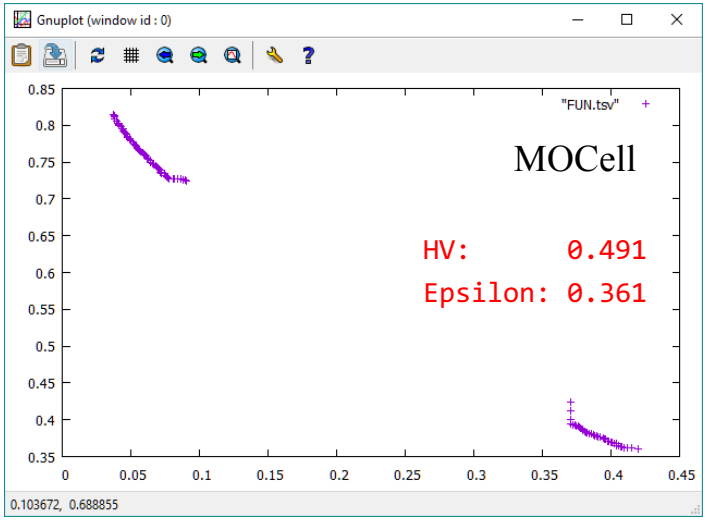
Common settings

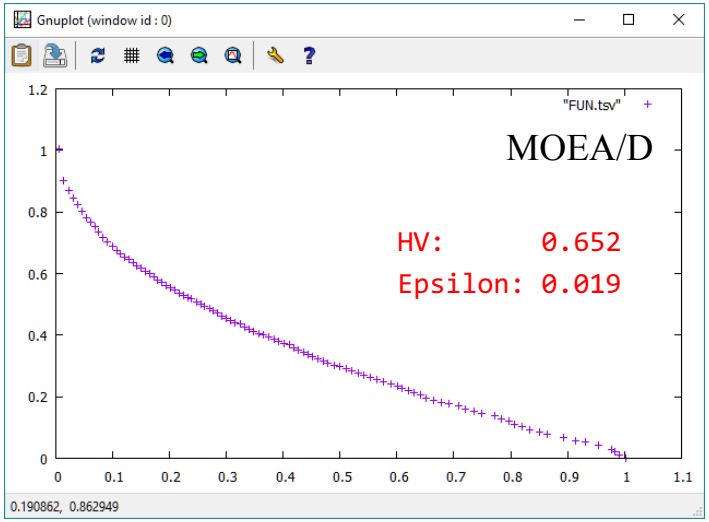
Pop. size: 100

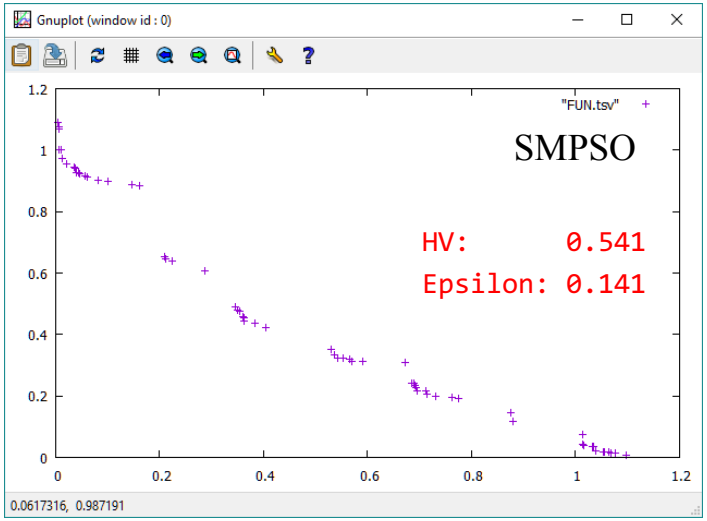
Evaluations: 25000

Example4: solving the LZ09-F2 problem







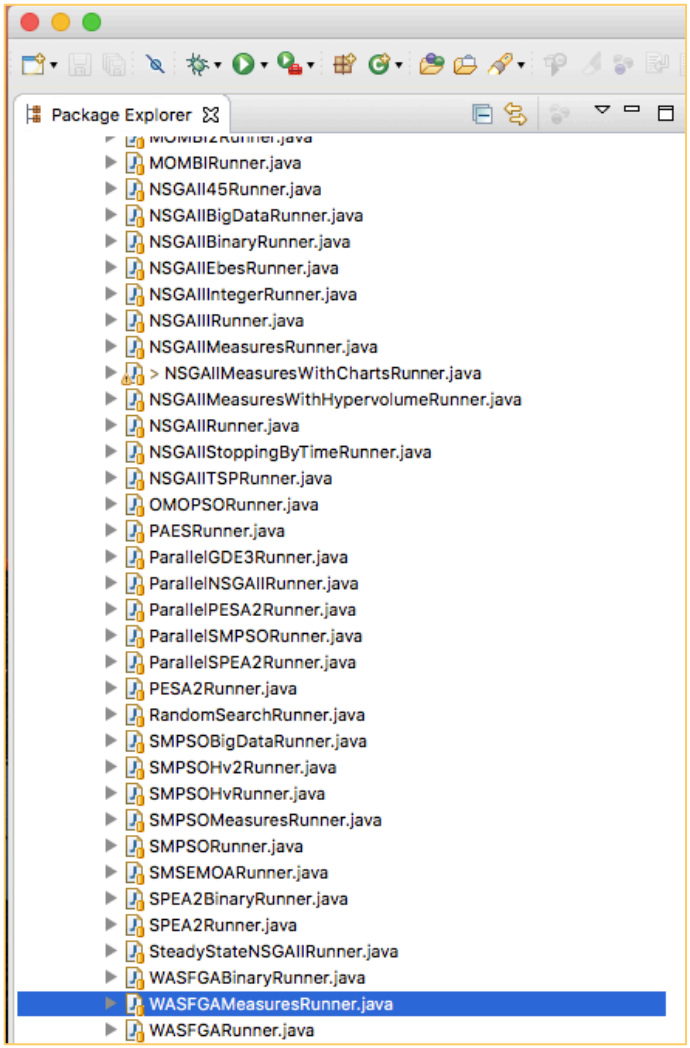


Common settings

Pop. size: 100

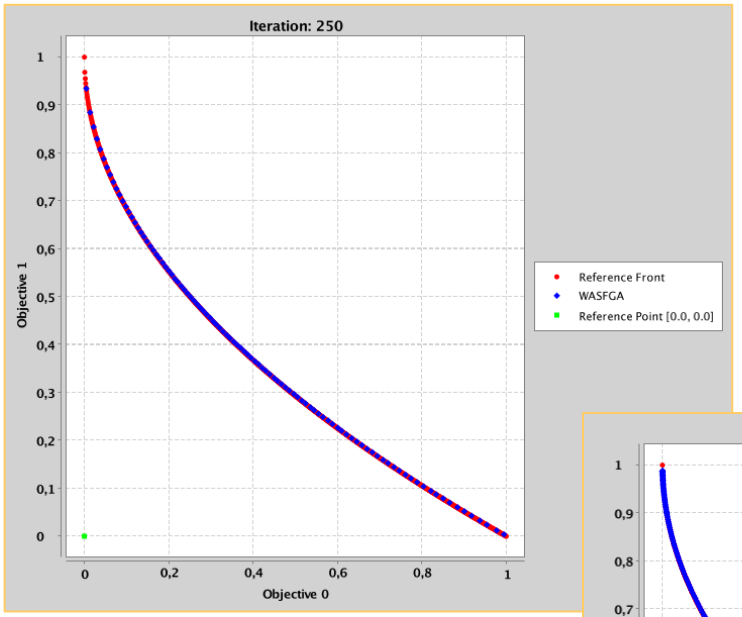
Evaluations: 150000

Example4: indicating preferences (WASF-GA)

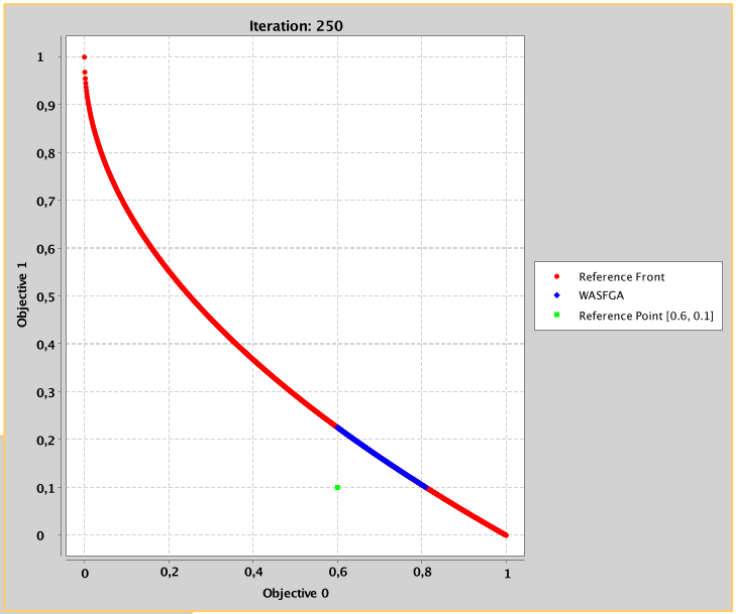



```java
public class WASFGAMeasuresRunner extends AbstractAlgorithmRunner {
  /**
   * @param args Command line arguments.
   */
  public static void main(String[] args) throws JMetalException, IOException {
    Problem<DoubleSolution> problem;
    Algorithm<List<DoubleSolution>> algorithm;
    CrossoverOperator<DoubleSolution> crossover;
    MutationOperator<DoubleSolution> mutation;
    SelectionOperator<List<DoubleSolution>, DoubleSolution> selection;
    String referenceParetoFront = "" ;
    List<Double> referencePoint = null;

    String problemName ;
    if (args.length == 1) {
      problemName = args[0];
    } else if (args.length == 2) {
      problemName = args[0] ;
      referenceParetoFront = args[1] ;
    } else {
      problemName = "org.uma.jmetal.problem.multiobjective.zdt.ZDT1";
      referenceParetoFront = "jmetal-problem/src/test/resources/pareto_fronts/ZDT1.pf" ;
    }

    problem = ProblemUtils.<DoubleSolution> loadProblem(problemName);

    referencePoint = new ArrayList<>();
    referencePoint.add(0.5);
    referencePoint.add(0.5);

    double crossoverProbability = 0.9 ;
    double crossoverDistributionIndex = 20.0 ;
    crossover = new SBXCrossover(crossoverProbability, crossoverDistributionIndex) ;

    double mutationProbability = 1.0 / problem.getNumberOfVariables() ;
    double mutationDistributionIndex = 20.0 ;
    mutation = new PolynomialMutation(mutationProbability, mutationDistributionIndex) ;
```

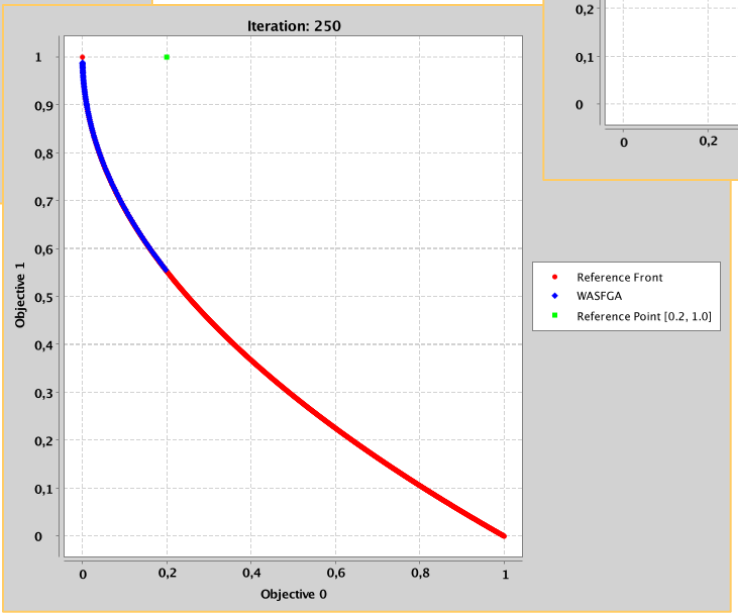
Example4: indicating preferences (WASF-GA)



Reference point: [0.0, 0.0]

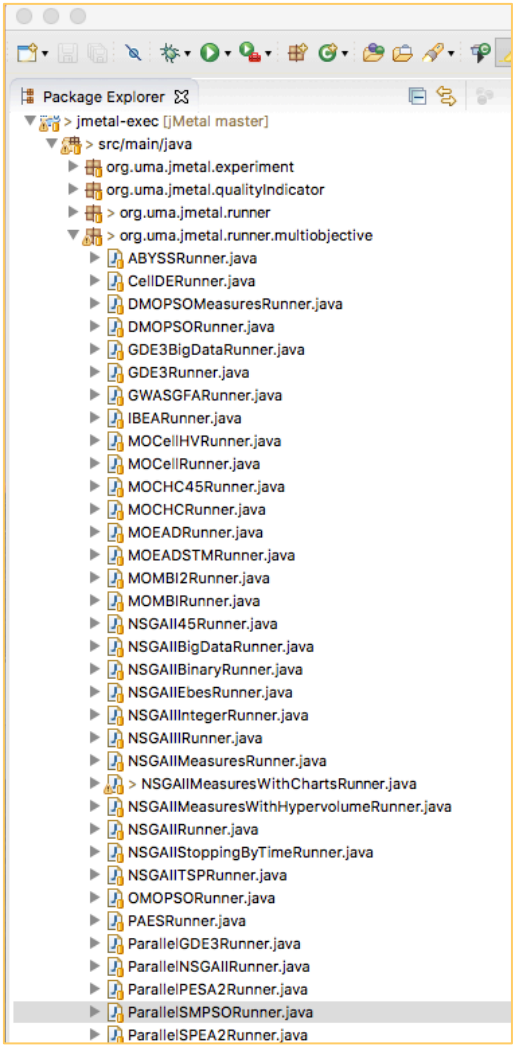


Reference point: [0.6, 0.1]



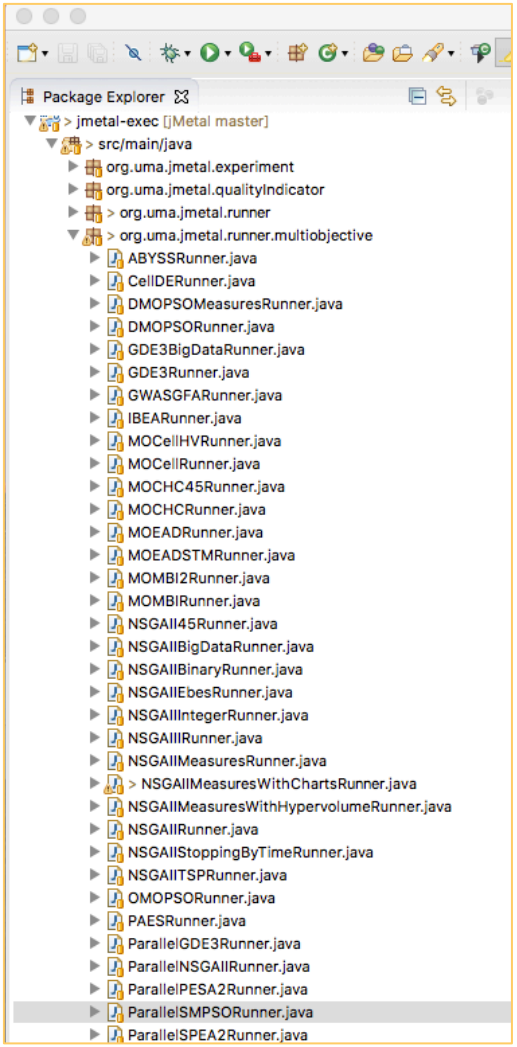
Reference point: [0.2, 1.0]

Example5: parallel execution




```java
public class ParallelSMPSORunner extends AbstractAlgorithmRunner {
  /**
   * @param args Command line arguments. The first (optional) argument specifies
   */
  public static void main(String[] args) throws Exception {
    DoubleProblem problem;
    Algorithm<List<DoubleSolution>> algorithm;
    MutationOperator<DoubleSolution> mutation;
    SolutionListEvaluator<DoubleSolution> evaluator ;

    String referenceParetoFront = "" ;

    String problemName ;
    if (args.length == 1) {
      problemName = args[0];
    } else if (args.length == 2) {
      problemName = args[0] ;
      referenceParetoFront = args[1] ;
    } else {
      problemName = "org.uma.jmetal.problem.multiobjective.zdt.ZDT1";
      referenceParetoFront = "jmetal-problem/src/test/resources/pareto_fronts/ZDT1.pf" ;
    }

    problem = (DoubleProblem) ProblemUtils.<DoubleSolution> loadProblem(problemName);

    BoundedArchive<DoubleSolution> archive = new CrowdingDistanceArchive<DoubleSolution>(100) ;

    double mutationProbability = 1.0 / problem.getNumberOfVariables() ;
    double mutationDistributionIndex = 20.0 ;
    mutation = new PolynomialMutation(mutationProbability, mutationDistributionIndex) ;

    evaluator = new MultithreadedSolutionListEvaluator<DoubleSolution>(0, problem) ;

    algorithm = new SMPSOBuilder(problem, archive)
            .setMutation(mutation)
            .setMaxIterations(250)
            .setSwarmSize(100)
            .setSolutionListEvaluator(evaluator)
            .build();
```

Example5: parallel execution




```
public class ParallelSMPSORunner extends AbstractAlgorithmRunner {
  /**
   * @param args Command line arguments. The first (optional) argument specifies
   */
  public static void main(String[] args) throws Exception {
    DoubleProblem problem;
    Algorithm<List<DoubleSolution>> algorithm;
    MutationOperator<DoubleSolution> mutation;
    SolutionListEvaluator<DoubleSolution> evaluator ;

    String referenceParetoFront = "" ;

    String problemName ;
    if (args.length == 1) {
      problemName = args[0];
    } else if (args.length == 2) {
      problemName = args[0] ;
      referenceParetoFront = args[1] ;
    } else {
      problemName = "org.uma.jmetal.problem.multiobjective.zdt.ZDT1";
      referenceParetoFront = "jmetal-problem/src/test/resources/pareto_fronts/ZDT1.pf" ;
    }

    problem = (DoubleProblem) ProblemUtils.<DoubleSolution> loadProblem(problemName);

    BoundedArchive<DoubleSolution> archive = new CrowdingDistanceArchive<DoubleSolution>(100) ;

    double mutationProbability = 1.0 / problem.getNumberOfVariables() ;
    double mutationDistributionIndex = 20.0 ;
    mutation = new PolynomialMutation(mutationProbability, mutationDistributionIndex) ;

    evaluator = new MultithreadedSolutionListEvaluator<DoubleSolution>(0, problem) ;

    algorithm = new SMPSOBuilder(problem, archive)
            .setMutation(mutation)
            .setMaxIterations(250)
            .setSwarmSize(100)
            .setSolutionListEvaluator(evaluator)
            .build();
```

Algorithm templates

Algorithm 1 Pseudo-code of an evolutionary algorithm

```
1: P(0) ← GenerateInitialSolutions()
2: t ← 0
3: Evaluate(P(0))
4: while not StoppingCriterion() do
5:    Q(t) ← Variation(P(t))
6:    Evaluate(Q(t))
7:    P(t + 1) ← Update(P(t), Q(t))
8:    t ← t + 1
9: end while
```

```
public abstract class AbstractEvolutionaryAlgorithm<S extends
Solution<?>, R>  implements Algorithm<R>{

 @Override public void run() {
     List<S> offspringPopulation;
     List<S> matingPopulation;

1:   population = createInitialPopulation();
3:   population = evaluatePopulation(population);
2:   initProgress();
4:    while (!isStoppingConditionReached()) {
       matingPopulation = selection(population);
5:     offspringPopulation = reproduction(matingPopulation);
6:     offspringPopulation = evaluatePopulation(offspringPopulation);
7:     population = replacement(population, offspringPopulation);
8:     updateProgress();
9:   }
   }
 }
```

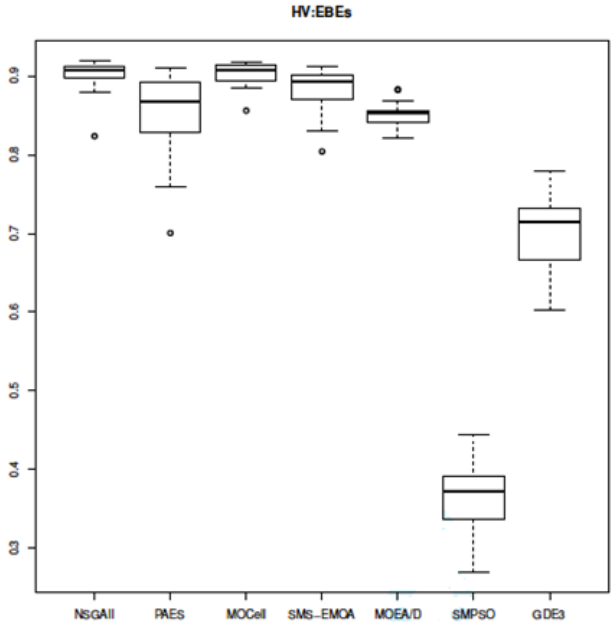
Algorithm templates: NSGA-II

```java
public class NSGAII<S extends Solution<?>> extends AbstractGeneticAlgorithm<S, List<S>> {
  protected final int maxEvaluations;

  protected final SolutionListEvaluator<S> evaluator;

  protected int evaluations;

  /**
   * Constructor
   */
  public NSGAII(Problem<S> problem, int maxEvaluations, int populationSize,
      CrossoverOperator<S> crossoverOperator, MutationOperator<S> mutationOperator,
      SelectionOperator<List<S>, S> selectionOperator, SolutionListEvaluator<S> evaluator) {
    super(problem);
    this.maxEvaluations = maxEvaluations;
    setMaxPopulationSize(populationSize); ;

    this.crossoverOperator = crossoverOperator;
    this.mutationOperator = mutationOperator;
    this.selectionOperator = selectionOperator;

    this.evaluator = evaluator;
  }

  @Override protected void initProgress() {
    evaluations = getMaxPopulationSize();
  }

  @Override protected void updateProgress() {
    evaluations += getMaxPopulationSize() ;
  }

  @Override protected boolean isStoppingConditionReached() {
    return evaluations >= maxEvaluations;
  }

  @Override protected List<S> evaluatePopulation(List<S> population) {
    population = evaluator.evaluate(population, getProblem());

    return population;
  }
```

```java
public class SteadyStateNSGAII<S extends Solution<?>> extends NSGAII<S> {
  /**
   * Constructor
   */
  public SteadyStateNSGAII(Problem<S> problem, int maxEvaluations, int populationSize,
      CrossoverOperator<S> crossoverOperator, MutationOperator<S> mutationOperator,
      SelectionOperator<List<S>, S> selectionOperator, SolutionListEvaluator<S> evaluator) {
    super(problem, maxEvaluations, populationSize, crossoverOperator, mutationOperator,
        selectionOperator, evaluator);
  }

  @Override protected void updateProgress() {
    evaluations ++ ;
  }


  @Override protected List<S> selection(List<S> population) {
    List<S> matingPopulation = new ArrayList<>(2);

    matingPopulation.add(selectionOperator.execute(population));
    matingPopulation.add(selectionOperator.execute(population));

    return matingPopulation;
  }

  @Override protected List<S> reproduction(List<S> population) {
    List<S> offspringPopulation = new ArrayList<>(1);

    List<S> parents = new ArrayList<>(2);
    parents.add(population.get(0));
    parents.add(population.get(1));

    List<S> offspring = crossoverOperator.execute(parents);

    mutationOperator.execute(offspring.get(0));

    offspringPopulation.add(offspring.get(0));
    return offspringPopulation;
  }
```

About chosing the proper algorithm

- Given a problem, finding the best algorithm to solve it is not trivial
  - No Free Lunch theorem
- Experience helps, but every problem is different:



Problem: Structural Design Optimization
Domain: Civil Engineering
Quality indicator: Hypervolume

Hypervolume (HV)			Epsilon ($I_{\epsilon+}$)		
Algorithm	Fri_{Rank}	$Holm_{Ap}$	Algorithm	Fri_{Rank}	$Holm_{Ap}$
***SMPSO**	**1.02**	-	***SMPSO**	**1.09**	-
MOEA/D	2.68	2.24e-03	MOEA/D	2.00	9.87e-02
GDE3	3.09	1.45e-04	GDE3	3.09	2.79e-04
NSGA-II	3.22	5.21e-05	NSGA-II	3.81	7.25e-07

Problem: Molecular Docking Optimization
Domain: Bioinformatics
Quality indicators: Hypervolume, Epsilon

About chosing the proper algorithm

- What to do when you have to solve a new problem?
- Questions:
  - Is the problem continuous or combinatorial?
    - If combinatorial DE and PSO are discarded
  - Is a constrained problem?
- Some hints:
  - Start with a simple algorithm (or one your are familiarized with)
    - Random search -> sanity check
    - PAES: (1+1) Evolution Strategy -> requires only mutation
    - NSGA-II without crossover
  - Once you have some reference fronts
    - Pilot tests with promising algorithms to adjusts their settings
    - Perform an empirical study

About chosing the proper algorithm

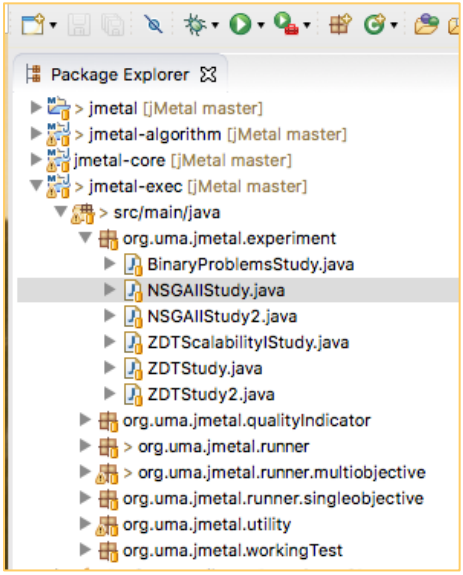
- Empirical studies
  - Idea: compare a number of algorithms over the problem/s to optimize
- Methodology
  - Select the metaheuristics to compare
  - Perform a number of independent runs per configuration (> 30)
  - Apply quality indicators for convergence and diversity
  - Report statistical data:
    - Mean/median + standard deviation/interquartile range
    - Statistical tests about the significance of the results among the algorithms: t-test, Wilcoxon, Kolmogorov Smirnoff, etc.
  - Analysis of the results

Table of contents

- Who, why, what, when
- Background
- Starting with jMetal
- **Experimental studies**
- Case study: the next release problem
- Further developments and related projects

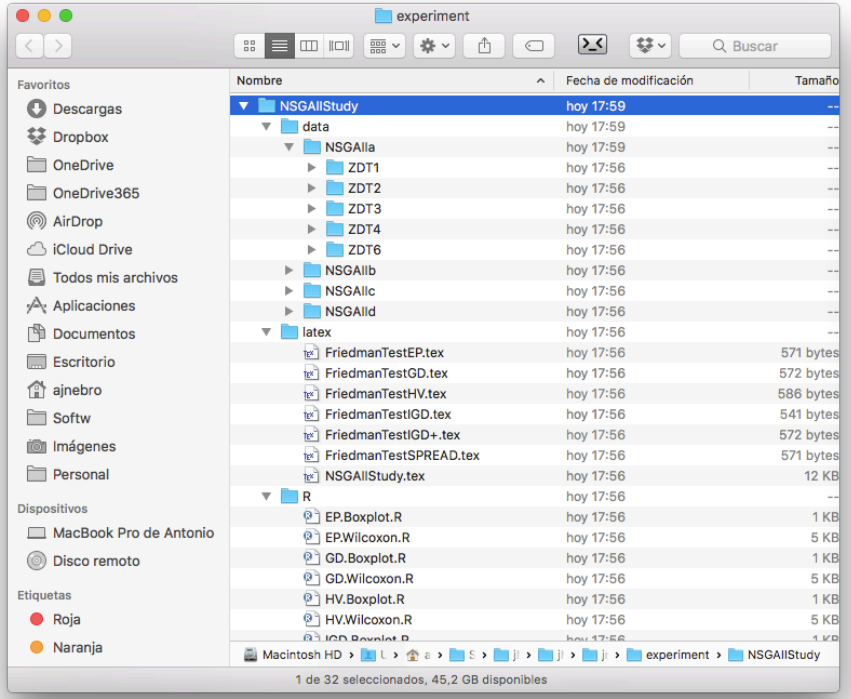
Experimental studies

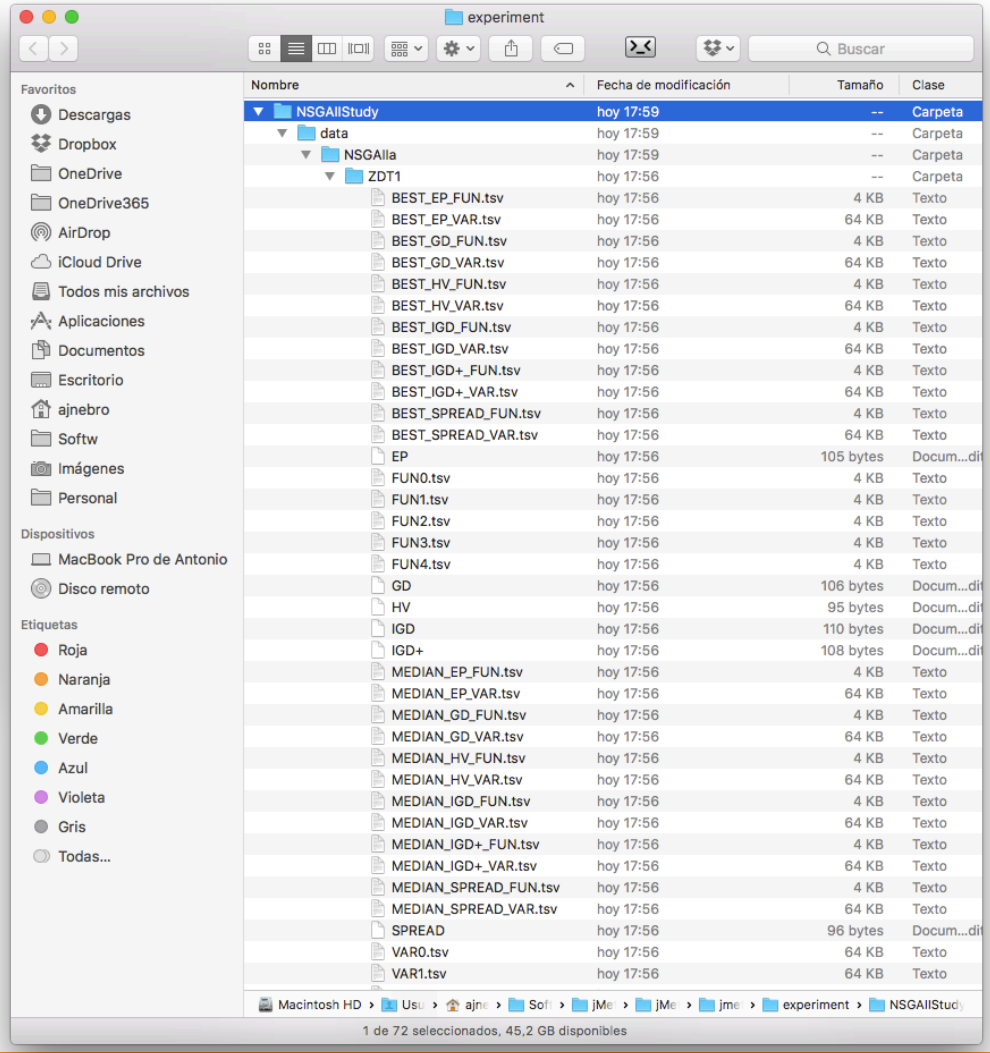
- jMetal provides a set of utilities to carry out experimental studies




```java
public class NSGAIIStudy {
  private static final int INDEPENDENT_RUNS = 25;

  public static void main(String[] args) throws IOException {
    if (args.length != 1) {
      throw new JMetalException("Missing argument: experimentBaseDirectory");
    }
    String experimentBaseDirectory = args[0];

    List<ExperimentProblem<DoubleSolution>> problemList = new ArrayList<>();
    problemList.add(new ExperimentProblem<>(new ZDT1()));
    problemList.add(new ExperimentProblem<>(new ZDT2()));
    problemList.add(new ExperimentProblem<>(new ZDT3()));
    problemList.add(new ExperimentProblem<>(new ZDT4()));
    problemList.add(new ExperimentProblem<>(new ZDT6()));

    List<ExperimentAlgorithm<DoubleSolution, List<DoubleSolution>>> algorithmList =
            configureAlgorithmList(problemList);

    List<String> referenceFrontFileNames = Arrays.asList("ZDT1.pf", "ZDT2.pf", "ZDT3.pf", "ZDT4.pf", "ZDT6.pf");

    Experiment<DoubleSolution, List<DoubleSolution>> experiment =
            new ExperimentBuilder<DoubleSolution, List<DoubleSolution>>("NSGAIIStudy")
                    .setAlgorithmList(algorithmList)
                    .setProblemList(problemList)
                    .setExperimentBaseDirectory(experimentBaseDirectory)
                    .setOutputParetoFrontFileName("FUN")
                    .setOutputParetoSetFileName("VAR")
                    .setReferenceFrontDirectory("/pareto_fronts")
                    .setReferenceFrontFileNames(referenceFrontFileNames)
                    .setIndicatorList(Arrays.asList(
                            new Epsilon<DoubleSolution>(),
                            new Spread<DoubleSolution>(),
                            new GenerationalDistance<DoubleSolution>(),
                            new PISAHypervolume<DoubleSolution>(),
                            new InvertedGenerationalDistance<DoubleSolution>(),
                            new InvertedGenerationalDistancePlus<DoubleSolution>()))
                    .setIndependentRuns(INDEPENDENT_RUNS)
                    .setNumberOfCores(8)
                    .build();

    new ExecuteAlgorithms<>(experiment).run();
    new ComputeQualityIndicators<>(experiment).run();
    new GenerateLatexTablesWithStatistics(experiment).run();
    new GenerateWilcoxonTestTablesWithR<>(experiment).run();
    new GenerateFriedmanTestTables<>(experiment).run();
    new GenerateBoxplotsWithR<>(experiment).setRows(3).setColumns(3).run();
```

Experimental studies: NSGAIIStudy





Output directories and files

Experimental studies: NSGAIIStudy

```
latex — -bash — 91×22
ajnebro ~/Softw/jMetal/jMetal/jmetal-exec/experiment/NSGAIIStudy $ cd latex
ajnebro ~/Softw/jMetal/jMetal/jmetal-exec/experiment/NSGAIIStudy/latex $ ls
FriedmanTestEP.tex      FriedmanTestIGD+.tex    NSGAIIStudy.tex
FriedmanTestGD.tex      FriedmanTestIGD.tex
FriedmanTestHV.tex      FriedmanTestSPREAD.tex
ajnebro ~/Softw/jMetal/jMetal/jmetal-exec/experiment/NSGAIIStudy/latex $ for i in *tex ; do
 pdflatex $i ; done
This is pdfTeX, Version 3.14159265-2.6-1.40.17 (TeX Live 2016) (preloaded format=pdflatex)
 restricted \write18 enabled.
entering extended mode
(./FriedmanTestEP.tex
LaTeX2e <2016/03/31>
Babel <3.9r> and hyphenation patterns for 83 language(s) loaded.
(/usr/local/texlive/2016/texmf-dist/tex/latex/base/article.cls
Document Class: article 2014/09/29 v1.4h Standard LaTeX document class
(/usr/local/texlive/2016/texmf-dist/tex/latex/base/size10.clo))
(/usr/local/texlive/2016/texmf-dist/tex/latex/graphics/graphicx.sty
(/usr/local/texlive/2016/texmf-dist/tex/latex/graphics/keyval.sty)
(/usr/local/texlive/2016/texmf-dist/tex/latex/graphics/graphics.sty
(/usr/local/texlive/2016/texmf-dist/tex/latex/graphics/trig.sty)
(/usr/local/texlive/2016/texmf-dist/tex/latex/graphics-cfg/graphics.cfg)
(/usr/local/texlive/2016/texmf-dist/tex/latex/pdftex-def/pdftex.def
```

Generating Latex tables

Table 1: Average ranking of the algorithms

Algorithm	Ranking
NSGAIIa	1.8
NSGAIIb	2.0
NSGAIIc	2.6
NSGAIId	3.6

Friedman statistic considering reduction performance (distributed according to chi-square with 3 degrees of freedom: 5.880000000000003).

Table 2: EP. Median and Interquartile Range

	NSGAIIa	NSGAIIb	NSGAIIc	NSGAIId
ZDT1	$1.29e-02_{4.5e-03}$	$1.37e-02_{2.8e-03}$	$1.40e-02_{3.1e-03}$	$1.54e-02_{2.3e-02}$
ZDT2	$1.18e-02_{1.8e-03}$	$1.25e-02_{3.8e-03}$	$1.85e-02_{4.6e-01}$	$4.53e-01_{8.7e-01}$
ZDT3	$7.44e-03_{1.2e-03}$	$7.88e-03_{2.2e-03}$	$8.48e-03_{3.3e-03}$	$1.02e-02_{1.8e-01}$
ZDT4	$1.59e-02_{6.9e-03}$	$1.41e-02_{4.3e-03}$	$1.74e-01_{1.9e-01}$	$7.83e-01_{4.4e-01}$
ZDT6	$1.94e-02_{3.4e-03}$	$1.73e-02_{3.1e-03}$	$1.78e-02_{2.2e-03}$	$1.82e-02_{1.9e-03}$

Table 7: HV. Mean and Standard Deviation

	NSGAIIa	NSGAIIb	NSGAIIc	NSGAIId
ZDT1	$6.59e-01_{3.0e-04}$	$6.59e-01_{3.7e-04}$	$6.59e-01_{3.2e-04}$	$6.56e-01_{9.2e-03}$
ZDT2	$3.26e-01_{2.6e-04}$	$3.26e-01_{3.3e-04}$	$2.98e-01_{4.0e-02}$	$1.89e-01_{1.4e-01}$
ZDT3	$5.14e-01_{1.8e-04}$	$5.14e-01_{8.9e-04}$	$5.13e-01_{8.2e-03}$	$5.00e-01_{2.1e-02}$
ZDT4	$6.49e-01_{1.5e-02}$	$6.55e-01_{3.4e-03}$	$6.17e-01_{3.8e-02}$	$2.21e-01_{1.8e-01}$
ZDT6	$3.86e-01_{1.7e-03}$	$3.89e-01_{1.1e-03}$	$3.90e-01_{1.5e-03}$	$3.89e-01_{2.2e-03}$

Experimental studies: NSGAIIStudy

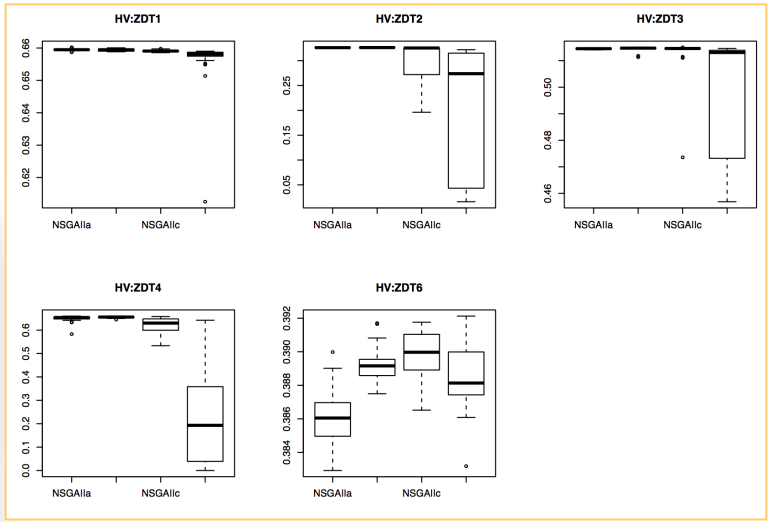
Running R scripts

```
R — -bash — 91×19
ajnebro ~/Softw/jMetal/jMetal/jmetal-exec/experiment/NSGAIIStudy/R $ !f
for i in *R; do Rscript $i ; done
Read 25 items
Read 25 items
Read 25 items
Read 25 items
Read 25 items
Read 25 items
Read 25 items
Read 25 items
Read 25 items
Read 25 items
```

```
R — -bash — 91×10
ajnebro ~/Softw/jMetal/jMetal/jmetal-exec/experiment/NSGAIIStudy/R $ ls
EP.Boxplot.R            HV.Boxplot.R            IGD.Boxplot.R
EP.Boxplot.eps          HV.Boxplot.eps          IGD.Boxplot.eps
EP.Wilcoxon.R           HV.Wilcoxon.R           IGD.Wilcoxon.R
EP.Wilcoxon.tex         HV.Wilcoxon.tex         IGD.Wilcoxon.tex
GD.Boxplot.R            IGD+.Boxplot.R          SPREAD.Boxplot.R
GD.Boxplot.eps          IGD+.Boxplot.eps        SPREAD.Boxplot.eps
GD.Wilcoxon.R           IGD+.Wilcoxon.R         SPREAD.Wilcoxon.R
GD.Wilcoxon.tex         IGD+.Wilcoxon.tex       SPREAD.Wilcoxon.tex
ajnebro ~/Softw/jMetal/jMetal/jmetal-exec/experiment/NSGAIIStudy/R $
```

```
R — -bash — 91×10
ajnebro ~/Softw/jMetal/jMetal/jmetal-exec/experiment/NSGAIIStudy/R $ for i in *tex ; do pdf
latex $i ; done
This is pdfTeX, Version 3.14159265-2.6-1.40.17 (TeX Live 2016) (preloaded format=pdflatex)
 restricted \write18 enabled.
entering extended mode
(./EP.Wilcoxon.tex
LaTeX2e <2016/03/31>
Babel <3.9r> and hyphenation patterns for 83 language(s) loaded.
(/usr/local/texlive/2016/texmf-dist/tex/latex/base/article.cls
Document Class: article 2014/09/29 v1.4h Standard LaTeX document class
```

Boxplots



Wilcoxon Rank-Sum Test

Table 1: ZDT1 ZDT2 ZDT3 ZDT4 ZDT6 .HV.

	NSGAIIb					NSGAIIc					NSGAIId				
NSGAIIa	–	–	▽	–	▽	▲	▲	–	▲	▽	▲	▲	▲	▲	▽
NSGAIIb						▲	▲	–	▲	–	▲	▲	▲	▲	–
NSGAIIc											▲	▲	▲	▲	–

Table of contents

- Who, why, what, when
- Background
- Starting with jMetal
- Experimental studies
- **Case study: the next release problem**
- Further research and related projects

Case study: the next release problem

- Next Release Problem (NRP)

> "One important issue addressed by software companies is to determine which features should be included in the next release of their products, in such a way that the highest possible number of customers get satisfied while entailing the minimum cost for the company. This problem is known as the Next Release Problem (NRP). Since minimizing the total cost of including new features into a software package and maximizing the total satisfaction of customers are contradictory objectives, the problem has a multi-objective nature."

Empirical Software Engineering

February 2011, Volume 16, Issue 1, pp 29–60

A study of the bi-objective next release problem

Authors | Authors and affiliations

Juan J. Durillo, Yuanyuan Zhang, Enrique Alba, Mark Harman, Antonio J. Nebro

Case study: the next release problem

- Step 1: Create a Maven project adding jMetal as a dependence

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>ssbse2017</groupId>
    <artifactId>nrp</artifactId>
    <version>1.0-SNAPSHOT</version>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.1</version>
                <configuration>
                    <!-- or whatever version you use -->
                    <source>1.8</source>
                    <target>1.8</target>
                    <encoding>UTF-8</encoding>
                </configuration>
            </plugin>
            <plugin>
                <artifactId>maven-assembly-plugin</artifactId>
                <version>2.4</version>
                <configuration>
                    <descriptorRefs>
                        <descriptorRef>jar-with-dependencies</descriptorRef>
                    </descriptorRefs>
                </configuration>
                <executions>
                    <execution>
                        <id>make-assembly</id>
                        <phase>package</phase>
                        <goals>
                            <goal>single</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
```

```xml
    <dependencies>
        <dependency>
            <groupId>org.uma.jmetal</groupId>
            <artifactId>jmetal-core</artifactId>
            <version>5.3</version>
        </dependency>
        <dependency>
            <groupId>org.uma.jmetal</groupId>
            <artifactId>jmetal-algorithm</artifactId>
            <version>5.3</version>
        </dependency>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

Case study: the next release problem

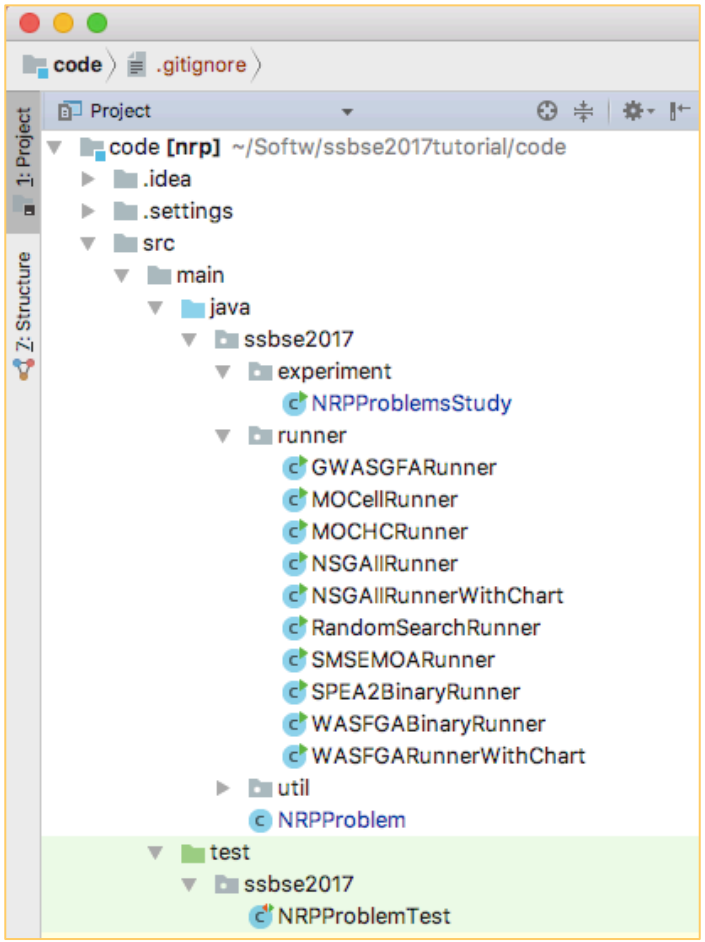
- Step 2: Define the problem
  - We have implemented a NRP problem generator

```
package ssbse2017;

import org.uma.jmetal.problem.impl.AbstractBinaryProblem;
import org.uma.jmetal.solution.BinarySolution;

import java.util.BitSet;
import java.util.Random;
import java.util.stream.IntStream;

public class NRPProblem extends AbstractBinaryProblem {
  private int numberOfRequirements ;
  private int highestClientSatisfactionValue ;
  private int highestRequirementCost ;
  private int numberOfClients ;
  private int highestImportanceValue ;
  private long randomSeed;

  private int requirementCost[] ;
  private int clientSatisfaction[] ;
  private int importanceMatrix[][] ;

  public NRPProblem(
          int numberOfClients,
          int numberOfRequirements,
          int highestClientSatisfactionValue,
          int highestRequirementCost,
          int highestImportanceValue,
          long randomSeed) {
    this.numberOfClients = numberOfClients ;
    this.numberOfRequirements = numberOfRequirements ;
    this.highestRequirementCost = highestRequirementCost ;
    this.highestClientSatisfactionValue = highestClientSatisfactionValue ;
    this.highestImportanceValue = highestImportanceValue ;
    this.randomSeed = randomSeed ;

    this.setNumberOfVariables(1);
    this.setNumberOfObjectives(2);
    this.setNumberOfConstraints(0);

    Random random = new Random(this.randomSeed) ;

    this.requirementCost = new int[this.numberOfRequirements] ;
    IntStream
            .range(0, this.numberOfRequirements)
            .forEach(i -> this.requirementCost[i] = random.nextInt(this.highestRequirementCost) + 1) ;

    this.clientSatisfaction = new int[this.numberOfClients] ;
```

```
@Override
public void evaluate(BinarySolution solution) {
  solution.setObjective(0, computeCost(solution));
  // satisfaction is a maximization objective: multiply by -1 to minimize
  solution.setObjective(1, -1.0 * computeSatisfaction(solution));
}

private double computeCost(BinarySolution solution) {
  double result = 0.0 ;
  BitSet bitset = solution.getVariableValue(0) ;

  for (int i = 0; i < this.numberOfRequirements; i++) {
    if (bitset.get(i)) {
      result += requirementCost[i] ;
    }
  }

  return result ;
}

private double computeSatisfaction(BinarySolution solution) {
  double result = 0.0 ;
  BitSet bitset = solution.getVariableValue(0) ;

  for (int client = 0; client < numberOfClients; client++) {
    double sum = 0.0 ;
    for (int i = 0; i < this.numberOfRequirements; i++) {
      if (bitset.get(i)) {
        sum += importanceMatrix[client][i] ;
      }
    }
    result += sum * clientSatisfaction[client] ;
  }

  return result ;
}
```

Case study: the next release problem

- Step 3: Choose the algorithms (binary encoding)



Encoding:

- Binary array with a bit per requirement (true if a requirement is selected, false otherwise)

Algorithm list:

- NSGA-II (classical GA, 2002)
- SPEA2 (classical GA, 2001)
- MOCHC (CHC, 2007)
- SMS-EMOA (GA, indicator based, 2007)
- G-WASF-GA (GA, decomposition based, 2017)
- Random search

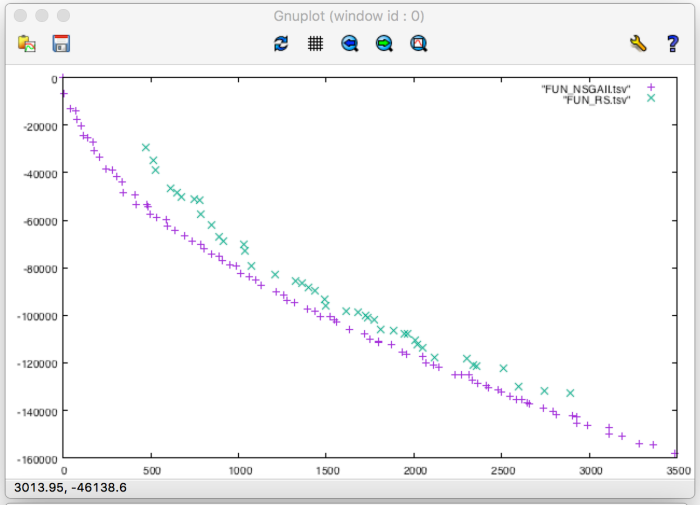
Case study: the next release problem

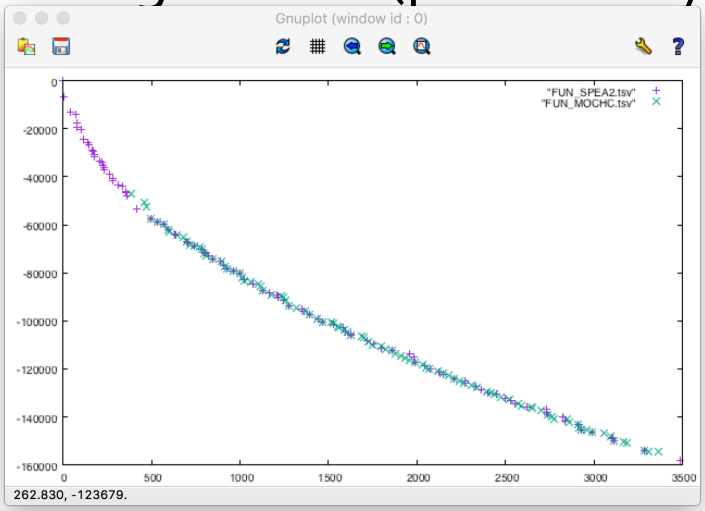
- Step 4: Configure and run the algorithms (preliminary experiments)

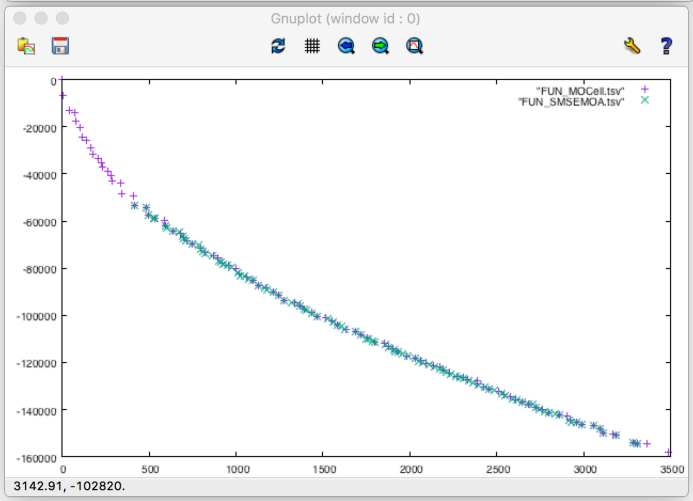
```
public class NSGAIIRunner {
  /**...*/
  public static void main(String[] args) throws JMetalException {
    Algorithm<List<BinarySolution>> algorithm;
    CrossoverOperator<BinarySolution> crossover;
    MutationOperator<BinarySolution> mutation;
    SelectionOperator<List<BinarySolution>, BinarySolution> selection;

    int numberOfClients = 20 ;
    int numberOfRequirements = 30 ;
    int highestClientSatisfactionValue = 10 ;
    int highestRequirementCost = 200 ;
    int highestImportanceValue = 100 ;
    int randomSeed = 1 ;
    NRPProblem problem = new NRPProblem(
            numberOfClients,
            numberOfRequirements,
            highestClientSatisfactionValue,
            highestRequirementCost,
            highestImportanceValue,
            randomSeed) ;

    double crossoverProbability = 0.9 ;
    crossover = new SinglePointCrossover(crossoverProbability) ;

    double mutationProbability = 1.0 / numberOfRequirements ;
    mutation = new BitFlipMutation(mutationProbability) ;

    selection = new BinaryTournamentSelection<>(
        new RankingAndCrowdingDistanceComparator<>());

    algorithm = new NSGAIIBuilder<>(problem, crossover, mutation)
        .setSelectionOperator(selection)
        .setMaxEvaluations(25000)
        .setPopulationSize(100)
        .build() ;

    AlgorithmRunner algorithmRunner = new AlgorithmRunner.Executor(algorithm)
        .execute() ;

    List<BinarySolution> population = algorithm.getResult() ;
    long computingTime = algorithmRunner.getComputingTime() ;
```

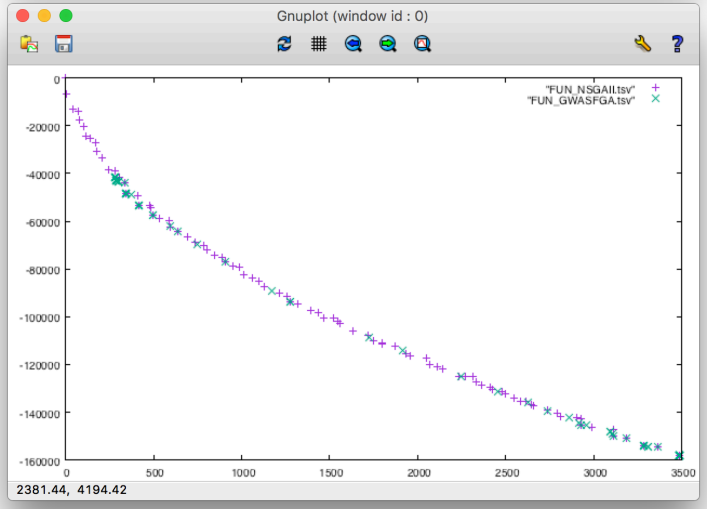
Case study: the next release problem

- Step 4: Configure and run the algorithms (preliminary experiments)









Case study: the next release problem

- Step 5: Prepare and run an experimental study

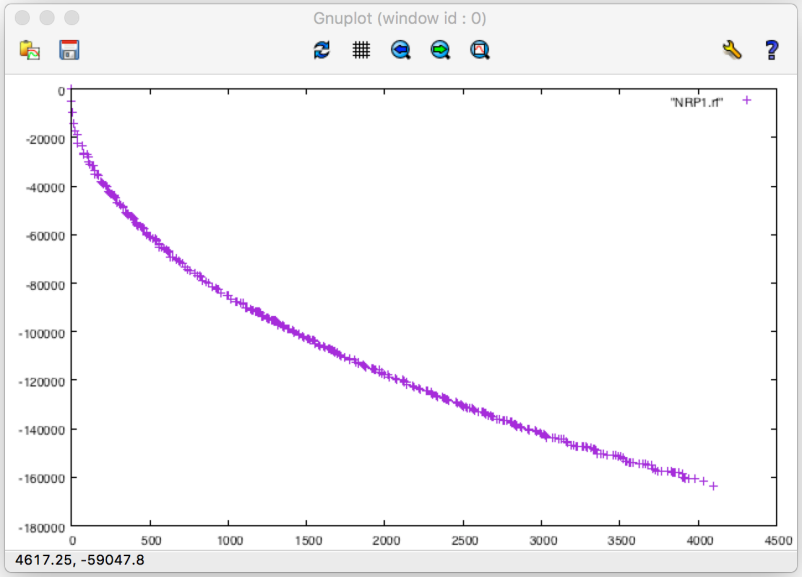
```
public class NRPProblemsStudy {
  private static final int INDEPENDENT_RUNS = 10;

  public static void main(String[] args) throws IOException {
    if (args.length != 1) {
      throw new JMetalException("Needed arguments: experimentBaseDirectory");
    }
    String experimentBaseDirectory = args[0];

    int highestClientSatisfactionValue = 10 ;
    int highestRequirementCost = 200 ;
    int highestImportanceValue = 100 ;
    int randomSeed = 1 ;
    NRPProblem problem1 = new NRPProblem(
            15,
            40,
            highestClientSatisfactionValue,
            highestRequirementCost,
            highestImportanceValue,
            randomSeed) ;

    NRPProblem problem2 = new NRPProblem(
            50,
            80,
            highestClientSatisfactionValue,
            highestRequirementCost,
            highestImportanceValue,
            randomSeed) ;

    NRPProblem problem3 = new NRPProblem(
            100,
            140,
            highestClientSatisfactionValue,
            highestRequirementCost,
            highestImportanceValue,
            randomSeed) ;

    NRPProblem problem4 = new NRPProblem(
            200,
            300,
            highestClientSatisfactionValue,
            highestRequirementCost,
            highestImportanceValue,
            randomSeed) ;

    List<ExperimentProblem<BinarySolution>> problemList = new ArrayList<>();
    problemList.add(new ExperimentProblem<>(problem1, "NRP1"));
    problemList.add(new ExperimentProblem<>(problem2, "NRP2"));
    problemList.add(new ExperimentProblem<>(problem3, "NRP3"));
    problemList.add(new ExperimentProblem<>(problem4, "NRP4"));
```

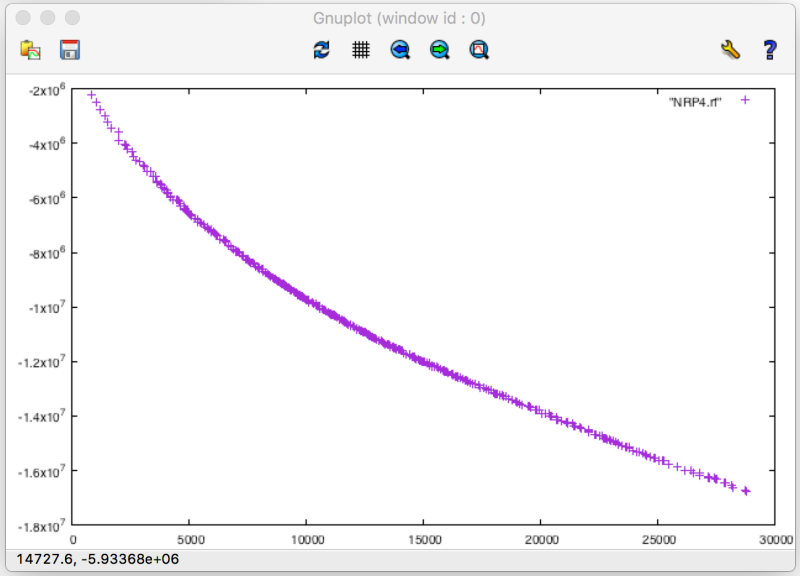
```
List<ExperimentAlgorithm<BinarySolution, List<BinarySolution>>> algorithmList =
        configureAlgorithmList(problemList);

Experiment<BinarySolution, List<BinarySolution>> experiment;
experiment = new ExperimentBuilder<BinarySolution, List<BinarySolution>>("NRPProblemsStudy")
        .setAlgorithmList(algorithmList)
        .setProblemList(problemList)
        .setExperimentBaseDirectory(experimentBaseDirectory)
        .setOutputParetoFrontFileName("FUN")
        .setOutputParetoSetFileName("VAR")
        .setReferenceFrontDirectory(experimentBaseDirectory+"/referenceFronts")
        .setIndicatorList(Arrays.asList(
                new Epsilon<BinarySolution>(),
                new Spread<BinarySolution>(),
                new PISAHypervolume<BinarySolution>(),
                new InvertedGenerationalDistance<BinarySolution>(),
                new InvertedGenerationalDistancePlus<BinarySolution>())
        )
        .setIndependentRuns(INDEPENDENT_RUNS)
        .setNumberOfCores(8)
        .build();

new ExecuteAlgorithms<>(experiment).run();
new GenerateReferenceParetoFront(experiment).run();
new ComputeQualityIndicators<>(experiment).run();
new GenerateLatexTablesWithStatistics(experiment).run();
new GenerateWilcoxonTestTablesWithR<>(experiment).run();
new GenerateFriedmanTestTables<>(experiment).run();
new GenerateBoxplotsWithR<>(experiment).setRows(2).setColumns(2).setDisplayNotch().run();
```

Case study: the next release problem

- Step 6: Analyze the results

```
experiment — -bash — 91×11

ajnebro ~/Softw/ssbse2017tutorial/experiment $ ls
NRPProblemsStudy        referenceFronts
ajnebro ~/Softw/ssbse2017tutorial/experiment $ ls referenceFronts/
NRP1.rf NRP2.rf NRP3.rf NRP4.rf
ajnebro ~/Softw/ssbse2017tutorial/experiment $ ls NRPProblemsStudy/
R       data    latex
ajnebro ~/Softw/ssbse2017tutorial/experiment $
```





Case study: the next release problem

- Step 6: Analyze the results

Table 2: EP. Median and Interquartile Range

	NSGAII	SPEA2	MOCell	MOCHC
NRP1	$1.51e-02_{3.4e-03}$	$1.91e-02_{1.0e-02}$	$8.07e-03_{1.6e-03}$	$9.53e-02_{2.2e-02}$
NRP2	$4.59e-02_{2.2e-02}$	$2.71e-02_{5.5e-02}$	$8.23e-03_{2.1e-03}$	$1.62e-01_{2.5e-02}$
NRP3	$1.11e-01_{2.6e-02}$	$7.50e-02_{7.2e-02}$	$1.43e-02_{2.8e-02}$	$2.10e-01_{2.4e-02}$
NRP4	$2.09e-01_{3.3e-02}$	$1.55e-01_{5.0e-02}$	$3.08e-02_{6.6e-02}$	$2.72e-01_{3.2e-02}$

Table 1: Average ranking of the algorithms

Algorithm	Ranking
NSGAII	2.5
SPEA2	2.5
MOCell	1.0
MOCHC	4.0

Table 1: NRP1 NRP2 NRP3 NRP4 .EP.

	SPEA2				MOCell				MOCHC			
NSGAII	–	–	▽	▽	▽	▽	▽	▽	▲	▲	▲	▲
SPEA2					▽	▽	▽	▽	▲	▲	▲	▲
MOCell									▲	▲	▲	▲

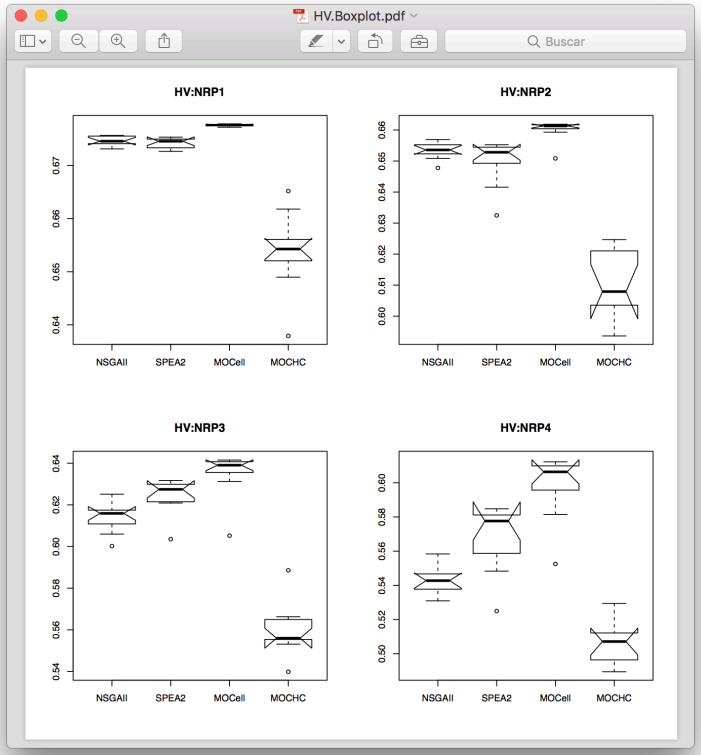


Table 6: HV. Median and Interquartile Range

	NSGAII	SPEA2	MOCell	MOCHC
NRP1	$6.75e-01_{1.5e-03}$	$6.75e-01_{1.8e-03}$	$6.78e-01_{2.7e-04}$	$6.54e-01_{6.2e-03}$
NRP2	$6.54e-01_{3.5e-03}$	$6.53e-01_{7.3e-03}$	$6.61e-01_{1.6e-03}$	$6.08e-01_{2.0e-02}$
NRP3	$6.16e-01_{8.5e-03}$	$6.27e-01_{8.9e-03}$	$6.39e-01_{6.4e-03}$	$5.56e-01_{1.0e-02}$
NRP4	$5.43e-01_{1.0e-02}$	$5.78e-01_{2.5e-02}$	$6.06e-01_{1.8e-02}$	$5.07e-01_{1.9e-02}$

Table 1: Average ranking of the algorithms

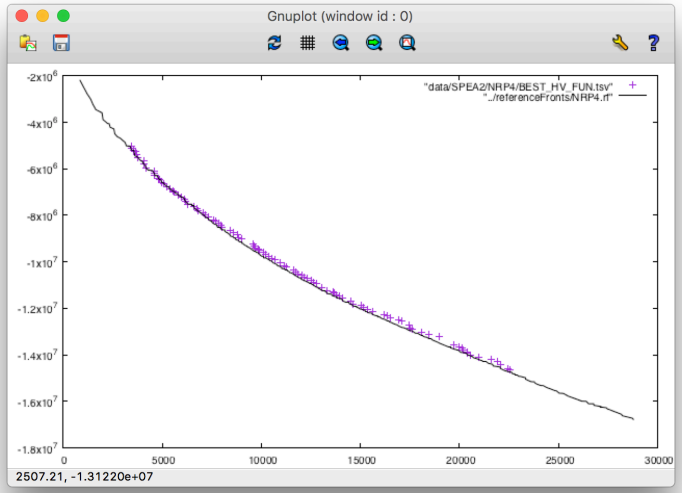
Algorithm	Ranking
NSGAII	2.5
SPEA2	2.5
MOCell	4.0
MOCHC	1.0

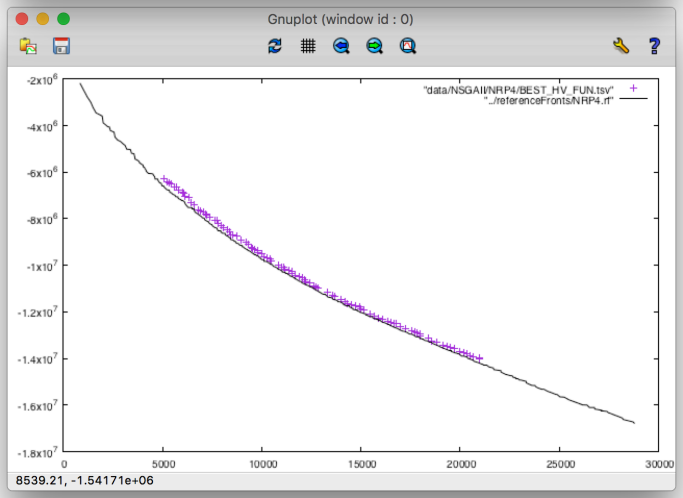
Table 1: NRP1 NRP2 NRP3 NRP4 .HV.

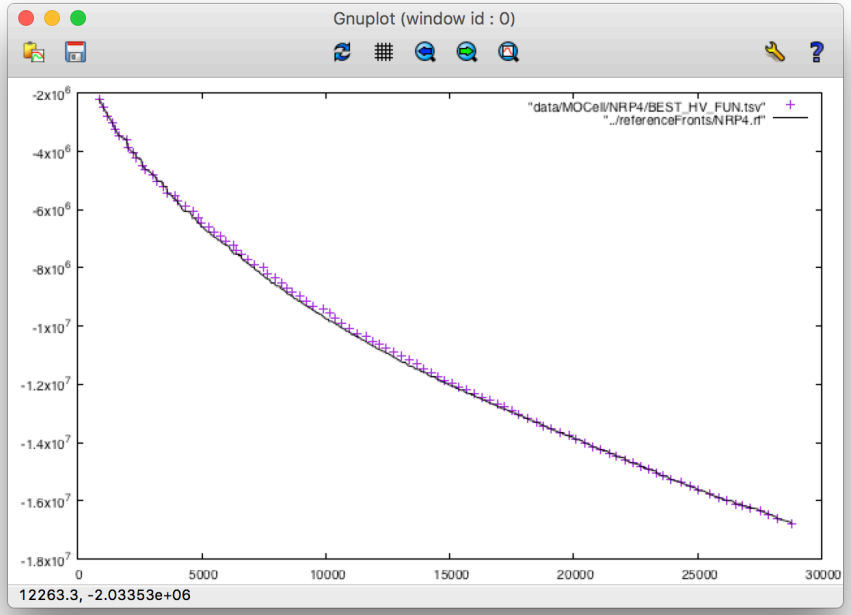
	SPEA2				MOCell				MOCHC			
NSGAII	–	–	▽	▽	▽	▽	▽	▽	▲	▲	▲	▲
SPEA2					▽	▽	▽	▽	▲	▲	▲	▲
MOCell									▲	▲	▲	▲

Case study: the next release problem

- Step 6: Analyze the results



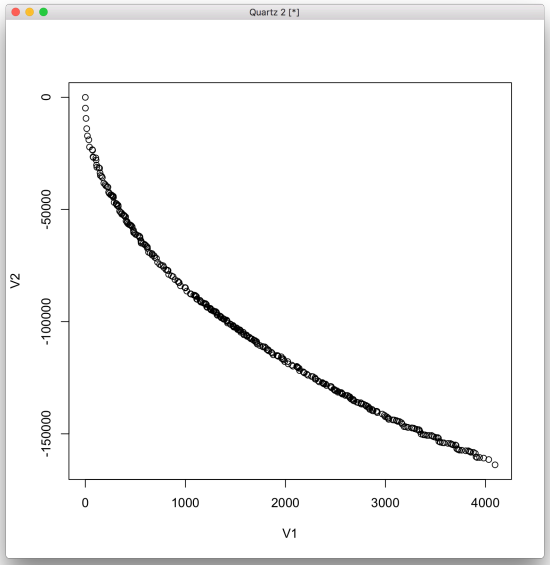




Case study: the next release problem

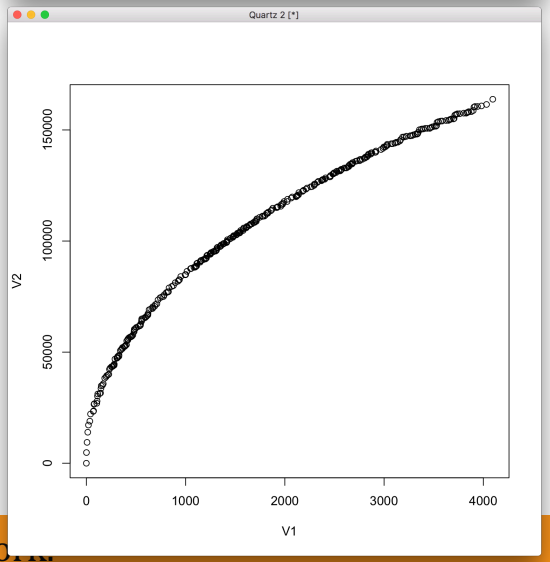
- Step 6: Analyze the results

```
referenceFronts — R — 91×12
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

[Previously saved workspace restored]

> nrp1 <- read.table("NRP1.rf")
> plot(nrp1)
>
```



Objective 2 must be multiplied by -1
to obtain the real front

```
referenceFronts — R — 91×11
> str(nrp1)
'data.frame':   370 obs. of  2 variables:
 $ V1: num  4095 1449 1380 1522 288 ...
 $ V2: num  -163814 -101032 -98735 -103521 -46874 ...
> nrp1$V2 <- nrp1$V2 * -1
> str(nrp1)
'data.frame':   370 obs. of  2 variables:
 $ V1: num  4095 1449 1380 1522 288 ...
 $ V2: num  163814 101032 98735 103521 46874 ...
> plot(nrp1)
>
```



Case study: the next release problem

- What can be concluded from the NRP study?
  - MOCell is the best performing algorithm in the context of:
    - The formulation of the NRP
    - The four NRP instances solved
    - The four selected metaheuristics
    - The parameter settings used to configure the algorithms
    - The quality indicators
  - But we cannot know if this conclusion will hold in case of:
    - A change in the formulation of the problem (e.g., adding some constraints or additional objectives)
    - Other NRP instances
  - We don't know the true Pareto front, so all the results are relative to the algorithms' performance -> quantitative vs qualitative

Table of contents

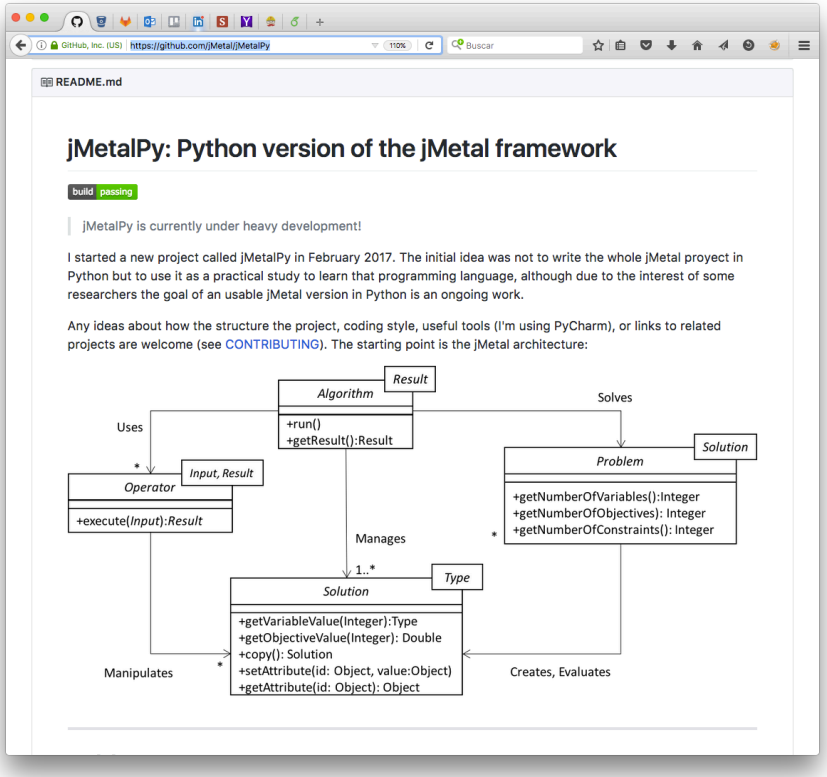
- Who, why, what, when
- Background
- Starting with jMetal
- Experimental studies
- Case study: the next release problem
- **Further developments and related projects**

Further development

- jMetal is continuously evolving
  - The experimental section is being redesigned
  - New algorithms are to be included
    - We are interested now in metaheuristics with support for
      - Decision making
      - Dynamic problems
  - The documentation is not complete
  - The project is not full covered with unit and integration tests

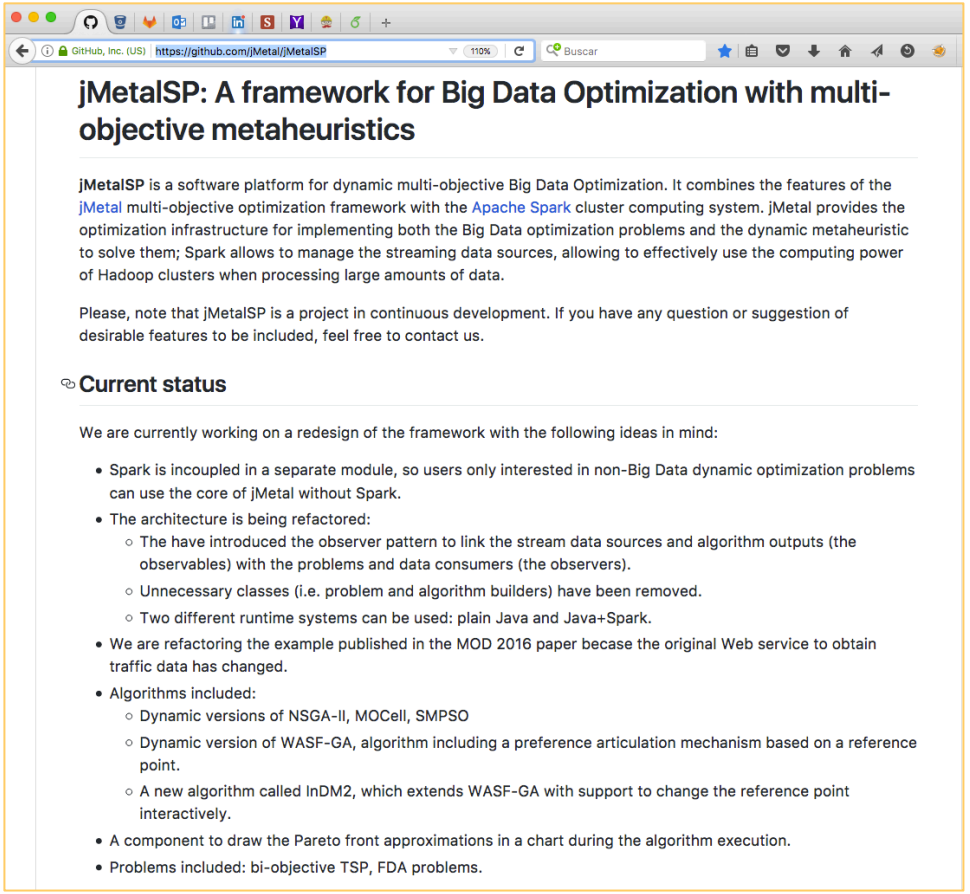
Related projects

- jMetalPy: Python version of jMetal
  - https://github.com/jMetal/jMetalPy
  - Motivation: our interest in Big Data and Bioinformatics



Related projects

- jMetalSP
  - https://github.com/jMetal/jMetalSP
  - Dynamic problems
  - Streaming data processing with Big Data technologies (Apache Spark)



jMetalSP: A framework for Big Data Optimization with multi-objective metaheuristics

jMetalSP is a software platform for dynamic multi-objective Big Data Optimization. It combines the features of the jMetal multi-objective optimization framework with the Apache Spark cluster computing system. jMetal provides the optimization infrastructure for implementing both the Big Data optimization problems and the dynamic metaheuristic to solve them; Spark allows to manage the streaming data sources, allowing to effectively use the computing power of Hadoop clusters when processing large amounts of data.

Please, note that jMetalSP is a project in continuous development. If you have any question or suggestion of desirable features to be included, feel free to contact us.

Current status

We are currently working on a redesign of the framework with the following ideas in mind:

- Spark is incoupled in a separate module, so users only interested in non-Big Data dynamic optimization problems can use the core of jMetal without Spark.
- The architecture is being refactored:
  - The have introduced the observer pattern to link the stream data sources and algorithm outputs (the observables) with the problems and data consumers (the observers).
  - Unnecessary classes (i.e. problem and algorithm builders) have been removed.
  - Two different runtime systems can be used: plain Java and Java+Spark.
- We are refactoring the example published in the MOD 2016 paper becase the original Web service to obtain traffic data has changed.
- Algorithms included:
  - Dynamic versions of NSGA-II, MOCell, SMPSO
  - Dynamic version of WASF-GA, algorithm including a preference articulation mechanism based on a reference point.
  - A new algorithm called InDM2, which extends WASF-GA with support to change the reference point interactively.
- A component to draw the Pareto front approximations in a chart during the algorithm execution.
- Problems included: bi-objective TSP, FDA problems.

Thanks for your attention ☺

- Comments and suggestions are welcome