PPSN 2018

8 – 12 September 2018 – Coimbra, Portugal

Multiobjective Optimization with the jMetal Framework

Antonio J. Nebro
PhD in Computer Science

ajnebro@uma.es

UNIVERSIDAD DE MÁLAGA

Khaos INVESTIGACIÓN

CC BY SA

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

Khaos INVESTIGACIÓN

About this tutorial

- jMetal and multi-objective optimization
  - jMetal is a tool widely used in the multi-objective optimization community

Cites

jMetal: A Java framework for multi-objective optimization
JJ Durillo, AJ Nebro
Advances in Engineering Software 42 (10), 760-771

763

The jMetal framework for multi-objective optimization: Design and architecture
JJ Durillo, AJ Nebro, E Alba
Evolutionary Computation (CEC), 2010 IEEE Congress on, 1-8

379

Redesigning the jMetal multi-objective optimization framework
AJ Nebro, JJ Durillo, M Vergne
Proceedings of the Companion Publication of the 2015 Annual Conference on …

47

1189

Source: Google Scholar, August 2018

Khaos INVESTIGACIÓN

About this tutorial

- Purpose of this tutorial
  - Offer a practical view of the jMetal framework
    - Description of its main components (algorithms, problems, operators, encodings, experiments, quality indicators)
    - Examples of classical algorithms and more modern techniques
  - Show how you can use jMetal to:
    - Configure and run multi-objective metaheuristics
    - Include your own problems
    - Implement a new algorithm
    - Develop experimental studies to assess the performance of algorithms

Khaos INVESTIGACIÓN

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/PPSN2018Tutorial

Table of contents

- **Who, why, what, when**
- Background
- Starting with jMetal
- jMetal components
- Experimental studies
- 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

Khaos INVESTIGACIÓN

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)
  - Scientific employee (Leibniz Supercomputing Centre of the Bavarian Academy of Sciencies and Humanities – Munich, Germany)

- Research interests
  - Multi-objective optimization
  - Software automatic tuning
  - Cloud computing
  - GPU computing
  - Machine learning
  - Parallel and distributed computing

Who, why, what, when

- Matthieu Vergne
  - PhD in ICT (2016)
  - Java developer since 2009
  - jMetal contributor since 2014
  - Micro-service developer since 2017
  - https://www.matthieu-vergne.fr/

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

Khaos INVESTIGACIÓN

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, GitHub)

Who, why, what, when

Khaos INVESTIGACIÓN

Table of contents

- Who, why, what, when
- **Background**
- Starting with jMetal
- jMetal components
- Experimental studies
- Further developments and related projects

Khaos INVESTIGACIÓN

What is multi-objective optimization?

- 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)

Civil Engineering Example: Cable Stayed Bridge Design

Gustavo R. Zavala, Antonio J. Nebro, Francisco Luna and Carlos A. Coello Coello, Structural design using multi-objective metaheuristics. Comparative study and application to a real-world problem. Structural and Multidisciplinary Optimization. Volume 53, Issue 3, pp 545-566. March 2016.

Civil Engineering Example: Cable Stayed Bridge Design

- The problem
  - Determine the dimensions of the bars
- Objectives to optimize
  - Miminize total weight
  - Minimize deformation
- Subject to a set of constraints
  - The bridge cannot be too week

Civil Engineering Example: Cable Stayed Bridge Design

Shape	Bars	Groups (gr)	Variables (var)	Geometric constraints	Mechanic constraints	Deflection constraints
Circle	6	2	1var/gr x 2gr = 2		3const/gr x 2gr = 6	
I-beam	14	1	4var/gr x 1gr = 4	4const/gr x 1gr = 4	3const/gr x 1gr = 3	20
Hollowed rectangle	15	5	4var/gr x 5gr = 20	4const/gr x 5gr = 20	3cont/gr x 5gr = 15	
Total	35	8	26	68		

Shape	Bars	Groups (gr)	Variables (var)	Geometric constraints	Mechanic constraints	Deflection constraints
Circle	46	8	1var/gr x 8gr = 8		3const/gr x 8gr = 24	
I-beam	39	4	4var/gr x 4gr = 16	4const/gr x 4gr = 16	3const/gr x 4gr = 12	68
Hollowed rectangle	136	21	4var/gr x 21gr = 84	4const/gr x 21gr = 84	3cont/gr x 21gr = 63	
Total	221	33	108	267		

Civil Engineering Example: Cable Stayed Bridge Design

- Results of the optimization of the bridge design

Structural design using multi-objective metaheuristics. Comparative study and application to a real-world problem
G Zavala, AJ Nebro, F Luna, CAC Coello
Structural and Multidisciplinary Optimization 53 (3), 545-566

Single vs Multi-Objective Optimization

- 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

Concepts: 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

Concepts: 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)

$$
\begin{aligned}
\text{Min } F &= (f_1(\vec{x}), f_2(\vec{x})) \\
f_1(\vec{x}) &= 4x_1^2 + 4x_2 \\
f_2(\vec{x}) &= (x_1 - 5)^2 + (x_2 - 5)^2
\end{aligned}
$$

Subject to:

$$
\begin{aligned}
g_1(\vec{x}) &= (x_1 - 5)^2 + x_2^2 - 25 && \leq 0 \\
g_2(\vec{x}) &= -(x_1 - 8)^2 - (x_2 + 3)^2 + 7.7 && \leq 0 \\
0 &\leq x_1 && \leq 5 \\
0 &\geq x_2 && \leq 3
\end{aligned}
$$

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

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

Khaos INVESTIGACIÓN

Table of contents

- Who, why, what, when
- Background
- **Starting with jMetal**
- jMetal components
- Experimental studies
- Further developments and related projects

Software requirements

- Requirements to run jMetal
  - Java JDK 8+
  - Maven
  - Gnuplot, R, … (data plotting)
  - R and Latex (algorithm performance comparison)

jMetal: a Maven Project Hosted in GitHub

https://jmetal.github.io/jMetal/

https://github.com/jMetal/jMetal

https://github.com/jMetal/jMetalDocumentation

Khaos INVESTIGACIÓN

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 — -bash — 91×22

```
ajnebro ~ $ cd jMetal
ajnebro ~/jMetal $ mvn package
[INFO] Scanning for projects...
[INFO] Inspecting build with total of 5 modules...
[INFO] Installing Nexus Staging features:
[INFO]   ... total of 5 executions of maven-deploy-plugin replaced with nexus-staging-maven
-plugin
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Build Order:
[INFO]
[INFO] org.uma.jmetal:jmetal
[INFO] org.uma.jmetal:jmetal-core
[INFO] org.uma.jmetal:jmetal-problem
[INFO] org.uma.jmetal:jmetal-algorithm
[INFO] org.uma.jmetal:jmetal-exec
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building org.uma.jmetal:jmetal 5.4-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- jacoco-maven-plugin:0.7.7.201606060606:prepare-agent (default) @ jmetal ---
[INFO] argLine set to -javaagent:/Users/ajnebro/.m2/repository/org/jacoco/org.jacoco.agent/
```

jMetal: a Maven Project Hosted in GitHub

- jMetal 5 is composed of four Maven sub-packages
- In the current version (jMetal 5.6):
  - Multi-Objective algorithms (25+)
  - Benchmark problems (80+)
  - Solution representations (6)
  - Quality indicators (10)

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>
```

Khaos INVESTIGACIÓN

jMetal features

- Multi-objective algoritms:
  - NSGA-II, SPEA2, PAES, PESA-II, OMOPSO, MOCell, AbYSS, GDE3, IBEA, SMPSO, SMPSOhv, SMPSO/RP, SMS-EMOA, MOEA/D-DE, MOEA/D-STM, MOEA/D-DD, MOCHC, MOMBI, MOMBI-II, NSGA-III, WASF-GA, GWASF-GA, R-NSGA-II , ESPEA, CDG-MOEA
- Single-objective algoritms:
  - genetic algorithm (variants: generational, steady-state), evolution strategy (variants: elitist or mu+lambda, non-elitist or mu, lambda), DE, CMA-ES, PSO (Stantard 2007, Standard 2011), coral reef optimization.
- Algorithms that can be executed in parallel:
  - NSGA-II, SMPSO, GDE3, SPEA2, PESA-II

Khaos INVESTIGACIÓN

jMetal features

- Included problems:
  - Problem families: ZDT, DTLZ, WFG, CEC2009, LZ09, GLT, MOP, CEC2018
  - Classical problems: Kursawe, Fonseca, Schaffer, Viennet2, Viennet3
  - Constrained problems: Srinivas, Tanaka, Osyczka2, Constr_Ex, Golinski, Water, Viennet4
  - Combinatorial problems: multi-objective TSP
  - Academic problems: OneMax, OneZeroMax
  - Real world: EBEs (structural design)
- Quality indicators: hypervolume, spread, generational distance, inverted generational distance, inverted generational distance plus, additive epsilon.
- Variable representations: binary, real, integer, permutation.

Example 1: configuring and running NSGA-II

workspake - Java - jmetal-exec/src/main/java/org/uma/jmetal/runner/multiobje

File Edit Source Refactor Navigate Search Project Run Window He

Package Explorer

- > jmetal [jMetal master ↑2]
- jmetal-algorithm [jMetal master ↑2]
- jmetal-core [jMetal master ↑2]
- > jmetal-exec [jMetal master ↑2]
  - > src/main/java
    - org.uma.jmetal.experiment
    - org.uma.jmetal.qualityIndicator
    - > org.uma.jmetal.runner
    - > org.uma.jmetal.runner.multiobjective
      - ABYSSRunner.java
      - CellDERunner.java
      - DMOPSOMeasuresRunner.java
      - DMOPSORunner.java
      - GDE3BigDataRunner.java
      - GDE3Runner.java
      - GWASGFARunner.java
      - IBEARunner.java
      - MOCellHVRunner.java
      - MOCellRunner.java
      - MOCHC45Runner.java
      - MOCHCRunner.java
      - MOEADRunner.java
      - MOEADSTMRunner.java
      - MOMBI2Runner.java
      - MOMBIRunner.java
      - NSGAII45Runner.java
      - NSGAIIBigDataRunner.java
      - NSGAIIBinaryRunner.java
      - NSGAIIEbesRunner.java
      - NSGAIIIntegerRunner.java
      - NSGAIIIRunner.java
      - NSGAIIMeasuresRunner.java
      - NSGAIIMeasuresWithChartsRunner.java
      - NSGAIIMeasuresWithHypervolumeRunner.java
      - NSGAIIRunner.java
      - NSGAIIStoppingByTimeRunner.java
      - NSGAIITSPRunner.java

```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) ;
    }
  }
}
```

Example 1: 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

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

Example 2: solving binary problems with NSGA-II

workspake - Java - jmetal-exec/src/main/java/org/uma

File Edit Source Refactor Navigate Search Proj

Package Explorer

- > jmetal [jMetal master ↑2]
- jmetal-algorithm [jMetal master ↑2]
- jmetal-core [jMetal master ↑2]
- > jmetal-exec [jMetal master ↑2]
  - > src/main/java
    - org.uma.jmetal.experiment
    - org.uma.jmetal.qualityIndicator
    - > org.uma.jmetal.runner
    - > org.uma.jmetal.runner.multiobjective
      - ABYSSRunner.java
      - CellDERunner.java
      - DMOPSOMeasuresRunner.java
      - DMOPSORunner.java
      - GDE3BigDataRunner.java
      - GDE3Runner.java
      - GWASGFARunner.java
      - IBEARunner.java
      - MOCellHVRunner.java
      - MOCellRunner.java
      - MOCHC45Runner.java
      - MOCHCRunner.java
      - MOEADRunner.java
      - MOEADSTMRunner.java
      - MOMBI2Runner.java
      - MOMBIRunner.java
      - NSGAII45Runner.java
      - NSGAIIBigDataRunner.java
      - NSGAIIBinaryRunner.java
      - NSGAIIEbesRunner.java
      - NSGAIIIntegerRunner.java
      - NSGAIIIRunner.java

```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) ;
    }
  }
```

Example 2: 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>
```

Example 3: solving the ZDT4 problem

Common settings

Pop. size: 100

Evaluations: 25000

Example 4: solving the LZ09-F2 problem

Common settings

Pop. size: 100

Evaluations: 150000

Example 4: indicating preferences (SMPSO/RP)

jMetal [~/Softw/jMetal/jMetal] - .../jmetal-exec/src/main/java/org/uma/jmetal/runner/multiobjective/SMPSORPWithMultipleReferencePointsAndChartsRunner.java [jmetal-exec]

java > org > uma > jmetal > runner > multiobjective > SMPSORPWithMultipleReferencePointsAndChart

Project

- MOMBIRunner
- NSGAII45Runner
- NSGAIIBigDataRunner
- NSGAIIBinaryRunner
- NSGAIIEbesRunner
- NSGAIIIntegerRunner
- NSGAIIIRunner
- NSGAIIMeasuresRunner
- NSGAIIMeasuresWithChartsRunner
- NSGAIIMeasuresWithHypervolumeRunner
- NSGAIIRunner
- NSGAIIStoppingByTimeRunner
- NSGAIITSPRunner
- OMOPSORunner
- PAESRunner
- ParallelGDE3Runner
- ParallelNSGAIIRunner
- ParallelPESA2Runner
- ParallelSMPSORunner
- ParallelSPEA2Runner
- PESA2Runner
- RandomSearchRunner
- RNSGAIIConstraintRunner
- RNSGAIIRunner
- SMPSOBigDataRunner
- SMPSOHv2Runner
- SMPSOHvRunner
- SMPSOMeasuresRunner
- SMPSOMeasuresWithChartsRunner
- SMPSORPChangingTheReferencePointsAndChartsRunner
- SMPSORPWithMultipleReferencePointsAndChartsRunner
- SMPSORPWithMultipleReferencePointsRunner
- SMPSORPWithOneReferencePointRunner
- SMPSORunner
- SMSEMOARunner
- SPEA2BinaryRunner
- SPEA2Runner

```java
1   package org.uma.jmetal.runner.multiobjective;
2
3   import ...
30
31  public class SMPSORPWithMultipleReferencePointsAndChartsRunner {
32    /**
33     * Program to run the SMPSORP algorithm with three reference points and plotting a graph duri
34     * execution. SMPSORP is described in "Extending the Speed-constrained Multi-Objective PSO (S
35     * Reference Point Based Preference Articulation". Accepted in PPSN 2018.
36     *
37     * @author Antonio J. Nebro
38     */
39    public static void main(String[] args) throws JMetalException, IOException {
40      DoubleProblem problem;
41      Algorithm<List<DoubleSolution>> algorithm;
42      MutationOperator<DoubleSolution> mutation;
43      String referenceParetoFront = "" ;
44
45      String problemName ;
46      if (args.length == 1) {
47        problemName = args[0];
48      } else if (args.length == 2) {
49        problemName = args[0] ;
50        referenceParetoFront = args[1] ;
51      } else {
52        problemName = "org.uma.jmetal.problem.multiobjective.zdt.ZDT1" ;
53        referenceParetoFront = "jmetal-problem/src/test/resources/pareto_fronts/ZDT1.pf" ;
54      }
55
56      problem = (DoubleProblem) ProblemUtils.<> loadProblem(problemName);
57
58      List<List<Double>> referencePoints;
59      referencePoints = new ArrayList<>();
60      referencePoints.add(Arrays.asList(0.0, 0.0)) ;
61      //referencePoints.add(Arrays.asList(0.3, 0.3)) ;
62      //referencePoints.add(Arrays.asList(0.8, 0.2)) ;
63
64      double mutationProbability = 1.0 / problem.getNumberOfVariables() ;
65      double mutationDistributionIndex = 20.0 ;
66      mutation = new PolynomialMutation(mutationProbability, mutationDistributionIndex) ;
67
68      int maxIterations = 250;
69      int swarmSize = 100 ;
```

A.J. Nebro, J.J. Durillo, J. García-Nieto, C. Barba-González, J. del Ser, C.A. Coello Coello, A. Benítez-Hidalgo, J.F. Aldana-Montes . Extending the Speed-Constrained Multi-objective PSO (SMPSO) with Reference Point Based Preference Articulation. PPSN 2018. LNCS Vol. 11101, pp: 298-310. September 2018.

Example 4: indicating preferences (SMPSO/RP)

Example 5: 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();
```

Example 5: 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();
```

Table of contents

- Who, why, what, when
- Background
- Starting with jMetal
- **jMetal components**
- Experimental studies
- Further developments and related projects

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

The *Algorithm* class

jMetal [~/Softw/jMetal/jMetal] - .../jmetal-core/src/main/java/org/uma/jmetal/algorithm/Algorithm.java [jmetal-core]

jMetal 〉 jmetal-core 〉 src 〉 main 〉 java 〉 org 〉 uma 〉 jmetal 〉 algorithm 〉 Algorithm 〉 SMPSORPWithMultipleReferencePointsAndChartsRunner

Project

- jmetal-core
  - .settings
  - src
    - main
      - java
        - org.uma.jmetal
          - algorithm
            - impl
              - AbstractCoralReefsOptimization
              - AbstractDifferentialEvolution
              - AbstractEvolutionaryAlgorithm
              - AbstractEvolutionStrategy
              - AbstractGeneticAlgorithm
              - AbstractParticleSwarmOptimization
              - AbstractScatterSearch
            - Algorithm
            - InteractiveAlgorithm
          - measure
          - operator
          - problem
          - qualityindicator
          - solution
          - util

Algorithm.java

```
1   package org.uma.jmetal.algorithm;
2
3   import ...
6
7   /**
8    * Interface representing an algorithm
9    * @author Antonio J. Nebro
10   * @version 0.1
11   * @param <Result> Result
12   */
13  public interface Algorithm<Result> extends Runnable, Serializable, DescribedEntity {
14    void run() ;
15    Result getResult() ;
16  }
17
```

Algorithm templates

Algorithm 1 Pseudo-code of an evolutionary algorithm

1: $P(0) \leftarrow$ GenerateInitialSolutions()
2: $t \leftarrow 0$
3: Evaluate($P(0)$)
4: **while not** StoppingCriterion() **do**
5: $Q(t) \leftarrow$ Variation($P(t)$)
6: Evaluate($Q(t)$)
7: $P(t+1) \leftarrow$ Update($P(t)$, $Q(t)$)
8: $t \leftarrow t+1$
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:  }
  }
 }
```

Hierarchy of *Algorithm* related classes

Algorithm templates: NSGA-II (Generational)

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

  protected int evaluations;
  protected Comparator<S> dominanceComparator ;

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

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

    this.evaluator = evaluator;
    this.dominanceComparator = dominanceComparator ;
  }

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

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

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

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

    return population;
  }

  @Override protected List<S> replacement(
      List<S> population, List<S> offspringPopulation) {
    List<S> jointPopulation = new ArrayList<>();
    jointPopulation.addAll(population);
    jointPopulation.addAll(offspringPopulation);

    RankingAndCrowdingSelection<S> rankingAndCrowdingSelection ;
    rankingAndCrowdingSelection = new RankingAndCrowdingSelection<S>() ;

    return rankingAndCrowdingSelection.execute(jointPopulation) ;
  }

  @Override public List<S> getResult() {
    return getNonDominatedSolutions(getPopulation());
  }
}
```

Algorithm templates: NSGA-II (Steady State)

```
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, Comparator<S> dominanceComparator, SolutionListEvaluator<S>
evaluator) {
    super(problem, maxEvaluations, populationSize, crossoverOperator, mutationOperator,
        selectionOperator, dominanceComparator, 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;
  }
```

The *Problem* class

jMetal [~/Softw/jMetal/jMetal] - .../jmetal-core/src/main/java/org/uma/jmetal/problem/Problem.java [jmetal-core]

jMetal 〉 jmetal-core 〉 src 〉 main 〉 java 〉 org 〉 uma 〉 jmetal 〉 problem 〉 Problem 〉 SMPSORPWithMultipleReferencePointsAndChartsRunner

Project

- jmetal-core
  - .settings
  - src
    - main
      - java
        - org.uma.jmetal
          - algorithm
          - measure
          - operator
          - problem
            - impl
              - AbstractBinaryProblem
              - AbstractDoubleProblem
              - AbstractGenericProblem
              - AbstractIntegerDoubleProblem
              - AbstractIntegerPermutationProblem
              - AbstractIntegerProblem
            - BinaryProblem
            - ConstrainedProblem
            - DoubleBinaryProblem
            - DoubleProblem
            - IntegerDoubleProblem
            - IntegerProblem
            - PermutationProblem
            - Problem
          - qualityindicator
          - solution
          - util
      - resources
    - test

Problem.java | AbstractEvolutionaryAlgorithm.java | AbstractGeneticAlgorithm.java

```
1  package org.uma.jmetal.problem;
2
3  import java.io.Serializable;
4
5  /**
6   * Interface representing a multi-objective optimization problem
7   *
8   * @author Antonio J. Nebro <antonio@lcc.uma.es>
9   *
10  * @param <S> Encoding
11  */
12 public interface Problem<S> extends Serializable {
13   /* Getters */
14   int getNumberOfVariables() ;
15   int getNumberOfObjectives() ;
16   int getNumberOfConstraints() ;
17   String getName() ;
18
19   /* Methods */
20   void evaluate(S solution) ;
21   S createSolution() ;
22 }
23
```

The *Problem* class

```java
public class Kursawe extends AbstractDoubleProblem {

  /**
   * Constructor.
   * Creates a default instance (3 variables) of the Kursawe problem
   */
  public Kursawe() {
    this(3);
  }

  /**
   * Constructor.
   * Creates a new instance of the Kursawe problem.
   *
   * @param numberOfVariables Number of variables of the problem
   */
  public Kursawe(Integer numberOfVariables) {
    setNumberOfVariables(numberOfVariables);
    setNumberOfObjectives(2);
    setName("Kursawe");

    List<Double> lowerLimit = new ArrayList<>(getNumberOfVariables()) ;
    List<Double> upperLimit = new ArrayList<>(getNumberOfVariables()) ;

    for (int i = 0; i < getNumberOfVariables(); i++) {
      lowerLimit.add(-5.0);
      upperLimit.add(5.0);
    }

    setLowerLimit(lowerLimit);
    setUpperLimit(upperLimit);
  }
```

```java
  /** Evaluate() method */
  public void evaluate(DoubleSolution solution){
    double aux, xi, xj;
    double[] fx = new double[getNumberOfObjectives()];
    double[] x = new double[getNumberOfVariables()];
    for (int i = 0; i < solution.getNumberOfVariables(); i++) {
      x[i] = solution.getVariableValue(i) ;
    }

    fx[0] = 0.0;
    for (int var = 0; var < solution.getNumberOfVariables() - 1; var++)
{
      xi = x[var] * x[var];
      xj = x[var + 1] * x[var + 1];
      aux = (-0.2) * Math.sqrt(xi + xj);
      fx[0] += (-10.0) * Math.exp(aux);
    }

    fx[1] = 0.0;

    for (int var = 0; var < solution.getNumberOfVariables(); var++) {
      fx[1] += Math.pow(Math.abs(x[var]), 0.8) +
        5.0 * Math.sin(Math.pow(x[var], 3.0));
    }

    solution.setObjective(0, fx[0]);
    solution.setObjective(1, fx[1]);
  }
}
```

The *Solution* class

jMetal [~/Softw/jMetal/jMetal] - .../jmetal-core/src/main/java/org/uma/jmetal/solution/Solution.java [jmetal-core]

jMetal 〉 jmetal-core 〉 src 〉 main 〉 java 〉 org 〉 uma 〉 jmetal 〉 solution 〉 Solution 〉 org.uma.jmetal.operator.impl in jmetal-core

Project

- jmetal-core
  - .settings
  - src
    - main
      - java
        - org.uma.jmetal
          - algorithm
          - measure
          - operator
          - problem
          - qualityindicator
          - solution
            - impl
              - AbstractGenericSolution
              - ArrayDoubleSolution
              - DefaultBinarySolution
              - DefaultDoubleBinarySolution
              - DefaultDoubleSolution
              - DefaultIntegerDoubleSolution
              - DefaultIntegerPermutationSolution
              - DefaultIntegerSolution
              - ObjectiveFactory
              - VariableFactory
            - util
            - BinarySolution
            - DoubleBinarySolution
            - DoubleSolution
            - IntegerDoubleSolution
            - IntegerSolution
            - PermutationSolution
            - Solution
            - SolutionBuilder
            - SolutionEvaluator
          - util

NSGAIIStoppingByTime.java | SBXCrossover.java | SBXCrossoverTest.java | Solution.java

```java
package org.uma.jmetal.solution;

import java.io.Serializable;

/**
 * Interface representing a Solution
 *
 * @author Antonio J. Nebro <antonio@lcc.uma.es>
 * @param <T> Type (Double, Integer, etc.)
 */
public interface Solution<T> extends Serializable {
  void setObjective(int index, double value) ;
  double getObjective(int index) ;
  double[] getObjectives() ;

  T getVariableValue(int index) ;
  void setVariableValue(int index, T value) ;
  String getVariableValueString(int index) ;

  int getNumberOfVariables() ;
  int getNumberOfObjectives() ;

  Solution<T> copy() ;

  void setAttribute(Object id, Object value) ;
  Object getAttribute(Object id) ;
}
```

The *Operator* class

jMetal [~/Softw/jMetal/jMetal] - .../jmetal-core/src/main/java/org/uma/jmetal/operator/Operator.java [jmetal-core]

jMetal 〉 jmetal-core 〉 src 〉 main 〉 java 〉 org 〉 uma 〉 jmetal 〉 operator 〉 Operator 〉 org.uma.jmetal.operator.impl in jmetal-core

Project

- jmetal-core
  - .settings
  - src
    - main
      - java
        - org.uma.jmetal
          - algorithm
          - measure
          - operator
            - impl
              - crossover
                - BLXAlphaCrossover
                - DifferentialEvolutionCrossover
                - HUXCrossover
                - IntegerSBXCrossover
                - NPointCrossover
                - NullCrossover
                - PMXCrossover
                - SBXCrossover
                - SinglePointCrossover
                - TwoPointCrossover
              - localsearch
              - mutation
                - BitFlipMutation
                - CDGMutation
                - IntegerPolynomialMutation
                - NonUniformMutation
                - NullMutation
                - PermutationSwapMutation
                - PolynomialMutation
                - SimpleRandomMutation
                - UniformMutation
              - selection
              - CrossoverOperator
              - LocalSearchOperator
              - MutationOperator
              - Operator
              - SelectionOperator

SBXCrossoverTest.java | Solution.java | Operator.java | BitFlipMutation.java | CountingMeasureTest.java

```
1  package org.uma.jmetal.operator;
2
3  import java.io.Serializable;
4
5  /** * Interface representing an operator
6   *
7   * @author Antonio J. Nebro <antonio@lcc.uma.es>
8   * @version 0.1
9
10  * @param <Source> Source Class of the object to be operated with
11  * @param <Result> Result Class of the result obtained after applying the operator
12  */
13 public interface Operator<Source, Result> extends Serializable {
14   /**
15    * @param source The data to process
16    */
17   Result execute(Source source) ;
18 }
19
```

Khaos INVESTIGACIÓN

Table of contents

- Who, why, what, when
- Background
- Starting with jMetal
- jMetal components
- **Experimental studies**
- Further developments and related projects

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

Khaos INVESTIGACIÓN

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

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);

    Experiment<DoubleSolution, List<DoubleSolution>> experiment =
            new ExperimentBuilder<DoubleSolution, List<DoubleSolution>>("NSGAIIStudy")
                    .setAlgorithmList(algorithmList)
                    .setProblemList(problemList)
                    .setExperimentBaseDirectory(experimentBaseDirectory)
                    .setOutputParetoFrontFileName("FUN")
                    .setOutputParetoSetFileName("VAR")
                    .setReferenceFrontDirectory("/pareto_fronts")
                    .setIndicatorList(Arrays.asList(
                            new Epsilon<DoubleSolution>(),
                            new Spread<DoubleSolution>(),
                            new PISAHypervolume<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
- jMetal components
- Experimental studies
- **Further developments and related projects**

Further developments

- jMetal is continuously evolving
  - New algorithms and features are continuously included
    - According to our research interests
      - Decision making
      - Dynamic problems
    - Contributions of users
  - 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.

Khaos INVESTIGACIÓN

Thanks for your attention ☺

- Comments and suggestions are welcome