jMetalPy Documentation

Release

Antonio J. Nebro

Contents:

1	Getting started	3
2	Contributing	9
3	About	19
4	User documentation	21
5	Installation steps	45
6	Features	47
Pv	thon Module Index	49

Warning: Documentation is WIP!! Some information may be missing.

Contents: 1

2 Contents:

CHAPTER 1

Getting started

1.1 NSGA-II

Common imports for these examples:

```
from jmetal.algorithm import NSGAII
from jmetal.operator import Polynomial, SBX, BinaryTournamentSelection
from jmetal.component import RankingAndCrowdingDistanceComparator
from jmetal.util import FrontPlot
```

1.1.1 NSGA-II with plotting

```
pareto_front.plot(front, reference_front=problem.reference_front)
pareto_front.to_html(filename='NSGAII-ZDT1')
```

1.1.2 NSGA-II stopping by time

```
class NSGA2b(NSGAII):
    def is_stopping_condition_reached(self):
        # Re-define the stopping condition
        return [False, True][self.get_current_computing_time() > 4]

problem = ZDT1()

algorithm = NSGA2b(
    problem=problem,
    population_size=100,
    max_evaluations=25000,
    mutation=Polynomial(probability=1.0/problem.number_of_variables, distribution_
-index=20),
    crossover=SBX(probability=1.0, distribution_index=20),

--selection=BinaryTournamentSelection(comparator=RankingAndCrowdingDistanceComparator()))

algorithm.run()
front = algorithm.get_result()
```

1.2 SMPSO

Common imports for these examples:

```
from jmetal.operator import Polynomial
from jmetal.util import FrontPlot
```

1.2.1 SMPSO with standard settings

```
from jmetal.algorithm import SMPSO
from jmetal.component import CrowdingDistanceArchive
from jmetal.problem import DTLZ1
problem = DTLZ1(number_of_objectives=5)
algorithm = SMPSO(
   problem=problem,
   swarm_size=100,
   max_evaluations=25000,
   mutation=Polynomial(probability=1.0/problem.number_of_variables, distribution_
\rightarrowindex=20),
    leaders=CrowdingDistanceArchive(100)
algorithm.run()
front = algorithm.get_result()
pareto_front = FrontPlot(plot_title='SMPSO-DTLZ1-5', axis_labels=problem.obj_labels)
pareto_front.plot(front, reference_front=problem.reference_front)
pareto_front.to_html(filename='SMPSO-DTLZ1-5')
```

1.2.2 SMPSO/RP with standard settings

```
from jmetal.algorithm import SMPSORP
from jmetal.component import CrowdingDistanceArchiveWithReferencePoint
from jmetal.problem import ZDT1

def points_to_solutions(points):
    solutions = []
    for i, _ in enumerate(points):
        point = problem.create_solution()
        point.objectives = points[i]
        solutions.append(point)

    return solutions

problem = ZDT1(rf_path='resources/reference_front/ZDT1.pf')
```

1.2. SMPSO 5

```
swarm\_size = 100
reference_points = [[0.8, 0.2], [0.4, 0.6]]
archives_with_reference_points = []
for point in reference_points:
    archives_with_reference_points.append(
        CrowdingDistanceArchiveWithReferencePoint(swarm_size, point)
algorithm = SMPSORP(
   problem=problem,
    swarm_size=swarm_size,
   max_evaluations=25000,
   mutation=Polynomial(probability=1.0/problem.number_of_variables, distribution_
\rightarrowindex=20).
    reference_points=reference_points,
    leaders=archives_with_reference_points
)
algorithm.run()
front = algorithm.get_result()
pareto_front = FrontPlot(plot_title='SMPSORP-ZDT1', axis_labels=problem.obj_labels)
pareto_front.plot(front, reference_front=problem.reference_front)
pareto_front.update(points_to_solutions(reference_points), legend='reference points')
pareto_front.to_html(filename='SMPSORP-ZDT1')
```

1.3 Observers

It is possible to attach any number of observers to a jMetalPy's algorithm to retrieve information from each iteration. For example, a basic algorithm observer will print the number of evaluations, the objectives from the best individual in the population and the computing time:

```
basic = BasicAlgorithmObserver(frequency=1.0)
algorithm.observable.register(observer=basic)
```

A full list of all available observer can be found at jmetal.component.observer module.

1.4 Experiments

This is an example of an experimental study based on solving two problems of the ZDT family with two versions of the same algorithm (NSGAII). The hypervolume indicator is used for performance assessment.

```
population_size=100,
          max_evaluations=25000,
         mutation=NullMutation(),
         crossover=SBX(probability=1.0, distribution_index=20),
→selection=BinaryTournamentSelection(comparator=RankingAndCrowdingDistanceComparator())
  algorithm_list.append(
      ('NSGAII_B',
      NSGAII(
         problem=problem,
         population_size=100,
         max_evaluations=25000,
         mutation=Polynomial(probability=1.0 / problem.number_of_variables,...
→distribution_index=20),
         crossover=SBX(probability=1.0, distribution_index=20),
→selection=BinaryTournamentSelection(comparator=RankingAndCrowdingDistanceComparator())
      ))
study = Experiment(algorithm_list, n_runs=2)
study.run()
# Compute quality indicators
metric_list = [HyperVolume(reference_point=[1, 1])]
print (study.compute_metrics(metric_list))
```

1.4. Experiments 7

Contributing

Contributions to the jMetalPy project are welcome. Please, take into account the following guidelines (all developers should follow these guidelines):

2.1 Git WorkFlow

We have a set of branches on the remote Git server. Some branches are temporary, and others are constant throughout the life of the repository.

- Branches always present in the repository:
 - master: You have the latest released to production, receive merges from the develop branch, or merge from a hot
 - * Do I have to put a TAG when doing a merge from develop to master? yes
 - * Do I have to put a TAG when doing a merge from a hotfix branch to master? yes
 - * After merge from a hotfix to master, do I have to merge from master to develop? yes
 - *develop*: It is considered the "Next Release", receives merges from branches of each developer, either corrections (*fix*) or new features (*feature*).
- Temporary branches:
 - feature/<task-id>-<description>: When we are doing a development, we create a local branch with the prefix "fea
 - * Where does this branch emerge? This branch always emerge from the develop branch
 - * When I finish the development in my feature branch, which branch to merge into?: You always merge feature branch into develop branch
 - fix/<task-id>-<description>: When we are making a correction, we create a local branch with the prefix "fix/", the
 - * Where does this branch emerge? This branch always emerge from the develop branch

- * When I finish the correction in my fix branch, which branch to merge into?: You always merge feature branch into develop branch
- hotfix/<task-id>-<description>: When we are correcting an emergency incidence in production, we create a local
 - * Where does this branch emerge?: This branch always emerge from the master branch
 - * When I finish the correction in my hotfix branch, which branch to merge into?: This branch always emerge from the master and develop branch
- Steps to follow when you are creating or going to work on a branch of any kind (feature / fix / hotfix):
 - 1. After you create your branch (feature / fix / hotfix) locally, upload it to the remote Git server. The integration system will verify your code from the outset.
 - 2. Each time you commit, as much as possible, you send a push to the server. Each push will trigger the automated launch of the tests, etc.
 - 3. Once the development is finished, having done a push to the remote Git server, and that the test phase has passed without problem, you create an pull request.

Note: Do not forget to remove your branch (feature / fix / hotfix) once the merge has been made.

Some useful Git commands:

• git fetch -prune: Cleaning branches removed and bringing new branches

2.2 PEP8!

It is really important to follow some standards when a team develops an application. If all team members format the code in the same format, then it is much easier to read the code. PEP8 is Python's style guide. It's a set of rules for how to format your Python code.

Some style rules:

- Package and module names: Modules should have short, **all-lowercase** names. Underscores can be used in the module name if it improves readability. Python packages should also have short, **all-lowercase** names, although the use of underscores is discouraged. In Python, a module is a file with the suffix '.py'.
- Class names: Class names should normally use the **CapWords** convention.
- Method names and instance variables: Lowercase with words separated by underscores as necessary to improve readability.

There are many more style standards in PEP8 so, please, refer to PEP8 documentation . The most appropriate is to use an IDE that has support for PEP8. For example, PyCharm.

2.3 Object-oriented programming

Object-oriented programming should be the single programming paradigm used. Avoiding as far as possible, imperative and functional programming.

```
# Object-oriented programming

class Imprint(object):

    def world(self) -> str:
        return "World"

    def hello(self) -> str:
        return "Hello"

    def hello_world(self) -> None:
        print(self.hello(), self.world())

imprint = Imprint()
imprint.hello_world()
```

```
# Functional programming

def world() -> str:
    return "World"

def hello() -> str:
    return "Hello"

def print_hello_world() -> None:
    print(hello(), world())
```

```
# Imperative programming
world = "World"
hello = "Hello"
hello_world = hello + ' ' + world
print(hello_world)
```

In classes, we directly access the attributes, which are usually defined as public.

```
class Circle(object):
    def __init__(self, radius: int):
        self.radius = radius
```

Only when we want to **implement additional logic in the accesses to the attributes** we define getter/setter methods, but **always by using the *property*** annotation or the ***property*** function:

```
class Circle(object):
    def __init__(self):
        self.__radius = None

@property
    def radius(self) -> int:
        print("Accessing the radius attribute by get")
        return self.__radius

@radius.setter
    def radius(self, radius: int) -> None:
        print("Accessing the radius attribute by set")
        # Logic to validate
        if radius < 0:
            raise ValueError("The radius value must be a positive integer")
        self.__radius = radius</pre>
```

```
class Circle(object):
    def __init__(self):
        self.__radius = None

def __get_radius(self) -> int:
        print("Accessing the radius attribute by get")
        return self.__radius

def __set_radius(self, radius: int) -> None:
        print("Accessing the radius attribute by set")
        # Logic to validate
        if radius < 0:
            raise ValueError("The radius value must be a positive integer")
        self.__radius = radius

radius = property(fget=__get_radius, fset=__set_radius)</pre>
```

By using *property*, we continue to access the attributes directly:

```
circle = Circle()
circle.radius = 3
print(circle.radius)
```

Do not use getter/setter methods without the *property* annotation or the *property* function:

```
class Circle(object):

    def __init__(self):
        self.__radius = None

    def get_radius(self) -> int:
        return self.__radius

    def set_radius(self, radius: int) -> None:
        # Logic to validate
        if radius < 0:
            raise ValueError("The radius value must be a positive integer")
        self.__radius = radius</pre>
```

Since this way of accessing the attribute is not commonly used in Python:

```
circle = Circle()
circle.set_radius(3)
print(circle.get_radius())
```

2.4 Structure

Python is not Java. In Java you cannot, by design, have more than one class in a file. In Python, you can do it.

In Python, it is appropriate to group several classes into a single .py file. For that reason, the .py files are called modules.

2.5 Python 3.6

We always define types in the parameters of the arguments and the return value:

2.4. Structure 13

```
class Car(object):

    def __init__(self):
        self.fuel = 0
        self.battery = 0

    def refuel(self, new_fuel: int):
        self.fuel += new_fuel

    def recharge(self, new_energy: int):
        self.battery += new_energy

    def status(self) -> Tuple[int, int]:
        return self.fuel, self.battery
```

We can define abstract classes (ABCs) in Python:

```
class AbstractClass(metaclass=ABCMeta):
    @abstractmethod
    def abstract_method(self) -> float:
        pass

class ImplementingClass(AbstractClass):
    def abstract_method(self) -> float:
        # implementation ...
```

In the case that we want to define an **interface** class, it is done in the same way. We just have to define all the methods of the class as abstract.

Example of use of generic types:

```
T = TypeVar('T') # <- Can be anything
S = TypeVar('S', int, float) # <- Must be int or float

class Car(object):

    def __init__(self, fuel: S, battery: S, model: T):
        self.fuel = fuel
        self.battery = battery
        self.model = model

    def refuel(self, new_fuel: S) -> None:
        self.fuel += new_fuel

    def recharge(self, new_energy: S) -> None:
        self.battery += new_energy

    def status(self) -> Tuple[S, S]:
        return self.fuel, self.battery
```

In the code below, the IDE displays a **warning**, since although the 2nd parameter is a float type, which is a type allowed in the definition of the generic type X, it is not of the same type as the first, since the first 2 parameters must be of the same generic type (S):

```
carl = Car(3, <u>3.44</u>, "FORD-F-150")
```

In the code below, the IDE displays a **warning**, since the 2nd parameter is a type not allowed in the definition of the generic type (*TypeVar('S', int, float)*):

```
car2 = Car(3, <u>"hello"</u>, "FORD-F-150")
```

When the class inherits from *Generic[...]*, the **class is defined as generic**. In this way we can indicate the types that will have the values of the generic types, when using the class as type. Look at the *add_car()* method of the *Parking* class.

Note: The generic classes inherit from abc.ABCMeta, so they are abstract classes and **abstract methods can be used**.

2.5. Python 3.6 15

```
T = TypeVar('T') # <- Can be anything
S = TypeVar('S', int, float) # <- Must be int or float

class CarGeneric(Generic[S, T]):

    def __init__(self, fuel: S, battery: S, model: T):
        self.fuel = fuel
        self.battery = battery
        self.model = model

def refuel(self, new_fuel: S) -> None:
        self.fuel += new_fuel

def recharge(self, new_energy: S) -> None:
        self.battery += new_energy

def status(self) -> Tuple[S, S]:
        return self.fuel, self.battery
```

```
class Parking(object):
    def __init__(self):
        self.car_list = list()

def add_car(self, new_car: CarGeneric[int, str]) -> None:
        self.car_list.append(new_car)
```

In the code below, the IDE displays a **warning** in the call to the *add_car()* method when adding the car, since the 3rd parameter of the init must be a *str* type, as defined in the *add_car()* method of the *Parking* class.

```
car3 = CarGeneric(3, 4, 777)
parking = Parking()
parking.add_car(car3)
```

When inheriting from generic classes, some type variables could be fixed:

```
T = TypeVar('T')

class MyClass(CarGeneric[str, T]):
    pass
```

Example of inheritance from non-generic class to generic class:

```
class A(object):
    pass

class B(A, Generic[S, T]):
    pass
```

Example of inheritance from generic class to another generic class:

```
class A(Generic[S, T]):
    pass

class B(A[S, T], Generic[S, T]):
    pass
```

2.6 Create automatic documentation files with Sphinx

First, you need to know how to correctly document your code. It is **important** to follow these simple rules in order to automatically create good documentation for the project.

When you create a new module file (testDoc.py in this example), you should mention it using this format:

```
"""
.. module:: testDoc
    :platform: Unix, Windows
    :synopsis: A useful module indeed.

.. moduleauthor:: Andrew Carter <andrew@invalid.com>
"""

class testDoc(object):
    """We use this as a public class example class.

This class is ruled by the very trendy important method :func:`public_fn_with_
    sphinxy_docstring`.
```

```
.. note::
    An example of intersphinx is this: you **cannot** use :mod:`pickle` on this
    →class.
    """

def __init__(self):
    pass
```

This code snippet generates the following documentation:

jmetal.algorithm.singleobjective.testDoc module

```
class jmetal.algorithm.singleobjective.testDoc.testDoc(foo: str, bar: str)
Bases: object
```

We use this as a public class example class.

This class is ruled by the very trendy important method public_fn_with_sphinxy_docstring().

Note: An example of intersphinx is this: you cannot use pickle on this class.

Now, you can document your methods using the following sintax:

```
def public_fn_with_sphinxy_docstring(self, name: str, state: bool = False) -> int:
    """This function does something.

    :param name: The name to use.
    :type name: str.
    :param state: Current state to be in.
    :type state: bool.
    :returns: int -- the return code.
    :raises: AttributeError, KeyError
    """
    return 0

def public_fn_without_docstring(self):
    return True
```

And the produced output doc will be:

```
public_fn_with_sphinxy_docstring(name: str, state: bool = False) \rightarrow int This function does something.
```

```
    name (str.) – The name to use.
    state (bool.) – Current state to be in.
    int – the return code.
    Raises: AttributeError, KeyError
```

public_fn_without_docstring()

As you may notice, if you don't use any docstring, the method documentation will be empty.

CHAPTER 3

About

jMetalPy is being developed by Antonio J. Nebro, associate professor at the University of Málaga, and Antonio Benítez-Hidalgo.

3.1 References

- 1. J.J. Durillo, A.J. Nebro jMetal: a Java Framework for Multi-Objective Optimization. Advances in Engineering Software 42 (2011) 760-771.
- 2. A.J. Nebro, J.J. Durillo, M. Vergne Redesigning the jMetal Multi-Objective Optimization Framework. GECCO (Companion) 2015, pp. 1093-1100. July 2015.
- 3. Nebro A.J. et al. (2018) Extending the Speed-Constrained Multi-objective PSO (SMPSO) with Reference Point Based Preference Articulation. In: Auger A., Fonseca C., Lourenço N., Machado P., Paquete L., Whitley D. (eds) Parallel Problem Solving from Nature PPSN XV. PPSN 2018. Lecture Notes in Computer Science, vol 11101. Springer, Cham

20 Chapter 3. About

CHAPTER 4

User documentation

4.1 Algorithms

4.1.1 Multiobjective algorithms

NSGA-II

Bases: jmetal.algorithm.singleobjective.evolutionaryalgorithm. GenerationalGeneticAlgorithm

NSGA-II implementation as described in

• K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, Apr 2002. doi: 10.1109/4235.996017

NSGA-II is a genetic algorithm (GA), i.e. it belongs to the evolutionary algorithms (EAs) family. The implementation of NSGA-II provided in jMetalPy follows the evolutionary algorithm template described in the algorithm module (jmetal.core.algorithm).

Parameters

- **problem** The problem to solve.
- population_size Size of the population.
- max evaluations Maximum number of evaluations/iterations.
- mutation Mutation operator (see jmetal.operator.mutation).
- **crossover** Crossover operator (see jmetal.operator.crossover).
- **selection Selection** operator (see jmetal.operator.selection).
- evaluator An evaluator object to evaluate the individuals of the population.

```
get_name() → str
get_result()
replacement(population: typing.List[S], offspring_population: typing.List[S]) → typ-
ing.List[typing.List[S]]
```

This method joins the current and offspring populations to produce the population of the next generation by applying the ranking and crowding distance selection.

Parameters

- **population** Parent population.
- offspring_population Offspring population.

Returns New population after ranking and crowding distance selection is applied.

```
jmetal.algorithm.multiobjective.nsgaii.R
```

SMPSO

This class implements the SMPSO algorithm as described in

- SMPSO: A new PSO-based metaheuristic for multi-objective optimization
- MCDM 2009. DOI: http://dx.doi.org/10.1109/MCDM.2009.4938830/.

The implementation of SMPSO provided in jMetalPy follows the algorithm template described in the algorithm templates section of the documentation.

Parameters

```
• swarm size - Swarm size.
                 • max_evaluations - Maximum number of evaluations.
                 • mutation – Mutation operator.
                 • leaders – Archive for leaders.
                 • evaluator – An evaluator object to evaluate the solutions in the population.
     \textbf{create\_initial\_swarm} () \rightarrow typing.List[jmetal.core.solution.FloatSolution]
                                            typing.List[jmetal.core.solution.FloatSolution])
     evaluate_swarm(swarm:
                                                                                                        typ-
                           ing.List[imetal.core.solution.FloatSolution]
     get_result () → typing.List[jmetal.core.solution.FloatSolution]
     init\_progress() \rightarrow None
     \verb|initialize_global_best| (\textit{swarm: typing.List[jmetal.core.solution.FloatSolution]})| \rightarrow None
     initialize\_particle\_best (swarm: typing.List[jmetal.core.solution.FloatSolution]) \rightarrow None
     initialize\_velocity (swarm: typing.List[imetal.core.solution.FloatSolution]) \rightarrow None
     is_stopping_condition_reached() \rightarrow bool
     perturbation (swarm: typing.List[jmetal.core.solution.FloatSolution]) \rightarrow None
     select\_global\_best() \rightarrow jmetal.core.solution.FloatSolution
     update\_global\_best (swarm: typing.List[jmetal.core.solution.FloatSolution]) \rightarrow None
     update\_particle\_best (swarm: typing.List[jmetal.core.solution.FloatSolution]) \rightarrow None
     update\_position (swarm: typing.List[jmetal.core.solution.FloatSolution]) \rightarrow None
     update\_progress() \rightarrow None
     update\_velocity (swarm: typing.List[imetal.core.solution.FloatSolution]) \rightarrow None
class jmetal.algorithm.multiobjective.smpso.SMPSORP (problem:
                                                                           jmetal.core.problem.FloatProblem,
                                                                           swarm_size:
                                                                                                        int,
                                                                           max_evaluations: int, mutation:
                                                                           jmetal.core.operator.Mutation[jmetal.core.solution.Float]
                                                                           reference_points:
                                                                                                       typ-
                                                                           ing.List[typing.List[float]],
                                                                           leaders:
                                                                                                       typ-
                                                                           ing.List[jmetal.component.archive.BoundedArchive[jmetal
                                                                           evaluator:
                                                                           jmetal.component.evaluator.Evaluator[jmetal.core.solution]
                                                                           <jmetal.component.evaluator.SequentialEvaluator</pre>
                                                                           object>)
     Bases: jmetal.algorithm.multiobjective.smpso.SMPSO
     This class implements the SMPSORP algorithm.
           Parameters
                 • problem – The problem to solve.
                  • swarm size-
```

• **problem** – The problem to solve.

4.1. Algorithms 23

```
• max evaluations -
           • mutation -
           • leaders – List of bounded archives.
           • evaluator – An evaluator object to evaluate the solutions in the population.
get_result () → typing.List[jmetal.core.solution.FloatSolution]
init\_progress() \rightarrow None
initialize\_global\_best (swarm: typing.List[jmetal.core.solution.FloatSolution]) \rightarrow None
select_global_best() \rightarrow jmetal.core.solution.FloatSolution
update\_global\_best (swarm: typing.List[jmetal.core.solution.FloatSolution]) \rightarrow None
update_leaders_density_estimator()
update\_progress() \rightarrow None
```

Random Search

```
class jmetal.algorithm.multiobjective.randomSearch.RandomSearch(problem:
                                                                                        jmetal.core.problem.Problem[S],
                                                                                        max evaluations:
                                                                                        int = 25000)
     Bases: typing. Generic
     static get_name() \rightarrow str
     \texttt{get\_result}() \rightarrow typing.List[S]
     run() \rightarrow None
```

4.1.2 Singleobjectives algorithms

jmetal.algorithm.multiobjective.randomSearch.S = ~S

Evolutionary Algorithm

```
class jmetal.algorithm.singleobjective.evolutionaryalgorithm.ElitistEvolutionStrategy (proble
```

```
Bases: jmetal.core.algorithm.EvolutionaryAlgorithm
create\_initial\_population() \rightarrow typing.List[S]
evaluate\_population (population: typing.List[S]) \rightarrow typing.List[S]
```

jmetal mu: int, lambd int. max_e int, тиtation: jmetal

```
init_progress()
      \verb|is_stopping_condition_reached|()| \rightarrow bool
      replacement (population: typing.List[S], offspring_population: typing.List[S]) \rightarrow typing.List[S]
      reproduction (population: typing.List[S]) \rightarrow typing.List[S]
      selection (population: typing.List[S]) \rightarrow typing.List[S]
      update_progress()
class jmetal.algorithm.singleobjective.evolutionaryalgorithm.GenerationalGeneticAlgorithm()
      Bases: jmetal.core.algorithm.EvolutionaryAlgorithm
      create\_initial\_population() \rightarrow typing.List[S]
      evaluate_population (population: typing.List[S])
      \texttt{get}\_\texttt{name}\left(\right) \to \mathsf{str}
      \texttt{get\_result}\,(\,)\,\to R
               Returns The best individual of the population.
      init_progress()
      \verb|is_stopping_condition_reached|()| \rightarrow bool
      replacement (population: typing.List[S], offspring_population: typing.List[S]) \rightarrow typing.List[S]
      reproduction (population: typing.List[S]) \rightarrow typing.List[S]
```

$$\begin{split} &\texttt{get_name} \; () \; \rightarrow str \\ &\texttt{get_result} \; () \; \rightarrow R \end{split}$$

4.1. Algorithms 25

selection (population: typing.List[S])

```
update_progress()
class jmetal.algorithm.singleobjective.evolutionaryalgorithm.NonElitistEvolutionStrategy (prince)
     Bases:
                                imetal.algorithm.singleobjective.evolutionaryalgorithm.
     ElitistEvolutionStrategy
     \texttt{get}\_\texttt{name} \,(\,) \, \to str
     \textbf{replacement} \ (\textit{population: typing.List[S]}, \textit{offspring\_population: typing.List[S]}) \ \rightarrow \textbf{typing.List[S]}
jmetal.algorithm.singleobjective.evolutionaryalgorithm.R = ~R
4.2 Components
4.2.1 Archive
class jmetal.component.archive.Archive
     Bases: typing. Generic
     add (solution: S) \rightarrow bool
     get(index: int) \rightarrow S
     \mathtt{get}\_\mathtt{name} \,(\,) \, \to \mathrm{str}
     size() \rightarrow int
class jmetal.component.archive.ArchiveWithReferencePoint(maximum_size:
                                                                                                 int,
                                                                             reference_point:
                                                                             typing.List[float],
                                                                             comparator:
                                                                             jmetal.component.comparator.Comparator[S],
                                                                             density_estimator:
                                                                             jmetal.component.density_estimator.DensityEstim
     Bases: jmetal.component.archive.BoundedArchive
     add (solution: S) \rightarrow bool
     get_reference_point() → typing.List[float]
class jmetal.component.archive.BoundedArchive (maximum_size:
                                                                                  int,
                                                                                         comparator:
                                                              jmetal.component.comparator.Comparator[S]
                                                                      None,
                                                                                   density_estimator:
                                                              jmetal.component.density_estimator.DensityEstimator
                                                              = None)
     Bases: jmetal.component.archive.Archive
```

in la in

in m ta tie jn

```
add (solution: S) \rightarrow bool
     compute_density_estimator()
class jmetal.component.archive.CrowdingDistanceArchive (maximum_size: int)
     Bases: jmetal.component.archive.BoundedArchive
class jmetal.component.archive.CrowdingDistanceArchiveWithReferencePoint (maximum size:
                                                                                          ref-
                                                                                          er-
                                                                                          ence_point:
                                                                                          typ-
                                                                                          ing.List[float])
     Bases: jmetal.component.archive.ArchiveWithReferencePoint
class jmetal.component.archive.NonDominatedSolutionListArchive
     Bases: jmetal.component.archive.Archive
     add (solution: S) \rightarrow bool
4.2.2 Comparator
class jmetal.component.comparator.Comparator
     Bases: typing. Generic
     compare (solution1: S, solution2: S) \rightarrow int
     \texttt{get}\_\texttt{name}\left(\right) \to \mathsf{str}
class jmetal.component.comparator.DominanceComparator(constraint_comparator=<jmetal.component.comparator
                                                                  object>)
     Bases: jmetal.component.comparator.Comparator
     compare (solution1: jmetal.core.solution.Solution, solution2: jmetal.core.solution.Solution) \rightarrow int
class jmetal.component.comparator.EqualSolutionsComparator
     Bases: jmetal.component.comparator.Comparator
     compare (solution1: jmetal.core.solution.Solution, solution2: jmetal.core.solution.Solution) \rightarrow int
class jmetal.component.comparator.RankingAndCrowdingDistanceComparator
     Bases: jmetal.component.comparator.Comparator
     compare (solution1: jmetal.core.solution.Solution, solution2: jmetal.core.solution.Solution) \rightarrow int
class jmetal.component.comparator.SolutionAttributeComparator(key:
                                                                            est_is_best: bool
                                                                             = True)
     Bases: jmetal.component.comparator.Comparator
     compare (solution1: jmetal.core.solution.Solution, solution2: jmetal.core.solution.Solution) \rightarrow int
4.2.3 Density Estimator
class jmetal.component.density_estimator.CrowdingDistance
     Bases: jmetal.component.density_estimator.DensityEstimator
```

4.2. Components 27

this class is compute density estimator().

This class implements a DensityEstimator based on the crowding distance. In consequence, the main method of

```
compute_density_estimator (front: typing.List[S])
          This function performs the computation of the crowding density estimation over the solution list.
          Note: This method assign the distance in the inner elements of the solution list.
              Parameters front – The list of solutions.
class jmetal.component.density_estimator.DensityEstimator
     Bases: typing.List
     This is the interface of any density estimator algorithm.
     compute\_density\_estimator (solution_list: typing.List[S]) \rightarrow float
jmetal.component.density_estimator.S = ~S
4.2.4 Evaluator
class jmetal.component.evaluator.Evaluator
     Bases: typing. Generic
     evaluate (solution\_list: typing.List[S], problem: jmetal.core.problem.Problem) \rightarrow typing.List[S]
     static evaluate solution (solution: S, problem: jmetal.core.problem.Problem) \rightarrow None
     get_name() \rightarrow str
class jmetal.component.evaluator.MapEvaluator(processes=None)
     Bases: jmetal.component.evaluator.Evaluator
     evaluate (solution\_list: typing.List[S], problem: jmetal.core.problem.Problem) <math>\rightarrow typing.List[S]
class jmetal.component.evaluator.SequentialEvaluator
     Bases: jmetal.component.evaluator.Evaluator
     evaluate (solution\_list: typing.List[S], problem: jmetal.core.problem.Problem) \rightarrow typing.List[S]
                                                                        None
     Bases: jmetal.core.observable.Observer
```

4.2.5 Observer

```
class jmetal.component.observer.BasicAlgorithmObserver(frequency: float = 1.0) \rightarrow
     Show the number of evaluations, best fitness and computing time.
          Parameters frequency – Display frequency.
     update (*args, **kwargs)
class jmetal.component.observer.ProgressBarObserver(step: int, maximum: int, desc: str
                                                                   = 'Progress') \rightarrow None
     Bases: jmetal.core.observable.Observer
     Show a smart progress meter with the number of evaluations and computing time.
          Parameters
```

• step – Initial counter value.

```
• maximum – Number of expected iterations.
```

• desc – Prefix for the progressbar.

Bases: jmetal.core.observable.Observer

Write function values of the front into files.

Parameters output_directory - Output directory. Each front will be saved on a file *FUN.x*.

```
update (*args, **kwargs)
```

```
jmetal.component.observer.jMetalPyLogger = <Logger jMetalPy (DEBUG)>
```

4.2.6 Quality indicator

```
class jmetal.component.quality_indicator.HyperVolume(reference_point: list)
    Bases: jmetal.component.quality_indicator.Metric
```

Hypervolume computation based on variant 3 of the algorithm in the paper:

C. M. Fonseca, L. Paquete, and M. Lopez-Ibanez. An improved dimension-sweep algorithm for the hyper-volume indicator. In IEEE Congress on Evolutionary Computation, pages 1157-1163, Vancouver, Canada, July 2006.

Minimization is implicitly assumed here!

Constructor.

```
compute (front: typing.List[jmetal.core.solution.Solution])
```

Before the HV computation, front and reference point are translated, so that the reference point is $[0, \ldots, 0]$.

Returns The hypervolume that is dominated by a non-dominated front.

```
get_name() → str

class jmetal.component.quality_indicator.Metric
    Bases: object
    compute(front: typing.List[jmetal.core.solution.Solution])
    get_name() → str

class jmetal.component.quality_indicator.MultiList(number_lists)
    Bases: object
```

A special front structure needed by FonsecaHyperVolume.

It consists of several doubly linked lists that share common nodes. So, every node has multiple predecessors and successors, one in every list.

Builds 'numberLists' doubly linked lists.

4.2. Components 29

```
class Node (number_lists, cargo=None)
          Bases: object
     append (node, index)
          Appends a node to the end of the list at the given index.
     extend(nodes, index)
          Extends the list at the given index with the nodes.
     get_length(i)
          Returns the length of the i-th list.
     reinsert (node, index, bounds)
          Inserts 'node' at the position it had in all lists in [0, 'index'] before it was removed. This method assumes
          that the next and previous nodes of the node that is reinserted are in the list.
     remove (node, index, bounds)
          Removes and returns 'node' from all lists in [0, 'index'].
4.2.7 Ranking
class jmetal.component.ranking.EfficientNonDominatedRanking
     Bases: jmetal.component.ranking.Ranking
     Class implementing the EDS (efficient non-dominated sorting) algorithm.
     compute_ranking (solution_list: typing.List[S])
class jmetal.component.ranking.FastNonDominatedRanking
     Bases: jmetal.component.ranking.Ranking
     Class implementing the non-dominated ranking of NSGA-II.
     compute_ranking (solution_list: typing.List[S])
class jmetal.component.ranking.Ranking
     Bases: typing.List
     compute_ranking (solution_list: typing.List[S])
     get_number_of_subfronts()
     get_subfront (rank: int)
4.3 Core
```

This subpackage store templates used in jMetalPy.

4.3.1 Algorithm

```
class jmetal.core.algorithm.Algorithm
      Bases: typing. Generic, threading. Thread
      get_current_computing_time() → float
      \mathtt{get\_evaluations}() \rightarrow \mathtt{int}
      \mathtt{get}\_\mathtt{name}\left(\right) \to \mathrm{str}
      \texttt{get\_result}\,(\,)\,\to R
```

Returns Final population.

```
class jmetal.core.algorithm.EvolutionaryAlgorithm
    Bases: jmetal.core.algorithm.Algorithm
    create_initial_population() → typing.List[S]
    evaluate_population(population: typing.List[S]) → typing.List[S]
    init_progress() → None
    is_stopping_condition_reached() → bool
    replacement(population: typing.List[S], offspring_population: typing.List[S]) → typing.List[S]
    reproduction(population: typing.List[S]) → typing.List[S]
    run()
```

- Step One: Generate the initial population of individuals randomly. (First generation)
- Step Two: Evaluate the fitness of each individual in that population
- Step Three: Repeat the following regenerational steps until termination
 - 1. Select the best-fit individuals for reproduction. (Parents)
 - 2. Breed new individuals through crossover and mutation operations to give birth to offspring.
 - 3. Evaluate the individual fitness of new individuals.
 - 4. Replace least-fit population with new individuals.

Note: To develop an EA, all the abstract the methods used in the run() method must be implemented.

```
selection (population: typing.List[S]) \rightarrow typing.List[S]
     update_progress()
class jmetal.core.algorithm.ParticleSwarmOptimization
     Bases: jmetal.core.algorithm.Algorithm
     create_initial_swarm() → typing.List[jmetal.core.solution.FloatSolution]
     evaluate swarm(swarm:
                                           typing.List[jmetal.core.solution.FloatSolution])
                                                                                                      typ-
                           ing.List[imetal.core.solution.FloatSolution]
     init\_progress() \rightarrow None
     initialize\_global\_best (swarm: typing.List[jmetal.core.solution.FloatSolution]) \rightarrow None
     initialize\_particle\_best (swarm: typing.List[jmetal.core.solution.FloatSolution]) \rightarrow None
     initialize\_velocity (swarm: typing.List[jmetal.core.solution.FloatSolution]) \rightarrow None
     \verb|is_stopping_condition_reached|()| \rightarrow bool
     perturbation (swarm: typing.List[jmetal.core.solution.FloatSolution]) \rightarrow None
     run()
     update\_global\_best (swarm: typing.List[imetal.core.solution.FloatSolution]) \rightarrow None
     update\_particle\_best (swarm: typing.List[imetal.core.solution.FloatSolution]) \rightarrow None
     update\_position (swarm: typing.List[jmetal.core.solution.FloatSolution]) \rightarrow None
     update\_progress() \rightarrow None
```

4.3. Core 31

```
update\_velocity (swarm: typing.List[imetal.core.solution.FloatSolution]) \rightarrow None
jmetal.core.algorithm.R = ~R
4.3.2 Operator
class jmetal.core.operator.Crossover(probability: float)
     Bases: jmetal.core.operator.Operator
     Class representing crossover operator.
     execute (source: S) \rightarrow R
     \mathtt{get}\_\mathtt{name}\left(\right) \to \mathrm{str}
     get_number_of_parents()
class jmetal.core.operator.Mutation(probability: float)
     Bases: jmetal.core.operator.Operator
     Class representing mutation operator.
     execute (source: S) \rightarrow R
     \texttt{get}\_\texttt{name} \,(\,) \, \to str
class jmetal.core.operator.Operator
     Bases: typing. Generic
     Class representing operator
     execute (source: S) \rightarrow R
     \texttt{get}\_\texttt{name} \, (\,) \, \to str
jmetal.core.operator.R = ~R
class jmetal.core.operator.Selection
     Bases: jmetal.core.operator.Operator
     Class representing selection operator.
     execute (source: S) \rightarrow R
     \mathtt{get}\_\mathtt{name}\left(\right) \to \mathrm{str}
4.3.3 Problem
class jmetal.core.problem.BinaryProblem(rf_path: str = None)
     Bases: jmetal.core.problem.Problem
     Class representing binary problems.
     create\_solution() \rightarrow jmetal.core.solution.BinarySolution
     evaluate (solution: imetal.core.solution.BinarySolution) \rightarrow imetal.core.solution.BinarySolution
class jmetal.core.problem.FloatProblem(rf_path: str = None)
     Bases: jmetal.core.problem.Problem
```

Class representing float problems.

```
create\_solution() \rightarrow jmetal.core.solution.FloatSolution
     evaluate (solution: jmetal.core.solution.FloatSolution) <math>\rightarrow jmetal.core.solution.FloatSolution
class jmetal.core.problem.IntegerProblem(rf_path: str = None)
     Bases: jmetal.core.problem.Problem
     Class representing integer problems.
     create\_solution() \rightarrow jmetal.core.solution.IntegerSolution
     evaluate (solution: jmetal.core.solution.IntegerSolution) \rightarrow jmetal.core.solution.IntegerSolution
class jmetal.core.problem.Problem(reference_front_path: str)
     Bases: typing. Generic
     Class representing problems.
     MAXIMIZE = 1
     MINIMIZE = -1
     create solution() \rightarrow S
           Creates a random solution to the problem.
               Returns Solution.
     evaluate (solution: S) \rightarrow S
           Evaluate a solution. For any new problem inheriting from Problem, this method should be replaced.
               Returns Evaluated solution.
     evaluate constraints (solution: S)
     \mathtt{get}\_\mathtt{name}\left(\right) \to \mathrm{str}
     static read_front_from_file (file_path: str) → typing.List[typing.List[float]]
           Reads a front from a file and returns a list.
               Returns List of solution points.
     static read_front_from_file_as_solutions (file\_path: str) \rightarrow typing.List[S]
           Reads a front from a file and returns a list of solution objects.
               Returns List of solution objects.
4.3.4 Solution
class jmetal.core.solution.BinarySolution(number_of_variables:
                                                                                                   num-
                                                                                          int.
                                                           ber_of_objectives: int, number_of_constraints:
                                                           int = 0)
     Bases: jmetal.core.solution.Solution
     Class representing float solutions
     get total number of bits() \rightarrow int
class jmetal.core.solution.FloatSolution(number of variables: int, number of objectives:
                                                         int, number_of_constraints: int, lower_bound:
                                                         typing.List[float],
                                                                               upper bound:
                                                                                                    typ-
                                                         ing.List[float])
     Bases: jmetal.core.solution.Solution
     Class representing float solutions
```

4.3. Core 33

```
class jmetal.core.solution.IntegerSolution(number_of_variables:
                                                                              int,
                                                                                      num-
                                                    ber_of_objectives:
                                                                             int.
                                                                                      num-
                                                    ber of constraints: int, lower bound: typ-
                                                    ing.List[int], upper_bound: typing.List[int])
     Bases: jmetal.core.solution.Solution
     Class representing integer solutions
class jmetal.core.solution.Solution(number_of_variables: int, number_of_objectives: int,
                                           number\_of\_constraints: int = 0)
     Bases: typing.Generic
     Class representing solutions
4.3.5 Observable
class jmetal.core.observable.DefaultObservable
     Bases: jmetal.core.observable.Observable
     deregister (observer: jmetal.core.observable.Observer)
     deregister_all()
     notify_all(*args, **kwargs)
     register (observer: jmetal.core.observable.Observer)
class jmetal.core.observable.Observable
     Bases: object
     deregister (observer)
     deregister_all()
     notify_all(*args, **kwargs)
     register (observer)
class jmetal.core.observable.Observer
     Bases: object
     update (*args, **kwargs)
         Update method
             Parameters
                 • args -
                 • kwargs -
             Returns
4.4 Operators
4.4.1 Crossover
class jmetal.operator.crossover.NullCrossover
     Bases: jmetal.core.operator.Crossover
     execute (parents:
                                    typing.List[jmetal.core.solution.Solution])
                                                                                       typ-
              ing.List[jmetal.core.solution.Solution]
```

```
get_name()
     get_number_of_parents()
class jmetal.operator.crossover.SBX (probability: float, distribution_index: float = 20.0)
     Bases: jmetal.core.operator.Crossover
     execute (parents:
                                   typing.List[jmetal.core.solution.FloatSolution])
                                                                                             typ-
               ing.List[jmetal.core.solution.FloatSolution]
     get_name()
     get_number_of_parents()
class jmetal.operator.crossover.SP (probability: float)
     Bases: jmetal.core.operator.Crossover
                                   typing.List[jmetal.core.solution.BinarySolution])
     execute (parents:
                                                                                             typ-
               ing.List[jmetal.core.solution.BinarySolution]
     get name()
     get_number_of_parents()
4.4.2 Mutation
class jmetal.operator.mutation.BitFlip(probability: float)
     Bases: jmetal.core.operator.Mutation
     execute (solution: jmetal.core.solution.BinarySolution) <math>\rightarrow jmetal.core.solution.BinarySolution
     get_name()
class jmetal.operator.mutation.IntegerPolynomial(probability: float, distribution_index:
                                                               float = 0.2)
     Bases: jmetal.core.operator.Mutation
     execute (solution: jmetal.core.solution.IntegerSolution) <math>\rightarrow jmetal.core.solution.IntegerSolution
     get_name()
class jmetal.operator.mutation.NullMutation
     Bases: jmetal.core.operator.Mutation
     execute (solution: imetal.core.solution.Solution) <math>\rightarrow imetal.core.solution.Solution
     get name()
class jmetal.operator.mutation.Polynomial (probability: float, distribution_index: float =
     Bases: jmetal.core.operator.Mutation
     execute (solution: jmetal.core.solution.FloatSolution) <math>\rightarrow jmetal.core.solution.FloatSolution
     get_name()
class jmetal.operator.mutation.SimpleRandom(probability: float)
     Bases: jmetal.core.operator.Mutation
     execute (solution: jmetal.core.solution.FloatSolution) <math>\rightarrow jmetal.core.solution.FloatSolution
     get_name()
class jmetal.operator.mutation.Uniform (probability: float, perturbation: float = 0.5)
     Bases: jmetal.core.operator.Mutation
```

4.4. Operators 35

```
\textbf{execute} (\textit{solution: jmetal.core.solution.FloatSolution}) \rightarrow \texttt{jmetal.core.solution.FloatSolution} \textbf{get\_name} ()
```

4.4.3 Selection

```
class jmetal.operator.selection.BestSolutionSelection
     Bases: jmetal.core.operator.Selection
     execute (front: typing.List[S]) \rightarrow S
     \texttt{get}\_\texttt{name}\left(\right) \to \mathsf{str}
class jmetal.operator.selection.BinaryTournament2Selection(comparator_list: typ-
                                                                                  ing.List[jmetal.component.comparator.Compar
     Bases: jmetal.core.operator.Selection
     execute (front: typing.List[S]) \rightarrow S
     \texttt{get}\_\texttt{name} \,(\,) \, \to str
class jmetal.operator.selection.BinaryTournamentSelection(comparator:
                                                                                jmetal.component.comparator.Comparator
                                                                                 <jmetal.component.comparator.DominanceCom</pre>
                                                                                 object>)
     Bases: imetal.core.operator.Selection
     execute (front: typing.List[S]) \rightarrow S
     \texttt{get}\_\texttt{name} \,(\,) \, \to str
class jmetal.operator.selection.NaryRandomSolutionSelection(number_of_solutions_to_be_returned:
                                                                                    int = 1)
     Bases: jmetal.core.operator.Selection
     execute (front: typing.List[S]) \rightarrow S
     \texttt{get}\_\texttt{name} \,(\,) \, \to str
class jmetal.operator.selection.RandomSolutionSelection
     Bases: jmetal.core.operator.Selection
     execute (front: typing.List[S]) \rightarrow S
     \texttt{get}\_\texttt{name} \,(\,) \, \to str
class jmetal.operator.selection.RankingAndCrowdingDistanceSelection(max_population_size:
     Bases: jmetal.core.operator.Selection
     execute (front: typing.List[S]) \rightarrow typing.List[S]
     \texttt{get}\_\texttt{name} \,(\,) \, \to str
jmetal.operator.selection.S = ~S
```

4.5 Problems

4.5.1 Multiobjective problems

Constrained

```
class jmetal.problem.multiobjective.constrained.Srinivas(rf_path: str = None)
     Bases: jmetal.core.problem.FloatProblem
     Class representing problem Srinivas.
     evaluate (solution: jmetal.core.solution.FloatSolution) <math>\rightarrow jmetal.core.solution.FloatSolution
     evaluate\_constraints (solution: jmetal.core.solution.FloatSolution) \rightarrow None
     get_name()
class jmetal.problem.multiobjective.constrained.Tanaka (rf_path: str = None)
     Bases: imetal.core.problem.FloatProblem
     Class representing problem Tanaka
     evaluate (solution: jmetal.core.solution.FloatSolution) \rightarrow jmetal.core.solution.FloatSolution
     evaluate_constraints (solution: jmetal.core.solution.FloatSolution) \rightarrow None
     get_name()
Unconstrained
class jmetal.problem.multiobjective.unconstrained.Fonseca (rf_path: str = None)
     Bases: jmetal.core.problem.FloatProblem
     evaluate (solution: jmetal.core.solution.FloatSolution) <math>\rightarrow jmetal.core.solution.FloatSolution
     get_name()
class jmetal.problem.multiobjective.unconstrained.Kursawe (number_of_variables:
                                                                              int = 3, rf path: <math>str =
                                                                              None)
     Bases: jmetal.core.problem.FloatProblem
     Class representing problem Kursawe.
     \textbf{evaluate} \ (\textit{solution: jmetal.core.solution.FloatSolution}) \ \rightarrow \ \textit{jmetal.core.solution.FloatSolution}
     get_name()
class jmetal.problem.multiobjective.unconstrained.Schaffer (rf_path: str = None)
     Bases: jmetal.core.problem.FloatProblem
     evaluate (solution: jmetal.core.solution.FloatSolution) \rightarrow jmetal.core.solution.FloatSolution
     get name()
class jmetal.problem.multiobjective.unconstrained.Viennet2(rf_path: str = None)
     Bases: jmetal.core.problem.FloatProblem
     \textbf{evaluate} \ (\textit{solution: jmetal.core.solution.FloatSolution}) \ \rightarrow \ \textit{jmetal.core.solution.FloatSolution}
     get_name()
```

4.5. Problems 37

DTLZ

```
class jmetal.problem.multiobjective.dtlz.DTLZ1 (number\_of\_variables: int = 7, number\_of\_objectives=3, rf\_path: str = None)
```

Bases: jmetal.core.problem.FloatProblem

Problem DTLZ1. Continuous problem having a flat Pareto front

Note: Unconstrained problem. The default number of variables and objectives are, respectively, 7 and 3.

Parameters

- number_of_variables number of decision variables of the problem.
- **rf_path** Path to the reference front file (if any). Default to None.

 $\textbf{evaluate} \ (solution: jmetal.core.solution.FloatSolution) \ \rightarrow \text{jmetal.core.solution.FloatSolution} \\ \textbf{get_name} \ ()$

Bases: jmetal.core.problem.FloatProblem

Problem DTLZ2. Continuous problem having a convex Pareto front

Note: Unconstrained problem. The default number of variables and objectives are, respectively, 12 and 3.

Parameters

- number_of_variables number of decision variables of the problem
- **rf_path** Path to the reference front file (if any). Default to None.

 $\textbf{evaluate} \ (\textit{solution: jmetal.core.solution.FloatSolution}) \ \rightarrow \textbf{jmetal.core.solution.FloatSolution}$ $\textbf{get_name} \ ()$

ZDT

Problem ZDT1.

Note: Bi-objective unconstrained problem. The default number of variables is 30.

Note: Continuous problem having a convex Pareto front

Parameters

```
• number_of_variables – Number of decision variables of the problem.
```

• **rf_path** – Path to the reference front file (if any). Default to None.

```
\textbf{evaluate} \ (\textit{solution: jmetal.core.solution.FloatSolution}) \ \rightarrow \ \texttt{jmetal.core.solution.FloatSolution}  \textbf{get\_name} \ ()
```

```
class jmetal.problem.multiobjective.zdt.ZDT2 (number\_of\_variables: int = 30, rf\_path: str = None)
```

Bases: jmetal.core.problem.FloatProblem

Problem ZDT2.

Note: Bi-objective unconstrained problem. The default number of variables is 30.

Note: Continuous problem having a non-convex Pareto front

Parameters

- number_of_variables Number of decision variables of the problem.
- **rf_path** Path to the reference front file (if any). Default to None.

```
\textbf{evaluate} \ (\textit{solution: jmetal.core.solution.FloatSolution}) \ \rightarrow \textbf{jmetal.core.solution.FloatSolution}  \textbf{get\_name} \ ()
```

```
class jmetal.problem.multiobjective.zdt.ZDT3 (number\_of\_variables: int = 30, rf\_path: str = None)
```

 $Bases: {\it jmetal.core.problem.FloatProblem}$

Problem ZDT3.

Note: Bi-objective unconstrained problem. The default number of variables is 30.

Note: Continuous problem having a partitioned Pareto front

Parameters

- number_of_variables Number of decision variables of the problem.
- **rf_path** Path to the reference front file (if any). Default to None.

 $\textbf{evaluate} \ (\textit{solution: jmetal.core.solution.FloatSolution}) \ \rightarrow \ \texttt{jmetal.core.solution.FloatSolution}$ $\textbf{get_name} \ ()$

```
class jmetal.problem.multiobjective.zdt.ZDT4 (number\_of\_variables: int = 10, rf\_path: str = None)
Bases: jmetal.core.problem.FloatProblem
```

Problem ZDT4.

4.5. Problems 39

Note: Bi-objective unconstrained problem. The default number of variables is 10.

Note: Continuous multi-modal problem having a convex Pareto front

Parameters

- number_of_variables Number of decision variables of the problem.
- **rf_path** Path to the reference front file (if any). Default to None.

 $\textbf{evaluate} \ (\textit{solution: jmetal.core.solution.FloatSolution}) \ \rightarrow \textbf{jmetal.core.solution.FloatSolution}$ $\textbf{get_name} \ ()$

```
class jmetal.problem.multiobjective.zdt.ZDT6 (number\_of\_variables: int = 10, rf\_path: str = None)
```

 $Bases: {\it jmetal.core.problem.FloatProblem}$

Problem ZDT6.

Note: Bi-objective unconstrained problem. The default number of variables is 10.

Note: Continuous problem having a non-convex Pareto front

Parameters

- number_of_variables Number of decision variables of the problem.
- **rf_path** Path to the reference front file (if any). Default to None.

 $\textbf{evaluate} \ (\textit{solution: jmetal.core.solution.FloatSolution}) \ \rightarrow \textbf{jmetal.core.solution.FloatSolution}$ $\textbf{get_name} \ ()$

4.5.2 Singleobjective problems

Unconstrained

```
\textbf{evaluate} \ (solution: jmetal.core.solution.FloatSolution) \ \rightarrow \text{jmetal.core.solution.FloatSolution}  \textbf{get\_name} \ () \ \rightarrow \text{str}
```

4.6 Utils

4.6.1 Graphic

```
class jmetal.util.graphic.FrontPlot (plot_title: str, axis_labels: list = None)
    Bases: jmetal.util.graphic.Plot
```

Creates a new FrontPlot instance. Suitable for problems with 2 or more objectives.

Parameters

- plot_title Title of the graph.
- axis_labels List of axis labels.

```
export (filename: str = ", include\_plotlyjs: bool = False) \rightarrow str Export as a div for embedding the graph in an HTML file.
```

Parameters

- **filename** Output file name (if desired, default to None).
- include_plotlyjs If True, include plot.ly JS script (default to False).

Returns Script as string.

plot (front: typing.List[S], reference_front: typing.List[S] = None, normalize: bool = False) \rightarrow None Plot a front of solutions (2D, 3D or parallel coordinates).

Parameters

- **front** List of solutions.
- reference_front Reference solution list (if any).
- **normalize** Normalize the input front between 0 and 1 (for problems with more than 3 objectives).

```
to_html (filename: str = 'front') \rightarrow str
```

Export the graph to an interactive HTML (solutions can be selected to show some metadata).

Parameters filename - Output file name.

Returns Script as string.

update (*data: typing.List[S], normalize: bool = False, legend: str* = ") \rightarrow None Update an already created graph with new data.

Parameters

- data List of solutions to be included.
- **legend** Legend to be included.
- **normalize** Normalize the input front between 0 and 1 (for problems with more than 3 objectives).

```
class jmetal.util.graphic.Plot(plot_title: str, axis_labels: list)
    Bases: object
```

4.6. Utils 41

```
static get_objectives (front: typing.List[S]) \rightarrow <Mock name='mock.DataFrame' id='140544273184248'> Get objectives for each solution of the front.
```

Parameters front – List of solutions.

Returns Pandas dataframe with one column for each objective and one row for each solution.

```
class jmetal.util.graphic.ScatterStreaming(plot_title: str, axis_labels: list = None)
    Bases: jmetal.util.graphic.Plot
```

Creates a new ScatterStreaming instance. Suitable for problems with 2 or 3 objectives in streaming.

Parameters

- plot_title Title of the diagram.
- axis_labels List of axis labels.

plot (front: typing.List[S], reference_front: typing.List[S], filename: str = ", $show: bool = True) \rightarrow$ None Plot a front of solutions (2D or 3D).

Parameters

- **front** List of solutions.
- reference_front Reference solution list (if any).
- **filename** If specified, save the plot into a file.
- **show** If True, show the final diagram (default to True).

update (front: typing.List[S], reference_front: typing.List[S], rename_title: str = ", persistence: bool = True) \rightarrow None Update an already created plot.

Parameters

- **front** List of solutions.
- reference_front Reference solution list (if any).
- rename title New title of the plot.
- persistence If True, keep old points; else, replace them with new values.

4.6.2 Lab of experiments

```
class jmetal.util.laboratory.Experiment (algorithm_list: list, n_runs: int = 1, m_workers: int = 6)

Bases: object
```

Parameters

- algorithm_list List of algorithms as Tuple(Algorithm, dic() with parameters).
- m_workers Maximum number of workers for ProcessPoolExecutor.

```
compute_metrics (metric_list: list) → dict
```

Parameters metric_list – List of metrics. Each metric should inherit from Metric or, at least,

contain a method compute.

export_to_file (base_directory: str = 'experiment', function_values_filename: str = 'FUN', variables_filename: str = 'VAR')

4.6.3 Solution list output

```
class jmetal.util.solution_list_output.S = ~S

class jmetal.util.solution_list_output.SolutionList
    Bases: typing.Generic

    static print_function_values_to_file(solution_list: list, file_name)
    static print_function_values_to_screen(solution_list: list)
    static print_variables_to_file(solution_list: list, file_name)
    static print_variables_to_screen(solution_list: list)
```

4.6. Utils 43

iMetalP v	/ Documentation.	Release
------------------	------------------	---------

CHAPTER 5

Installation steps

Via pip:

\$ pip install jmetalpy

Via Github:

```
$ git clone https://github.com/jMetal/jMetalPy.git
```

- \$ pip install -r requirements.txt
 \$ python setup.py install

iMetalP v	/ Documentation.	Release
------------------	------------------	---------

CHAPTER 6

Features

The current release of jMetalPy (v0.5.1) contains the following components:

- Algorithms: random search, NSGA-II, SMPSO, SMPSO/RP
- Benchmark problems: ZDT1-6, DTLZ1-2, unconstrained (Kursawe, Fonseca, Schaffer, Viennet2), constrained (Srinivas, Tanaka).
- Encodings: real, binary
- Operators: selection (binary tournament, ranking and crowding distance, random, nary random, best solution), crossover (single-point, SBX), mutation (bit-blip, polynomial, uniform, random)
- Quality indicators: hypervolume
- Density estimator: crowding distance
- Graphics: Pareto front plotting (2 or more objectives)
- Laboratory: Experiment class for performing studies.

48 Chapter 6. Features

Python Module Index

```
a
                                            jmetal.problem.singleobjective.unconstrained,
algorithm (Unix, Windows), 32
                                            jmetal.util.graphic,41
C
                                            jmetal.util.laboratory, 42
                                            jmetal.util.solution_list_output,43
crowding_distance (Unix, Windows), 28
е
                                            laboratory (Unix, Windows), 43
evolutionary_algorithm (Unix, Windows), 26
                                            n
                                            NSGA-II (Unix, Windows), 22
jmetal.algorithm.multiobjective.nsgaii,
jmetal.algorithm.multiobjective.randomSearch,
                                            observer (Unix, Windows), 29
                                            Operator (Unix, Windows), 32
jmetal.algorithm.multiobjective.smpso,
jmetal.algorithm.singleobjective.evolutionaryalgorithm,
                                            RamdomSearch (Unix, Windows), 24
jmetal.component.archive, 26
jmetal.component.comparator, 27
                                            selection (Unix, Windows), 36
jmetal.component.density_estimator, 27
                                            SMPSO (Unix, Windows), 22
jmetal.component.evaluator, 28
                                            solution_list (Unix, Windows), 43
jmetal.component.observer, 28
jmetal.component.quality_indicator, 29
jmetal.component.ranking, 30
jmetal.core.algorithm, 30
jmetal.core.observable, 34
jmetal.core.operator, 32
jmetal.core.problem, 32
jmetal.core.solution, 33
jmetal.operator.crossover, 34
jmetal.operator.mutation, 35
jmetal.operator.selection, 36
jmetal.problem.multiobjective.constrained,
jmetal.problem.multiobjective.dtlz, 38
jmetal.problem.multiobjective.unconstrained,
jmetal.problem.multiobjective.zdt, 38
```

50 Python Module Index

Α		$compare () \ (jmetal.component.comparator. Ranking And Crowding Distance \ and \ an alternative for the comparator of the component of the component of the comparator of the component of the $
add() (jr	netal.component.archive.Archive method), 26	method), 27
add() (jn	netal.component.archive.ArchiveWithReferencePo	incompare() (jmetal.component.comparator.SolutionAttributeComparator
	method), 26	method), 27
add()	(jmetal.component.archive.BoundedArchive	compute() (jmetal.component.quality_indicator.HyperVolume
	method), 26	method), 29
add() (jn	netal.component.archive.NonDominatedSolutionL	iscampute() (jmetal.component.quality_indicator.Metric method), 29
4.1	method), 27	compute_density_estimator()
	m (class in jmetal.core.algorithm), 30	(jmetal.component.archive.BoundedArchive
_	m (module), 32	method), 27
appende) (jmetal.component.quality_indicator.MultiList method), 30	compute_density_estimator()
Archive	(class in jmetal.component.archive), 26	(jmetal.component.density_estimator.CrowdingDistance
	WithReferencePoint (class in	method), 27
7 Hemve	jmetal.component.archive), 26	compute_density_estimator()
	J	(jmetal.component.density_estimator.DensityEstimator
В		method), 28
BasicAl	gorithmObserver (class in	compute_metrics() (jmetal.util.laboratory.Experiment
	jmetal.component.observer), 28	method), 42
BestSolı	ationSelection (class in jmetal.operator.selection),	compute_ranking() (jmetal.component.ranking.EfficientNonDominatedRa
	36	method), 30
BinaryP	roblem (class in jmetal.core.problem), 32	compute_ranking() (jmetal.component.ranking.FastNonDominatedRanking)
	olution (class in jmetal.core.solution), 33	method), 30 compute_ranking() (jmetal.component.ranking.Ranking
BinaryT	ournament2Selection (class in	method), 30
	jmetal.operator.selection), 36	create_initial_population()
BinaryT	ournamentSelection (class in	(jmetal.algorithm.singleobjective.evolutionaryalgorithm.ElitistE
D'/El' /	jmetal.operator.selection), 36	method), 24
	class in jmetal operator.mutation), 35	create_initial_population()
Doulided	dArchive (class in jmetal.component.archive), 26	(jmetal.algorithm.singleobjective.evolutionaryalgorithm.Genera
С		method), 25
	oton (class in imptal commonant commonator) 27	create_initial_population()
compare	ator (class in jmetal.component.comparator), 27 (jmetal.component.comparator.Comparator	(jmetal.core.algorithm.EvolutionaryAlgorithm
compare	method), 27	method), 31
compare	e() (jmetal.component.comparator.DominanceCom	create_initial_swarm() (jmetal.algorithm.multiobjective.smpso.SMPSO
Company	method), 27	method), 25
compare	c() (jmetal.component.comparator.EqualSolutionsC	create_initial_swarm() (jmetal.core.algorithm.ParticleSwarmOptimization Comparator_mathod) 31
1	method), 27	create_solution() (jmetal.core.problem.BinaryProblem
		method) 32

method), 32	evaluate() (jmetal.core.problem.Problem method), 33 evaluate() (jmetal.problem.multiobjective.constrained.Srinivas
create_solution() (jmetal.core.problem.IntegerProblem method), 33	method), 37 evaluate() (jmetal.problem.multiobjective.constrained.Tanaka
create_solution() (jmetal.core.problem.Problem method),	method), 37
33	evaluate() (jmetal.problem.multiobjective.dtlz.DTLZ1
$create_solution() \ (jmetal.problem.single objective.unconstrated and the problem of the probl$	
method), 40	evaluate() (jmetal.problem.multiobjective.dtlz.DTLZ2
Crossover (class in jmetal.core.operator), 32	method), 38
crowding_distance (module), 28	evaluate() (jmetal.problem.multiobjective.unconstrained.Fonseca
CrowdingDistance (class in	method), 37
jmetal.component.density_estimator), 27	evaluate() (jmetal.problem.multiobjective.unconstrained.Kursawe
CrowdingDistanceArchive (class in	method), 37
jmetal.component.archive), 27	evaluate() (jmetal.problem.multiobjective.unconstrained.Schaffer
CrowdingDistanceArchiveWithReferencePoint (class in jmetal.component.archive), 27	method), 37 evaluate() (jmetal.problem.multiobjective.unconstrained.Viennet2
Jinetar.component.arcmve), 27	method), 37
D	evaluate() (jmetal.problem.multiobjective.zdt.ZDT1
	method), 39
DefaultObservable (class in jmetal.core.observable), 34 DensityEstimator (class in	evaluate() (jmetal.problem.multiobjective.zdt.ZDT2
DensityEstimator (class in jmetal.component.density_estimator), 28	method), 39
deregister() (jmetal.core.observable.DefaultObservable	evaluate() (jmetal.problem.multiobjective.zdt.ZDT3
method), 34	method), 39
deregister() (jmetal.core.observable.Observable method),	evaluate() (jmetal.problem.multiobjective.zdt.ZDT4
34	method), 40
deregister_all() (jmetal.core.observable.DefaultObservable	
method), 34	method), 40
deregister_all() (jmetal.core.observable.Observable	$evaluate()\ (jmet al.problem.single objective.unconstrained. One Max$
method), 34	method), 40
DominanceComparator (class in	evaluate() (jmetal.problem.singleobjective.unconstrained.Sphere
jmetal.component.comparator), 27	method), 40
DTLZ1 (class in jmetal.problem.multiobjective.dtlz), 38	evaluate_constraints() (jmetal.core.problem.Problem
DTLZ2 (class in jmetal.problem.multiobjective.dtlz), 38	method), 33
Г	evaluate_constraints() (jmetal.problem.multiobjective.constrained.Srinivas
E	method), 37
EfficientNonDominatedRanking (class in	evaluate_constraints() (jmetal.problem.multiobjective.constrained.Tanaka
jmetal.component.ranking), 30	method), 37 evaluate_population() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.singleobjective.ev
EntistEvolutionstrategy (class in	mathad) 24
jmetal.algorithm.singleobjective.evolutionaryalgo	orithm), methody, 24 evaluate_population() (jmetal.algorithm.singleobjective.evolutionaryalgorithm
24	method), 25
EqualSolutionsComparator (class in	evaluate_population() (jmetal.core.algorithm.EvolutionaryAlgorithm
jmetal.component.comparator), 27	method), 31
evaluate() (jmetal.component.evaluator.Evaluator	evaluate_solution() (jmetal.component.evaluator.Evaluator
method), 28	static method), 28
evaluate() (jmetal.component.evaluator.MapEvaluator method), 28	evaluate_swarm() (jmetal.algorithm.multiobjective.smpso.SMPSO
evaluate() (jmetal.component.evaluator.SequentialEvaluator	mathad) 22
method), 28	evaluate_swarm() (jmetal.core.algorithm.ParticleSwarmOptimization
evaluate() (jmetal.core.problem.BinaryProblem method),	method), 31
32	Evaluator (class in jmetal.component.evaluator), 28
evaluate() (jmetal.core.problem.FloatProblem method),	evolutionary_algorithm (module), 26
33	EvolutionaryAlgorithm (class in jmetal.core.algorithm),
evaluate() (jmetal.core.problem.IntegerProblem method),	31
33	execute() (jmetal.core.operator.Crossover method), 32

execute() (jmetal.core.operator.Mutation method), 32	get_evaluations() (jmetal.core.algorithm.Algorithm
execute() (jmetal.core.operator.Mutation method), 32 execute() (jmetal.core.operator.Operator method), 32	method), 30
	get_length() (jmetal.component.quality_indicator.MultiList
execute() (jmetal.operator.crossover.NullCrossover	method), 30
	get_name() (jmetal.algorithm.multiobjective.nsgaii.NSGAII
execute() (jmetal.operator.crossover.SBX method), 35	method), 22
	$get_name() \ (jmetal.algorithm.multiobjective.randomSearch.RandomSearch. \\$
execute() (jmetal.operator.mutation.BitFlip method), 35	static method), 24
execute() (jmetal.operator.mutation.IntegerPolynomial method), 35	get_name() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.ElitistEmethod), 24
execute() (jmetal.operator.mutation.NullMutation method), 35	get_name() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.General method), 25
execute() (jmetal.operator.mutation.Polynomial method), 35	get_name() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.NonElimethod), 26
execute() (jmetal.operator.mutation.SimpleRandom method), 35	get_name() (jmetal.component.archive.Archive method), 26
	get_name() (jmetal.component.comparator.Comparator
execute() (jmetal.operator.selection.BestSolutionSelection	method), 27
method), 36	get_name() (jmetal.component.evaluator.Evaluator
execute() (jmetal.operator.selection.BinaryTournament2Sele	ection method), 28
	get_name() (jmetal.component.quality_indicator.HyperVolume
execute() (jmetal.operator.selection.BinaryTournamentSelec	
	get_name() (jmetal.component.quality_indicator.Metric
execute() (jmetal.operator.selection.NaryRandomSolutionSemethod), 36	election method), 29 get_name() (jmetal.core.algorithm.Algorithm method),
$execute() \ (jmetal.operator.selection. Random Solution Selection \ (in the context of the con$	
	get_name() (jmetal.core.operator.Crossover method), 32
execute() (jmetal.operator.selection.RankingAndCrowdingD	
	get_name() (jmetal.core.operator.Operator method), 32
	get_name() (jmetal.core.operator.Selection method), 32
	get_name() (jmetal.core.problem.Problem method), 33
method), 42	get_name() (jmetal.operator.crossover.NullCrossover method), 34
	get_name() (jmetal.operator.crossover.SBX method), 35
	get_name() (jmetal.operator.crossover.SP method), 35
	get_name() (jmetal.operator.mutation.BitFlip method), 35
	get_name() (jmetal.operator.mutation.IntegerPolynomial
FastNonDominatedRanking (class in	method), 35
jmetal.component.ranking), 30	get_name() (jmetal.operator.mutation.NullMutation
FloatProblem (class in jmetal.core.problem), 32	method), 35
,,	get_name() (jmetal.operator.mutation.Polynomial
Fonseca (class in jmetal.problem.multiobjective.unconstrain 37	ed), method), 35 get_name() (jmetal.operator.mutation.SimpleRandom
FrontPlot (class in jmetal.util.graphic), 41	method), 35
	get_name() (jmetal.operator.mutation.Uniform method),
G	36
GenerationalGeneticAlgorithm (class in	get_name() (jmetal.operator.selection.BestSolutionSelection
jmetal.algorithm.singleobjective.evolutionaryalgo	rithm), method), 36
23	get_name() (jmetal.operator.selection.BinaryTournament2Selection
get() (jmetal.component.archive.Archive method), 26	method), 36 get_name() (jmetal.operator.selection.BinaryTournamentSelection
get_eurrent_eomputing_time()	method), 36
(jmetal.core.algorithm.Algorithm method),	get_name() (jmetal.operator.selection.NaryRandomSolutionSelection
30	method), 36

```
get name() (imetal.operator.selection.RandomSolutionSelection.result() (imetal.algorithm.multiobjective.smpso.SMPSO
          method), 36
                                                                    method), 23
get name() (jmetal.operator.selection.RankingAndCrowdingDistarsub(Se)(enternal.algorithm.multiobjective.smpso.SMPSORP
          method), 36
                                                                    method), 24
get_name() (jmetal.problem.multiobjective.constrained.Sringats_result() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.ElitistE
         method), 37
                                                                    method), 25
get name() (jmetal.problem.multiobjective.constrained.Tanakat result() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.Genera
          method), 37
                                                                    method), 25
get_name() (jmetal.problem.multiobjective.dtlz.DTLZ1 get_result() (jmetal.core.algorithm.Algorithm method),
          method), 38
                                                                    30
get_name() (jmetal.problem.multiobjective.dtlz.DTLZ2
                                                          get_subfront()
                                                                               (jmetal.component.ranking.Ranking
          method), 38
                                                                    method), 30
get_name() (jmetal.problem.multiobjective.unconstrained.Foresteetatal number of bits()
                                                                    (jmetal.core.solution.BinarySolution method),
         method), 37
get_name() (jmetal.problem.multiobjective.unconstrained.Kursawe
          method), 37
get\_name() \ (jmetal.problem.multiobjective.unconstrained.Schaffer
          method), 37
                                                          HyperVolume
                                                                                          (class
                                                                                                                 in
get_name() (jmetal.problem.multiobjective.unconstrained.Viennet2 jmetal.component.quality_indicator), 29
          method), 37
get_name()
                (jmetal.problem.multiobjective.zdt.ZDT1
                                                          init progress() (jmetal.algorithm.multiobjective.smpso.SMPSO
                (jmetal.problem.multiobjective.zdt.ZDT2
get name()
                                                                    method), 23
          method), 39
                                                          init progress() (jmetal.algorithm.multiobjective.smpso.SMPSORP
                (jmetal.problem.multiobjective.zdt.ZDT3
get name()
                                                                    method), 24
          method), 39
                                                          init progress() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.Elit
                (jmetal.problem.multiobjective.zdt.ZDT4
get_name()
                                                                    method), 25
          method), 40
                                                          init progress() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.Ger
                (jmetal.problem.multiobjective.zdt.ZDT6
get_name()
                                                                    method), 25
method), 40 init_progress() (jmetal.core.algorithm.EvolutionaryAlgorithm get_name() (jmetal.problem.singleobjective.unconstrained.OneMax method). 31
          method), 40
                                                                    method), 31
          method), 40
                                                          init\_progress() (jmetal.core.algorithm.ParticleSwarmOptimization
get_name() (jmetal.problem.singleobjective.unconstrained
                                                         Sphere
                                                                    method), 31
          method), 41
                                                          initialize_global_best() (jmetal.algorithm.multiobjective.smpso.SMPSO
get_number_of_parents() (jmetal.core.operator.Crossover
                                                                    method), 23
         method), 32
                                                          initialize_global_best() (jmetal.algorithm.multiobjective.smpso.SMPSORP
get number of parents()
                                                                    method), 24
         (jmetal.operator.crossover.NullCrossover
                                                          initialize global best() (jmetal.core.algorithm.ParticleSwarmOptimization
          method), 35
                                                                    method), 31
get_number_of_parents() (jmetal.operator.crossover.SBX
                                                          initialize particle best() (jmetal.algorithm.multiobjective.smpso.SMPSO
         method), 35
                                                                    method), 23
get number of parents()
                           (jmetal.operator.crossover.SP
                                                          initialize_particle_best() (jmetal.core.algorithm.ParticleSwarmOptimization
          method), 35
                                                                    method), 31
get_number_of_subfronts()
                                                          initialize velocity() (jmetal.algorithm.multiobjective.smpso.SMPSO
          (jmetal.component.ranking.Ranking method),
                                                                    method), 23
                                                          initialize_velocity() (jmetal.core.algorithm.ParticleSwarmOptimization
get_objectives() (jmetal.util.graphic.Plot static method),
                                                                    method), 31
                                                          IntegerPolynomial (class in jmetal.operator.mutation), 35
get_reference_point() (jmetal.component.archive.ArchiveWater)
                                                          thReferromenitelass in jmetal.core.problem), 33
          method), 26
                                                          IntegerSolution (class in imetal.core.solution), 33
get\_result() \ (jmetal.algorithm.multiobjective.nsgaii.NSGAII_{is\_stopping\_condition\_reached()}
         method), 22
                                                                    (jmetal.algorithm.multiobjective.smpso.SMPSO
get_result() (jmetal.algorithm.multiobjective.randomSearch.RandomSearch.23)
          method), 24
```

is_stopping_condition_reached() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.singleobjective.evolutionaryalgorithm.singleobjective.evolutionaryalgorithm.	L gqrithm,ElitistEyqlutjonStrategy
method), 25	inobiatory (modulo), 15
is_stopping_condition_reached()	M
(jmetal.algorithm.singleobjective.evolutionaryal	gorithm.GenerationalGeneticAlgorithm MapEvaluator (class in jmetal component evaluator), 28
method), 25	MAXIMIZE (jmetal.core.problem.Problem attribute), 33
is_stopping_condition_reached()	Metric (class in jmetal.component.quality_indicator), 29
(jmetal.core.algorithm.EvolutionaryAlgorithm	MINIMIZE (jmetal.core.problem.Problem attribute), 33
method), 31	MultiList (class in jmetal.component.quality_indicator),
is_stopping_condition_reached()	29
(jmetal.core.algorithm.ParticleSwarmOptimizati method), 31	
method), 51	jmetal.component.quality_indicator), 29
J	Mutation (class in jmetal.core.operator), 32
jmetal.algorithm.multiobjective.nsgaii (module), 21	N
jmetal.algorithm.multiobjective.randomSearch (module),	NaryRandomSolutionSelection (class in
24	jmetal.operator.selection), 36
jmetal.algorithm.multiobjective.smpso (module), 22	NonDominatedSolutionListArchive (class in
jmetal.algorithm.singleobjective.evolutionaryalgorithm	jmetal.component.archive), 27
(module), 24	NonElitistEvolutionStrategy (class in
jmetal.component.archive (module), 26 jmetal.component.comparator (module), 27	jmetal.algorithm.singleobjective.evolutionaryalgorithm),
jmetal.component.density_estimator (module), 27	26
jmetal.component.evaluator (module), 28	notify_all() (jmetal.core.observable.DefaultObservable
jmetal.component.observer (module), 28	method), 34
jmetal.component.quality_indicator (module), 29	notify_all() (jmetal.core.observable.Observable method),
jmetal.component.ranking (module), 30	NSCA II (modulo) 22
jmetal.core.algorithm (module), 30	NSGA-II (module), 22 NSGAII (class in jmetal.algorithm.multiobjective.nsgaii),
jmetal.core.observable (module), 34	21
jmetal.core.operator (module), 32	NullCrossover (class in jmetal.operator.crossover), 34
jmetal.core.problem (module), 32	NullMutation (class in jmetal.operator.mutation), 35
jmetal.core.solution (module), 33	
jmetal.operator.crossover (module), 34 jmetal.operator.mutation (module), 35	0
jmetal.operator.selection (module), 36	Observable (class in jmetal.core.observable), 34
jmetal.problem.multiobjective.constrained (module), 37	Observer (class in jmetal.core.observable), 34
jmetal.problem.multiobjective.dtlz (module), 38	observer (module), 29
jmetal.problem.multiobjective.unconstrained (module),	OneMax (class in jmetal.problem.singleobjective.unconstrained),
37	40
jmetal.problem.multiobjective.zdt (module), 38	Operator (class in jmetal.core.operator), 32
jmetal.problem.singleobjective.unconstrained (module),	Operator (module), 32
40	P
jmetal.util.graphic (module), 41	-
jmetal.util.laboratory (module), 42	ParticleSwarmOptimization (class in
jmetal.util.solution_list_output (module), 43	jmetal.core.algorithm), 31 perturbation() (jmetal.algorithm.multiobjective.smpso.SMPSO
jMetalPyLogger (in module jmetal.component.observer),	method), 23
jMetalPyLogger (in module jmetal.util.laboratory), 43	perturbation() (jmetal.core.algorithm.ParticleSwarmOptimization
K	method), 31 Plot (class in jmetal.util.graphic), 41
	not (crass in jinctar.um.graphic), 41 not() (imetal util graphic FrontPlot method) 41
Kursawe (class in jmetal.problem.multiobjective.unconstra	plot() (jmetal.util.graphic.ScatterStreaming method), 42
37	Polynomial (class in jmetal.operator.mutation), 35

print_function_values_to_file() (jmetal.util.solution_list_output.SolutionList	replacement() (jmetal.core.algorithm.EvolutionaryAlgorithm method), 31
static method), 43	reproduction() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.Elit
print_function_values_to_screen()	method), 25
(jmetal.util.solution_list_output.SolutionList	reproduction() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.Gen
static method), 43	method), 25
	oketiondlustion() (jmetal.core.algorithm.EvolutionaryAlgorithm
static method), 43	method), 31
print_variables_to_screen()	run() (jmetal.algorithm.multiobjective.randomSearch.RandomSearch
$(jmetal.util.solution_list_output.SolutionList$	method), 24
static method), 43	run() (jmetal.core.algorithm.EvolutionaryAlgorithm
Problem (class in jmetal.core.problem), 33	method), 31
ProgressBarObserver (class in	run() (jmetal.core.algorithm.ParticleSwarmOptimization
jmetal.component.observer), 28	method), 31
R	run() (jmetal.util.laboratory.Experiment method), 43
R (in module jmetal.algorithm.multiobjective.nsgaii), 22	S
R (in module jmetal.algorithm.multiobjective.smpso), 22	S (in module jmetal.algorithm.multiobjective.randomSearch),
R (in module jmetal.algorithm.singleobjective.evolutionary	
26	S (in module jmetal.component.density_estimator), 28
R (in module jmetal.core.algorithm), 32	S (in module jmetal.operator.selection), 36
R (in module jmetal.core.operator), 32	S (in module jmetal.util.solution_list_output), 43
RamdomSearch (module), 24	SBX (class in jmetal.operator.crossover), 35
RandomSearch (class in	ScatterStreaming (class in jmetal.util.graphic), 42
jmetal.algorithm.multiobjective.randomSearch), 24	Schaffer (class in jmetal.problem.multiobjective.unconstrained), 37
RandomSolutionSelection (class in	select_global_best() (jmetal.algorithm.multiobjective.smpso.SMPSO
jmetal.operator.selection), 36	method), 23
Ranking (class in jmetal.component.ranking), 30	select_global_best() (jmetal.algorithm.multiobjective.smpso.SMPSORP
RankingAndCrowdingDistanceComparator (class in	method), 24
jmetal.component.comparator), 27	Selection (class in jmetal.core.operator), 32
RankingAndCrowdingDistanceSelection (class in	selection (module), 36
jmetal.operator.selection), 36	selection() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.ElitistE
read_front_from_file() (jmetal.core.problem.Problem	method), 25
static method), 33	$selection () \ (jmetal. algorithm. single objective. evolutionary algorithm. Generating the property of the $
read_front_from_file_as_solutions()	method), 25
(jmetal.core.problem.Problem static method),	selection() (jmetal.core.algorithm.EvolutionaryAlgorithm
33	method), 31
register() (jmetal.core.observable.DefaultObservable	SequentialEvaluator (class in
method), 34	jmetal.component.evaluator), 28
register() (jmetal.core.observable.Observable method), 34	SimpleRandom (class in jmetal.operator.mutation), 35
reinsert() (jmetal.component.quality_indicator.MultiList	size() (jmetal.component.archive.Archive method), 26
method), 30	SMPSO (class in jmetal.algorithm.multiobjective.smpso),
remove() (jmetal.component.quality_indicator.MultiList	22 SMRSQ (m. 1.1.) 22
method), 30	SMPSO (module), 22
method), 22	ASIMPSORP (class in jmetal.algorithm.multiobjective.smpso), 23
replacement() (jmetal.algorithm.singleobjective.evolutiona	rySodentiithr(cEditistEprodutaloroSeratelegion), 34
method), 25	solution_list (module), 43
replacement() (jmetal.algorithm.singleobjective.evolutiona	
method), 25	jmetal.component.comparator), 27
replacement() (jmetal.algorithm.singleobjective.evolutiona	
method), 26	SP (class in jmetal.operator.crossover), 35

```
Sphere (class in jmetal.problem.singleobjective.unconstrained) date velocity() (jmetal.algorithm.multiobjective.smpso.SMPSO
                                                                                                                        method), 23
Srinivas (class in jmetal.problem.multiobjective.constrained)update velocity() (jmetal.core.algorithm.ParticleSwarmOptimization
                                                                                                                        method), 31
Т
Tanaka (class in jmetal.problem.multiobjective.constrained), Viennet2 (class in jmetal.problem.multiobjective.unconstrained),
to html() (jmetal.util.graphic.FrontPlot method), 41
                                                                                                       VisualizerObserver (class in jmetal.component.observer),
U
                                                                                                       W
Uniform (class in jmetal.operator.mutation), 35
update() \ (jmetal.component.observer. Basic Algorithm Observ \ Write Front To File Observer \ Algorithm Observer \ Write Front To File Observer \ Algorithm Observer \ Write Front To File Observer \ Algorithm Observer \ Write Front To File Observer \ Algorithm Observer \ Write Front To File Observer \ Algorithm Observer \ Write Front To File Observer \ Algorithm Observer \ Write Front To File Obser
                                                                                                                                                                         (class
                                                                                                                                                                                                      in
                                                                                                                        jmetal.component.observer), 29
                 method), 28
update() (jmetal.component.observer.ProgressBarObserver
                                                                                                       Z
                 method), 29
update() (jmetal.component.observer.VisualizerObserver
                                                                                                       ZDT1 (class in jmetal.problem.multiobjective.zdt), 38
                 method), 29
                                                                                                       ZDT2 (class in jmetal.problem.multiobjective.zdt), 39
update() (jmetal.component.observer.WriteFrontToFileObserver3 (class in jmetal.problem.multiobjective.zdt), 39
                 method), 29
                                                                                                       ZDT4 (class in jmetal.problem.multiobjective.zdt), 39
update() (jmetal.core.observable.Observer method), 34
                                                                                                       ZDT6 (class in jmetal.problem.multiobjective.zdt), 40
update() (jmetal.util.graphic.FrontPlot method), 41
update() (jmetal.util.graphic.ScatterStreaming method),
update_global_best() (jmetal.algorithm.multiobjective.smpso.SMPSO
                 method), 23
update global best() (jmetal.algorithm.multiobjective.smpso.SMPSORP
                 method), 24
update_global_best() (jmetal.core.algorithm.ParticleSwarmOptimization
                 method), 31
update_leaders_density_estimator()
                 (jmetal.algorithm.multiobjective.smpso.SMPSORP
                 method), 24
update_particle_best() (jmetal.algorithm.multiobjective.smpso.SMPSO
                 method), 23
update\_particle\_best() \ (jmetal.core.algorithm.ParticleSwarmOptimization
                 method), 31
update position() (jmetal.algorithm.multiobjective.smpso.SMPSO
                 method), 23
update_position() (jmetal.core.algorithm.ParticleSwarmOptimization
                 method), 31
update_progress() (jmetal.algorithm.multiobjective.smpso.SMPSO
                 method), 23
update progress() (jmetal.algorithm.multiobjective.smpso.SMPSORP
                 method), 24
update_progress() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.ElitistEvolutionStrategy
                 method), 25
update_progress() (jmetal.algorithm.singleobjective.evolutionaryalgorithm.GenerationalGeneticAlgorithm
                 method), 25
update_progress() (jmetal.core.algorithm.EvolutionaryAlgorithm
                 method), 31
update_progress() (jmetal.core.algorithm.ParticleSwarmOptimization
                 method), 31
```