

practica5

Generado por Doxygen 1.15.0



# Chapter 1

## Jerarquía de directorios

### 1.1 Directories

acceptation_type . . . . .	31
AcceptableCandidate.java . . . . .	??
AcceptAnyone.java . . . . .	??
AcceptBest.java . . . . .	??
AcceptMulticase.java . . . . .	??
AcceptNotBad.java . . . . .	??
AcceptNotBadT.java . . . . .	??
AcceptNotBadU.java . . . . .	??
AcceptNotDominated.java . . . . .	??
AcceptNotDominatedTabu.java . . . . .	??
AcceptType.java . . . . .	??
Dominance.java . . . . .	??
candidate_type . . . . .	32
CandidateType.java . . . . .	??
CandidateValue.java . . . . .	??
GreaterCandidate.java . . . . .	??
NotDominatedCandidate.java . . . . .	??
RandomCandidate.java . . . . .	??
SearchCandidate.java . . . . .	??
SmallerCandidate.java . . . . .	??
complement . . . . .	32
AIOMutation.java . . . . .	??
Crossover.java . . . . .	??
CrossoverType.java . . . . .	??
Distribution.java . . . . .	??
DistributionType.java . . . . .	??
FatherSelection.java . . . . .	??
GenerationalReplace.java . . . . .	??
Mutation.java . . . . .	??
MutationType.java . . . . .	??
OnePointCrossover.java . . . . .	??
OnePointMutation.java . . . . .	??
ProbabilisticSampling.java . . . . .	??
Probability.java . . . . .	??
Range.java . . . . .	??
Replace.java . . . . .	??
ReplaceType.java . . . . .	??

RouletteSelection.java . . . . .	??
Sampling.java . . . . .	??
SamplingType.java . . . . .	??
SelectionType.java . . . . .	??
SteadyStateReplace.java . . . . .	??
TowPointsMutation.java . . . . .	??
TruncationSelection.java . . . . .	??
UniformCrossover.java . . . . .	??
Univariate.java . . . . .	??
complement . . . . .	33
StopExecute.java . . . . .	??
StrategyType.java . . . . .	??
TabuSolutions.java . . . . .	??
UpdateParameter.java . . . . .	??
config . . . . .	34
tspDynamic . . . . .	43
TSPState.java . . . . .	??
definition . . . . .	34
Codification.java . . . . .	??
ObjetiveFunction.java . . . . .	??
Operator.java . . . . .	??
Problem.java . . . . .	??
State.java . . . . .	??
es . . . . .	35
ull . . . . .	43
App.java . . . . .	??
evolutionary_algorithms . . . . .	35
complement . . . . .	32
AIOMutation.java . . . . .	??
Crossover.java . . . . .	??
CrossoverType.java . . . . .	??
Distribution.java . . . . .	??
DistributionType.java . . . . .	??
FatherSelection.java . . . . .	??
GenerationalReplace.java . . . . .	??
Mutation.java . . . . .	??
MutationType.java . . . . .	??
OnePointCrossover.java . . . . .	??
OnePointMutation.java . . . . .	??
ProbabilisticSampling.java . . . . .	??
Probability.java . . . . .	??
Range.java . . . . .	??
Replace.java . . . . .	??
ReplaceType.java . . . . .	??
RouletteSelection.java . . . . .	??
Sampling.java . . . . .	??
SamplingType.java . . . . .	??
SelectionType.java . . . . .	??
SteadyStateReplace.java . . . . .	??
TowPointsMutation.java . . . . .	??
TruncationSelection.java . . . . .	??
UniformCrossover.java . . . . .	??
Univariate.java . . . . .	??
extension . . . . .	36
FactoresPonderados.java . . . . .	??
MetricasMultiobjetivo.java . . . . .	??

MultiObjetivoPuro.java . . . . .	??
SolutionMethod.java . . . . .	??
TypeSolutionMethod.java . . . . .	??
factory_interface . . . . .	36
IFFactoryAcceptCandidate.java . . . . .	??
IFFactoryCandidate.java . . . . .	??
IFFactoryCrossover.java . . . . .	??
IFFactoryDistribution.java . . . . .	??
IFFactoryFatherSelection.java . . . . .	??
IFFactoryGenerator.java . . . . .	??
IFFactoryMutation.java . . . . .	??
IFFactoryReplace.java . . . . .	??
IFFactorySolutionMethod.java . . . . .	??
IFFSampling.java . . . . .	??
factory_method . . . . .	37
FactoryAcceptCandidate.java . . . . .	??
FactoryCandidate.java . . . . .	??
FactoryCrossover.java . . . . .	??
FactoryDistribution.java . . . . .	??
FactoryFatherSelection.java . . . . .	??
FactoryGenerator.java . . . . .	??
FactoryLoader.java . . . . .	??
FactoryMutation.java . . . . .	??
FactoryReplace.java . . . . .	??
FactorySampling.java . . . . .	??
FactorySolutionMethod.java . . . . .	??
generators . . . . .	37
DistributionEstimationAlgorithm.java . . . . .	??
EvolutionStrategies.java . . . . .	??
Generator.java . . . . .	??
GeneratorType.java . . . . .	??
GeneticAlgorithm.java . . . . .	??
HillClimbing.java . . . . .	??
HillClimbingRestart.java . . . . .	??
InstanceDE.java . . . . .	??
InstanceEE.java . . . . .	??
InstanceGA.java . . . . .	??
LimitRoulette.java . . . . .	??
LimitThreshold.java . . . . .	??
MultiCaseSimulatedAnnealing.java . . . . .	??
MultiGenerator.java . . . . .	??
MultiobjectiveHillClimbingDistance.java . . . . .	??
MultiobjectiveHillClimbingRestart.java . . . . .	??
MultiobjectiveStochasticHillClimbing.java . . . . .	??
MultiobjectiveTabuSearch.java . . . . .	??
Particle.java . . . . .	??
ParticleSwarmOptimization.java . . . . .	??
RandomSearch.java . . . . .	??
SimulatedAnnealing.java . . . . .	??
TabuSearch.java . . . . .	??
java . . . . .	38
config . . . . .	34
tspDynamic . . . . .	43
TSPState.java . . . . .	??
es . . . . .	35
ull . . . . .	43
App.java . . . . .	??

evolutionary_algorithms . . . . .	35
complement . . . . .	32
AIOMutation.java . . . . .	??
Crossover.java . . . . .	??
CrossoverType.java . . . . .	??
Distribution.java . . . . .	??
DistributionType.java . . . . .	??
FatherSelection.java . . . . .	??
GenerationalReplace.java . . . . .	??
Mutation.java . . . . .	??
MutationType.java . . . . .	??
OnePointCrossover.java . . . . .	??
OnePointMutation.java . . . . .	??
ProbabilisticSampling.java . . . . .	??
Probability.java . . . . .	??
Range.java . . . . .	??
Replace.java . . . . .	??
ReplaceType.java . . . . .	??
RouletteSelection.java . . . . .	??
Sampling.java . . . . .	??
SamplingType.java . . . . .	??
SelectionType.java . . . . .	??
SteadyStateReplace.java . . . . .	??
TowPointsMutation.java . . . . .	??
TruncationSelection.java . . . . .	??
UniformCrossover.java . . . . .	??
Univariate.java . . . . .	??
factory_interface . . . . .	36
IFFactoryAcceptCandidate.java . . . . .	??
IFFactoryCandidate.java . . . . .	??
IFFactoryCrossover.java . . . . .	??
IFFactoryDistribution.java . . . . .	??
IFFactoryFatherSelection.java . . . . .	??
IFFactoryGenerator.java . . . . .	??
IFFactoryMutation.java . . . . .	??
IFFactoryReplace.java . . . . .	??
IFFactorySolutionMethod.java . . . . .	??
IFFSampling.java . . . . .	??
factory_method . . . . .	37
FactoryAcceptCandidate.java . . . . .	??
FactoryCandidate.java . . . . .	??
FactoryCrossover.java . . . . .	??
FactoryDistribution.java . . . . .	??
FactoryFatherSelection.java . . . . .	??
FactoryGenerator.java . . . . .	??
FactoryLoader.java . . . . .	??
FactoryMutation.java . . . . .	??
FactoryReplace.java . . . . .	??
FactorySampling.java . . . . .	??
FactorySolutionMethod.java . . . . .	??
local_search . . . . .	39
acceptation_type . . . . .	31
AcceptableCandidate.java . . . . .	??
AcceptAnyone.java . . . . .	??
AcceptBest.java . . . . .	??
AcceptMulticase.java . . . . .	??
AcceptNotBad.java . . . . .	??
AcceptNotBadT.java . . . . .	??

AcceptNotBadU.java . . . . .	??
AcceptNotDominated.java . . . . .	??
AcceptNotDominatedTabu.java . . . . .	??
AcceptType.java . . . . .	??
Dominance.java . . . . .	??
candidate_type . . . . .	32
CandidateType.java . . . . .	??
CandidateValue.java . . . . .	??
GreaterCandidate.java . . . . .	??
NotDominatedCandidate.java . . . . .	??
RandomCandidate.java . . . . .	??
SearchCandidate.java . . . . .	??
SmallerCandidate.java . . . . .	??
complement . . . . .	33
StopExecute.java . . . . .	??
StrategyType.java . . . . .	??
TabuSolutions.java . . . . .	??
UpdateParameter.java . . . . .	??
metaheuristics . . . . .	40
strategy . . . . .	42
Strategy.java . . . . .	??
metaheuristics . . . . .	40
generators . . . . .	37
DistributionEstimationAlgorithm.java . . . . .	??
EvolutionStrategies.java . . . . .	??
Generator.java . . . . .	??
GeneratorType.java . . . . .	??
GeneticAlgorithm.java . . . . .	??
HillClimbing.java . . . . .	??
HillClimbingRestart.java . . . . .	??
InstanceDE.java . . . . .	??
InstanceEE.java . . . . .	??
InstanceGA.java . . . . .	??
LimitRoulette.java . . . . .	??
LimitThreshold.java . . . . .	??
MultiCaseSimulatedAnnealing.java . . . . .	??
MultiGenerator.java . . . . .	??
MultiobjectiveHillClimbingDistance.java . . . . .	??
MultiobjectiveHillClimbingRestart.java . . . . .	??
MultiobjectiveStochasticHillClimbing.java . . . . .	??
MultiobjectiveTabuSearch.java . . . . .	??
Particle.java . . . . .	??
ParticleSwarmOptimization.java . . . . .	??
RandomSearch.java . . . . .	??
SimulatedAnnealing.java . . . . .	??
TabuSearch.java . . . . .	??
problem . . . . .	41
definition . . . . .	34
Codification.java . . . . .	??
ObjetiveFunction.java . . . . .	??
Operator.java . . . . .	??
Problem.java . . . . .	??
State.java . . . . .	??
extension . . . . .	36
FactoresPonderados.java . . . . .	??
MetricasMultiobjetivo.java . . . . .	??
MultiObjetivoPuro.java . . . . .	??
SolutionMethod.java . . . . .	??

TypeSolutionMethod.java . . . . .	??
problem_operators . . . . .	41
MutationOperator.java . . . . .	??
local_search . . . . .	39
acceptation_type . . . . .	31
AcceptableCandidate.java . . . . .	??
AcceptAnyone.java . . . . .	??
AcceptBest.java . . . . .	??
AcceptMulticase.java . . . . .	??
AcceptNotBad.java . . . . .	??
AcceptNotBadT.java . . . . .	??
AcceptNotBadU.java . . . . .	??
AcceptNotDominated.java . . . . .	??
AcceptNotDominatedTabu.java . . . . .	??
AcceptType.java . . . . .	??
Dominance.java . . . . .	??
candidate_type . . . . .	32
CandidateType.java . . . . .	??
CandidateValue.java . . . . .	??
GreaterCandidate.java . . . . .	??
NotDominatedCandidate.java . . . . .	??
RandomCandidate.java . . . . .	??
SearchCandidate.java . . . . .	??
SmallerCandidate.java . . . . .	??
complement . . . . .	33
StopExecute.java . . . . .	??
StrategyType.java . . . . .	??
TabuSolutions.java . . . . .	??
UpdateParameter.java . . . . .	??
main . . . . .	39
java . . . . .	38
config . . . . .	34
tspDynamic . . . . .	43
TSPState.java . . . . .	??
es . . . . .	35
ull . . . . .	43
App.java . . . . .	??
evolutionary_algorithms . . . . .	35
complement . . . . .	32
AIOMutation.java . . . . .	??
Crossover.java . . . . .	??
CrossoverType.java . . . . .	??
Distribution.java . . . . .	??
DistributionType.java . . . . .	??
FatherSelection.java . . . . .	??
GenerationalReplace.java . . . . .	??
Mutation.java . . . . .	??
MutationType.java . . . . .	??
OnePointCrossover.java . . . . .	??
OnePointMutation.java . . . . .	??
ProbabilisticSampling.java . . . . .	??
Probability.java . . . . .	??
Range.java . . . . .	??
Replace.java . . . . .	??
ReplaceType.java . . . . .	??
RouletteSelection.java . . . . .	??
Sampling.java . . . . .	??

SamplingType.java . . . . .	??
SelectionType.java . . . . .	??
SteadyStateReplace.java . . . . .	??
TowPointsMutation.java . . . . .	??
TruncationSelection.java . . . . .	??
UniformCrossover.java . . . . .	??
Univariate.java . . . . .	??
factory_interface . . . . .	36
IFFactoryAcceptCandidate.java . . . . .	??
IFFactoryCandidate.java . . . . .	??
IFFactoryCrossover.java . . . . .	??
IFFactoryDistribution.java . . . . .	??
IFFactoryFatherSelection.java . . . . .	??
IFFactoryGenerator.java . . . . .	??
IFFactoryMutation.java . . . . .	??
IFFactoryReplace.java . . . . .	??
IFFactorySolutionMethod.java . . . . .	??
IFFSampling.java . . . . .	??
factory_method . . . . .	37
FactoryAcceptCandidate.java . . . . .	??
FactoryCandidate.java . . . . .	??
FactoryCrossover.java . . . . .	??
FactoryDistribution.java . . . . .	??
FactoryFatherSelection.java . . . . .	??
FactoryGenerator.java . . . . .	??
FactoryLoader.java . . . . .	??
FactoryMutation.java . . . . .	??
FactoryReplace.java . . . . .	??
FactorySampling.java . . . . .	??
FactorySolutionMethod.java . . . . .	??
local_search . . . . .	39
acceptation_type . . . . .	31
AcceptableCandidate.java . . . . .	??
AcceptAnyone.java . . . . .	??
AcceptBest.java . . . . .	??
AcceptMulticase.java . . . . .	??
AcceptNotBad.java . . . . .	??
AcceptNotBadT.java . . . . .	??
AcceptNotBadU.java . . . . .	??
AcceptNotDominated.java . . . . .	??
AcceptNotDominatedTabu.java . . . . .	??
AcceptType.java . . . . .	??
Dominance.java . . . . .	??
candidate_type . . . . .	32
CandidateType.java . . . . .	??
CandidateValue.java . . . . .	??
GreaterCandidate.java . . . . .	??
NotDominatedCandidate.java . . . . .	??
RandomCandidate.java . . . . .	??
SearchCandidate.java . . . . .	??
SmallerCandidate.java . . . . .	??
complement . . . . .	33
StopExecute.java . . . . .	??
StrategyType.java . . . . .	??
TabuSolutions.java . . . . .	??
UpdateParameter.java . . . . .	??
metaheuristics . . . . .	40
strategy . . . . .	42

Strategy.java	??
metaheuristics	40
generators	37
DistributionEstimationAlgorithm.java	??
EvolutionStrategies.java	??
Generator.java	??
GeneratorType.java	??
GeneticAlgorithm.java	??
HillClimbing.java	??
HillClimbingRestart.java	??
InstanceDE.java	??
InstanceEE.java	??
InstanceGA.java	??
LimitRoulette.java	??
LimitThreshold.java	??
MultiCaseSimulatedAnnealing.java	??
MultiGenerator.java	??
MultiobjectiveHillClimbingDistance.java	??
MultiobjectiveHillClimbingRestart.java	??
MultiobjectiveStochasticHillClimbing.java	??
MultiobjectiveTabuSearch.java	??
Particle.java	??
ParticleSwarmOptimization.java	??
RandomSearch.java	??
SimulatedAnnealing.java	??
TabuSearch.java	??
problem	41
definition	34
Codification.java	??
ObjetiveFunction.java	??
Operator.java	??
Problem.java	??
State.java	??
extension	36
FactoresPonderados.java	??
MetricasMultiobjetivo.java	??
MultiObjetivoPuro.java	??
SolutionMethod.java	??
TypeSolutionMethod.java	??
problem_operators	41
MutationOperator.java	??
metaheuristics	40
strategy	42
Strategy.java	??
metaheuristics	40
generators	37
DistributionEstimationAlgorithm.java	??
EvolutionStrategies.java	??
Generator.java	??
GeneratorType.java	??
GeneticAlgorithm.java	??
HillClimbing.java	??
HillClimbingRestart.java	??
InstanceDE.java	??
InstanceEE.java	??
InstanceGA.java	??
LimitRoulette.java	??

LimitThreshold.java . . . . .	??
MultiCaseSimulatedAnnealing.java . . . . .	??
MultiGenerator.java . . . . .	??
MultiobjectiveHillClimbingDistance.java . . . . .	??
MultiobjectiveHillClimbingRestart.java . . . . .	??
MultiobjectiveStochasticHillClimbing.java . . . . .	??
MultiobjectiveTabuSearch.java . . . . .	??
Particle.java . . . . .	??
ParticleSwarmOptimization.java . . . . .	??
RandomSearch.java . . . . .	??
SimulatedAnnealing.java . . . . .	??
TabuSearch.java . . . . .	??
practica5 . . . . .	40
src . . . . .	42
main . . . . .	39
java . . . . .	38
config . . . . .	34
tspDynamic . . . . .	43
TSPState.java . . . . .	??
es . . . . .	35
ull . . . . .	43
App.java . . . . .	??
evolutionary_algorithms . . . . .	35
complement . . . . .	32
AIOMutation.java . . . . .	??
Crossover.java . . . . .	??
CrossoverType.java . . . . .	??
Distribution.java . . . . .	??
DistributionType.java . . . . .	??
FatherSelection.java . . . . .	??
GenerationalReplace.java . . . . .	??
Mutation.java . . . . .	??
MutationType.java . . . . .	??
OnePointCrossover.java . . . . .	??
OnePointMutation.java . . . . .	??
ProbabilisticSampling.java . . . . .	??
Probability.java . . . . .	??
Range.java . . . . .	??
Replace.java . . . . .	??
ReplaceType.java . . . . .	??
RouletteSelection.java . . . . .	??
Sampling.java . . . . .	??
SamplingType.java . . . . .	??
SelectionType.java . . . . .	??
SteadyStateReplace.java . . . . .	??
TowPointsMutation.java . . . . .	??
TruncationSelection.java . . . . .	??
UniformCrossover.java . . . . .	??
Univariate.java . . . . .	??
factory_interface . . . . .	36
IFFactoryAcceptCandidate.java . . . . .	??
IFFactoryCandidate.java . . . . .	??
IFFactoryCrossover.java . . . . .	??
IFFactoryDistribution.java . . . . .	??
IFFactoryFatherSelection.java . . . . .	??
IFFactoryGenerator.java . . . . .	??
IFFactoryMutation.java . . . . .	??
IFFactoryReplace.java . . . . .	??

IFFFactorySolutionMethod.java	??
IFFSampling.java	??
factory_method	37
FactoryAcceptCandidate.java	??
FactoryCandidate.java	??
FactoryCrossover.java	??
FactoryDistribution.java	??
FactoryFatherSelection.java	??
FactoryGenerator.java	??
FactoryLoader.java	??
FactoryMutation.java	??
FactoryReplace.java	??
FactorySampling.java	??
FactorySolutionMethod.java	??
local_search	39
acceptation_type	31
AcceptableCandidate.java	??
AcceptAnyone.java	??
AcceptBest.java	??
AcceptMulticase.java	??
AcceptNotBad.java	??
AcceptNotBadT.java	??
AcceptNotBadU.java	??
AcceptNotDominated.java	??
AcceptNotDominatedTabu.java	??
AcceptType.java	??
Dominance.java	??
candidate_type	32
CandidateType.java	??
CandidateValue.java	??
GreaterCandidate.java	??
NotDominatedCandidate.java	??
RandomCandidate.java	??
SearchCandidate.java	??
SmallerCandidate.java	??
complement	33
StopExecute.java	??
StrategyType.java	??
TabuSolutions.java	??
UpdateParameter.java	??
metaheuristics	40
strategy	42
Strategy.java	??
metaheuristics	40
generators	37
DistributionEstimationAlgorithm.java	??
EvolutionStrategies.java	??
Generator.java	??
GeneratorType.java	??
GeneticAlgorithm.java	??
HillClimbing.java	??
HillClimbingRestart.java	??
InstanceDE.java	??
InstanceEE.java	??
InstanceGA.java	??
LimitRoulette.java	??
LimitThreshold.java	??
MultiCaseSimulatedAnnealing.java	??

MultiGenerator.java . . . . .	??
MultiobjectiveHillClimbingDistance.java . . . . .	??
MultiobjectiveHillClimbingRestart.java . . . . .	??
MultiobjectiveStochasticHillClimbing.java . . . . .	??
MultiobjectiveTabuSearch.java . . . . .	??
Particle.java . . . . .	??
ParticleSwarmOptimization.java . . . . .	??
RandomSearch.java . . . . .	??
SimulatedAnnealing.java . . . . .	??
TabuSearch.java . . . . .	??
problem . . . . .	41
definition . . . . .	34
Codification.java . . . . .	??
ObjetiveFunction.java . . . . .	??
Operator.java . . . . .	??
Problem.java . . . . .	??
State.java . . . . .	??
extension . . . . .	36
FactoresPonderados.java . . . . .	??
MetricasMultiobjetivo.java . . . . .	??
MultiObjetivoPuro.java . . . . .	??
SolutionMethod.java . . . . .	??
TypeSolutionMethod.java . . . . .	??
problem_operators . . . . .	41
MutationOperator.java . . . . .	??
problem . . . . .	41
definition . . . . .	34
Codification.java . . . . .	??
ObjetiveFunction.java . . . . .	??
Operator.java . . . . .	??
Problem.java . . . . .	??
State.java . . . . .	??
extension . . . . .	36
FactoresPonderados.java . . . . .	??
MetricasMultiobjetivo.java . . . . .	??
MultiObjetivoPuro.java . . . . .	??
SolutionMethod.java . . . . .	??
TypeSolutionMethod.java . . . . .	??
problem_operators . . . . .	41
MutationOperator.java . . . . .	??
src . . . . .	42
main . . . . .	39
java . . . . .	38
config . . . . .	34
tspDynamic . . . . .	43
TSPState.java . . . . .	??
es . . . . .	35
ull . . . . .	43
App.java . . . . .	??
evolutionary_algorithms . . . . .	35
complement . . . . .	32
AIOMutation.java . . . . .	??
Crossover.java . . . . .	??
CrossoverType.java . . . . .	??
Distribution.java . . . . .	??
DistributionType.java . . . . .	??
FatherSelection.java . . . . .	??

GenerationalReplace.java . . . . .	??
Mutation.java . . . . .	??
MutationType.java . . . . .	??
OnePointCrossover.java . . . . .	??
OnePointMutation.java . . . . .	??
ProbabilisticSampling.java . . . . .	??
Probability.java . . . . .	??
Range.java . . . . .	??
Replace.java . . . . .	??
ReplaceType.java . . . . .	??
RouletteSelection.java . . . . .	??
Sampling.java . . . . .	??
SamplingType.java . . . . .	??
SelectionType.java . . . . .	??
SteadyStateReplace.java . . . . .	??
TowPointsMutation.java . . . . .	??
TruncationSelection.java . . . . .	??
UniformCrossover.java . . . . .	??
Univariate.java . . . . .	??
factory_interface . . . . .	36
IFFactoryAcceptCandidate.java . . . . .	??
IFFactoryCandidate.java . . . . .	??
IFFactoryCrossover.java . . . . .	??
IFFactoryDistribution.java . . . . .	??
IFFactoryFatherSelection.java . . . . .	??
IFFactoryGenerator.java . . . . .	??
IFFactoryMutation.java . . . . .	??
IFFactoryReplace.java . . . . .	??
IFFactorySolutionMethod.java . . . . .	??
IFFSampling.java . . . . .	??
factory_method . . . . .	37
FactoryAcceptCandidate.java . . . . .	??
FactoryCandidate.java . . . . .	??
FactoryCrossover.java . . . . .	??
FactoryDistribution.java . . . . .	??
FactoryFatherSelection.java . . . . .	??
FactoryGenerator.java . . . . .	??
FactoryLoader.java . . . . .	??
FactoryMutation.java . . . . .	??
FactoryReplace.java . . . . .	??
FactorySampling.java . . . . .	??
FactorySolutionMethod.java . . . . .	??
local_search . . . . .	39
acceptation_type . . . . .	31
AcceptableCandidate.java . . . . .	??
AcceptAnyone.java . . . . .	??
AcceptBest.java . . . . .	??
AcceptMulticase.java . . . . .	??
AcceptNotBad.java . . . . .	??
AcceptNotBadT.java . . . . .	??
AcceptNotBadU.java . . . . .	??
AcceptNotDominated.java . . . . .	??
AcceptNotDominatedTabu.java . . . . .	??
AcceptType.java . . . . .	??
Dominance.java . . . . .	??
candidate_type . . . . .	32
CandidateType.java . . . . .	??
CandidateValue.java . . . . .	??

GreaterCandidate.java . . . . .	??
NotDominatedCandidate.java . . . . .	??
RandomCandidate.java . . . . .	??
SearchCandidate.java . . . . .	??
SmallerCandidate.java . . . . .	??
complement . . . . .	33
StopExecute.java . . . . .	??
StrategyType.java . . . . .	??
TabuSolutions.java . . . . .	??
UpdateParameter.java . . . . .	??
metaheuristics . . . . .	40
strategy . . . . .	42
Strategy.java . . . . .	??
metaheuristics . . . . .	40
generators . . . . .	37
DistributionEstimationAlgorithm.java . . . . .	??
EvolutionStrategies.java . . . . .	??
Generator.java . . . . .	??
GeneratorType.java . . . . .	??
GeneticAlgorithm.java . . . . .	??
HillClimbing.java . . . . .	??
HillClimbingRestart.java . . . . .	??
InstanceDE.java . . . . .	??
InstanceEE.java . . . . .	??
InstanceGA.java . . . . .	??
LimitRoulette.java . . . . .	??
LimitThreshold.java . . . . .	??
MultiCaseSimulatedAnnealing.java . . . . .	??
MultiGenerator.java . . . . .	??
MultiobjectiveHillClimbingDistance.java . . . . .	??
MultiobjectiveHillClimbingRestart.java . . . . .	??
MultiobjectiveStochasticHillClimbing.java . . . . .	??
MultiobjectiveTabuSearch.java . . . . .	??
Particle.java . . . . .	??
ParticleSwarmOptimization.java . . . . .	??
RandomSearch.java . . . . .	??
SimulatedAnnealing.java . . . . .	??
TabuSearch.java . . . . .	??
problem . . . . .	41
definition . . . . .	34
Codification.java . . . . .	??
ObjetiveFunction.java . . . . .	??
Operator.java . . . . .	??
Problem.java . . . . .	??
State.java . . . . .	??
extension . . . . .	36
FactoresPonderados.java . . . . .	??
MetricasMultiobjetivo.java . . . . .	??
MultiObjetivoPuro.java . . . . .	??
SolutionMethod.java . . . . .	??
TypeSolutionMethod.java . . . . .	??
problem_operators . . . . .	41
MutationOperator.java . . . . .	??
strategy . . . . .	42
Strategy.java . . . . .	??
tspDynamic . . . . .	43
TSPState.java . . . . .	??

ull	.....	43
App.java	.....	??

## Chapter 2

# Índice de espacios de nombres

### 2.1 Lista de espacios de nombres

Lista de los espacios de nombres documentados, con breves descripciones:

config.tspDynamic . . . . .	45
es.ull . . . . .	45
evolutionary_algorithms.complement . . . . .	45
factory_interface	
@(#) IFFactoryAcceptCandidate.java . . . . .	46
factory_method	
@(#) FactoryAcceptCandidate.java . . . . .	47
local_search.acceptation_type	
@(#) AcceptAnyone.java . . . . .	48
local_search.candidate_type	
@(#) TypeCandidate.java . . . . .	49
local_search.complement	
@(#) Strategy.java . . . . .	50
metaheuristics.strategy . . . . .	50
metaheuristics.generators . . . . .	50
problem.definition . . . . .	51
problem.extension . . . . .	52
problem_operators . . . . .	52



# Chapter 3

## Índice jerárquico

### 3.1 Jerarquía de clases

Este listado de herencia está ordenado de forma general pero no está en orden alfabético estricto:

local_search.acceptation_type.AcceptableCandidate . . . . .	53
local_search.acceptation_type.AcceptAnyone . . . . .	55
local_search.acceptation_type.AcceptBest . . . . .	58
local_search.acceptation_type.AcceptMulticase . . . . .	60
local_search.acceptation_type.AcceptNotBad . . . . .	65
local_search.acceptation_type.AcceptNotBadT . . . . .	67
local_search.acceptation_type.AcceptNotBadU . . . . .	70
local_search.acceptation_type.AcceptNotDominated . . . . .	72
local_search.acceptation_type.AcceptNotDominatedTabu . . . . .	75
local_search.acceptation_type.AcceptType . . . . .	77
es.ull.App . . . . .	85
local_search.candidate_type.CandidateType . . . . .	86
local_search.candidate_type.CandidateValue . . . . .	87
Cloneable	
metaheuristics.generators.MultiGenerator . . . . .	289
problem.definition.State . . . . .	??
problem.definition.Codification . . . . .	92
evolutionary_algorithms.complement.Crossover . . . . .	95
evolutionary_algorithms.complement.OnePointCrossover . . . . .	388
evolutionary_algorithms.complement.UniformCrossover . . . . .	??
evolutionary_algorithms.complement.CrossoverType . . . . .	96
evolutionary_algorithms.complement.Distribution . . . . .	98
evolutionary_algorithms.complement.Univariate . . . . .	??
evolutionary_algorithms.complement.DistributionType . . . . .	118
local_search.acceptation_type.Dominance . . . . .	119
factory_method.FactoryLoader . . . . .	156
evolutionary_algorithms.complement.FatherSelection . . . . .	167
evolutionary_algorithms.complement.RouletteSelection . . . . .	??
evolutionary_algorithms.complement.TruncationSelection . . . . .	??
metaheuristics.generators.Generator . . . . .	171
metaheuristics.generators.DistributionEstimationAlgorithm . . . . .	99
metaheuristics.generators.EvolutionStrategies . . . . .	123
metaheuristics.generators.GeneticAlgorithm . . . . .	182
metaheuristics.generators.HillClimbing . . . . .	200

metaheuristics.generators.HillClimbingRestart . . . . .	213
metaheuristics.generators.LimitThreshold . . . . .	255
metaheuristics.generators.MultiCaseSimulatedAnnealing . . . . .	274
metaheuristics.generators.MultiGenerator . . . . .	289
metaheuristics.generators.MutiobjectiveHillClimbingDistance . . . . .	316
metaheuristics.generators.MutiobjectiveHillClimbingRestart . . . . .	332
metaheuristics.generators.MutiobjectiveStochasticHillClimbing . . . . .	347
metaheuristics.generators.MutiobjectiveTabuSearch . . . . .	360
metaheuristics.generators.Particle . . . . .	396
metaheuristics.generators.ParticleSwarmOptimization . . . . .	??
metaheuristics.generators.RandomSearch . . . . .	??
metaheuristics.generators.SimulatedAnnealing . . . . .	??
metaheuristics.generators.TabuSearch . . . . .	??
metaheuristics.generators.GeneratorType . . . . .	178
factory_interface.IFFactoryAcceptCandidate . . . . .	227
factory_method.FactoryAcceptCandidate . . . . .	141
factory_interface.IFFactoryCandidate . . . . .	228
factory_method.FactoryCandidate . . . . .	144
factory_interface.IFFactoryCrossover . . . . .	229
factory_method.FactoryCrossover . . . . .	146
factory_interface.IFFactoryDistribution . . . . .	231
factory_method.FactoryDistribution . . . . .	148
factory_interface.IFFactoryFatherSelection . . . . .	232
factory_method.FactoryFatherSelection . . . . .	150
factory_interface.IFFactoryGenerator . . . . .	233
factory_method.FactoryGenerator . . . . .	153
factory_interface.IFFactoryMutation . . . . .	235
factory_method.FactoryMutation . . . . .	158
factory_interface.IFFactoryReplace . . . . .	236
factory_method.FactoryReplace . . . . .	160
factory_interface.IFFactorySolutionMethod . . . . .	237
factory_method.FactorySolutionMethod . . . . .	164
factory_interface.IFFSampling . . . . .	239
factory_method.FactorySampling . . . . .	162
metaheuristics.generators.LimitRoulette . . . . .	250
problem.extension.MetricasMultiobjetivo . . . . .	268
evolutionary_algorithms.complement.Mutation . . . . .	375
evolutionary_algorithms.complement.AIOMutation . . . . .	80
evolutionary_algorithms.complement.OnePointMutation . . . . .	391
evolutionary_algorithms.complement.TowPointsMutation . . . . .	??
evolutionary_algorithms.complement.MutationType . . . . .	380
problem.definition.ObjetiveFunction . . . . .	384
problem.definition.Operator . . . . .	393
problem_operators.MutationOperator . . . . .	377
evolutionary_algorithms.complement.Probability . . . . .	??
problem.definition.Problem . . . . .	??
problem.definition.Problem.ProblemType . . . . .	??
evolutionary_algorithms.complement.Range . . . . .	??
evolutionary_algorithms.complement.Replace . . . . .	??
evolutionary_algorithms.complement.GenerationalReplace . . . . .	168
evolutionary_algorithms.complement.SteadyStateReplace . . . . .	??
evolutionary_algorithms.complement.ReplaceType . . . . .	??
Runnable	
metaheuristics.generators.InstanceDE . . . . .	240

metaheuristics.generators.InstanceEE . . . . .	244
metaheuristics.generators.InstanceGA . . . . .	247
evolutionary_algorithms.complement.Sampling . . . . .	??
evolutionary_algorithms.complement.ProbabilisticSampling . . . . .	??
evolutionary_algorithms.complement.SamplingType . . . . .	??
local_search.candidate_type.SearchCandidate . . . . .	??
local_search.candidate_type.GreaterCandidate . . . . .	198
local_search.candidate_type.NotDominatedCandidate . . . . .	382
local_search.candidate_type.RandomCandidate . . . . .	??
local_search.candidate_type.SmallerCandidate . . . . .	??
evolutionary_algorithms.complement.SelectionType . . . . .	??
problem.extension.SolutionMethod . . . . .	??
problem.extension.FactoresPonderados . . . . .	139
problem.extension.MultiObjetivoPuro . . . . .	373
local_search.complement.StopExecute . . . . .	??
metaheuristics.strategy.Strategy . . . . .	??
local_search.complement.StrATEGYType . . . . .	??
local_search.complement.TabuSolutions . . . . .	??
config.tspDynamic.TSPState . . . . .	??
problem.extension.TypeSolutionMethod . . . . .	??
local_search.complement.UpdateParameter . . . . .	??



# Chapter 4

## Índice de clases

### 4.1 Lista de clases

Lista de clases, estructuras, uniones e interfaces con breves descripciones:

local_search.acception_type.AcceptableCandidate	AcceptableCandidate - Base type for acception strategies . . . . .	53
local_search.acception_type.AcceptAnyone	AcceptAnyone - Accept any candidate solution . . . . .	55
local_search.acception_type.AcceptBest	AcceptBest - Accept the candidate only if it improves (or ties) the current state . . . . .	58
local_search.acception_type.AcceptMulticase	AcceptMulticase - Multi-case acception strategy for multiobjective algorithms . . . . .	60
local_search.acception_type.AcceptNotBad	AcceptNotBad - Accept candidate if it is not worse than the current state . . . . .	65
local_search.acception_type.AcceptNotBadT	AcceptNotBadT - Temperature based acception (simulated annealing style) . . . . .	67
local_search.acception_type.AcceptNotBadU	AcceptNotBadU - Threshold based acception strategy . . . . .	70
local_search.acception_type.AcceptNotDominated	AcceptNotDominated - Accept candidate if it is not dominated by current non-dominated set . . . . .	72
local_search.acception_type.AcceptNotDominatedTabu	AcceptNotDominatedTabu - Acception that integrates a tabu-like Pareto list . . . . .	75
local_search.acception_type.AcceptType	AcceptType - Enumeration of the available acception strategies . . . . .	77
evolutionary_algorithms.complement.AIOMutation	AIOMutation - applies the AIOMutation to the state . . . . .	80
es.uji.App	Hello world! . . . . .	85
local_search.candidate_type.CandidateType	CandidateType - enumeration of available candidate selection strategies . . . . .	86
local_search.candidate_type.CandidateValue	CandidateValue - Factory/wrapper for creating and using SearchCandidate selection strategies . . . . .	87
problem.definition.Codification	Codification . . . . .	92
evolutionary_algorithms.complement.Crossover	Crossover - applies the crossover operation to two parent states . . . . .	95
evolutionary_algorithms.complement.CrossoverType	CrossoverType - descripcion (añade detalles) . . . . .	96
evolutionary_algorithms.complement.Distribution	Distribution - applies a probability distribution to a set of fathers . . . . .	98

metaheuristics.generators.DistributionEstimationAlgorithm	
DistributionEstimationAlgorithm - class for distribution estimation algorithms . . . . .	99
evolutionary_algorithms.complement.DistributionType	
DistributionType - descripción (añade detalles) . . . . .	118
local_search.acceptation_type.Dominance	
Dominance - Utilities for Pareto dominance comparisons . . . . .	119
metaheuristics.generators.EvolutionStrategies	
EvolutionStrategies - class that implements the Evolution Strategies generator . . . . .	123
problem.extension.FactoresPonderados	
FactoresPonderados . . . . .	139
factory_method.FactoryAcceptCandidate	
FactoryAcceptCandidate - Interface for creating acceptable candidates . . . . .	141
factory_method.FactoryCandidate	
FactoryCandidate - Interface for creating search candidates . . . . .	144
factory_method.FactoryCrossover	
FactoryCrossover - Interface for creating crossover strategies . . . . .	146
factory_method.FactoryDistribution	
FactoryDistribution - Interface for creating distribution strategies . . . . .	148
factory_method.FactoryFatherSelection	
FactoryFatherSelection - Interface for creating father selection strategies . . . . .	150
factory_method.FactoryGenerator	
FactoryGenerator - Interface for creating generator strategies . . . . .	153
factory_method.FactoryLoader	
FactoryLoader - Class responsible for loading factory classes . . . . .	156
factory_method.FactoryMutation	
FactoryMutation - Class responsible for creating mutation strategies . . . . .	158
factory_method.FactoryReplace	
FactoryReplace - Class responsible for creating replacement strategies . . . . .	160
factory_method.FactorySampling	
FactorySampling - Class responsible for creating sampling strategies . . . . .	162
factory_method.FactorySolutionMethod	
FactorySolutionMethod - Class responsible for creating solution method strategies . . . . .	164
evolutionary_algorithms.complement.FatherSelection	
FatherSelection - selects the best fathers from the population . . . . .	167
evolutionary_algorithms.complement.GenerationalReplace	
GenerationalReplace - replaces the worst individual in the population . . . . .	168
metaheuristics.generators.Generator	
Generator - abstract base class for solution generators in metaheuristic algorithms . . . . .	171
metaheuristics.generators.GeneratorType	
GeneratorType - enumeration of different types of solution generators used in metaheuristic algorithms . . . . .	178
metaheuristics.generators.GeneticAlgorithm	
GeneticAlgorithm - descripción (añade detalles) . . . . .	182
local_search.candidate_type.GreaterCandidate	
GreaterCandidate - choose the neighbor with the greatest evaluation . . . . .	198
metaheuristics.generators.HillClimbing	
HillClimbing - class that implements the Hill Climbing metaheuristic . . . . .	200
metaheuristics.generators.HillClimbingRestart	
HillClimbingRestart - class that implements the Hill Climbing Restart metaheuristic . . . . .	213
factory_interface.IFFactoryAcceptCandidate	
IFFactoryAcceptCandidate - Interface for creating acceptable candidates . . . . .	227
factory_interface.IFFactoryCandidate	
IFFactoryCandidate - Interface for creating search candidates . . . . .	228
factory_interface.IFFactoryCrossover	
IFFactoryCrossover - Interface for creating crossover operators . . . . .	229
factory_interface.IFFactoryDistribution	
IFFactoryDistribution - Interface for creating distribution strategies . . . . .	231

factory_interface.IFFactoryFatherSelection	
IFFactoryFatherSelection - Interface for creating father selection strategies . . . . .	232
factory_interface.IFFactoryGenerator	
IFFactoryGenerator - Interface for creating generator instances . . . . .	233
factory_interface.IFFactoryMutation	
IFFactoryMutation - Interface for creating mutation strategies . . . . .	235
factory_interface.IFFactoryReplace	
IFFactoryReplace - Interface for creating replacement strategies . . . . .	236
factory_interface.IFFactorySolutionMethod	
IFFactorySolutionMethod - Interface for creating solution methods . . . . .	237
factory_interface.IFFSampling	
IFFSampling - Interface for creating sampling strategies . . . . .	239
metaheuristics.generators.InstanceDE	
InstanceDE - class that implements the Runnable interface to create an instance of the Distribution Estimation Algorithm generator . . . . .	240
metaheuristics.generators.InstanceEE	
InstanceEE - class that implements the Runnable interface to create an instance of . . . . .	244
metaheuristics.generators.InstanceGA	
InstanceGA - class that implements the Runnable interface to create an instance of the Genetic Algorithm generator . . . . .	247
metaheuristics.generators.LimitRoulette	
LimitRoulette - class that implements the Limit Roulette structure . . . . .	250
metaheuristics.generators.LimitThreshold	
LimitThreshold - class that implements the Limit Threshold metaheuristic . . . . .	255
problem.extension.MetricasMultiobjetivo	
MetricasMultiobjetivo . . . . .	268
metaheuristics.generators.MultiCaseSimulatedAnnealing	
MultiCaseSimulatedAnnealing - class that implements the Multi-Case Simulated Annealing metaheuristic . . . . .	274
metaheuristics.generators.MultiGenerator	
MultiGenerator - class that implements the Multi-Generator metaheuristic . . . . .	289
metaheuristics.generators.MultiobjectiveHillClimbingDistance	
MultiobjectiveHillClimbingDistance - class that implements the Multiobjective Hill Climbing Distance metaheuristic . . . . .	316
metaheuristics.generators.MultiobjectiveHillClimbingRestart	
MultiobjectiveHillClimbingRestart - class that implements the Multiobjective Hill Climbing with Restart metaheuristic . . . . .	332
metaheuristics.generators.MultiobjectiveStochasticHillClimbing	
MultiobjectiveStochasticHillClimbing - class that implements the Multiobjective Stochastic Hill Climbing metaheuristic . . . . .	347
metaheuristics.generators.MultiobjectiveTabuSearch	
MultiobjectiveTabuSearch - class that implements the Multiobjective Tabu Search metaheuristic . . . . .	360
problem.extension.MultiObjetivoPuro	
MultiObjetivoPuro . . . . .	373
evolutionary_algorithms.complement.Mutation	
Mutation - applies a mutation to a given state . . . . .	375
problem_operators.MutationOperator	
MutationOperator - defines a mutation operator for generating new states . . . . .	377
evolutionary_algorithms.complement.MutationType	
MutationType - descripcion (añade detalles) . . . . .	380
local_search.candidate_type.NotDominatedCandidate	
NotDominatedCandidate - select a non-dominated neighbor for multi-objective . . . . .	382
problem.definition.ObjetiveFunction	
ObjetiveFunction . . . . .	384
evolutionary_algorithms.complement.OnePointCrossover	
OnePointCrossover - applies the one-point crossover operator . . . . .	388
evolutionary_algorithms.complement.OnePointMutation	
OnePointMutation - applies the one-point mutation operator . . . . .	391

problem.definition.Operator	
Operator	393
metaheuristics.generators.Particle	
Particle - class that represents a particle in the Particle Swarm Optimization algorithm	396
metaheuristics.generators.ParticleSwarmOptimization	
ParticleSwarmOptimization - class that implements the Particle Swarm Optimization metaheuristic	??
evolutionary_algorithms.complement.ProbabilisticSampling	
ProbabilisticSampling - applies probabilistic sampling to select individuals	??
evolutionary_algorithms.complement.Probability	
Probability - represents the probability of a certain event	??
problem.definition.Problem	
Problem	??
problem.definition.Problem.ProblemType	
ProblemType - descripción (añade detalles)	??
local_search.candidate_type.RandomCandidate	
RandomCandidate - select a neighbor at random	??
metaheuristics.generators.RandomSearch	
RandomSearch - class that implements the Random Search metaheuristic	??
evolutionary_algorithms.complement.Range	
Range - represents a range of values with associated probabilities	??
evolutionary_algorithms.complement.Replace	
Replace - applies replacement strategies for individuals in the population	??
evolutionary_algorithms.complement.ReplaceType	
ReplaceType - represents the type of replacement strategy	??
evolutionary_algorithms.complement.RouletteSelection	
RouletteSelection - applies roulette selection to choose parents	??
evolutionary_algorithms.complement.Sampling	
Sampling - represents a sampling strategy for selecting individuals	??
evolutionary_algorithms.complement.SamplingType	
SamplingType - represents the type of sampling strategy	??
local_search.candidate_type.SearchCandidate	
SearchCandidate - abstract base class for candidate selection strategies	??
evolutionary_algorithms.complement.SelectionType	
SelectionType - represents the type of selection strategy	??
metaheuristics.generators.SimulatedAnnealing	
SimulatedAnnealing - class that implements the Simulated Annealing metaheuristic	??
local_search.candidate_type.SmallerCandidate	
SmallerCandidate - choose the neighbor with the smallest evaluation	??
problem.extension.SolutionMethod	
SolutionMethod	??
problem.definition.State	
State	??
evolutionary_algorithms.complement.SteadyStateReplace	
SteadyStateReplace - applies steady-state replacement strategy	??
local_search.complement.StopExecute	
StopExecute - termination predicate for local search loops	??
metaheuristics.strategy.Strategy	
Strategy	??
local_search.complement.StrATEGYType	
StrategyType - enumeration of local search strategy modes	??
metaheuristics.generators.TabuSearch	
TabuSearch - class that implements the Tabu Search metaheuristic	??
local_search.complement.TabuSolutions	
TabuSolutions - helper that manages a tabu list and filters neighborhoods	??
evolutionary_algorithms.complement.TowPointsMutation	
TowPointsMutation - applies two-points mutation	??

<a href="#">evolutionary_algorithms.complement.TruncationSelection</a>		
TruncationSelection - applies truncation selection strategy	.....	??
<a href="#">config.tspDynamic.TSPState</a>		
TSPState - descripcion (añade detalles)	.....	??
<a href="#">problem.extension.TypeSolutionMethod</a>		
TypeSolutionMethod	.....	??
<a href="#">evolutionary_algorithms.complement.UniformCrossover</a>		
UniformCrossover - complements uniform crossover strategy	.....	??
<a href="#">evolutionary_algorithms.complement.Univariate</a>		
Univariate - applies univariate distribution strategy	.....	??
<a href="#">local_search.complement.UpdateParameter</a>		
UpdateParameter - utility that updates generator parameters during search	.....	??



# Chapter 5

## Índice de archivos

### 5.1 Lista de archivos

Lista de todos los archivos con breves descripciones:

TSPState.java . . . . .	??
App.java . . . . .	??
AIMutation.java . . . . .	??
Crossover.java . . . . .	??
CrossoverType.java . . . . .	??
Distribution.java . . . . .	??
DistributionType.java . . . . .	??
FatherSelection.java . . . . .	??
GenerationalReplace.java . . . . .	??
Mutation.java . . . . .	??
MutationType.java . . . . .	??
OnePointCrossover.java . . . . .	??
OnePointMutation.java . . . . .	??
ProbabilisticSampling.java . . . . .	??
Probability.java . . . . .	??
Range.java . . . . .	??
Replace.java . . . . .	??
ReplaceType.java . . . . .	??
RouletteSelection.java . . . . .	??
Sampling.java . . . . .	??
SamplingType.java . . . . .	??
SelectionType.java . . . . .	??
SteadyStateReplace.java . . . . .	??
TowPointsMutation.java . . . . .	??
TruncationSelection.java . . . . .	??
UniformCrossover.java . . . . .	??
Univariate.java . . . . .	??
IFFactoryAcceptCandidate.java . . . . .	??
IFFactoryCandidate.java . . . . .	??
IFFactoryCrossover.java . . . . .	??
IFFactoryDistribution.java . . . . .	??
IFFactoryFatherSelection.java . . . . .	??
IFFactoryGenerator.java . . . . .	??
IFFactoryMutation.java . . . . .	??
IFFactoryReplace.java . . . . .	??

IFFFactorySolutionMethod.java	??
IFFSampling.java	??
FactoryAcceptCandidate.java	??
FactoryCandidate.java	??
FactoryCrossover.java	??
FactoryDistribution.java	??
FactoryFatherSelection.java	??
FactoryGenerator.java	??
FactoryLoader.java	??
FactoryMutation.java	??
FactoryReplace.java	??
FactorySampling.java	??
FactorySolutionMethod.java	??
AcceptableCandidate.java	??
AcceptAnyone.java	??
AcceptBest.java	??
AcceptMulticase.java	??
AcceptNotBad.java	??
AcceptNotBadT.java	??
AcceptNotBadU.java	??
AcceptNotDominated.java	??
AcceptNotDominatedTabu.java	??
AcceptType.java	??
Dominance.java	??
CandidateType.java	??
CandidateValue.java	??
GreaterCandidate.java	??
NotDominatedCandidate.java	??
RandomCandidate.java	??
SearchCandidate.java	??
SmallerCandidate.java	??
StopExecute.java	??
StrategyType.java	??
TabuSolutions.java	??
UpdateParameter.java	??
Strategy.java	??
DistributionEstimationAlgorithm.java	??
EvolutionStrategies.java	??
Generator.java	??
GeneratorType.java	??
GeneticAlgorithm.java	??
HillClimbing.java	??
HillClimbingRestart.java	??
InstanceDE.java	??
InstanceEE.java	??
InstanceGA.java	??
LimitRoulette.java	??
LimitThreshold.java	??
MultiCaseSimulatedAnnealing.java	??
MultiGenerator.java	??
MultiobjectiveHillClimbingDistance.java	??
MultiobjectiveHillClimbingRestart.java	??
MultiobjectiveStochasticHillClimbing.java	??
MultiobjectiveTabuSearch.java	??
Particle.java	??
ParticleSwarmOptimization.java	??
RandomSearch.java	??
SimulatedAnnealing.java	??

TabuSearch.java	??
Codification.java	??
ObjetiveFunction.java	??
Operator.java	??
Problem.java	??
State.java	??
FactoresPonderados.java	??
MetricasMultiobjetivo.java	??
MultiObjetivoPuro.java	??
SolutionMethod.java	??
TypeSolutionMethod.java	??
MutationOperator.java	??

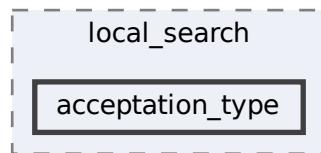


## Chapter 6

# Documentación de directorios

### 6.1 Referencia del directorio `acceptation_type`

Gráfico de dependencias de directorios de `acceptation_type`:



#### Archivos

- archivo `AcceptableCandidate.java`
- archivo `AcceptAnyone.java`
- archivo `AcceptBest.java`
- archivo `AcceptMulticase.java`
- archivo `AcceptNotBad.java`
- archivo `AcceptNotBadT.java`
- archivo `AcceptNotBadU.java`
- archivo `AcceptNotDominated.java`
- archivo `AcceptNotDominatedTabu.java`
- archivo `AcceptType.java`
- archivo `Dominance.java`

## 6.2 Referencia del directorio candidate\_type

Gráfico de dependencias de directorios de candidate\_type:

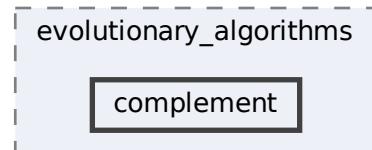


### Archivos

- archivo CandidateType.java
- archivo CandidateValue.java
- archivo GreaterCandidate.java
- archivo NotDominatedCandidate.java
- archivo RandomCandidate.java
- archivo SearchCandidate.java
- archivo SmallerCandidate.java

## 6.3 Referencia del directorio complement

Gráfico de dependencias de directorios de complement:



### Archivos

- archivo AIOMutation.java
- archivo Crossover.java
- archivo CrossoverType.java
- archivo Distribution.java
- archivo DistributionType.java
- archivo FatherSelection.java
- archivo GenerationalReplace.java
- archivo Mutation.java
- archivo MutationType.java
- archivo OnePointCrossover.java
- archivo OnePointMutation.java
- archivo ProbabilisticSampling.java
- archivo Probability.java
- archivo Range.java
- archivo Replace.java
- archivo ReplaceType.java
- archivo RouletteSelection.java
- archivo Sampling.java
- archivo SamplingType.java
- archivo SelectionType.java
- archivo SteadyStateReplace.java
- archivo TowPointsMutation.java
- archivo TruncationSelection.java
- archivo UniformCrossover.java
- archivo Univariate.java

## 6.4 Referencia del directorio complement

Gráfico de dependencias de directorios de complement:

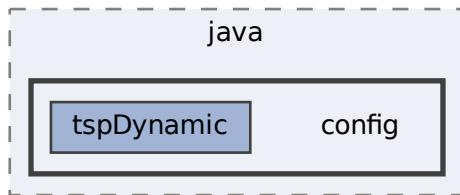


### Archivos

- archivo StopExecute.java
- archivo StrategyType.java
- archivo TabuSolutions.java
- archivo UpdateParameter.java

## 6.5 Referencia del directorio config

Gráfico de dependencias de directorios de config:

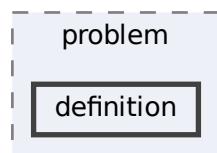


### Directorios

- directorio [tspDynamic](#)

## 6.6 Referencia del directorio definition

Gráfico de dependencias de directorios de definition:

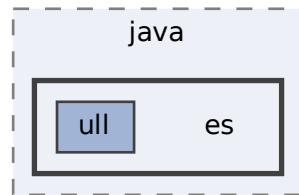


### Archivos

- archivo [Codification.java](#)
- archivo [ObjetiveFunction.java](#)
- archivo [Operator.java](#)
- archivo [Problem.java](#)
- archivo [State.java](#)

## 6.7 Referencia del directorio es

Gráfico de dependencias de directorios de es:

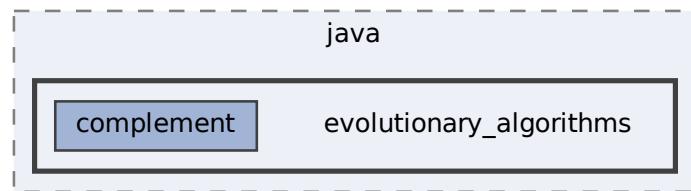


### Directories

- directorio [ull](#)

## 6.8 Referencia del directorio evolutionary\_algorithms

Gráfico de dependencias de directorios de evolutionary\_algorithms:

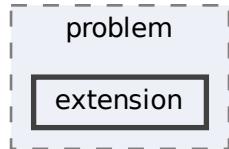


### Directories

- directorio [complement](#)

## 6.9 Referencia del directorio extension

Gráfico de dependencias de directorios de extension:



### Archivos

- archivo FactoresPonderados.java
- archivo MetricasMultiobjetivo.java
- archivo MultiObjetivoPuro.java
- archivo SolutionMethod.java
- archivo TypeSolutionMethod.java

## 6.10 Referencia del directorio factory\_interface

Gráfico de dependencias de directorios de factory\_interface:

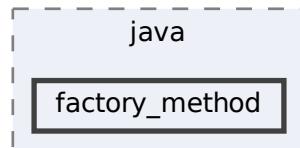


### Archivos

- archivo IFFactoryAcceptCandidate.java
- archivo IFFactoryCandidate.java
- archivo IFFactoryCrossover.java
- archivo IFFactoryDistribution.java
- archivo IFFactoryFatherSelection.java
- archivo IFFactoryGenerator.java
- archivo IFFactoryMutation.java
- archivo IFFactoryReplace.java
- archivo IFFactorySolutionMethod.java
- archivo IFFSampling.java

## 6.11 Referencia del directorio factory\_method

Gráfico de dependencias de directorios de factory\_method:

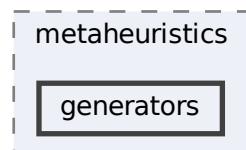


### Archivos

- archivo FactoryAcceptCandidate.java
- archivo FactoryCandidate.java
- archivo FactoryCrossover.java
- archivo FactoryDistribution.java
- archivo FactoryFatherSelection.java
- archivo FactoryGenerator.java
- archivo FactoryLoader.java
- archivo FactoryMutation.java
- archivo FactoryReplace.java
- archivo FactorySampling.java
- archivo FactorySolutionMethod.java

## 6.12 Referencia del directorio generators

Gráfico de dependencias de directorios de generators:



## Archivos

- archivo DistributionEstimationAlgorithm.java
- archivo EvolutionStrategies.java
- archivo Generator.java
- archivo GeneratorType.java
- archivo GeneticAlgorithm.java
- archivo HillClimbing.java
- archivo HillClimbingRestart.java
- archivo InstanceDE.java
- archivo InstanceEE.java
- archivo InstanceGA.java
- archivo LimitRoulette.java
- archivo LimitThreshold.java
- archivo MultiCaseSimulatedAnnealing.java
- archivo MultiGenerator.java
- archivo MultiobjectiveHillClimbingDistance.java
- archivo MultiobjectiveHillClimbingRestart.java
- archivo MultiobjectiveStochasticHillClimbing.java
- archivo MultiobjectiveTabuSearch.java
- archivo Particle.java
- archivo ParticleSwarmOptimization.java
- archivo RandomSearch.java
- archivo SimulatedAnnealing.java
- archivo TabuSearch.java

## 6.13 Referencia del directorio java

Gráfico de dependencias de directorios de java:

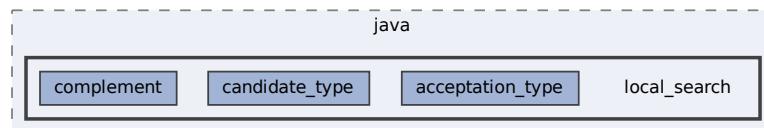


## Directorio

- directorio config
- directorio es
- directorio evolutionary\_algorithms
- directorio factory\_interface
- directorio factory\_method
- directorio local\_search
- directorio metaheuristics
- directorio metaheuristics
- directorio problem
- directorio problem\_operators

## 6.14 Referencia del directorio local\_search

Gráfico de dependencias de directorios de local\_search:

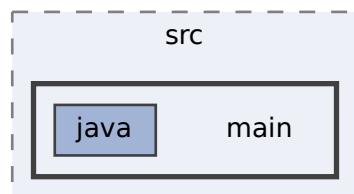


### Directarios

- directorio [acceptation\\_type](#)
- directorio [candidate\\_type](#)
- directorio [complement](#)

## 6.15 Referencia del directorio main

Gráfico de dependencias de directorios de main:

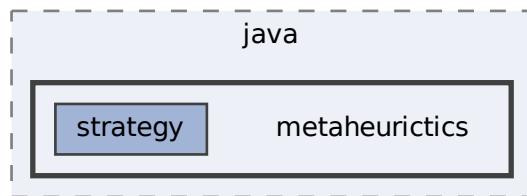


### Directarios

- directorio [java](#)

## 6.16 Referencia del directorio metaheurictics

Gráfico de dependencias de directorios de metaheurictics:

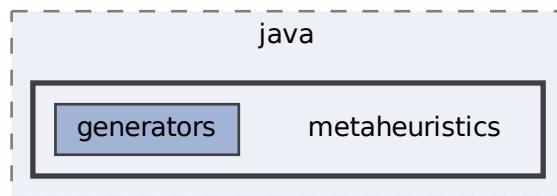


### Directarios

- directorio [strategy](#)

## 6.17 Referencia del directorio metaheuristics

Gráfico de dependencias de directorios de metaheuristics:



### Directarios

- directorio [generators](#)

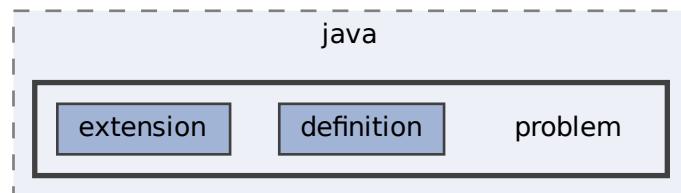
## 6.18 Referencia del directorio practica5

### Directarios

- directorio [src](#)

## 6.19 Referencia del directorio problem

Gráfico de dependencias de directorios de problem:

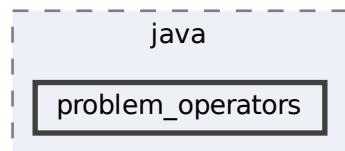


### Directarios

- directorio [definition](#)
- directorio [extension](#)

## 6.20 Referencia del directorio problem\_operators

Gráfico de dependencias de directorios de problem\_operators:

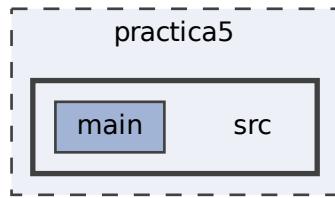


### Archivos

- archivo [MutationOperator.java](#)

## 6.21 Referencia del directorio src

Gráfico de dependencias de directorios de src:

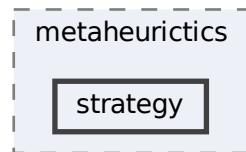


### Directarios

- directorio [main](#)

## 6.22 Referencia del directorio strategy

Gráfico de dependencias de directorios de strategy:

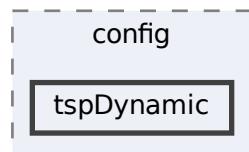


### Archivos

- archivo [Strategy.java](#)

## 6.23 Referencia del directorio `tspDynamic`

Gráfico de dependencias de directorios de `tspDynamic`:



### Archivos

- archivo [TSPState.java](#)

## 6.24 Referencia del directorio `ull`

Gráfico de dependencias de directorios de `ull`:



### Archivos

- archivo [App.java](#)



# Chapter 7

## Documentación de espacios de nombres

### 7.1 Paquete config.tspDynamic

#### Clases

- class [TSPState](#)

*TSPState* - descripción (añade detalles).

### 7.2 Paquete es.ull

#### Clases

- class [App](#)

*Hello world!*

### 7.3 Paquete evolutionary\_algorithms.complement

#### Clases

- class [AIOMutation](#)

*AIOMutation* - applies the [AIOMutation](#) to the state.

- class [Crossover](#)

*Crossover* - applies the crossover operation to two parent states.

- enum [CrossoverType](#)

*CrossoverType* - descripción (añade detalles).

- class [Distribution](#)

*Distribution* - applies a probability distribution to a set of fathers.

- enum [DistributionType](#)

*DistributionType* - descripción (añade detalles).

- class [FatherSelection](#)

*FatherSelection* - selects the best fathers from the population.

- class [GenerationalReplace](#)

*GenerationalReplace* - replaces the worst individual in the population.

- class [Mutation](#)  
*Mutation* - applies a mutation to a given state.
- enum [MutationType](#)  
*MutationType* - descripción (añade detalles).
- class [OnePointCrossover](#)  
*OnePointCrossover* - applies the one-point crossover operator.
- class [OnePointMutation](#)  
*OnePointMutation* - applies the one-point mutation operator.
- class [ProbabilisticSampling](#)  
*ProbabilisticSampling* - applies probabilistic sampling to select individuals.
- class [Probability](#)  
*Probability* - represents the probability of a certain event.
- class [Range](#)  
*Range* - represents a range of values with associated probabilities.
- class [Replace](#)  
*Replace* - applies replacement strategies for individuals in the population.
- enum [ReplaceType](#)  
*ReplaceType* - represents the type of replacement strategy.
- class [RouletteSelection](#)  
*RouletteSelection* - applies roulette selection to choose parents.
- class [Sampling](#)  
*Sampling* - represents a sampling strategy for selecting individuals.
- enum [SamplingType](#)  
*SamplingType* - represents the type of sampling strategy.
- enum [SelectionType](#)  
*SelectionType* - represents the type of selection strategy.
- class [SteadyStateReplace](#)  
*SteadyStateReplace* - applies steady-state replacement strategy.
- class [TowPointsMutation](#)  
*TowPointsMutation* - applies two-points mutation.
- class [TruncationSelection](#)  
*TruncationSelection* - applies truncation selection strategy.
- class [UniformCrossover](#)  
*UniformCrossover* - complements uniform crossover strategy.
- class [Univariate](#)  
*Univariate* - applies univariate distribution strategy.

## 7.4 Paquete factory\_interface

@(#)[IFFactoryAcceptCandidate.java](#)

## Clases

- interface [IFFactoryAcceptCandidate](#)  
*IFFactoryAcceptCandidate - Interface for creating acceptable candidates.*
- interface [IFFactoryCandidate](#)  
*IFFactoryCandidate - Interface for creating search candidates.*
- interface [IFFactoryCrossover](#)  
*IFFactoryCrossover - Interface for creating crossover operators.*
- interface [IFFactoryDistribution](#)  
*IFFactoryDistribution - Interface for creating distribution strategies.*
- interface [IFFactoryFatherSelection](#)  
*IFFactoryFatherSelection - Interface for creating father selection strategies.*
- interface [IFFactoryGenerator](#)  
*IFFactoryGenerator - Interface for creating generator instances.*
- interface [IFFactoryMutation](#)  
*IFFactoryMutation - Interface for creating mutation strategies.*
- interface [IFFactoryReplace](#)  
*IFFactoryReplace - Interface for creating replacement strategies.*
- interface [IFFactorySolutionMethod](#)  
*IFFactorySolutionMethod - Interface for creating solution methods.*
- interface [IFFSampling](#)  
*IFFSampling - Interface for creating sampling strategies.*

### 7.4.1 Descripción detallada

@(#)[IFFactoryAcceptCandidate.java](#)

@(#)[IFFactoryCandidate.java](#)

## 7.5 Paquete factory\_method

@(#)[FactoryAcceptCandidate.java](#)

## Clases

- class [FactoryAcceptCandidate](#)  
*FactoryAcceptCandidate - Interface for creating acceptable candidates.*
- class [FactoryCandidate](#)  
*FactoryCandidate - Interface for creating search candidates.*
- class [FactoryCrossover](#)  
*FactoryCrossover - Interface for creating crossover strategies.*
- class [FactoryDistribution](#)  
*FactoryDistribution - Interface for creating distribution strategies.*
- class [FactoryFatherSelection](#)  
*FactoryFatherSelection - Interface for creating father selection strategies.*
- class [FactoryGenerator](#)  
*FactoryGenerator - Interface for creating generator strategies.*
- class [FactoryLoader](#)

- class [FactoryLoader](#) - Class responsible for loading factory classes.
- class [FactoryMutation](#)  
*FactoryMutation* - Class responsible for creating mutation strategies.
- class [FactoryReplace](#)  
*FactoryReplace* - Class responsible for creating replacement strategies.
- class [FactorySampling](#)  
*FactorySampling* - Class responsible for creating sampling strategies.
- class [FactorySolutionMethod](#)  
*FactorySolutionMethod* - Class responsible for creating solution method strategies.

### 7.5.1 Descripción detallada

@(#) [FactoryAcceptCandidate.java](#)

@(#) [FactoryCandidate.java](#)

## 7.6 Paquete local\_search.acceptation\_type

@(#)[AcceptAnyone.java](#)

### Clases

- class [AcceptableCandidate](#)  
*AcceptableCandidate* - Base type for acceptation strategies.
- class [AcceptAnyone](#)  
*AcceptAnyone* - Accept any candidate solution.
- class [AcceptBest](#)  
*AcceptBest* - Accept the candidate only if it improves (or ties) the current state.
- class [AcceptMulticase](#)  
*AcceptMulticase* - Multi-case acceptation strategy for multiobjective algorithms.
- class [AcceptNotBad](#)  
*AcceptNotBad* - Accept candidate if it is not worse than the current state.
- class [AcceptNotBadT](#)  
*AcceptNotBadT* - Temperature based acceptation (simulated annealing style).
- class [AcceptNotBadU](#)  
*AcceptNotBadU* - Threshold based acceptation strategy.
- class [AcceptNotDominated](#)  
*AcceptNotDominated* - Accept candidate if it is not dominated by current non-dominated set.
- class [AcceptNotDominatedTabu](#)  
*AcceptNotDominatedTabu* - Acceptation that integrates a tabu-like Pareto list.
- enum [AcceptType](#)  
*AcceptType* - Enumeration of the available acceptation strategies.
- class [Dominance](#)  
*Dominance* - Utilities for Pareto dominance comparisons.

### 7.6.1 Descripción detallada

@(#) [AcceptAnyone.java](#)

@(#) [TypeAcceptation.java](#)

@(#) [AcceptNoBadU.java](#)

@(#) [AcceptNoBadT.java](#)

@(#) [AcceptNoBad.java](#)

@(#) [AcceptBest.java](#)

## 7.7 Paquete local\_search.candidate\_type

@(#) [TypeCandidate.java](#)

### Clases

- enum [CandidateType](#)

*CandidateType* - enumeration of available candidate selection strategies.

- class [CandidateValue](#)

*CandidateValue* - Factory/wrapper for creating and using [SearchCandidate](#) selection strategies.

- class [GreaterCandidate](#)

*GreaterCandidate* - choose the neighbor with the greatest evaluation.

- class [NotDominatedCandidate](#)

*NotDominatedCandidate* - select a non-dominated neighbor for multi-objective.

- class [RandomCandidate](#)

*RandomCandidate* - select a neighbor at random.

- class [SearchCandidate](#)

*SearchCandidate* - abstract base class for candidate selection strategies.

- class [SmallerCandidate](#)

*SmallerCandidate* - choose the neighbor with the smallest evaluation.

### 7.7.1 Descripción detallada

@(#)[TypeCandidate.java](#)

@(#)[SmallerCandidate.java](#)

@(#)[SearchCandidate.java](#)

@(#)[AleatoryCandidate.java](#)

@(#)[MajorCandidate.java](#)

@(#)[CandidateValue.java](#)

## 7.8 Paquete local\_search.complement

@(#)[Strategy.java](#)

### Clases

- class [StopExecute](#)  
*StopExecute* - termination predicate for local search loops.
- enum [StrategyType](#)  
*StrategyType* - enumeration of local search strategy modes.
- class [TabuSolutions](#)  
*TabuSolutions* - helper that manages a tabu list and filters neighborhoods.
- class [UpdateParameter](#)  
*UpdateParameter* - utility that updates generator parameters during search.

### 7.8.1 Descripción detallada

@(#)[Strategy.java](#)

## 7.9 Paquete metaheuristics.strategy

### Clases

- class [Strategy](#)  
*Strategy* -

## 7.10 Paquete metaheuristics.generators

### Clases

- class [DistributionEstimationAlgorithm](#)  
*DistributionEstimationAlgorithm* - class for distribution estimation algorithms.
- class [EvolutionStrategies](#)  
*EvolutionStrategies* - class that implements the Evolution Strategies generator.
- class [Generator](#)  
*Generator* - abstract base class for solution generators in metaheuristic algorithms.
- enum [GeneratorType](#)  
*GeneratorType* - enumeration of different types of solution generators used in metaheuristic algorithms.
- class [GeneticAlgorithm](#)  
*GeneticAlgorithm* - descripción (añade detalles).
- class [HillClimbing](#)  
*HillClimbing* - class that implements the Hill Climbing metaheuristic.
- class [HillClimbingRestart](#)  
*HillClimbingRestart* - class that implements the Hill Climbing Restart metaheuristic.
- class [InstanceDE](#)

- class [InstanceDE](#)  
*InstanceDE* - class that implements the Runnable interface to create an instance of the Distribution Estimation Algorithm generator.
- class [InstanceEE](#)  
*InstanceEE* - class that implements the Runnable interface to create an instance of.
- class [InstanceGA](#)  
*InstanceGA* - class that implements the Runnable interface to create an instance of the Genetic Algorithm generator.
- class [LimitRoulette](#)  
*LimitRoulette* - class that implements the Limit Roulette structure.
- class [LimitThreshold](#)  
*LimitThreshold* - class that implements the Limit Threshold metaheuristic.
- class [MultiCaseSimulatedAnnealing](#)  
*MultiCaseSimulatedAnnealing* - class that implements the Multi-Case Simulated Annealing metaheuristic.
- class [MultiGenerator](#)  
*MultiGenerator* - class that implements the Multi-Generator metaheuristic.
- class [MultiobjectiveHillClimbingDistance](#)  
*MultiobjectiveHillClimbingDistance* - class that implements the Multiobjective Hill Climbing Distance metaheuristic.
- class [MultiobjectiveHillClimbingRestart](#)  
*MultiobjectiveHillClimbingRestart* - class that implements the Multiobjective Hill Climbing with Restart metaheuristic.
- class [MultiobjectiveStochasticHillClimbing](#)  
*MultiobjectiveStochasticHillClimbing* - class that implements the Multiobjective Stochastic Hill Climbing metaheuristic.
- class [MultiobjectiveTabuSearch](#)  
*MultiobjectiveTabuSearch* - class that implements the Multiobjective Tabu Search metaheuristic.
- class [Particle](#)  
*Particle* - class that represents a particle in the [Particle](#) Swarm Optimization algorithm.
- class [ParticleSwarmOptimization](#)  
*ParticleSwarmOptimization* - class that implements the [Particle](#) Swarm Optimization metaheuristic.
- class [RandomSearch](#)  
*RandomSearch* - class that implements the Random Search metaheuristic.
- class [SimulatedAnnealing](#)  
*SimulatedAnnealing* - class that implements the Simulated Annealing metaheuristic.
- class [TabuSearch](#)  
*TabuSearch* - class that implements the Tabu Search metaheuristic.

## 7.11 Paquete problem.definition

### Clases

- class [Codification](#)  
*Codification.*
- class [ObjetiveFunction](#)  
*ObjetiveFunction.*
- class [Operator](#)  
*Operator.*
- class [Problem](#)  
*Problem.*
- class [State](#)  
*State.*

## 7.12 Paquete problem.extension

### Clases

- class FactoresPonderados  
*FactoresPonderados.*
- class MetricasMultiobjetivo  
*MetricasMultiobjetivo.*
- class MultiObjetivoPuro  
*MultiObjetivoPuro.*
- class SolutionMethod  
*SolutionMethod.*
- enum TypeSolutionMethod  
*TypeSolutionMethod.*

## 7.13 Paquete problem\_operators

### Clases

- class MutationOperator  
*MutationOperator* - defines a mutation operator for generating new states.

# Chapter 8

## Documentación de clases

### 8.1 Referencia de la clase

#### `local_search.acceptation_type.AcceptableCandidate`

`AcceptableCandidate` - Base type for acceptation strategies.

Diagrama de herencia de `local_search.acceptation_type.AcceptableCandidate`

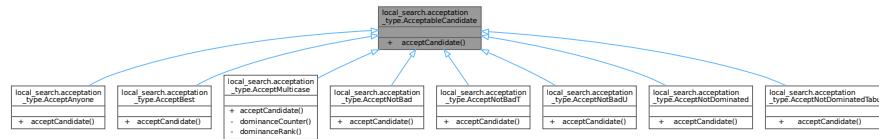
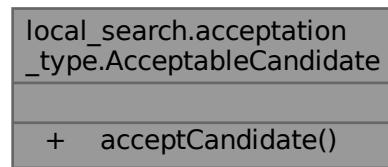


Diagrama de colaboración de `local_search.acceptation_type.AcceptableCandidate`:



### Métodos públicos

- abstract Boolean `acceptCandidate (State stateCurrent, State stateCandidate)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*Evaluate whether a candidate state should be accepted.*

## 8.1.1 Descripción detallada

[AcceptableCandidate](#) - Base type for acceptation strategies.

Abstract contract that all acceptation strategies must implement.

Definición en la línea 12 del archivo [AcceptableCandidate.java](#).

## 8.1.2 Documentación de funciones miembro

### 8.1.2.1 acceptCandidate()

```
abstract Boolean local_search.acceptation_type.AcceptableCandidate.acceptCandidate (
    State stateCurrent,
    State stateCandidate) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException [abstract]
```

Evaluate whether a candidate state should be accepted.

#### Parámetros

<code>stateCurrent</code>	the current state
<code>stateCandidate</code>	the candidate state

#### Devuelve

true if the candidate should be accepted, false otherwise

#### Excepciones

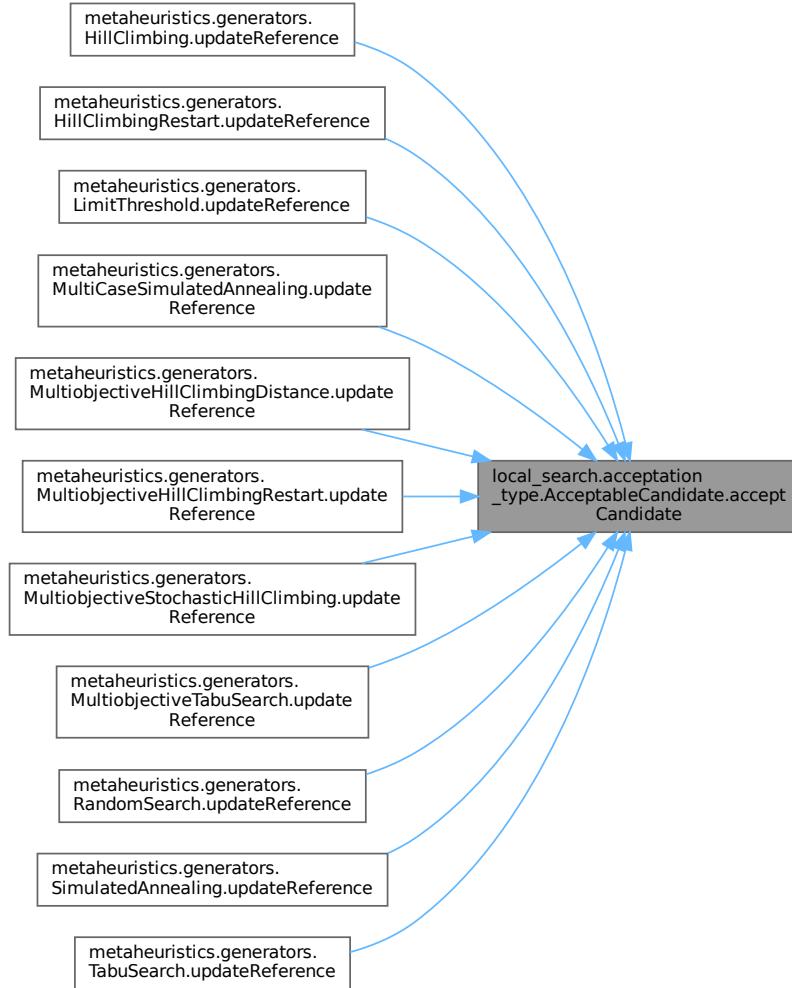
<code>IllegalArgumentException</code>	on invalid arguments
<code>SecurityException</code>	on reflective access problems
<code>ClassNotFoundException</code>	when a referenced class cannot be found
<code>InstantiationException</code>	on reflective instantiation failures
<code>IllegalAccessException</code>	on illegal reflective access
<code>InvocationTargetException</code>	when a reflective call throws
<code>NoSuchMethodException</code>	when a reflective method lookup fails

Reimplementado en [local\\_search.acceptation\\_type.AcceptAnyone](#), [local\\_search.acceptation\\_type.AcceptBest](#), [local\\_search.acceptation\\_type.AcceptMulticase](#), [local\\_search.acceptation\\_type.AcceptNotBad](#), [local\\_search.acceptation\\_type.AcceptNotBadU](#), [local\\_search.acceptation\\_type.AcceptNotDominated](#) y [local\\_search.acceptation\\_type.AcceptNotDominating](#).

Referenciado por [metaheuristics.generators.HillClimbing.updateReference\(\)](#), [metaheuristics.generators.HillClimbingRestart.updateReference\(\)](#), [metaheuristics.generators.LimitThreshold.updateReference\(\)](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing.updateReference\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance.updateReference\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingUpdateReference\(\)](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing.updateReference\(\)](#), [metaheuristics.generators.MultiobjectiveTabuSearch.updateReference\(\)](#).

`metaheuristics.generators.RandomSearch.updateReference()`, `metaheuristics.generators.SimulatedAnnealing.updateReference()` y `metaheuristics.generators.TabuSearch.updateReference()`.

Gráfico de llamadas a esta función:



La documentación de esta clase está generada del siguiente archivo:

- [AcceptableCandidate.java](#)

## 8.2 Referencia de la clase local\_search.acceptation\_type.AcceptAnyone

`AcceptAnyone` - Accept any candidate solution.

Diagrama de herencia de local\_search.acceptation\_type.AcceptAnyone

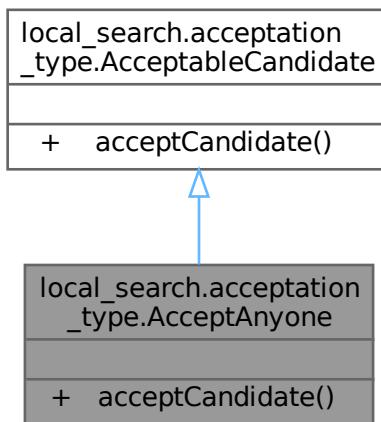
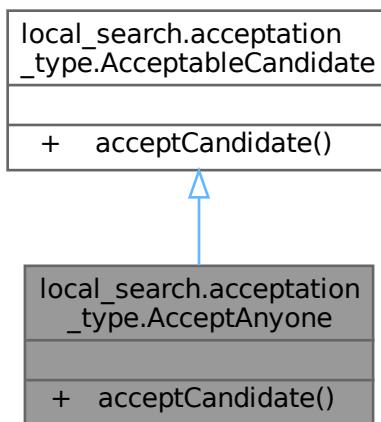


Diagrama de colaboración de local\_search.acceptation\_type.AcceptAnyone:



## Métodos públicos

- Boolean acceptCandidate ([State](#) stateCurrent, [State](#) stateCandidate)  
*Decide whether to accept the candidate state.*

## 8.2.1 Descripción detallada

[AcceptAnyone](#) - Accept any candidate solution.

Simple acceptation strategy that always accepts the candidate solution provided to the local search operator.

Definición en la línea 15 del archivo [AcceptAnyone.java](#).

## 8.2.2 Documentación de funciones miembro

### 8.2.2.1 acceptCandidate()

```
Boolean local_search.acceptation_type.AcceptAnyone.acceptCandidate (
    State stateCurrent,
    State stateCandidate) [inline]
```

Decide whether to accept the candidate state.

#### Parámetros

<i>stateCurrent</i>	the current state
<i>stateCandidate</i>	the candidate state to evaluate

#### Devuelve

true always (accept any candidate)

Reimplementado de [local\\_search.acceptation\\_type.AcceptableCandidate](#).

Definición en la línea 25 del archivo [AcceptAnyone.java](#).

```
00025
00026     Boolean accept = true;
00027     return accept;
00028 }
```

La documentación de esta clase está generada del siguiente archivo:

- [AcceptAnyone.java](#)

### 8.3 Referencia de la clase local\_search.acceptation\_type.AcceptBest

[AcceptBest](#) - Accept the candidate only if it improves (or ties) the current state.

Diagrama de herencia de local\_search.acceptation\_type.AcceptBest

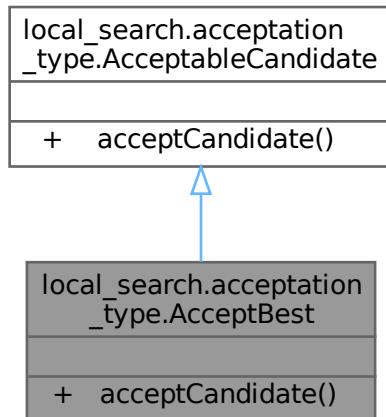
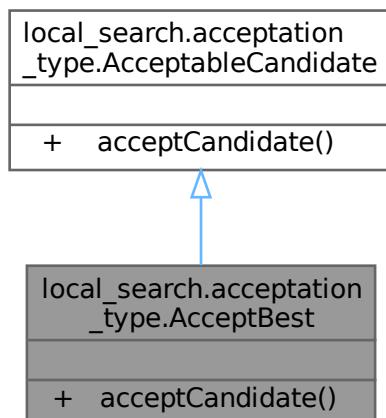


Diagrama de colaboración de local\_search.acceptation\_type.AcceptBest:



#### Métodos públicos

- Boolean `acceptCandidate (State stateCurrent, State stateCandidate)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*Evaluate and decide acceptance of a candidate state.*

### 8.3.1 Descripción detallada

[AcceptBest](#) - Accept the candidate only if it improves (or ties) the current state.

Accepts the candidate state when it is better according to the problem objective type (maximize/minimize).

Definición en la línea 20 del archivo [AcceptBest.java](#).

### 8.3.2 Documentación de funciones miembro

#### 8.3.2.1 acceptCandidate()

```
Boolean local_search.acceptation_type.AcceptBest.acceptCandidate (
    State stateCurrent,
    State stateCandidate) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

Evaluate and decide acceptance of a candidate state.

#### Parámetros

<i>stateCurrent</i>	current state
<i>stateCandidate</i>	candidate state to evaluate

#### Devuelve

true if candidate should be accepted according to the problem objective (maximization/minimization), false otherwise

#### Excepciones

<i>IllegalArgumentException</i>	when arguments are invalid
<i>SecurityException</i>	on reflective access problems
<i>ClassNotFoundException</i>	when a referenced class cannot be found
<i>InstantiationException</i>	when a class cannot be instantiated reflectively
<i>IllegalAccessException</i>	on illegal reflective access
<i>InvocationTargetException</i>	when a reflective call throws an exception
<i>NoSuchMethodException</i>	when a reflective method lookup fails

Reimplementado de [local\\_search.acceptation\\_type.AcceptableCandidate](#).

Definición en la línea 38 del archivo [AcceptBest.java](#).

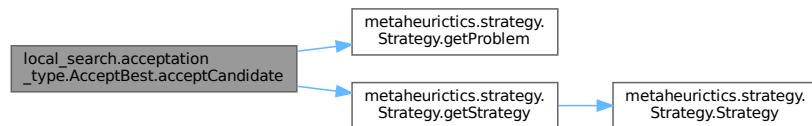
```
00038
{
00039     Boolean accept = null;
00040     Problem problem = Strategy.getStrategy().getProblem();
00041     if(problem.getTypeProblem().equals(ProblemType.Maximizar)) {
00042         if (stateCandidate.getEvaluation().get(0) >= stateCurrent.getEvaluation().get(0)) {
00043             accept = true;
00044         } else {
```

```

00045         accept = false;
00046     }
00047     } else {
00048         if (stateCandidate.getEvaluation().get(0) <= stateCurrent.getEvaluation().get(0)) {
00049             accept = true;
00050         } else {
00051             accept = false;
00052         }
00053     }
00054     return accept;
00055 }
```

Hace referencia a [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#) y [problem.definition.ProblemType.Maximizar](#).

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [AcceptBest.java](#)

## 8.4 Referencia de la clase

### **local\_search.acceptation\_type.AcceptMulticase**

[AcceptMulticase](#) - Multi-case acceptation strategy for multiobjective algorithms.

Diagrama de herencia de `local_search.acceptation_type.AcceptMulticase`

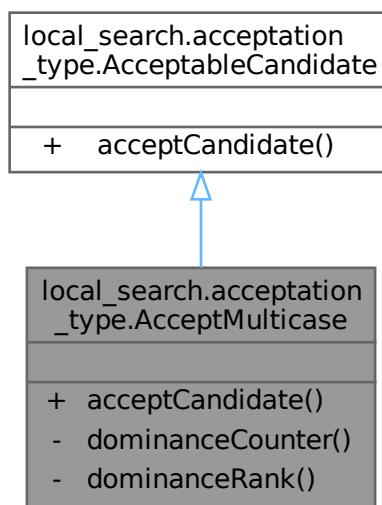
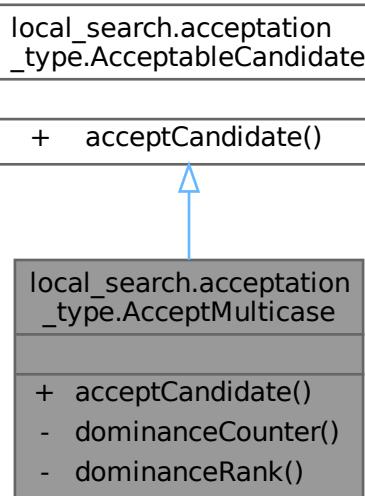


Diagrama de colaboración de local\_search.acceptation\_type.AcceptMulticase:



## Métodos públicos

- Boolean `acceptCandidate (State stateCurrent, State stateCandidate)`  
*Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.*

## Métodos privados

- int `dominanceCounter (State stateCandidate, List< State > list)`
- int `dominanceRank (State stateCandidate, List< State > list)`

### 8.4.1 Descripción detallada

`AcceptMulticase` - Multi-case acceptance strategy for multiobjective algorithms.

Decides whether to accept a candidate state based on dominance, rank and simulated annealing like probabilities.

Definición en la línea 17 del archivo [AcceptMulticase.java](#).

### 8.4.2 Documentación de funciones miembro

#### 8.4.2.1 `acceptCandidate()`

```
Boolean local_search.acceptation_type.AcceptMulticase.acceptCandidate (
    State stateCurrent,
    State stateCandidate) [inline]
```

Generado por Doxygen

Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.

This RNG is used to decide whether a mutation occurs in the evolutionary algorithm and is not used for security-sensitive purposes. Suppress Sonar security hotspot S2245 for this usage. Evaluate a candidate for acceptance

## Parámetros

<i>stateCurrent</i>	the current state
<i>stateCandidate</i>	the candidate state

## Devuelve

true if the candidate should be accepted, false otherwise

Reimplementado de [local\\_search.acceptation\\_type.AcceptableCandidate](#).

Definición en la línea 34 del archivo [AcceptMulticase.java](#).

```

00034
00035      // TODO Auto-generated method stub
00036      Boolean accept = false;
00037      List<State> list = Strategy.getStrategy().listRefPoblacFinal;
00038
00039      if(list.size() == 0){
00040          list.add(stateCurrent.clone());
00041      }
00042      Double T = MultiCaseSimulatedAnnealing.getTinitial();
00043      double pAccept = 0;
00044      // use ThreadLocalRandom to avoid creating a new Random instance on each call
00045      Dominance dominance= new Dominance();
00046      //Verificando si la soluci n candidata domina a la soluci n actual
00047      //Si la soluci n candidata domina a la soluci n actual
00048      if(dominance.dominance(stateCandidate, stateCurrent) == true){
00049          //Se asigna como soluci n actual la soluci n candidata con probabilidad 1
00050          pAccept = 1;
00051      }
00052      else if(dominance.dominance(stateCandidate, stateCurrent)== false){
00053          if(dominanceCounter(stateCandidate, list) > 0){
00054              pAccept = 1;
00055          }
00056          else if(dominanceRank(stateCandidate, list) == 0){
00057              pAccept = 1;
00058          }
00059          else if(dominanceRank(stateCandidate, list) < dominanceRank(stateCurrent, list)){
00060              pAccept = 1;
00061          }
00062          else if(dominanceRank(stateCandidate, list) == dominanceRank(stateCurrent, list)){
00063              //Calculando la probabilidad de aceptaci n
00064              List<Double> evaluations = stateCurrent.getEvaluation();
00065              double total = 0;
00066              for (int i = 0; i < evaluations.size()-1; i++) {
00067                  Double evalA = evaluations.get(i);
00068                  Double evalB = stateCandidate.getEvaluation().get(i);
00069                  if (evalA != 0 && evalB != 0) {
00070                      total += (evalA - evalB)/((evalA + evalB)/2);
00071                  }
00072              }
00073              pAccept = Math.exp(-(1-total)/T);
00074          }
00075          else if (dominanceRank(stateCandidate, list) > dominanceRank(stateCurrent, list) &&
dominanceRank(stateCurrent, list)!= 0){
00076              // avoid integer division -> compute ranks once and use floating point division
00077              int rankCandidate = dominanceRank(stateCandidate, list);
00078              int rankCurrent = dominanceRank(stateCurrent, list);
00079              float value = (rankCurrent == 0) ? 0f : ((float) rankCandidate) / ((float)
rankCurrent);
00080              pAccept = Math.exp(-((double)value+1.0)/T);
00081          }
00082      else{
00083          //Calculando la probabilidad de aceptaci n
00084          List<Double> evaluations = stateCurrent.getEvaluation();
00085          double total = 0;
00086          for (int i = 0; i < evaluations.size()-1; i++) {
00087              Double evalA = evaluations.get(i);
00088              Double evalB = stateCandidate.getEvaluation().get(i);
00089              if (evalA != 0 && evalB != 0) {
00090                  total += (evalA - evalB)/((evalA + evalB)/2);
00091              }
00092          }
00093          pAccept = Math.exp(-(1-total)/T);
00094      }
00095  }
00096  //Generar un n mero aleatorio
}

```

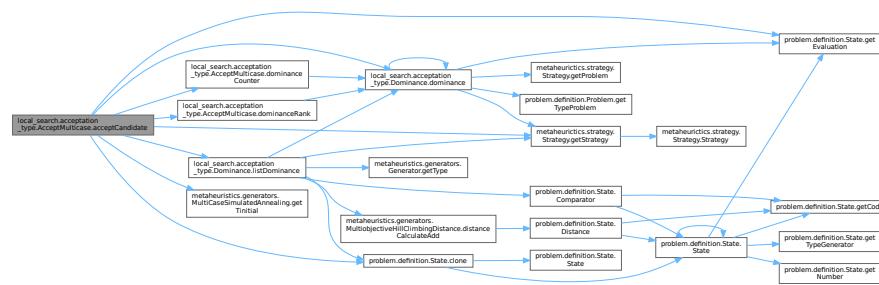
```

00097         if((ThreadLocalRandom.current().nextFloat() < pAccept) {
00098             // Don't assign to the parameter reference (dead store). The caller should handle state
00099             updates.
00100             // Verificando que la solución candidata domina a alguna de las soluciones
00101             accept = dominance.listDominance(stateCandidate, list);
00102         }
00103     }

```

Hace referencia a `problem.definition.State.clone()`, `local_search.acceptation_type.Dominance.dominance()`, `dominanceCounter()`, `dominanceRank()`, `problem.definition.State.getEvaluation()`, `metaheuristics.strategy.Strategy.getStrategy()`, `metaheuristics.generators.MultiCaseSimulatedAnnealing.getTinitial()`, `local_search.acceptation_type.Dominance.listDominance()` y `metaheuristics.strategy.Strategy.listRefPoblacFinal`.

Gráfico de llamadas de esta función:



#### 8.4.2.2 dominanceCounter()

```

int local_search.acceptation_type.AcceptMulticase.dominanceCounter (
    State stateCandidate,
    List< State > list) [inline], [private]

```

Definición en la línea 106 del archivo `AcceptMulticase.java`.

```

00106     soluciones de Pareto que son dominados por la nueva solución
00107     int counter = 0;
00108     for (int i = 0; i < list.size(); i++) {
00109         State solution = list.get(i);
00110         Dominance dominance = new Dominance();
00111         if(dominance.dominance(stateCandidate, solution) == true)
00112             counter++;
00113     }
00114     return counter;
00115 }

```

Hace referencia a `local_search.acceptation_type.Dominance.dominance()`.

Referenciado por `acceptCandidate()`.

Gráfico de llamadas de esta función:

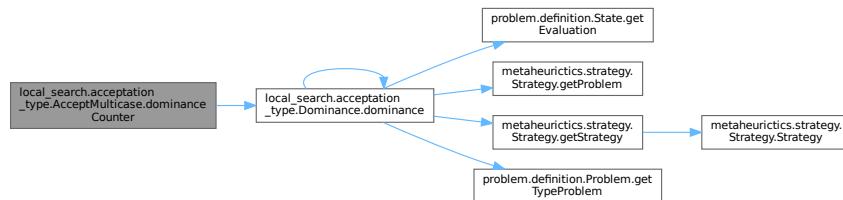


Gráfico de llamadas a esta función:



#### 8.4.2.3 dominanceRank()

```
int local_search.acceptation_type.AcceptMulticase.dominanceRank (
    State stateCandidate,
    List< State > list) [inline], [private]
```

Definición en la línea 117 del archivo [AcceptMulticase.java](#).

```
00117     soluciones en el conjunto de Pareto que dominan a la solución { //calculando el número de
00118         int rank = 0;
00119         for (int i = 0; i < list.size(); i++) {
00120             State solution = list.get(i);
00121             Dominance dominance = new Dominance();
00122             if(dominance.dominance(solution, stateCandidate) == true){
00123                 rank++;
00124             }
00125         }
00126     }
00127     return rank;
00128 }
```

Hace referencia a [local\\_search.acceptation\\_type.Dominance.dominance\(\)](#).

Referenciado por [acceptCandidate\(\)](#).

Gráfico de llamadas de esta función:

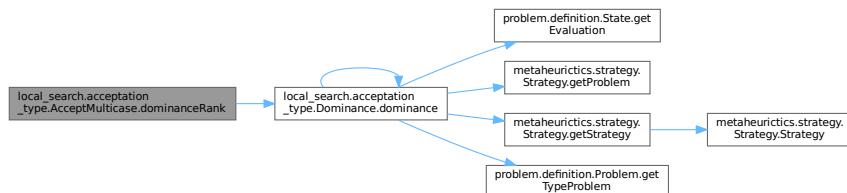


Gráfico de llamadas a esta función:



La documentación de esta clase está generada del siguiente archivo:

- [AcceptMulticase.java](#)

## 8.5 Referencia de la clase local\_search.acceptation\_type.AcceptNotBad

[AcceptNotBad](#) - Accept candidate if it is not worse than the current state.

Diagrama de herencia de local\_search.acceptation\_type.AcceptNotBad

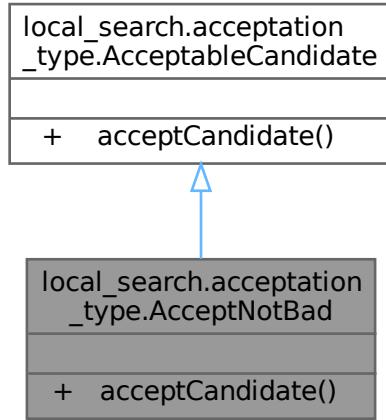
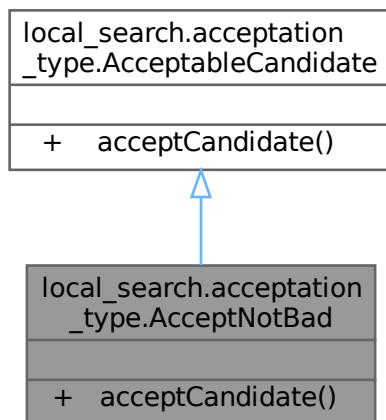


Diagrama de colaboración de local\_search.acceptation\_type.AcceptNotBad:



### Métodos públicos

- Boolean [acceptCandidate](#) ([State](#) stateCurrent, [State](#) stateCandidate)  
*Determine acceptance by comparing candidate and current state.*

## 8.5.1 Descripción detallada

[AcceptNotBad](#) - Accept candidate if it is not worse than the current state.

[Strategy](#) that accepts a candidate if it is at least as good as the current state according to the problem objective.

Definición en la línea 18 del archivo [AcceptNotBad.java](#).

## 8.5.2 Documentación de funciones miembro

### 8.5.2.1 acceptCandidate()

```
Boolean local_search.acceptation_type.AcceptNotBad.acceptCandidate (
    State stateCurrent,
    State stateCandidate) [inline]
```

Determine acceptance by comparing candidate and current state.

#### Parámetros

<i>stateCurrent</i>	the current state
<i>stateCandidate</i>	the candidate state

#### Devuelve

true if candidate is not worse than current state, false otherwise

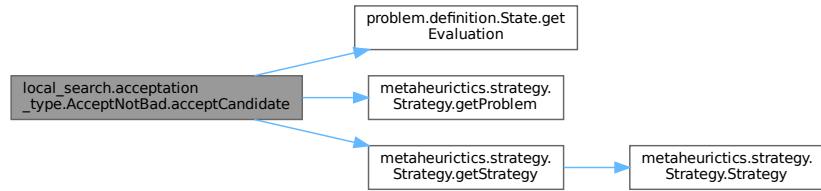
Reimplementado de [local\\_search.acceptation\\_type.AcceptableCandidate](#).

Definición en la línea 28 del archivo [AcceptNotBad.java](#).

```
00028
00029     Boolean accept = null;
00030     Problem problem = Strategy.getStrategy().getProblem();
00031     for (int i = 0; i < problem.getFunction().size(); i++) {
00032         if (problem.getTypeProblem().equals(ProblemType.Maximizar)) {
00033             if (stateCandidate.getEvaluation().get(0) >= stateCurrent.getEvaluation().get(0)) {
00034                 accept = true;
00035             } else {
00036                 accept = false;
00037             }
00038         } else {
00039             if (stateCandidate.getEvaluation().get(0) <= stateCurrent.getEvaluation().get(0)) {
00040                 accept = true;
00041             } else {
00042                 accept = false;
00043             }
00044         }
00045     }
00046     return accept;
}
```

Hace referencia a `problem.definition.State.getEvaluation()`, `metaheuristics.strategy.Strategy.getProblem()`, `metaheuristics.strategy.Strategy.getStrategy()` y `problem.definition.Problem.ProblemType.Maximizar`.

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [AcceptNotBad.java](#)

## 8.6 Referencia de la clase local\_search.acceptation\_type.AcceptNotBadT

[AcceptNotBadT](#) - Temperature based acceptation (simulated annealing style).

Diagrama de herencia de local\_search.acceptation\_type.AcceptNotBadT

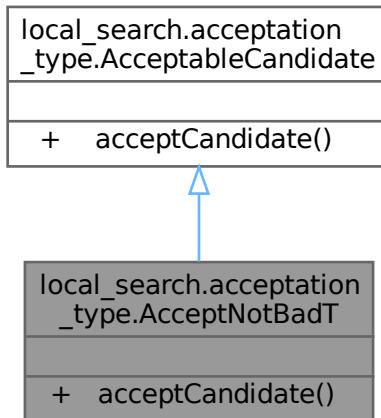
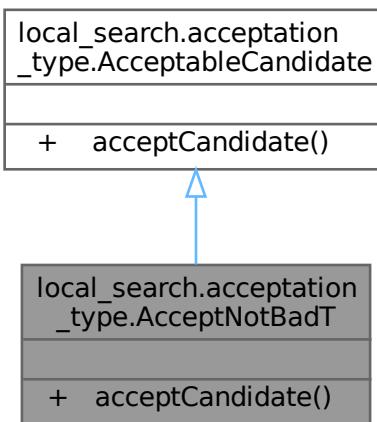


Diagrama de colaboración de local\_search.acceptation\_type.AcceptNotBadT:



## Métodos públicos

- Boolean `acceptCandidate (State stateCurrent, State stateCandidate)` throws `IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

*Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.*

### 8.6.1 Descripción detallada

[AcceptNotBadT](#) - Temperature based acceptation (simulated annealing style).

Accepts candidates with a probability depending on temperature and the evaluation difference when the candidate is worse.

Definición en la línea [22](#) del archivo [AcceptNotBadT.java](#).

### 8.6.2 Documentación de funciones miembro

#### 8.6.2.1 acceptCandidate()

```

Boolean local_search.acceptation_type.AcceptNotBadT.acceptCandidate (
    State stateCurrent,
    State stateCandidate) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
  
```

*Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.*

This RNG is used to decide acceptance in simulated annealing and is not used for security-sensitive purposes.  
Suppress Sonar S2245 for this usage. Decide acceptance using temperature-based probability. Generado por Doxygen

## Parámetros

<i>stateCurrent</i>	the current state
<i>stateCandidate</i>	the candidate state

Devuelve

true if accepted, false otherwise

## Excepciones

<i>IllegalArgumentException</i>	on invalid arguments
<i>SecurityException</i>	on reflective access problems
<i>ClassNotFoundException</i>	when a referenced class cannot be found
<i>InstantiationException</i>	on reflective instantiation failures
<i>IllegalAccessException</i>	on illegal reflective access
<i>InvocationTargetException</i>	when a reflective call throws
<i>NoSuchMethodException</i>	when a reflective method lookup fails

Reimplementado de [local\\_search.acceptation\\_type.AcceptableCandidate](#).

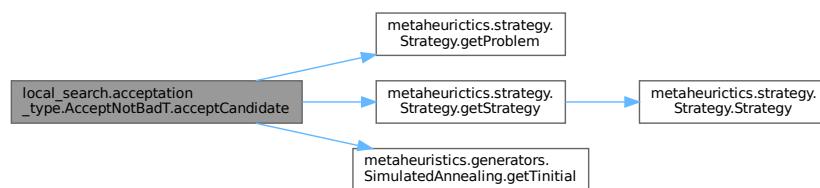
Definición en la línea 45 del archivo [AcceptNotBadT.java](#).

```

00045
00046     {
00047         Boolean accept = null;
00048         Problem problem = Strategy.getStrategy().getProblem();
00049         if (problem.getTypeProblem() .equals(ProblemType.Maximizar)) {
00050             double result = (stateCandidate.getEvaluation().get(0) -
00051                 stateCurrent.getEvaluation().get(0)) / SimulatedAnnealing.getTinitial();
00052             double probaleatory = ThreadLocalRandom.current().nextDouble();
00053             double exp = Math.exp(result);
00054             if ((stateCandidate.getEvaluation().get(0) >= stateCurrent.getEvaluation().get(0))
00055                 || (probaleatory < exp))
00056                 accept = true;
00057             else
00058                 accept = false;
00059         } else {
00060             double result_min = (stateCandidate.getEvaluation().get(0) -
00061                 stateCurrent.getEvaluation().get(0)) / SimulatedAnnealing.getTinitial();
00062             if ((stateCandidate.getEvaluation().get(0) <= stateCurrent.getEvaluation().get(0))
00063                 || (ThreadLocalRandom.current().nextDouble() < Math.exp(result_min)))
00064                 accept = true;
00065             else
00066                 accept = false;
00067         }
00068     }
00069     return accept;
00070 }
```

Hace referencia a [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [metaheuristics.generators.SimulatedAnnealing.getTinitial\(\)](#) y [problem.definition.Problem.ProblemType.Maximizar](#).

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [AcceptNotBadT.java](#)

## 8.7 Referencia de la clase

### **local\_search.acceptation\_type.AcceptNotBadU**

[AcceptNotBadU](#) - Threshold based acceptation strategy.

Diagrama de herencia de local\_search.acceptation\_type.AcceptNotBadU

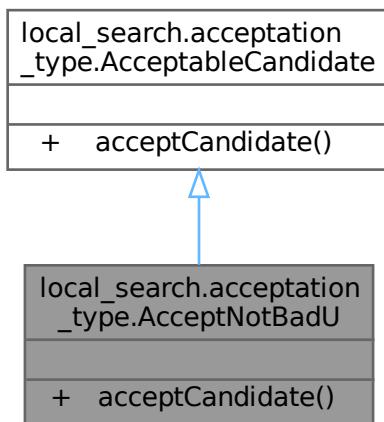
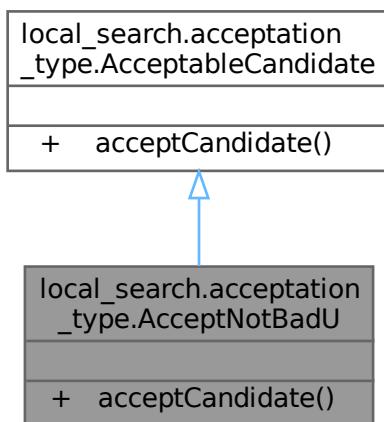


Diagrama de colaboración de local\_search.acceptation\_type.AcceptNotBadU:



## Métodos públicos

- Boolean `acceptCandidate (State stateCurrent, State stateCandidate)` throws `IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

*Decide acceptance using a configured threshold in the strategy.*

### 8.7.1 Descripción detallada

[AcceptNotBadU](#) - Threshold based acceptation strategy.

Accepts a candidate when the evaluation difference between the current and candidate state is below/above a configured threshold (depending on maximization/minimization).

Definición en la línea [21](#) del archivo [AcceptNotBadU.java](#).

### 8.7.2 Documentación de funciones miembro

#### 8.7.2.1 acceptCandidate()

```
Boolean local_search.acceptation_type.AcceptNotBadU.acceptCandidate (
    State stateCurrent,
    State stateCandidate) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

Decide acceptance using a configured threshold in the strategy.

#### Parámetros

<code>stateCurrent</code>	the current state
<code>stateCandidate</code>	the candidate state

#### Devuelve

true if candidate is accepted according to threshold, false otherwise

#### Excepciones

<code>IllegalArgumentException</code>	on invalid arguments
<code>SecurityException</code>	on reflective access problems
<code>ClassNotFoundException</code>	when a referenced class cannot be found
<code>InstantiationException</code>	on reflective instantiation failures
<code>IllegalAccessException</code>	on illegal reflective access
<code>InvocationTargetException</code>	when a reflective call throws
<code>NoSuchMethodException</code>	when a reflective method lookup fails

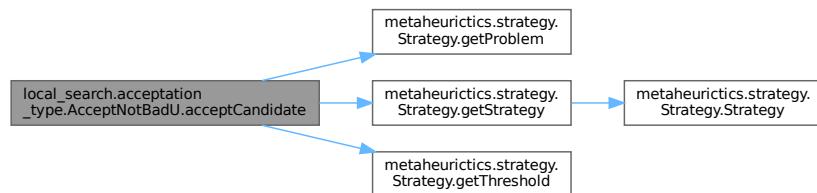
Reimplementado de [local\\_search.acceptation\\_type.AcceptableCandidate](#).

Definición en la línea 38 del archivo [AcceptNotBadU.java](#).

```
00038
{
00039     Boolean accept = null;
00040     Problem problem = Strategy.getStrategy().getProblem();
00041     if (problem.getTypeProblem() .equals(ProblemType.Maximizar)) {
00042         Double result = stateCurrent.getEvaluation() .get(0) -
00043             stateCandidate.getEvaluation() .get(0);
00044         if (result < Strategy.getStrategy().getThreshold())
00045             accept = true;
00046         else
00047             accept = false;
00048     } else {
00049         Double result_min = stateCurrent.getEvaluation() .get(0) -
00050             stateCandidate.getEvaluation() .get(0);
00051         if (result_min > Strategy.getStrategy().getThreshold())
00052             accept = true;
00053         else
00054             accept = false;
00055     }
00056 }
```

Hace referencia a [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [metaheuristics.strategy.Strategy.getThreshold\(\)](#) y [problem.definition.Problem.ProblemType.Maximizar](#).

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [AcceptNotBadU.java](#)

## 8.8 Referencia de la clase

### **local\_search.acceptation\_type.AcceptNotDominated**

[AcceptNotDominated](#) - Accept candidate if it is not dominated by current non-dominated set.

Diagrama de herencia de local\_search.acceptation\_type.AcceptNotDominated

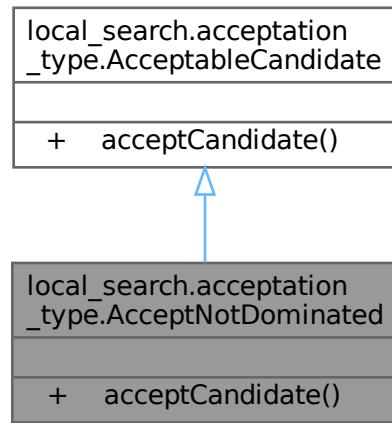
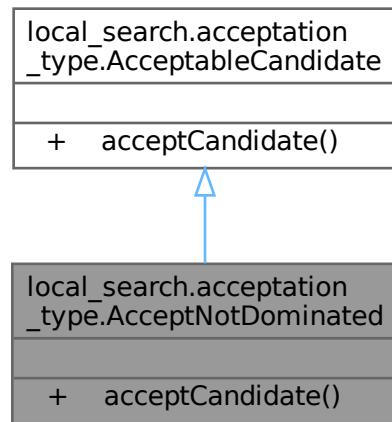


Diagrama de colaboración de local\_search.acceptation\_type.AcceptNotDominated:



## Métodos públicos

- Boolean `acceptCandidate (State stateCurrent, State stateCandidate)`

*Evaluate and accept a candidate based on Pareto dominance against the stored set of non-dominated solutions.*

## 8.8.1 Descripción detallada

[AcceptNotDominated](#) - Accept candidate if it is not dominated by current non-dominated set.

Multi-objective acceptation strategy that updates and consults a Pareto front (`listRefPoblacFinal`) to decide acceptance.

Definición en la línea 16 del archivo [AcceptNotDominated.java](#).

## 8.8.2 Documentación de funciones miembro

### 8.8.2.1 acceptCandidate()

```
Boolean local_search.acceptation_type.AcceptNotDominated.acceptCandidate (
    State stateCurrent,
    State stateCandidate) [inline]
```

Evaluate and accept a candidate based on Pareto dominance against the stored set of non-dominated solutions.

#### Parámetros

<code>stateCurrent</code>	the current state
<code>stateCandidate</code>	the candidate state

#### Devuelve

true if candidate should be accepted (becomes part of Pareto set), false otherwise

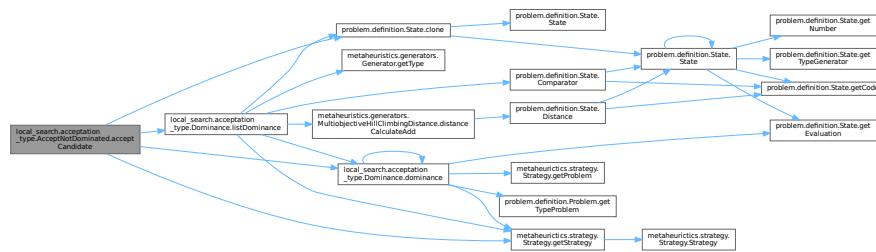
Reimplementado de [local\\_search.acceptation\\_type.AcceptableCandidate](#).

Definición en la línea 28 del archivo [AcceptNotDominated.java](#).

```
00028
00029     Boolean accept = false;
00030     Dominance dominace = new Dominance();
00031
00032     if(Strategy.getStrategy().listRefPoblacFinal.size() == 0){
00033         Strategy.getStrategy().listRefPoblacFinal.add(stateCurrent.clone());
00034     }
00035     if(dominace.dominance(stateCurrent, stateCandidate)== false)
00036     {
00037         //Verificando si la solución candidata domina a alguna de las soluciones de la lista de
00038         //soluciones no dominadas
00039         //De ser así se eliminan de la lista y se adiciona la nueva solución en la lista
00040         //De lo contrario no se adiciona la solución candidata a la lista
00041         //Si fue insertada en la lista entonces la solución candidata se convierte en solución
00042         //actual
00043         if(dominace.listDominance(stateCandidate, Strategy.getStrategy().listRefPoblacFinal) ==
00044             true){
00045             //Se pone la solución candidata como solución actual
00046             accept = true;
00047         }
00048     }
00049     return accept;
00050 }
```

Hace referencia a `problem.definition.State.clone()`, `local_search.acceptation_type.Dominance.dominance()`, `metaheuristics.strategy.Strategy.getStrategy()`, `local_search.acceptation_type.Dominance.listDominance()` y `metaheuristics.strategy.listRefPoblacFinal`.

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [AcceptNotDominated.java](#)

## 8.9 Referencia de la clase local\_search.acceptation\_type.AcceptNotDominatedTabu

[AcceptNotDominatedTabu](#) - Acceptation that integrates a tabu-like Pareto list.

Diagrama de herencia de local\_search.acceptation\_type.AcceptNotDominatedTabu

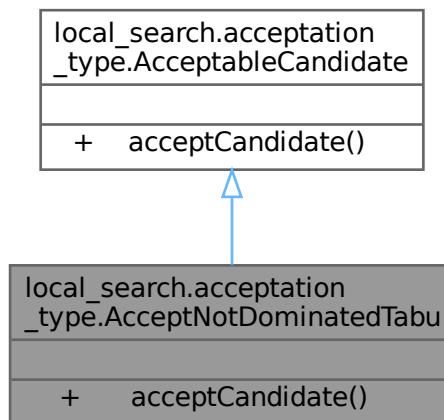
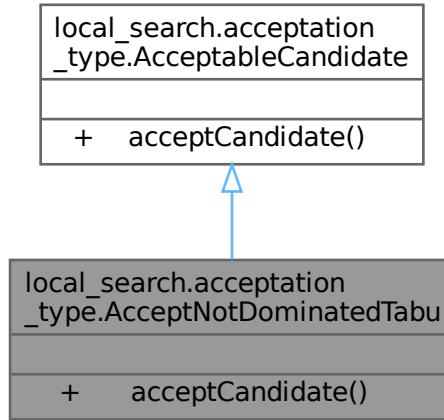


Diagrama de colaboración de local\_search.acceptation\_type.AcceptNotDominatedTabu:



## Métodos públicos

- Boolean [acceptCandidate \(State stateCurrent, State stateCandidate\)](#)  
*Evaluate candidate against the non-dominated list and update it.*

### 8.9.1 Descripción detallada

[AcceptNotDominatedTabu](#) - Acceptation that integrates a tabu-like Pareto list.

Maintains a list of non-dominated solutions and checks whether the candidate should be accepted into that list.  
 Intended for tabu-like multi-objective variants.

Definición en la línea 16 del archivo [AcceptNotDominatedTabu.java](#).

### 8.9.2 Documentación de funciones miembro

#### 8.9.2.1 acceptCandidate()

```
Boolean local_search.acceptation_type.AcceptNotDominatedTabu.acceptCandidate (
    State stateCurrent,
    State stateCandidate) [inline]
```

Evaluate candidate against the non-dominated list and update it.

#### Parámetros

<code>stateCurrent</code>	the current state
---------------------------	-------------------

<i>stateCandidate</i>	the candidate state
-----------------------	---------------------

### Devuelve

true if candidate processing succeeded (this method returns true in current implementation), false if not

Reimplementado de [local\\_search.acceptation\\_type.AcceptableCandidate](#).

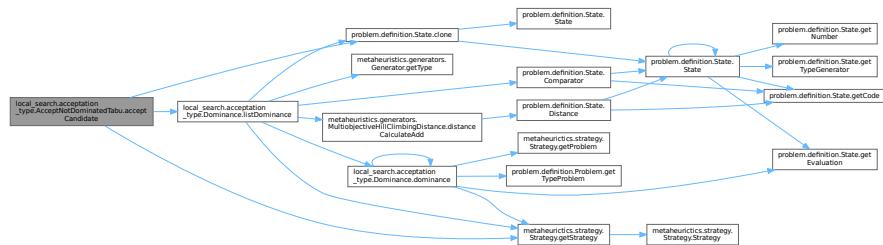
Definición en la línea 27 del archivo [AcceptNotDominatedTabu.java](#).

```

00027
00028     List<State> list = Strategy.getStrategy().listRefPoblacFinal;
00029
00030     Dominance dominance = new Dominance();
00031     if(list.size() == 0){
00032         list.add(stateCurrent.clone());
00033     }
00034     //Verificando si la solución candidata domina a alguna de las soluciones de la lista de
00035     //soluciones no dominadas
00036     //De ser así se eliminan de la lista y se adiciona la nueva solución en la lista
00037     //De lo contrario no se adiciona la solución candidata a la lista
00038     //Si fue insertada en la lista entonces la solución candidata se convierte en solución actual
00039     dominance.listDominance(stateCandidate, list);
00040     return true;
00041 }
```

Hace referencia a [problem.definition.State.clone\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [local\\_search.acceptation\\_type.Dominance](#) y [metaheuristics.strategy.Strategy.listRefPoblacFinal](#).

Gráfico de llamadas de esta función:



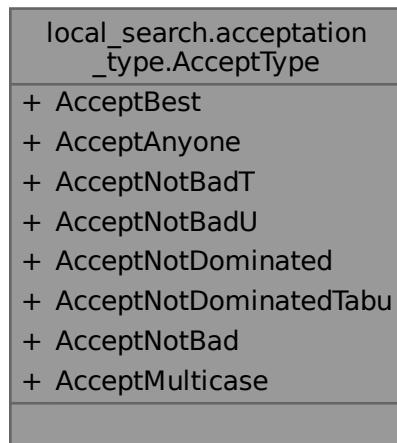
La documentación de esta clase está generada del siguiente archivo:

- [AcceptNotDominatedTabu.java](#)

## 8.10 Referencia de la enumeración local\_search.acceptation\_type.AcceptType

[AcceptType](#) - Enumeration of the available acceptation strategies.

Diagrama de colaboración de local\_search.acceptation\_type.AcceptType:



### Atributos públicos

- [AcceptBest](#)
- [AcceptAnyone](#)
- [AcceptNotBadT](#)
- [AcceptNotBadU](#)
- [AcceptNotDominated](#)
- [AcceptNotDominatedTabu](#)
- [AcceptNotBad](#)
- [AcceptMulticase](#)

### 8.10.1 Descripción detallada

[AcceptType](#) - Enumeration of the available acceptation strategies.

Enumerates the strategies supported by the local search framework for deciding whether to accept candidate solutions.

Definición en la línea 13 del archivo [AcceptType.java](#).

### 8.10.2 Documentación de datos miembro

#### 8.10.2.1 AcceptAnyone

`local_search.acceptation_type.AcceptType.AcceptAnyone`

Definición en la línea 15 del archivo [AcceptType.java](#).

Referenciado por [metaheuristics.generators.TabuSearch.TabuSearch\(\)](#).

### 8.10.2.2 AcceptBest

```
local_search.acceptation_type.AcceptType.AcceptBest
```

Definición en la línea 15 del archivo [AcceptType.java](#).

Referenciado por [metaheuristics.generators.HillClimbing.HillClimbing\(\)](#), [metaheuristics.generators.HillClimbingRestart.HillClimbingRestart\(\)](#) y [metaheuristics.generators.RandomSearch.RandomSearch\(\)](#).

### 8.10.2.3 AcceptMulticase

```
local_search.acceptation_type.AcceptType.AcceptMulticase
```

Definición en la línea 15 del archivo [AcceptType.java](#).

Referenciado por [metaheuristics.generators.MultiCaseSimulatedAnnealing.MultiCaseSimulatedAnnealing\(\)](#).

### 8.10.2.4 AcceptNotBad

```
local_search.acceptation_type.AcceptType.AcceptNotBad
```

Definición en la línea 15 del archivo [AcceptType.java](#).

### 8.10.2.5 AcceptNotBadT

```
local_search.acceptation_type.AcceptType.AcceptNotBadT
```

Definición en la línea 15 del archivo [AcceptType.java](#).

Referenciado por [metaheuristics.generators.SimulatedAnnealing.SimulatedAnnealing\(\)](#).

### 8.10.2.6 AcceptNotBadU

```
local_search.acceptation_type.AcceptType.AcceptNotBadU
```

Definición en la línea 15 del archivo [AcceptType.java](#).

Referenciado por [metaheuristics.generators.LimitThreshold.LimitThreshold\(\)](#).

### 8.10.2.7 AcceptNotDominated

```
local_search.acceptation_type.AcceptType.AcceptNotDominated
```

Definición en la línea 15 del archivo [AcceptType.java](#).

Referenciado por [metaheuristics.generators.MultiobjectiveHillClimbingDistance.MultiobjectiveHillClimbingDistance\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart.MultiobjectiveHillClimbingRestart\(\)](#) y [metaheuristics.generators.Multiobje](#)

### 8.10.2.8 AcceptNotDominatedTabu

local\_search.acceptation\_type.AcceptType.AcceptNotDominatedTabu

Definición en la línea 15 del archivo [AcceptType.java](#).

Referenciado por [metaheuristics.generators.MultiobjectiveTabuSearch.MultiobjectiveTabuSearch\(\)](#).

La documentación de esta enumeración está generada del siguiente archivo:

- [AcceptType.java](#)

## 8.11 Referencia de la clase **evolutionary\_algorithms.complement.AIOMutation**

[AIOMutation](#) - applies the [AIOMutation](#) to the state.

Diagrama de herencia de `evolutionary_algorithms.complement.AIOMutation`

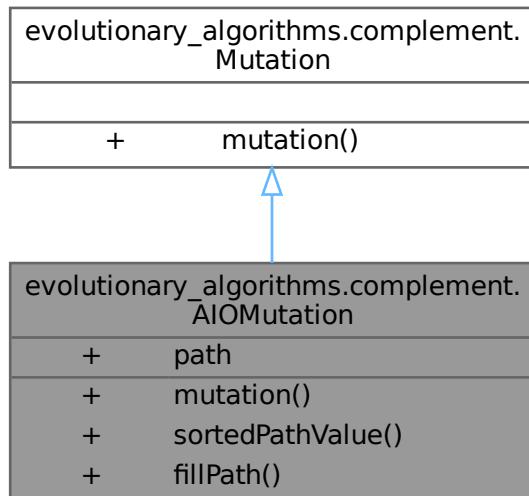
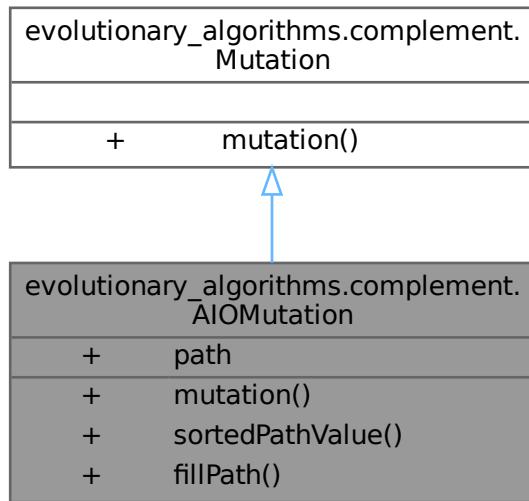


Diagrama de colaboración de `evolutionary_algorithms.complement.AIOMutation`:



## Métodos públicos

- `State mutation (State state, double PM)`  
*mutation - applies the mutation to the state.*
- `void sortedPathValue (State state)`  
*sortedPathValue - sorts the path values based on the state.*

## Métodos públicos estáticos

- `static void fillPath ()`  
*fillPath - fills the path with the variable indices.*

## Atributos públicos estáticos

- `static final List< Object > path = Collections.synchronizedList(new ArrayList<Object>())`  
*path - stores the path of the TSP problem.*

### 8.11.1 Descripción detallada

`AIOMutation` - applies the `AIOMutation` to the state.

Definición en la línea 17 del archivo `AIOMutation.java`.

## 8.11.2 Documentación de funciones miembro

### 8.11.2.1 fillPath()

```
void evolutionary_algorithms.complement.AIOMutation.fillPath () [inline], [static]
```

fillPath - fills the path with the variable indices.

Definición en la línea 85 del archivo [AIOMutation.java](#).

```
00085         {
00086             for(int k = 0; k < Strategy.getStrategy().getProblem().getCodification().getVariableCount();
00087                 k++) {
00088                 path.add(k);
00089             }
}
```

Hace referencia a [metaheuristics.strategy.Strategy.getStrategy\(\)](#) y [path](#).

Gráfico de llamadas de esta función:



### 8.11.2.2 mutation()

```
State evolutionary_algorithms.complement.AIOMutation.mutation (
    State state,
    double PM) [inline]
```

mutation - applies the mutation to the state.

#### Parámetros

<i>state</i>	
<i>PM</i>	

#### Devuelve

Reimplementado de [evolutionary\\_algorithms.complement.Mutation](#).

Definición en la línea 31 del archivo [AIOMutation.java](#).

```
00031         {
00032             // TODO Auto-generated method stub
00033             int key = Strategy.getStrategy().getProblem().getCodification().getAleatoryKey();
00034             //seleccionar aleatoriamente una ciudad
00035             int key1 = 0;
00036             boolean found = false;
00037             while (found == false){
00038                 key1 = Strategy.getStrategy().getProblem().getCodification().getAleatoryKey();
00039                 if(key1 != key)
```

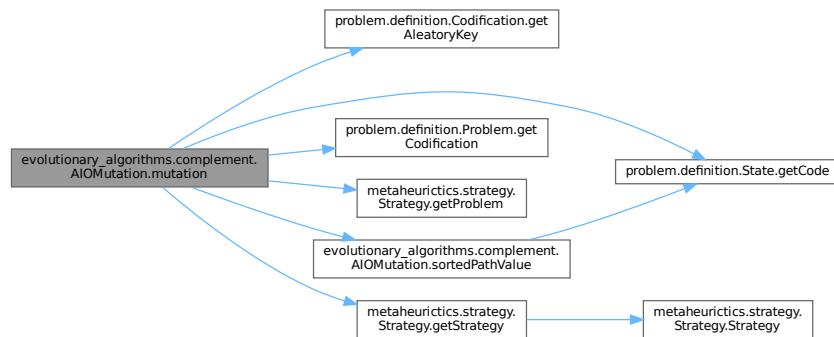
```

00039         found = true;
00040     }
00041     sortedPathValue(state);
00042     int p1 = 0;
00043     int p2 = 0;
00044     if(key > key1){
00045         p2 = key;
00046         p1 = key1;
00047     }
00048     else{
00049         p1 = key;
00050         p2 = key1;
00051     }
00052     int length = (p2 - p1) / 2;
00053     for (int i = 1; i <= length + 1; i++) {
00054         int tempC = ((TSPState) state.getCode().get(p1 + i - 1)).getIdCity();
00055         ((TSPState) state.getCode()).get(p1 + i - 1).setIdCity(((TSPState) state.getCode()).get(p2 -
00056             i + 1).getIdCity());
00057         ((TSPState) state.getCode()).get(p2 - i + 1).setIdCity(tempC);
00058     }
00059     path.clear();
00060     return state;
00060 }

```

Hace referencia a [problem.definition.Codification.getAleatoryKey\(\)](#), [problem.definition.State.getCode\(\)](#), [problem.definition.Problem.get](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [path](#) y [sortedPathValue\(\)](#).

Gráfico de llamadas de esta función:



### 8.11.2.3 sortedPathValue()

```
void evolutionary_algorithms.complement.AIOMutation.sortedPathValue (
    State state) [inline]
```

sortedPathValue - sorts the path values based on the state.

#### Parámetros

state	
-------	--

Definición en la línea 66 del archivo [AIOMutation.java](#).

```

00066         {
00067             for(int k = 0; k < state.getCode().size(); k++){
00068                 path.add( state.getCode().get(k));
00069             }
00070             for(int i = 1; i < path.size(); i++){
00071                 for(int j = 0; j < i; j++){
00072                     Integer data1 = ((TSPState)state.getCode().get(i)).getValue();

```

```

00073         Integer data2 = ((TSPState)state.getCode().get(j)).getValue();
00074         if(data1 < data2){
00075             state.getCode().add(j, state.getCode().remove(i));
00076             path.add(j, path.remove(i));
00077         }
00078     }
00079 }
00080
00081 }
```

Hace referencia a [problem.definition.State.getCode\(\)](#) y [path](#).

Referenciado por [mutation\(\)](#).

Gráfico de llamadas de esta función:



Gráfico de llamadas a esta función:



### 8.11.3 Documentación de datos miembro

#### 8.11.3.1 path

```
final List<Object> evolutionary_algorithms.complement.AIOMutation.path = Collections.synchronizedList(new ArrayList<Object>()) [static]
```

path - stores the path of the TSP problem.

Definición en la línea 22 del archivo [AIOMutation.java](#).

Referenciado por [fillPath\(\)](#), [mutation\(\)](#) y [sortedPathValue\(\)](#).

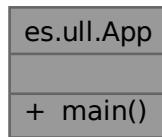
La documentación de esta clase está generada del siguiente archivo:

- [AIOMutation.java](#)

## 8.12 Referencia de la clase es.ull.App

Hello world!

Diagrama de colaboración de es.ull.App:



### Métodos públicos estáticos

- static void [main](#) (String[ ] args)

*Main method.*

### 8.12.1 Descripción detallada

Hello world!

[App](#) - descripción (añade detalles).

Definición en la línea [10](#) del archivo [App.java](#).

### 8.12.2 Documentación de funciones miembro

#### 8.12.2.1 [main\(\)](#)

```
void es.ull.App.main (
    String[ ] args) [inline], [static]
```

Main method.

##### Parámetros

<i>args</i>	
-------------	--

Definición en la línea [15](#) del archivo [App.java](#).

```
00016   {
00017       System.out.println( "Hello World!" );
00018   }
```

La documentación de esta clase está generada del siguiente archivo:

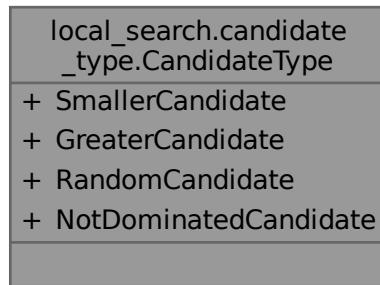
- [App.java](#)

## 8.13 Referencia de la enumeración

### local\_search.candidate\_type.CandidateType

[CandidateType](#) - enumeration of available candidate selection strategies.

Diagrama de colaboración de local\_search.candidate\_type.CandidateType:



#### Atributos públicos

- [SmallerCandidate](#)  
*Select the smallest/best candidate according to objective.*
- [GreaterCandidate](#)  
*Select the largest/best candidate according to objective.*
- [RandomCandidate](#)  
*Select a candidate at random from the neighborhood.*
- [NotDominatedCandidate](#)  
*Select a candidate that is not dominated (multi-objective).*

#### 8.13.1 Descripción detallada

[CandidateType](#) - enumeration of available candidate selection strategies.

Enumerates strategies used to pick a candidate state from a neighborhood during local search operators.

Definición en la línea 13 del archivo [CandidateType.java](#).

#### 8.13.2 Documentación de datos miembro

##### 8.13.2.1 GreaterCandidate

local\_search.candidate\_type.CandidateType.GreaterCandidate

Select the largest/best candidate according to objective.

Definición en la línea 18 del archivo [CandidateType.java](#).

Referenciado por [metaheuristics.generators.HillClimbing.HillClimbing\(\)](#), [metaheuristics.generators.HillClimbingRestart.HillClimbingRestart\(\)](#), [metaheuristics.generators.LimitThreshold.LimitThreshold\(\)](#) y [metaheuristics.generators.TabuSearch.TabuSearch\(\)](#).

### 8.13.2.2 NotDominatedCandidate

`local_search.candidate_type.CandidateType.NotDominatedCandidate`

Select a candidate that is not dominated (multi-objective).

Definición en la línea 22 del archivo [CandidateType.java](#).

Referenciado por [metaheuristics.generators.MultiobjectiveHillClimbingDistance.MultiobjectiveHillClimbingDistance\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart.MultiobjectiveHillClimbingRestart\(\)](#) y [metaheuristics.generators.MultiobjectiveHillClimbingRestart.MultiobjectiveHillClimbingRestart\(\)](#).

### 8.13.2.3 RandomCandidate

`local_search.candidate_type.CandidateType.RandomCandidate`

Select a candidate at random from the neighborhood.

Definición en la línea 20 del archivo [CandidateType.java](#).

Referenciado por [metaheuristics.generators.MultiCaseSimulatedAnnealing.MultiCaseSimulatedAnnealing\(\)](#), [metaheuristics.generators.MultiobjectiveTabuSearch.MultiobjectiveTabuSearch\(\)](#), [metaheuristics.generators.RandomSearch.RandomSearch\(\)](#) y [metaheuristics.generators.SimulatedAnnealing.SimulatedAnnealing\(\)](#).

### 8.13.2.4 SmallerCandidate

`local_search.candidate_type.CandidateType.SmallerCandidate`

Select the smallest/best candidate according to objective.

Definición en la línea 16 del archivo [CandidateType.java](#).

Referenciado por [metaheuristics.generators.HillClimbing.HillClimbing\(\)](#), [metaheuristics.generators.HillClimbingRestart.HillClimbingRestart\(\)](#), [metaheuristics.generators.LimitThreshold.LimitThreshold\(\)](#) y [metaheuristics.generators.TabuSearch.TabuSearch\(\)](#).

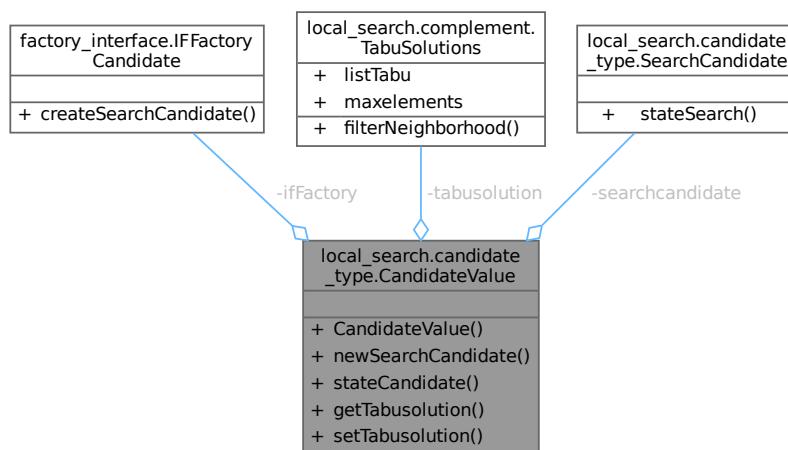
La documentación de esta enumeración está generada del siguiente archivo:

- [CandidateType.java](#)

## 8.14 Referencia de la clase local\_search.candidate\_type.CandidateValue

`CandidateValue` - Factory/wrapper for creating and using [SearchCandidate](#) selection strategies.

Diagrama de colaboración de `local_search.candidate_type.CandidateValue`:



## Métodos públicos

- [CandidateValue \(\)](#)
- [SearchCandidate newSearchCandidate \(CandidateType typecandidate\) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException](#)

*Create a new SearchCandidate instance for the provided candidate type.*
- [State stateCandidate \(State stateCurrent, CandidateType typeCandidate, StrategyType strategy, Integer operatornumber, List< State > neighborhood\) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException](#)

*Choose a candidate state from the provided neighborhood according to the specified candidate selection type and strategy.*
- [TabuSolutions getTabusolution \(\)](#)

*Get the currently stored tabu solutions helper instance.*
- [void setTabusolution \(TabuSolutions tabusolution\)](#)

*Set the TabuSolutions helper used by this CandidateValue.*

## Atributos privados

- [IFFactoryCandidate ifFactory](#)
- [TabuSolutions tabusolution](#)
- [SearchCandidate searchcandidate](#)

### 8.14.1 Descripción detallada

[CandidateValue](#) - Factory/wrapper for creating and using [SearchCandidate](#) selection strategies.

Responsible for creating concrete [SearchCandidate](#) implementations (via factories), filtering neighborhoods using tabu lists and returning candidate states chosen by the selected strategy.

Definición en la línea 29 del archivo [CandidateValue.java](#).

### 8.14.2 Documentación de constructores y destructores

#### 8.14.2.1 CandidateValue()

```
local_search.candidate_type.CandidateValue.CandidateValue () [inline]
```

Definición en la línea 40 del archivo [CandidateValue.java](#).

```
00040 { }
```

### 8.14.3 Documentación de funciones miembro

#### 8.14.3.1 getTabusolution()

```
TabuSolutions local_search.candidate_type.CandidateValue.getTabusolution () [inline]
```

Get the currently stored tabu solutions helper instance.

Devuelve

[TabuSolutions](#) manager used for neighborhood filtering

Definición en la línea 104 del archivo [CandidateValue.java](#).

```
00104
00105     return tabusolution;
00106 }
```

Hace referencia a [tabusolution](#).

### 8.14.3.2 newSearchCandidate()

```
SearchCandidate local_search.candidate_type.CandidateValue.newSearchCandidate (
    CandidateType typecandidate) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
Exception, NoSuchMethodException [inline]
```

Create a new `SearchCandidate` instance for the provided candidate type.

#### Parámetros

<code>typecandidate</code>	the enum value describing which concrete <code>SearchCandidate</code> to create
----------------------------	---

#### Devuelve

a new `SearchCandidate` instance

#### Excepciones

<code>ReflectiveOperationException</code>	when the factory cannot create the class
---	--

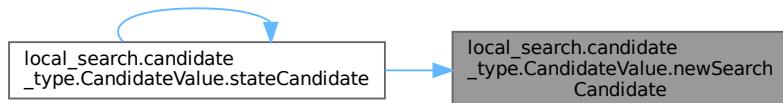
Definición en la línea 52 del archivo `CandidateValue.java`.

```
00052
{
00053     ifFactory = new FactoryCandidate();
00054     searchcandidate = ifFactory.createSearchCandidate(typecandidate);
00055     return searchcandidate;
00056 }
```

Hace referencia a `ifFactory` y `searchcandidate`.

Referenciado por `stateCandidate()`.

Gráfico de llamadas a esta función:



### 8.14.3.3 setTabusolution()

```
void local_search.candidate_type.CandidateValue.setTabusolution (
    TabuSolutions tabusolution) [inline]
```

Set the `TabuSolutions` helper used by this `CandidateValue`.

## Parámetros

<i>tabusolution</i>	<a href="#">TabuSolutions</a> instance to use for neighborhood filtering
---------------------	--

Definición en la línea 113 del archivo [CandidateValue.java](#).

```
00113
00114     this.tabusolution = tabusolution;
00115 }
```

Hace referencia a [tabusolution](#).

### 8.14.3.4 stateCandidate()

```
State local_search.candidate_type.CandidateValue.stateCandidate (
    State stateCurrent,
    CandidateType typeCandidate,
    StrategyType strategy,
    Integer operatornumber,
    List< State > neighborhood) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

Choose a candidate state from the provided neighborhood according to the specified candidate selection type and strategy.

If the [StrategyType](#) is TABU, the neighborhood is first filtered through the tabu list. On exception the method will attempt to regenerate the neighborhood via the strategy's problem operator and retry.

## Parámetros

<i>stateCurrent</i>	the current state
<i>typeCandidate</i>	which candidate selection strategy to use
<i>strategy</i>	the local search strategy type (e.g. TABU)
<i>operatornumber</i>	operator index used to generate new states if needed
<i>neighborhood</i>	list of neighbor states to select from

Devuelve

the chosen candidate state

## Excepciones

<i>ReflectiveOperationException</i>	when factory/reflection creation fails
-------------------------------------	--

Definición en la línea 74 del archivo [CandidateValue.java](#).

```
00074
00075     {
00076         //Problem problem = ExecuteGenerator.getExecuteGenerator().getProblem();
00077         State stateCandidate;
00078         List<State> auxList = new ArrayList<State>();
00079         for (int i = 0; i < neighborhood.size(); i++) {
00080             auxList.add(neighborhood.get(i));
```

```

00080      }
00081      this.tabusolution = new TabuSolutions();
00082      if (strategy.equals(StrategyType.TABU)) {
00083          try {
00084              auxList = this.tabusolution.filterNeighborhood(auxList);
00085          }
00086          catch (Exception e) {
00087              Strategy strategys = Strategy.getStrategy();
00088              if(strategys.getProblem() != null) {
00089                  neighborhood =
00090                      strategys.getProblem().getOperator().generatedNewState(neighborhood.get(0), operatornumber);
00091              }
00092          }
00093      }
00094      SearchCandidate searchCand = newSearchCandidate(typeCandidate);
00095      stateCandidate = searchCand.stateSearch(auxList);
00096      return stateCandidate;
00097  }

```

Hace referencia a `local_search.complement.TabuSolutions.filterNeighborhood()`, `problem.definition.Operator.generatedNewState()`, `problem.definition.Problem.getOperator()`, `metaheuristics.strategy.Strategy.getProblem()`, `metaheuristics.strategy.Strategy.getStrategy()`, `newSearchCandidate()`, `stateCandidate()`, `local_search.candidate_type.SearchCandidate.stateSearch()` y `local_search.complement.StrategyType.TABU`.

Referenciado por [stateCandidate\(\)](#).

Gráfico de llamadas de esta función:

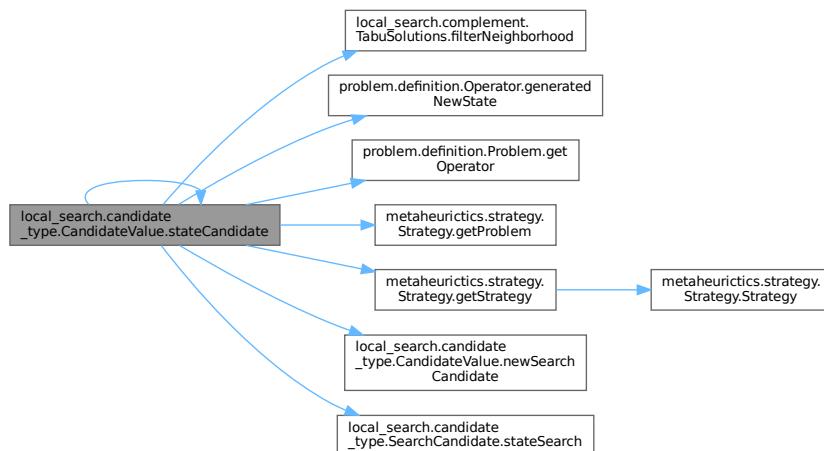
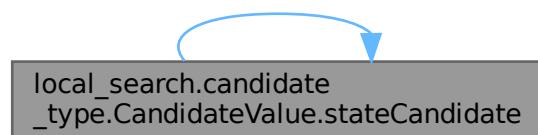


Gráfico de llamadas a esta función:



## 8.14.4 Documentación de datos miembro

### 8.14.4.1 ifFactory

```
IFFactoryCandidate local_search.candidate_type.CandidateValue.ifFactory [private]
```

Definición en la línea 32 del archivo [CandidateValue.java](#).

Referenciado por [newSearchCandidate\(\)](#).

### 8.14.4.2 searchcandidate

```
SearchCandidate local_search.candidate_type.CandidateValue.searchcandidate [private]
```

Definición en la línea 38 del archivo [CandidateValue.java](#).

Referenciado por [newSearchCandidate\(\)](#).

### 8.14.4.3 tabusolution

```
TabuSolutions local_search.candidate_type.CandidateValue.tabusolution [private]
```

Definición en la línea 36 del archivo [CandidateValue.java](#).

Referenciado por [getTabusolution\(\)](#) y [setTabusolution\(\)](#).

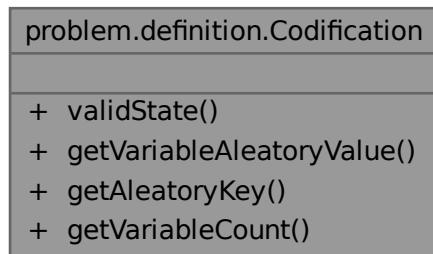
La documentación de esta clase está generada del siguiente archivo:

- [CandidateValue.java](#)

## 8.15 Referencia de la clase problem.definition.Codification

[Codification](#).

Diagrama de colaboración de `problem.definition.Codification`:



## Métodos públicos

- abstract boolean [validState \(State state\)](#)
- abstract Object [getVariableAleatoryValue \(int key\)](#)
- abstract int [getAleatoryKey \(\)](#)
- abstract int [getVariableCount \(\)](#)

### 8.15.1 Descripción detallada

[Codification](#).

Interfaz abstracta para la codificación de variables de un problema.

Implementaciones concretas deben validar estados y generar valores aleatorios para variables según el dominio del problema.

Definición en la línea 11 del archivo [Codification.java](#).

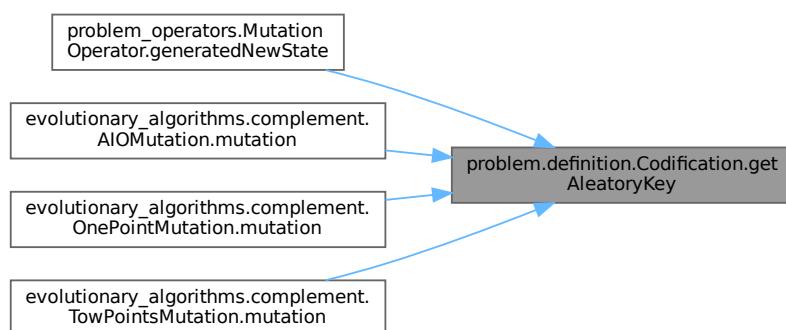
### 8.15.2 Documentación de funciones miembro

#### 8.15.2.1 [getAleatoryKey\(\)](#)

```
abstract int problem.definition.Codification.getAleatoryKey () [abstract]
```

Referenciado por [problem\\_operators.MutationOperator.generatedNewState\(\)](#), [evolutionary\\_algorithms.complement.AIOMutation.mutation\(\)](#), [evolutionary\\_algorithms.complement.OnePointMutation.mutation\(\)](#) y [evolutionary\\_algorithms.complement.TowPointsMutation.mutation\(\)](#)

Gráfico de llamadas a esta función:

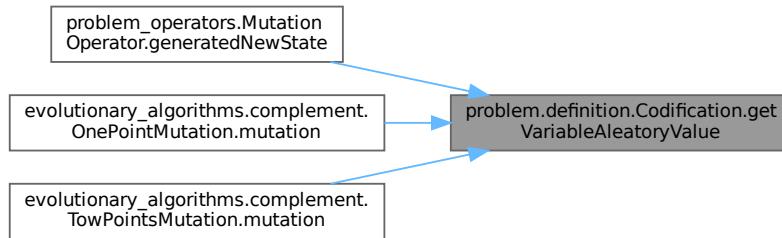


### 8.15.2.2 getVariableAleatoryValue()

```
abstract Object problem.definition.Codification.getVariableAleatoryValue (
    int key) [abstract]
```

Referenciado por [problem\\_operators.MutationOperator.generatedNewState\(\)](#), [evolutionary\\_algorithms.complement.OnePointMutation.mutation\(\)](#) y [evolutionary\\_algorithms.complement.TowPointsMutation.mutation\(\)](#).

Gráfico de llamadas a esta función:

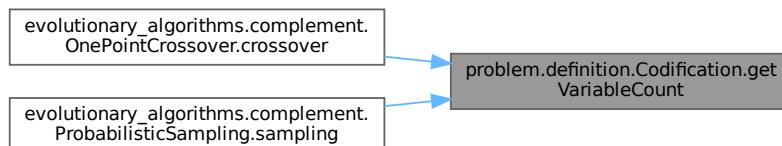


### 8.15.2.3 getCount()

```
abstract int problem.definition.Codification.getCount () [abstract]
```

Referenciado por [evolutionary\\_algorithms.complement.OnePointCrossover.crossover\(\)](#) y [evolutionary\\_algorithms.complement.ProbabilisticSampling.sampling\(\)](#).

Gráfico de llamadas a esta función:



### 8.15.2.4 validState()

```
abstract boolean problem.definition.Codification.validState (
    State state) [abstract]
```

La documentación de esta clase está generada del siguiente archivo:

- [Codification.java](#)

## 8.16 Referencia de la clase evolutionary\_algorithms.complement.Crossover

[Crossover](#) - applies the crossover operation to two parent states.

Diagrama de herencia de `evolutionary_algorithms.complement.Crossover`

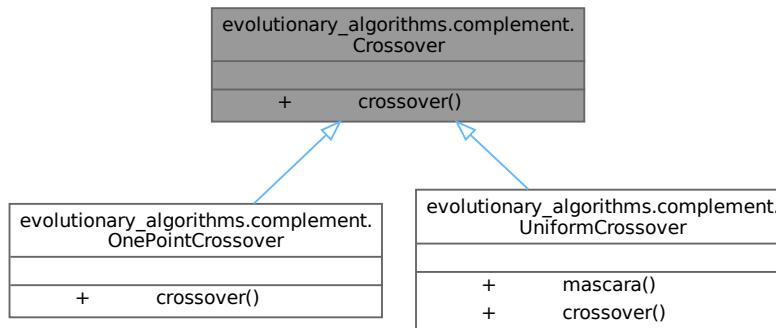
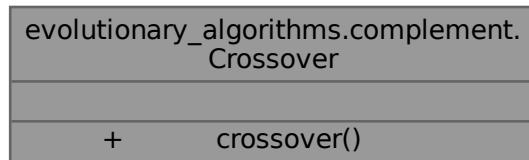


Diagrama de colaboración de `evolutionary_algorithms.complement.Crossover`:



### Métodos públicos

- abstract [State crossover \(State father1, State father2, double PC\)](#)  
*crossover - applies the crossover operation to two parent states.*

#### 8.16.1 Descripción detallada

[Crossover](#) - applies the crossover operation to two parent states.

Definición en la línea 9 del archivo [Crossover.java](#).

## 8.16.2 Documentación de funciones miembro

### 8.16.2.1 crossover()

```
abstract State evolutionary_algorithms.complement.Crossover.crossover (
    State father1,
    State father2,
    double PC) [abstract]
```

crossover - applies the crossover operation to two parent states.

#### Parámetros

<i>father1</i>	
<i>father2</i>	
<i>PC</i>	

#### Devuelve

returns the offspring resulting from the crossover between father1 and father2

Reimplementado en [evolutionary\\_algorithms.complement.OnePointCrossover](#) y [evolutionary\\_algorithms.complement.UniformCrossover](#).

Referenciado por [metaheuristics.generators.GeneticAlgorithm.generate\(\)](#).

Gráfico de llamadas a esta función:



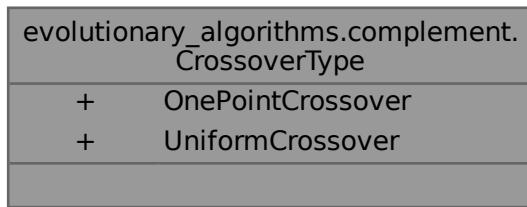
La documentación de esta clase está generada del siguiente archivo:

- [Crossover.java](#)

## 8.17 Referencia de la enumeración **evolutionary\_algorithms.complement.CrossoverType**

[CrossoverType](#) - descripción (añade detalles).

Diagrama de colaboración de evolutionary\_algorithms.complement.CrossoverType:



### Atributos públicos

- [OnePointCrossover](#)
- [UniformCrossover](#)

### 8.17.1 Descripción detallada

[CrossoverType](#) - descripción (añade detalles).

Definición en la línea [6](#) del archivo [CrossoverType.java](#).

### 8.17.2 Documentación de datos miembro

#### 8.17.2.1 OnePointCrossover

`evolutionary_algorithms.complement.CrossoverType.OnePointCrossover`

Definición en la línea [7](#) del archivo [CrossoverType.java](#).

#### 8.17.2.2 UniformCrossover

`evolutionary_algorithms.complement.CrossoverType.UniformCrossover`

Definición en la línea [7](#) del archivo [CrossoverType.java](#).

La documentación de esta enumeración está generada del siguiente archivo:

- [CrossoverType.java](#)

## 8.18 Referencia de la clase

### **evolutionary\_algorithms.complement.Distribution**

[Distribution](#) - applies a probability distribution to a set of fathers.

Diagrama de herencia de `evolutionary_algorithms.complement.Distribution`

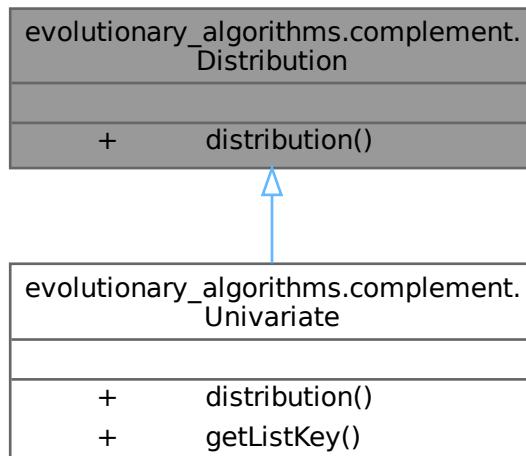
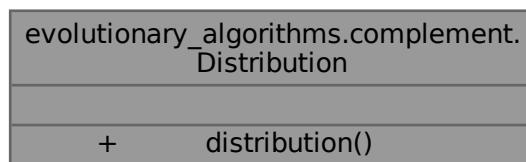


Diagrama de colaboración de `evolutionary_algorithms.complement.Distribution`:



#### Métodos públicos

- abstract List<[Probability](#)> `distribution` (List<[State](#)> fathers)  
*distribution - applies the distribution to the given fathers.*

#### 8.18.1 Descripción detallada

[Distribution](#) - applies a probability distribution to a set of fathers.

Definición en la línea 11 del archivo [Distribution.java](#).

## 8.18.2 Documentación de funciones miembro

### 8.18.2.1 distribution()

```
abstract List< Probability > evolutionary_algorithms.complement.Distribution.distribution (   
    List< State > fathers) [abstract]
```

distribution - applies the distribution to the given fathers.

#### Parámetros

<i>fathers</i>	
----------------	--

#### Devuelve

returns a list of probabilities associated to the given fathers

Reimplementado en [evolutionary\\_algorithms.complement.Univariate](#).

La documentación de esta clase está generada del siguiente archivo:

- [Distribution.java](#)

## 8.19 Referencia de la clase metaheuristics.generators.DistributionEstimationAlgorithm

[DistributionEstimationAlgorithm](#) - class for distribution estimation algorithms.

Diagrama de herencia de metaheuristics.generators.DistributionEstimationAlgorithm

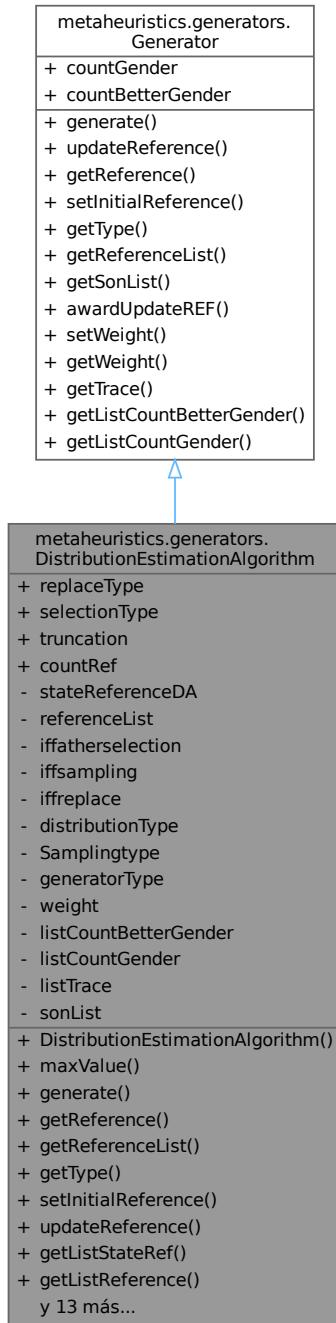
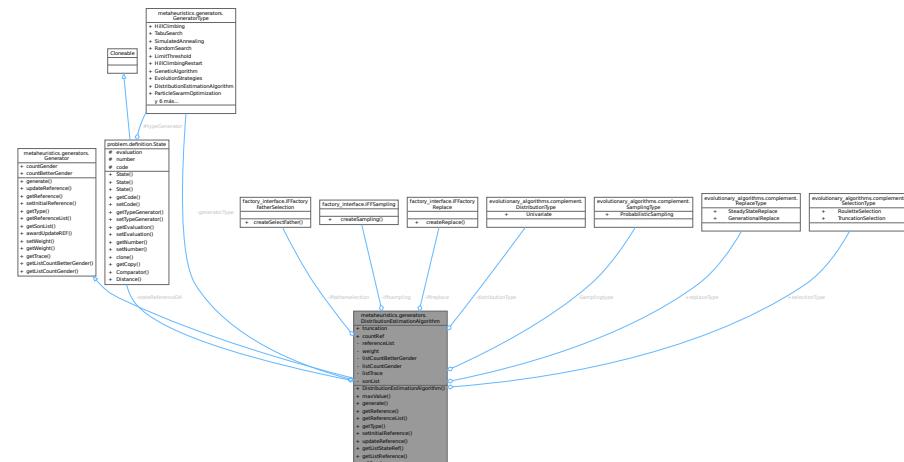


Diagrama de colaboración de `metaheuristics.generators.DistributionEstimationAlgorithm`:



## Métodos públicos

- **DistributionEstimationAlgorithm ()**  
*DistributionEstimationAlgorithm* - constructor for distribution estimation algorithms.
  - **State maxValue (List< State > listInd)**  
*maxValue* - method to find the state with the maximum value.
  - **State generate (Integer operatornumber) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException**  
*generate* - method to generate a new state.
  - **State getReference ()**  
*getReference* - method to get the current reference state.
  - **List< State > getReferenceList ()**  
*getReferenceList* - method to get the list of reference states.
  - **GeneratorType getType ()**  
*getType* - method to get the generator type.
  - **void setInitialReference (State stateInitialRef)**  
*setInitialReference* - method to set the initial reference state.
  - **void updateReference (State stateCandidate, Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException**  
*updateReference* - method to update the reference state.
  - **List< State > getListStateRef ()**  
*getListStateRef* - method to get the list of reference states.
  - **List< State > getListReference ()**  
*getListReference* - method to get the list of reference states.
  - **void setListReference (List< State > listReference)**  
*setListReference* - method to set the list of reference states.
  - **GeneratorType getGeneratorType ()**  
*getGeneratorType* - method to get the generator type.
  - **void setGeneratorType (GeneratorType generatorType)**  
*setGeneratorType* - method to set the generator type.

- List< [State](#) > [getfathersList](#) () throws [IllegalArgumentException](#), [SecurityException](#), [ClassNotFoundException](#), [InstantiationException](#), [IllegalAccessException](#), [InvocationTargetException](#), [NoSuchMethodException](#)  
*getfathersList - method to get the list of father states.*
- List< [State](#) > [getSonList](#) ()  
*getSonList - method to get the list of son states.*
- boolean [awardUpdateREF](#) ([State](#) stateCandidate)  
*awardUpdateREF - method to award the update of the reference state.*
- float [getWeight](#) ()  
*getWeight - method to get the weight.*
- void [setWeight](#) (float [weight](#))  
*setWeight - method to set the weight.*
- [DistributionType](#) [getDistributionType](#) ()  
*getDistributionType - method to get the distribution type.*
- void [setDistributionType](#) ([DistributionType](#) [distributionType](#))  
*setDistributionType - method to set the distribution type.*
- int[] [getListCountBetterGender](#) ()  
*getListCountBetterGender - method to get the list of count better gender.*
- int[] [getListCountGender](#) ()  
*getListCountGender - method to get the list of count gender.*
- float[] [getTrace](#) ()  
*getTrace - method to get the trace.*

### Atributos públicos estáticos

- static final [ReplaceType](#) [replaceType](#) = [ReplaceType.GenerationalReplace](#)
- static final [SelectionType](#) [selectionType](#) = [SelectionType.TruncationSelection](#)
- static final int [truncation](#) = 0
- static final int [countRef](#) = 0

### Atributos privados

- [State](#) [stateReferenceDA](#)
- List< [State](#) > [referenceList](#) = new ArrayList< [State](#) >()
- [IFFactoryFatherSelection](#) [iffatherselection](#)
- [IFFSampling](#) [iiffsampling](#)
- [IFFactoryReplace](#) [iffreplace](#)
- [DistributionType](#) [distributionType](#)
- [SamplingType](#) [Samplingtype](#)
- [GeneratorType](#) [generatorType](#)
- float [weight](#)
- int[] [listCountBetterGender](#) = new int[10]
- int[] [listCountGender](#) = new int[10]
- float[] [listTrace](#) = new float[1200000]

### Atributos estáticos privados

- static final List< [State](#) > [sonList](#) = new ArrayList< [State](#) >()

## Otros miembros heredados

### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- int [countGender](#)
- int [countBetterGender](#)

## 8.19.1 Descripción detallada

[DistributionEstimationAlgorithm](#) - class for distribution estimation algorithms.

Definición en la línea 30 del archivo [DistributionEstimationAlgorithm.java](#).

## 8.19.2 Documentación de constructores y destructores

### 8.19.2.1 DistributionEstimationAlgorithm()

```
metaheuristics.generators.DistributionEstimationAlgorithm.DistributionEstimationAlgorithm ()  
[inline]
```

[DistributionEstimationAlgorithm](#) - constructor for distribution estimation algorithms.

Definición en la línea 60 del archivo [DistributionEstimationAlgorithm.java](#).

```
00060         super();  
00061         this.referenceList = getListStateRef(); // llamada al método que devuelve la lista.  
00062         this.generatorType = GeneratorType.DistributionEstimationAlgorithm;  
00063         this.distributionType = DistributionType.Univariate;  
00064         this.Samplingtype = SamplingType.ProbabilisticSampling;  
00065         this.weight = 50;  
00066         listTrace[0] = weight;  
00067         listCountBetterGender[0] = 0;  
00068         listCountGender[0] = 0;  
00069     }  
00070 }
```

Hace referencia a [metaheuristics.generators.GeneratorType.DistributionEstimationAlgorithm](#), [getListStateRef\(\)](#), [listCountBetterGender](#), [listCountGender](#), [listTrace](#), [evolutionary\\_algorithms.complement.SamplingType.ProbabilisticSampling](#), [evolutionary\\_algorithms.complement.DistributionType.Univariate](#) y [weight](#).

Referenciado por [getListStateRef\(\)](#).

Gráfico de llamadas de esta función:

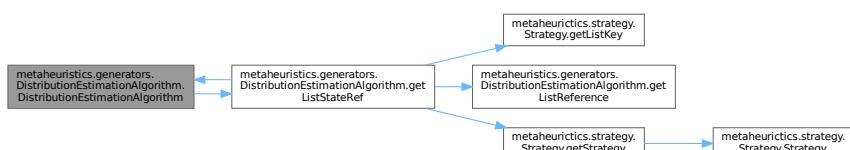


Gráfico de llamadas a esta función:



### 8.19.3 Documentación de funciones miembro

#### 8.19.3.1 awardUpdateREF()

```
boolean metaheuristics.generators.DistributionEstimationAlgorithm.awardUpdateREF (
    State stateCandidate) [inline]
```

awardUpdateREF - method to award the update of the reference state.

##### Parámetros

<i>stateCandidate</i>	<input type="button" value=""/>
-----------------------	---------------------------------

##### Devuelve

returns true if the candidate state is already in the reference list, false otherwise.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 269 del archivo [DistributionEstimationAlgorithm.java](#).

```
00269
00270     boolean find = false;
00271     int i = 0;
00272     while (find == false && i < this.referenceList.size()) {
00273         if(stateCandidate.equals(this.referenceList.get(i)))
00274             find = true;
00275         else i++;
00276     }
00277     return find;
00278 }
```

#### 8.19.3.2 generate()

```
State metaheuristics.generators.DistributionEstimationAlgorithm.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

generate - method to generate a new state.

##### Parámetros

<i>operatornumber</i>	<input type="button" value=""/>
-----------------------	---------------------------------

Devuelve

returns the generated state.

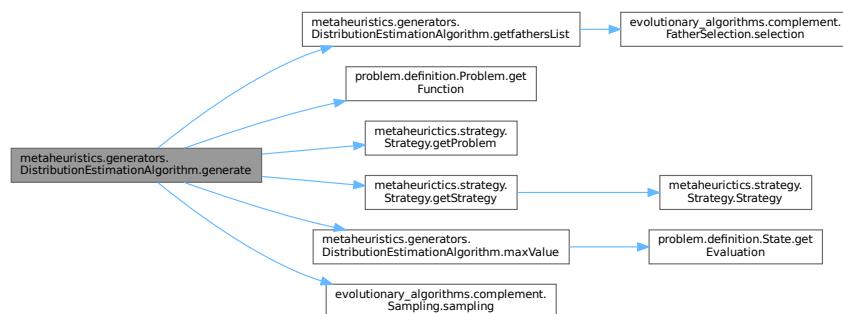
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 95 del archivo [DistributionEstimationAlgorithm.java](#).

```
00095
00096 {
00097     List<State> fathers = getfathersList();
00098     iffsampling = new FactorySampling();
00099     Sampling samplingG = iffsampling.createSampling(Samplingtype);
00100     List<State> ind = samplingG.sampling(fathers, operatornumber);
00101     State candidate = null;
00102     if(ind.size() > 1){
00103         for (int i = 0; i < ind.size(); i++) {
00104             double evaluation =
00105                 Strategy.getStrategy().getProblem().getFunction().get(0).Evaluation(ind.get(i));
00106             ArrayList<Double> listEval = new ArrayList<Double>();
00107             listEval.add(evaluation);
00108             ind.get(0).setEvaluation(listEval);
00109         }
00110         candidate = maxValue(ind);
00111     } else{
00112         candidate = ind.get(0);
00113     }
00114
00115     return candidate;
00116 }
00117 }
```

Hace referencia a [getfathersList\(\)](#), [problem.definition.Problem.getFunction\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [iffsampling](#), [maxValue\(\)](#), [evolutionary\\_algorithms.complement.Sampling.sampling\(\)](#) y [Samplingtype](#).

Gráfico de llamadas de esta función:



### 8.19.3.3 getDistributionType()

```
DistributionType metaheuristics.generators.DistributionEstimationAlgorithm.getDistributionType()
() [inline]
```

`getDistributionType` - method to get the distribution type.

Devuelve

returns the distribution type.

Definición en la línea 304 del archivo [DistributionEstimationAlgorithm.java](#).

```
00304
00305     return distributionType;
00306 }
```

Hace referencia a [distributionType](#).

### 8.19.3.4 getfathersList()

```
List< State > metaheuristics.generators.DistributionEstimationAlgorithm.getfathersList ()  
throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,  
IllegalAccessException, InvocationTargetException, NoSuchMethodException [inline]  
  
getfathersList - method to get the list of father states.
```

Devuelve

returns the list of father states.

Definición en la línea 246 del archivo [DistributionEstimationAlgorithm.java](#).

```
00246 {  
00247     List<State> refList = new ArrayList<State>(this.referenceList);  
00248     iffatherselection = new FactoryFatherSelection();  
00249     FatherSelection selection = iffatherselection.createSelectFather(selectionType);  
00250     List<State> fathers = selection.selection(refList, truncation);  
00251     return fathers;  
00252 }
```

Hace referencia a [iffatherselection](#), [evolutionary\\_algorithms.complement.FatherSelection.selection\(\)](#), [selectionType](#) y [truncation](#).

Referenciado por [generate\(\)](#).

Gráfico de llamadas de esta función:



Gráfico de llamadas a esta función:



### 8.19.3.5 getGeneratorType()

```
GeneratorType metaheuristics.generators.DistributionEstimationAlgorithm.getGeneratorType ()  
[inline]
```

getGeneratorType - method to get the generator type.

Devuelve

returns the generator type.

Definición en la línea 230 del archivo [DistributionEstimationAlgorithm.java](#).

```
00230 {  
00231     return generatorType;  
00232 }
```

Hace referencia a [generatorType](#).

### 8.19.3.6 getListCountBetterGender()

```
int[] metaheuristics.generators.DistributionEstimationAlgorithm.getListCountBetterGender ()  
[inline]
```

getListCountBetterGender - method to get the list of count better gender.

Devuelve

returns the list of count better gender.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 321 del archivo [DistributionEstimationAlgorithm.java](#).

```
00321 {  
00322     // TODO Auto-generated method stub  
00323     return (this.listCountBetterGender == null) ? new int[0] :  
00324         Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);  
00325 }
```

### 8.19.3.7 getListCountGender()

```
int[] metaheuristics.generators.DistributionEstimationAlgorithm.getListCountGender () [inline]
```

getListCountGender - method to get the list of count gender.

Devuelve

returns the list of count gender.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 331 del archivo [DistributionEstimationAlgorithm.java](#).

```
00331 {  
00332     // TODO Auto-generated method stub  
00333     return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,  
00334         this.listCountGender.length);  
00335 }
```

### 8.19.3.8 getListReference()

```
List< State > metaheuristics.generators.DistributionEstimationAlgorithm.getListReference ()  
[inline]
```

getListReference - method to get the list of reference states.

Devuelve

returns the list of reference states.

Definición en la línea 214 del archivo [DistributionEstimationAlgorithm.java](#).

```
00214 {  
00215     return referenceList;  
00216 }
```

Hace referencia a [referenceList](#).

Referenciado por [getListStateRef\(\)](#).

Gráfico de llamadas a esta función:



### 8.19.3.9 getListStateRef()

```
List< State > metaheuristics.generators.DistributionEstimationAlgorithm.getListStateRef ()
[inline]
```

getListStateRef - method to get the list of reference states.

Devuelve

returns the list of reference states.

Definición en la línea 189 del archivo [DistributionEstimationAlgorithm.java](#).

```
00189         Boolean found = false;
00190         List<String> key = Strategy.getStrategy().getListKey();
00191         int count = 0;
00192         while((found.equals(false)) && (Strategy.getStrategy().mapGenerators.size() > count)){
00193             if(key.get(count).equals(GeneratorType.DistributionEstimationAlgorithm.toString())){
00194                 GeneratorType keyGenerator = GeneratorType.valueOf(String.valueOf(key.get(count)));
00195                 DistributionEstimationAlgorithm generator =
00196                     (DistributionEstimationAlgorithm)Strategy.getStrategy().mapGenerators.get(keyGenerator);
00197                 if(generator.getListReference().isEmpty()){
00198                     referenceList.addAll(RandomSearch.listStateReference);
00199                 }
00200             else{
00201                 referenceList = generator.getListReference();
00202             }
00203             found = true;
00204         }
00205         count++;
00206     }
00207     return referenceList;
00208 }
```

Hace referencia a [DistributionEstimationAlgorithm\(\)](#), [metaheuristics.generators.GeneratorType.DistributionEstimationAlgorithm](#), [metaheuristics.strategy.Strategy.getListKey\(\)](#), [getListReference\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [metaheuristics.generators.RandomSearch.listStateReference](#), [metaheuristics.strategy.Strategy.mapGenerators](#) y [referenceList](#).

Referenciado por [DistributionEstimationAlgorithm\(\)](#).

Gráfico de llamadas de esta función:

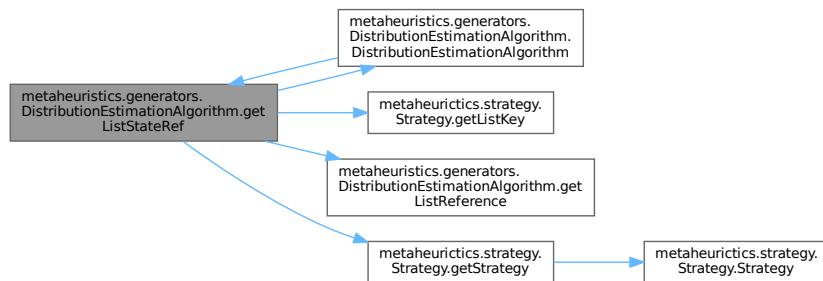


Gráfico de llamadas a esta función:



### 8.19.3.10 getReference()

```
State metaheuristics.generators.DistributionEstimationAlgorithm.getReference () [inline]
```

getReference - method to get the current reference state.

Devuelve

returns the current reference state.

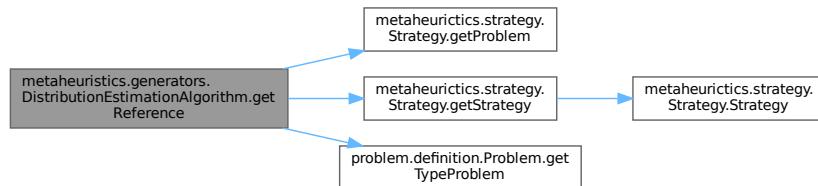
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 124 del archivo [DistributionEstimationAlgorithm.java](#).

```
00124         {
00125             stateReferenceDA = referenceList.get (0);
00126             if(Strategy.getStrategy ().getProblem ().getTypeProblem ().equals (ProblemType.Maximizar )) {
00127                 for (int i = 1; i < referenceList.size (); i++) {
00128                     if(stateReferenceDA.getEvaluation ().get (0) <
00129                         referenceList.get (i).getEvaluation ().get (0))
00130                         stateReferenceDA = referenceList.get (i);
00131                 }
00132             else{
00133                 for (int i = 1; i < referenceList.size (); i++) {
00134                     if(stateReferenceDA.getEvaluation ().get (0) >
00135                         referenceList.get (i).getEvaluation ().get (0))
00136                         stateReferenceDA = referenceList.get (i);
00137                 }
00138             }
00139         }
00139 }
```

Hace referencia a [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Problem.getTypeProblem\(\)](#), [problem.definition.Problem.ProblemType.Maximizar](#), [referenceList](#) y [stateReferenceDA](#).

Gráfico de llamadas de esta función:



### 8.19.3.11 getReferenceList()

```
List< State > metaheuristics.generators.DistributionEstimationAlgorithm.getReferenceList () [inline]
```

getReferenceList - method to get the list of reference states.

**Devuelve**

returns the list of reference states.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 146 del archivo [DistributionEstimationAlgorithm.java](#).

```
00146             {
00147     List<State> ReferenceList = new ArrayList<State>();
00148     for (int i = 0; i < referenceList.size(); i++) {
00149         State value = referenceList.get(i);
00150         ReferenceList.add( value );
00151     }
00152     return ReferenceList;
00153 }
```

Hace referencia a [referenceList](#).

**8.19.3.12 getSonList()**

```
List< State > metaheuristics.generators.DistributionEstimationAlgorithm.getSonList () [inline]
```

getSonList - method to get the list of son states.

**Devuelve**

returns the list of son states.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 259 del archivo [DistributionEstimationAlgorithm.java](#).

```
00259             {
00260     // return a defensive copy to avoid exposing internal mutable list
00261     return new ArrayList<State>(sonList);
00262 }
```

Hace referencia a [sonList](#).

**8.19.3.13 getTrace()**

```
float[ ] metaheuristics.generators.DistributionEstimationAlgorithm.getTrace () [inline]
```

getTrace - method to get the trace.

**Devuelve**

returns the trace.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 341 del archivo [DistributionEstimationAlgorithm.java](#).

```
00341             {
00342     // TODO Auto-generated method stub
00343     return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00344         this.listTrace.length);
00344 }
```

### 8.19.3.14 `getType()`

```
GeneratorType metaheuristics.generators.DistributionEstimationAlgorithm.getType () [inline]
```

`getType` - method to get the generator type.

**Devuelve**

returns the generator type.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 160 del archivo [DistributionEstimationAlgorithm.java](#).

```
00160                     {
00161             return this.generatorType;
00162         }
```

### 8.19.3.15 `getWeight()`

```
float metaheuristics.generators.DistributionEstimationAlgorithm.getWeight () [inline]
```

`getWeight` - method to get the weight.

**Devuelve**

returns the weight.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 285 del archivo [DistributionEstimationAlgorithm.java](#).

```
00285                     {
00286             // TODO Auto-generated method stub
00287             return 0;
00288         }
```

### 8.19.3.16 `maxValue()`

```
State metaheuristics.generators.DistributionEstimationAlgorithm maxValue (
    List< State > listInd) [inline]
```

`maxValue` - method to find the state with the maximum value.

#### Parámetros

<i>listInd</i>	<input type="text"/>
----------------	----------------------

**Devuelve**

returns the state with the maximum evaluation value.

Definición en la línea 77 del archivo [DistributionEstimationAlgorithm.java](#).

```
00077         {
00078     State state = new State(listInd.get(0));
00079     double max = state.getEvaluation().get(0);
00080     for (int i = 1; i < listInd.size(); i++) {
00081         if (listInd.get(i).getEvaluation().get(0) > max) {
00082             max = listInd.get(i).getEvaluation().get(0);
00083             state = new State(listInd.get(i));
00084         }
00085     }
00086     return state;
00087 }
```

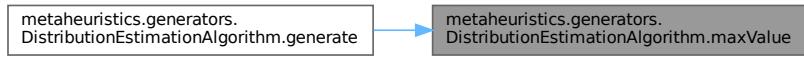
Hace referencia a [problem.definition.State.getEvaluation\(\)](#).

Referenciado por [generate\(\)](#).

Gráfico de llamadas de esta función:



Gráfico de llamadas a esta función:



### 8.19.3.17 setDistributionType()

```
void metaheuristics.generators.DistributionEstimationAlgorithm.setDistributionType (
    DistributionType distributionType) [inline]
```

`setDistributionType` - method to set the distribution type.

#### Parámetros

<code>distributionType</code>	the distribution type to set
-------------------------------	------------------------------

Definición en la línea 312 del archivo [DistributionEstimationAlgorithm.java](#).

```
00312
00313     this.distributionType = distributionType;
00314 }
```

Hace referencia a [distributionType](#).

### 8.19.3.18 setGeneratorType()

```
void metaheuristics.generators.DistributionEstimationAlgorithm.setGeneratorType (
    GeneratorType generatorType) [inline]
```

setGeneratorType - method to set the generator type.

#### Parámetros

<i>generatorType</i>	<input type="button" value=""/>
----------------------	---------------------------------

Definición en la línea 238 del archivo [DistributionEstimationAlgorithm.java](#).

```
00238
00239     this.generatorType = generatorType;
00240 }
```

Hace referencia a [generatorType](#).

### 8.19.3.19 setInitialReference()

```
void metaheuristics.generators.DistributionEstimationAlgorithm.setInitialReference (
    State stateInitialRef) [inline]
```

setInitialReference - method to set the initial reference state.

#### Parámetros

<i>stateInitialRef</i>	<input type="button" value=""/>
------------------------	---------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 169 del archivo [DistributionEstimationAlgorithm.java](#).

```
00169
00170     this.stateReferenceDA = stateInitialRef;
00171 }
```

### 8.19.3.20 setListReference()

```
void metaheuristics.generators.DistributionEstimationAlgorithm.setListReference (
    List< State > listReference) [inline]
```

setListReference - method to set the list of reference states.

#### Parámetros

<i>listReference</i>	<input type="button" value=""/>
----------------------	---------------------------------

Definición en la línea 222 del archivo [DistributionEstimationAlgorithm.java](#).

```
00222
00223     referenceList = listReference;
00224 }
```

## Parámetros

<i>weight</i>	
---------------	--

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 295 del archivo [DistributionEstimationAlgorithm.java](#).

```
00295         {
00296             // TODO Auto-generated method stub
00297
00298     }
```

Hace referencia a [weight](#).

### 8.19.3.22 updateReference()

```
void metaheuristics.generators.DistributionEstimationAlgorithm.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

`updateReference` - method to update the reference state.

## Parámetros

<i>stateCandidate</i>	
<i>countIterationsCurrent</i>	

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 179 del archivo [DistributionEstimationAlgorithm.java](#).

```
00179
{
00180     iffreplace = new FactoryReplace();
00181     Replace replace = iffreplace.createReplace(replaceType);
00182     referenceList = replace.replace(stateCandidate, referenceList);
00183 }
```

Hace referencia a [iffreplace](#), [referenceList](#), [evolutionary\\_algorithms.complement.Replace.replace\(\)](#) y [replaceType](#).

Gráfico de llamadas de esta función:



## 8.19.4 Documentación de datos miembro

### 8.19.4.1 countRef

```
final int metaheuristics.generators.DistributionEstimationAlgorithm.countRef = 0 [static]
```

Definición en la línea 48 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [metaheuristics.generators.RandomSearch.generate\(\)](#), [metaheuristics.strategy.Strategy.update\(\)](#) y [local\\_search.complement.UpdateParameter.updateParameter\(\)](#).

### 8.19.4.2 distributionType

```
DistributionType metaheuristics.generators.DistributionEstimationAlgorithm.distributionType  
[private]
```

Definición en la línea 38 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [getDistributionType\(\)](#) y [setDistributionType\(\)](#).

### 8.19.4.3 generatorType

```
GeneratorType metaheuristics.generators.DistributionEstimationAlgorithm.generatorType [private]
```

Definición en la línea 45 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [getGeneratorType\(\)](#) y [setGeneratorType\(\)](#).

### 8.19.4.4 iffatherselection

```
IFFactoryFatherSelection metaheuristics.generators.DistributionEstimationAlgorithm.iffatherselection  
[private]
```

Definición en la línea 35 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [getfathersList\(\)](#).

### 8.19.4.5 iffreplace

```
IFFactoryReplace metaheuristics.generators.DistributionEstimationAlgorithm.iffreplace [private]
```

Definición en la línea 37 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.19.4.6 iffSampling

```
IFFSampling metaheuristics.generators.DistributionEstimationAlgorithm.iffSampling [private]
```

Definición en la línea 36 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [generate\(\)](#).

#### 8.19.4.7 listCountBetterGender

```
int [] metaheuristics.generators.DistributionEstimationAlgorithm.listCountBetterGender = new int[10] [private]
```

Definición en la línea 52 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [DistributionEstimationAlgorithm\(\)](#).

#### 8.19.4.8 listCountGender

```
int [] metaheuristics.generators.DistributionEstimationAlgorithm.listCountGender = new int[10] [private]
```

Definición en la línea 53 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [DistributionEstimationAlgorithm\(\)](#).

#### 8.19.4.9 listTrace

```
float [] metaheuristics.generators.DistributionEstimationAlgorithm.listTrace = new float[1200000] [private]
```

Definición en la línea 54 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [DistributionEstimationAlgorithm\(\)](#).

#### 8.19.4.10 referenceList

```
List<State> metaheuristics.generators.DistributionEstimationAlgorithm.referenceList = new ArrayList<State>() [private]
```

Definición en la línea 33 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [getListReference\(\)](#), [getListStateRef\(\)](#), [getReference\(\)](#), [getReferenceList\(\)](#), [setListReference\(\)](#) y [updateReference\(\)](#).

#### 8.19.4.11 replaceType

```
final ReplaceType metaheuristics.generators.DistributionEstimationAlgorithm.replaceType = ReplaceType.GenerationalReplace [static]
```

Definición en la línea 42 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.19.4.12 Samplingtype

```
SamplingType metaheuristics.generators.DistributionEstimationAlgorithm.Samplingtype [private]
```

Definición en la línea 39 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [generate\(\)](#).

#### 8.19.4.13 selectionType

```
final SelectionType metaheuristics.generators.DistributionEstimationAlgorithm.selectionType =  
SelectionType.TruncationSelection [static]
```

Definición en la línea 43 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [getfathersList\(\)](#).

#### 8.19.4.14 sonList

```
final List<State> metaheuristics.generators.DistributionEstimationAlgorithm.sonList = new  
ArrayList<State>() [static], [private]
```

Definición en la línea 34 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [getSonList\(\)](#).

#### 8.19.4.15 stateReferenceDA

```
State metaheuristics.generators.DistributionEstimationAlgorithm.stateReferenceDA [private]
```

Definición en la línea 32 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [getReference\(\)](#).

#### 8.19.4.16 truncation

```
final int metaheuristics.generators.DistributionEstimationAlgorithm.truncation = 0 [static]
```

Definición en la línea 47 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [getfathersList\(\)](#).

#### 8.19.4.17 weight

```
float metaheuristics.generators.DistributionEstimationAlgorithm.weight [private]
```

Definición en la línea 49 del archivo [DistributionEstimationAlgorithm.java](#).

Referenciado por [DistributionEstimationAlgorithm\(\)](#) y [setWeight\(\)](#).

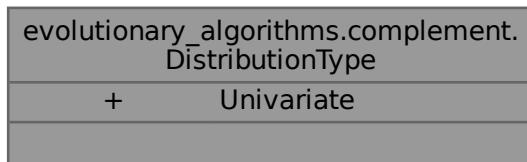
La documentación de esta clase está generada del siguiente archivo:

- [DistributionEstimationAlgorithm.java](#)

## 8.20 Referencia de la enumeración `evolutionary_algorithms.complement.DistributionType`

[DistributionType](#) - descripción (añade detalles).

Diagrama de colaboración de `evolutionary_algorithms.complement.DistributionType`:



### Atributos públicos

- [Univariate](#)

### 8.20.1 Descripción detallada

[DistributionType](#) - descripción (añade detalles).

Definición en la línea 6 del archivo [DistributionType.java](#).

### 8.20.2 Documentación de datos miembro

#### 8.20.2.1 Univariate

`evolutionary_algorithms.complement.DistributionType.Univariate`

Definición en la línea 7 del archivo [DistributionType.java](#).

Referenciado por [metaheuristics.generators.DistributionEstimationAlgorithm.DistributionEstimationAlgorithm\(\)](#).

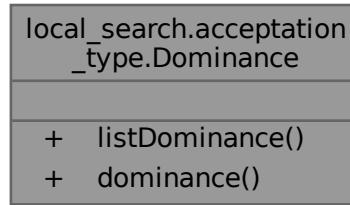
La documentación de esta enumeración está generada del siguiente archivo:

- [DistributionType.java](#)

## 8.21 Referencia de la clase local\_search.acceptation\_type.Dominance

[Dominance](#) - Utilities for Pareto dominance comparisons.

Diagrama de colaboración de local\_search.acceptation\_type.Dominance:



### Métodos públicos

- boolean [listDominance](#) ([State](#) solutionX, List< [State](#) > list)  
*Check whether solutionX dominates members of the supplied list and update the list accordingly.*
- boolean [dominance](#) ([State](#) solutionX, [State](#) solutionY)  
*Determine whether solutionX Pareto-dominates solutionY.*

### 8.21.1 Descripción detallada

[Dominance](#) - Utilities for Pareto dominance comparisons.

Provides helpers to test dominance relations between multi-objective solution states and to maintain a Pareto front (list of non-dominated solutions).

Definición en la línea 18 del archivo [Dominance.java](#).

### 8.21.2 Documentación de funciones miembro

#### 8.21.2.1 dominance()

```
boolean local_search.acceptation_type.Dominance.dominance (
    State solutionX,
    State solutionY) [inline]
```

Determine whether solutionX Pareto-dominates solutionY.

#### Parámetros

<code>solutionX</code>	the candidate solution
<code>solutionY</code>	the reference solution

**Devuelve**

true if solutionX dominates solutionY according to the problem objective type, false otherwise

Definición en la línea 87 del archivo [Dominance.java](#).

```

00087
00088     boolean dominance = false;
00089     int countBest = 0;
00090     int countEquals = 0;
00091     //Si solutionX domina a solutionY
00092     if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)) {
00093         //Recorriendo las evaluaciones de las funciones objetivo
00094         for (int i = 0; i < solutionX.getEvaluation().size(); i++) {
00095             if(solutionX.getEvaluation().get(i).floatValue() >
00096                 solutionY.getEvaluation().get(i).floatValue())
00097                 countBest++;
00098             if(solutionX.getEvaluation().get(i).floatValue() ==
00099                 solutionY.getEvaluation().get(i).floatValue())
00100                 countEquals++;
00101         }
00102     }
00103     else{
00104         //Recorriendo las evaluaciones de las funciones objetivo
00105         for (int i = 0; i < solutionX.getEvaluation().size(); i++) {
00106             if(solutionX.getEvaluation().get(i).floatValue() <
00107                 solutionY.getEvaluation().get(i).floatValue())
00108                 countBest++;
00109             if(solutionX.getEvaluation().get(i).floatValue() ==
00110                 solutionY.getEvaluation().get(i).floatValue())
00111                 countEquals++;
00112         }
00113     }
00114     if((countBest >= 1) && (countEquals + countBest == solutionX.getEvaluation().size())) {
00115         dominance = true;
00116     }
00117     return dominance;
00118 }
```

Hace referencia a [dominance\(\)](#), [problem.definition.State.getEvaluation\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Problem.getTypeProblem\(\)](#) y [problem.definition.Problem.ProblemType](#).

Referenciado por [local\\_search.acceptation\\_type.AcceptMulticase.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotDominated\(\)](#), [local\\_search.acceptation\\_type.AcceptMulticase.dominanceCounter\(\)](#), [local\\_search.acceptation\\_type.AcceptMulticase.countListDominance\(\)](#) y [local\\_search.candidate\\_type.NotDominatedCandidate.stateSearch\(\)](#).

Gráfico de llamadas de esta función:

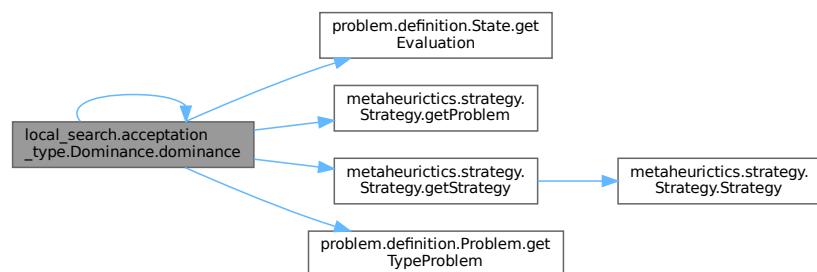
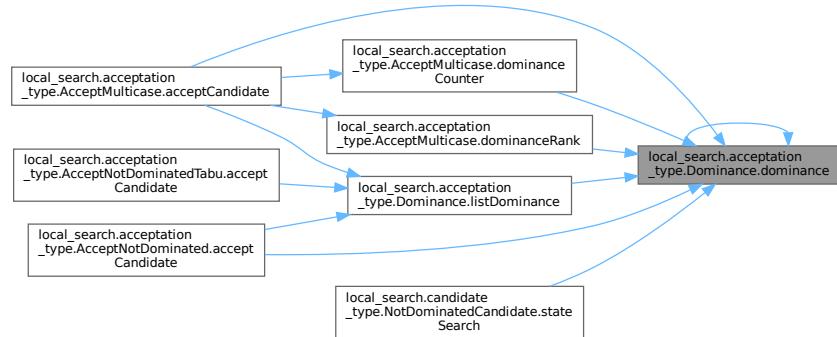


Gráfico de llamadas a esta función:



### 8.21.2.2 listDominance()

```
boolean local_search.acceptation_type.Dominance.listDominance (
    State solutionX,
    List< State > list) [inline]
```

Check whether solutionX dominates members of the supplied list and update the list accordingly.

If solutionX is not dominated and not present it is added.

#### Parámetros

<i>solutionX</i>	the candidate solution to test
<i>list</i>	the list of non-dominated solutions (Pareto front)

#### Devuelve

true if the solution was added to the list (i.e. it is non-dominated and not duplicate), false otherwise

Definición en la línea 33 del archivo [Dominance.java](#).

```

00033
00034     boolean domain = false;
00035     for (int i = 0; i < list.size() && domain == false; i++) {
00036         //Si la soluci&on X domina a la soluci&on de la lista
00037         if(dominance(solutionX, list.get(i)) == true){
00038             //Se elimina el elemento de la lista
00039             list.remove(i);
00040             if (i!=0) {
00041                 i--;
00042             }
00043             if
00044                 (Strategy.getStrategy().generator.getType() .equals(GeneratorType.MultiobjectiveHillClimbingDistance)&&list.size() !=0)
00045             {
00046                 MultiobjectiveHillClimbingDistance.distanceCalculateAdd(list);
00047             }
00048             if (list.size()>0) {
00049                 if(dominance(list.get(i), solutionX) == true){
00050                     domain = true;
00051                 }
00052             }
}

```

```

00053         }
00054         //Si la solución X no fue dominada
00055         if(domain == false){
00056             //Comprobando que la solución no exista
00057             boolean found = false;
00058             for (int k = 0; k < list.size() && found == false; k++) {
00059                 State element = list.get(k);
00060                 found = solutionX.Comparator(element);
00061             }
00062             //Si la solución no existe
00063             if(found == false){
00064                 //Se guarda la solución candidata en la lista de soluciones óptimas de Pareto
00065                 list.add(solutionX.clone());
00066                 if
00067                     (Strategy.getStrategy().generator.getType().equals(GeneratorType.MultiobjectiveHillClimbingDistance))
00068                 {
00069                     MultiobjectiveHillClimbingDistance.distanceCalculateAdd(list);
00070                 }
00071             }
00072         return false;
00073     }
00074 }
00075 }
```

Hace referencia a [problem.definition.State.clone\(\)](#), [problem.definition.State.Comparator\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance.dominance\(\)](#), [metaheuristics.strategy.Strategy.generator](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [metaheuristics.generators.GeneratorType](#) y [metaheuristics.generators.MultiobjectiveHillClimbingDistance](#).

Referenciado por [local\\_search.acceptation\\_type.AcceptMulticase.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotDominated\(\)](#) y [local\\_search.acceptation\\_type.AcceptNotDominatedTabu.acceptCandidate\(\)](#).

Gráfico de llamadas de esta función:

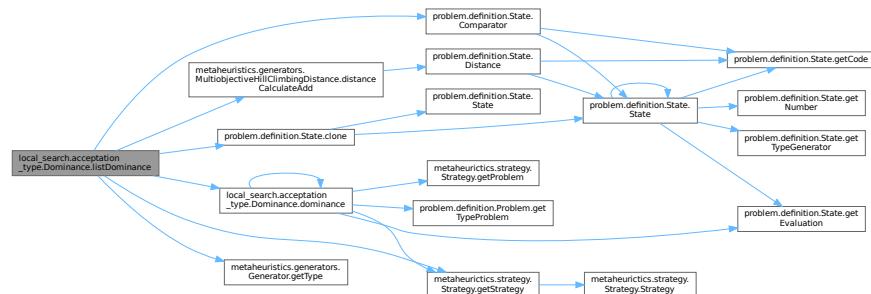
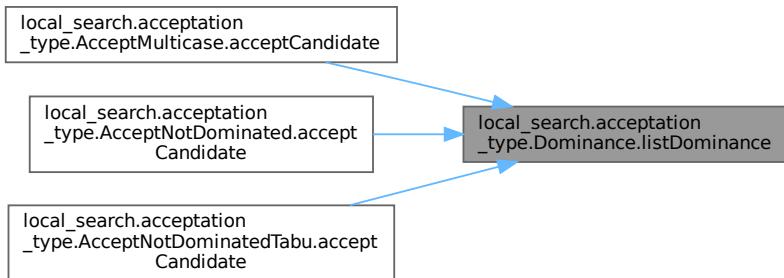


Gráfico de llamadas a esta función:



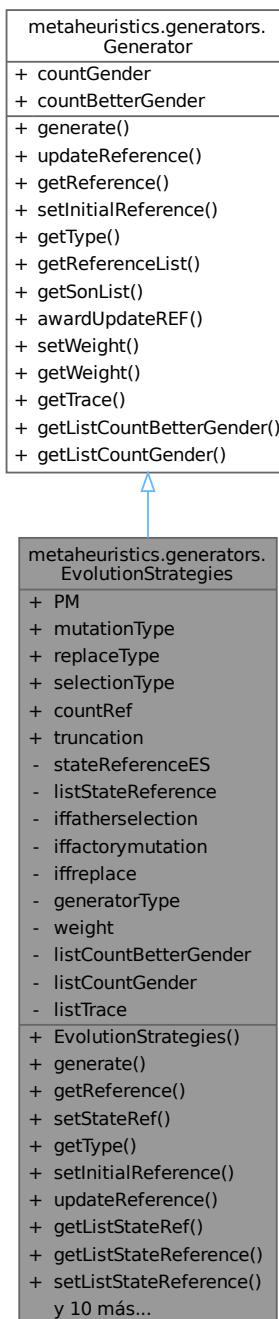
La documentación de esta clase está generada del siguiente archivo:

- [Dominance.java](#)

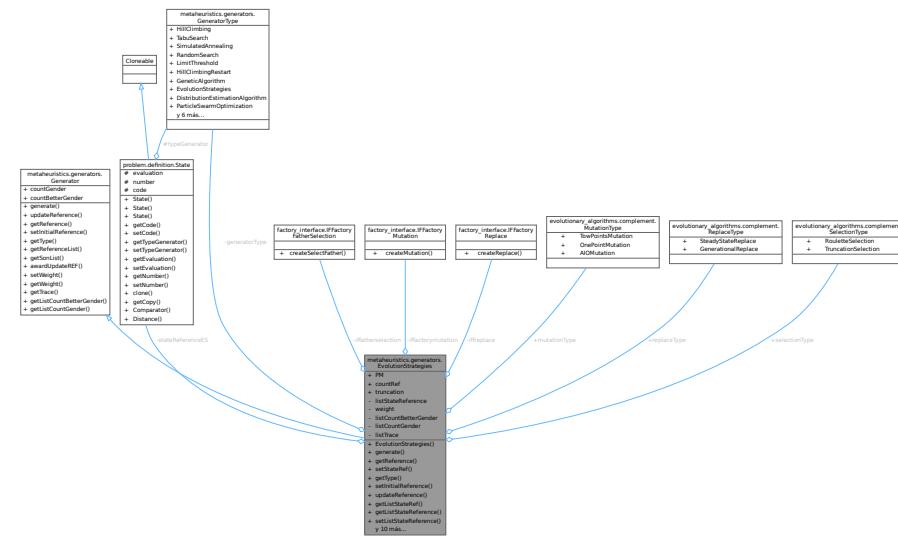
## 8.22 Referencia de la clase metaheuristics.generators.EvolutionStrategies

[EvolutionStrategies](#) - class that implements the Evolution Strategies generator.

Diagrama de herencia de metaheuristics.generators.EvolutionStrategies



## Diagrama de colaboración de metaheuristics.generators.EvolutionStrategies:



## Métodos públicos

- EvolutionStrategies ()

*EvolutionStrategies* - method to create an instance of *EvolutionStrategies*.

- `State generate (Integer operatornumber)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.*

- State getReference ()

*getReference - method to get the current reference state.*

- void **setStateRef** (State stateRef)

*setStateRef - method to set the current reference state*

- `GeneratorType getType()`

*getType - method to get the generator type*

- void **setInitialReference** (**State** stateInitialRef)

*setInitialReference* – method to set the initial reference state

- void `updateReference` (State stateCandidate, Integer countIterationsCurrent) throws `IllegalArgumentException`, `Exception`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*updateReference - method to update the reference state*

- List< State > getIlistStateRef ()

*getListStateRef - method to get the list of reference states*

- List< State > getListStateReference ()

`getListOfStateReferences` method to get the list of reference states.

- void setListStateReference (List< State > listStateReference)

This is the first page of the list of state references.

- ### **setListStateReference - Method to set**

`setTimeGenerator`—method to set the generator time

- `void setTypeGenerator (GeneratorType generatorType)`

• **Set-type generator** (Generator type generality)

- `set typeGenerator - method to set`

- `getReferenceList` - method to get the list of reference states.
- `List< State > getSonList ()`  
*getSonList - method to get the list of son states.*
- `boolean awardUpdateREF (State stateCandidate)`  
*awardUpdateREF - method to award the update of the reference state.*
- `float getWeight ()`  
*getWeight - method to get the weight.*
- `void setWeight (float weight)`  
*setWeight - method to set the weight.*
- `int[] getListCountBetterGender ()`  
*getListCountBetterGender - method to get the list of count better gender.*
- `int[] getListCountGender ()`  
*getListCountGender - method to get the list of count gender.*
- `float[] getTrace ()`  
*getTrace - method to get the trace.*

### Atributos públicos estáticos

- `static final double PM = 0.0`
- `static final MutationType mutationType = MutationType.OnePointMutation`
- `static final ReplaceType replaceType = ReplaceType.GenerationalReplace`
- `static final SelectionType selectionType = SelectionType.TruncationSelection`
- `static final int countRef = 0`
- `static final int truncation = 0`

### Atributos privados

- `State stateReferenceES`
- `List< State > listStateReference = new ArrayList<State>()`
- `IFFactoryFatherSelection iffatherselection`
- `IFFactoryMutation iffactorymutation`
- `IFFactoryReplace iffreplace`
- `GeneratorType generatorType`
- `float weight = 50`
- `int[] listCountBetterGender = new int[10]`
- `int[] listCountGender = new int[10]`
- `float[] listTrace = new float[1200000]`

### Otros miembros heredados

#### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- `int countGender`
- `int countBetterGender`

### 8.22.1 Descripción detallada

[EvolutionStrategies](#) - class that implements the Evolution Strategies generator.

Definición en la línea 29 del archivo [EvolutionStrategies.java](#).

## 8.22.2 Documentación de constructores y destructores

### 8.22.2.1 EvolutionStrategies()

```
metaheuristics.generators.EvolutionStrategies.EvolutionStrategies () [inline]
```

[EvolutionStrategies](#) - method to create an instance of [EvolutionStrategies](#).

Definición en la línea 54 del archivo [EvolutionStrategies.java](#).

```
00054      super();  
00055      this.listStateReference = getListStateRef();  
00056      this.generatorType = GeneratorType.EvolutionStrategies;  
00057      this.weight = 50;  
00058      listTrace[0] = this.weight;  
00059      listCountBetterGender[0] = 0;  
00060      listCountGender[0] = 0;  
00061  }  
00062 }
```

Hace referencia a [metaheuristics.generators.GeneratorType.EvolutionStrategies](#), [getListStateRef\(\)](#), [listCountBetterGender](#), [listCountGender](#) y [listTrace](#).

Referenciado por [getListStateRef\(\)](#).

Gráfico de llamadas de esta función:

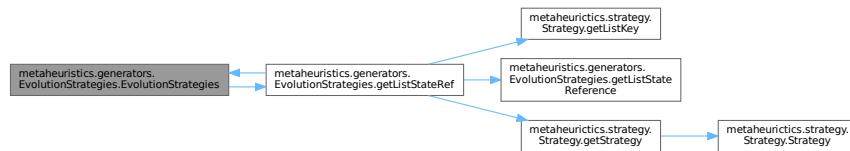


Gráfico de llamadas a esta función:



## 8.22.3 Documentación de funciones miembro

### 8.22.3.1 awardUpdateREF()

```
boolean metaheuristics.generators.EvolutionStrategies.awardUpdateREF (  
    State stateCandidate) [inline]
```

[awardUpdateREF](#) - method to award the update of the reference state.

## Parámetros

<i>stateCandidate</i>	<input type="button" value=""/>
-----------------------	---------------------------------

### Devuelve

returns true if the update is awarded, false otherwise.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 247 del archivo [EvolutionStrategies.java](#).

```
00247         // TODO Auto-generated method stub
00248         return false;
00249     }
00250 }
```

### 8.22.3.2 generate()

```
State metaheuristics.generators.EvolutionStrategies.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.

This RNG is used to decide whether a mutation occurs in the evolutionary algorithm and is not used for security-sensitive purposes. Suppress Sonar security hotspot S2245 for this usage. generate - method to generate a new state.

## Parámetros

<i>operatornumber</i>	<input type="button" value=""/>
-----------------------	---------------------------------

### Devuelve

returns the generated state.

Reimplementado de [metaheuristics.generators.Generator](#).

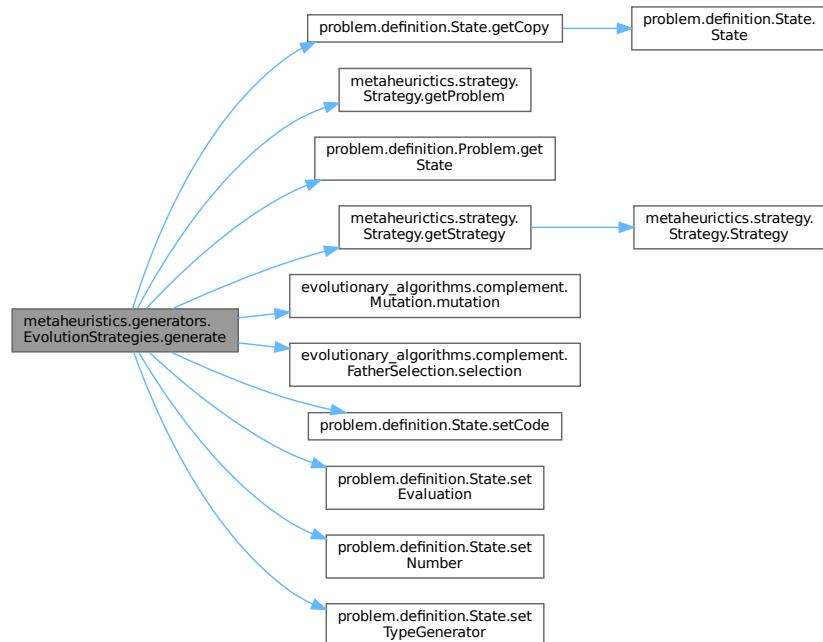
Definición en la línea 76 del archivo [EvolutionStrategies.java](#).

```
00076 {
00077     iffatherselection = new FactoryFatherSelection();
00078     FatherSelection selection = iffatherselection.createSelectFather(selectionType);
00079     List<State> fathers = selection.selection(this.listStateReference, truncation);
00080     int pos1 = (fathers.size() > 0) ? ThreadLocalRandom.current().nextInt(fathers.size()) : 0;
00081     State candidate = (State) Strategy.getStrategy().getProblem().getState().getCopy();
00082     candidate.setCode(new ArrayList<Object>(fathers.get(pos1).getCode()));
00083     candidate.setEvaluation(fathers.get(pos1).getEvaluation());
00084     candidate.setNumber(fathers.get(pos1).getNumber());
00085     candidate.setTypeGenerator(fathers.get(pos1).getTypeGenerator());
00086
00087     //*****mutacion*****
00088     iffactorymutation = new FactoryMutation();
00089     Mutation mutation = iffactorymutation.createMutation(mutationType);
00090     candidate = mutation.mutation(candidate, PM);
00091     //list.add(candidate);
00092
00093     return candidate;
```

```
00094 }
```

Hace referencia a `problem.definition.State.getCopy()`, `metaheuristics.strategy.Strategy.getProblem()`, `problem.definition.Problem.getState()`, `metaheuristics.strategy.Strategy.getStrategy()`, `iffactorymutation`, `iffatherselection`, `listStateReference`, `evolutionary_algorithms.complement.MutationType`, `PM`, `evolutionary_algorithms.complement.FatherSelection.selection()`, `selectionType`, `problem.definition.State.setCode()`, `problem.definition.State.setEvaluation()`, `problem.definition.State.setNumber()`, `problem.definition.State.setTypeGenerator()` y `truncation`.

Gráfico de llamadas de esta función:



### 8.22.3.3 getListCountBetterGender()

```
int[] metaheuristics.generators.EvolutionStrategies.getListCountBetterGender () [inline]
```

`getListCountBetterGender` - method to get the list of count better gender.

Devuelve

Reimplementado de `metaheuristics.generators.Generator`.

Definición en la línea 277 del archivo `EvolutionStrategies.java`.

```
00277
00278     // TODO Auto-generated method stub
00279     return (this.listCountBetterGender == null) ? new int[0] :
00280         Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
```

### 8.22.3.4 getListCountGender()

```
int[] metaheuristics.generators.EvolutionStrategies.getListCountGender () [inline]
```

getListCountGender - method to get the list of count gender.

#### Devuelve

returns the list of count gender.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 287 del archivo [EvolutionStrategies.java](#).

```
00287      {
00288          // TODO Auto-generated method stub
00289          return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00290              this.listCountGender.length);
00290      }
```

### 8.22.3.5 getListStateRef()

```
List< State > metaheuristics.generators.EvolutionStrategies.getListStateRef () [inline]
```

getListStateRef - method to get the list of reference states.

#### Devuelve

returns the list of reference states.

Definición en la línea 164 del archivo [EvolutionStrategies.java](#).

```
00164      {
00165          Boolean found = false;
00166          List<String> key = Strategy.getStrategy().getListKey ();
00167          int count = 0;
00168          while((found.equals(false) && (Strategy.getStrategy().mapGenerators.size() > count)){
00169              if(key.get(count).equals(GeneratorType.EvolutionStrategies.toString())){
00170                  GeneratorType keyGenerator = GeneratorType.valueOf(String.valueOf(key.get(count)));
00171                  EvolutionStrategies generator = (EvolutionStrategies)
00172                      Strategy.getStrategy().mapGenerators.get(keyGenerator);
00173                  if(generator.getListStateReference().isEmpty()){
00174                      listStateReference.addAll(RandomSearch.listStateReference);
00175                  }
00176                  else{
00177                      listStateReference = generator.getListStateReference();
00178                  }
00179                  found = true;
00180              }
00181              count++;
00182          }
00183          return (listStateReference == null) ? new ArrayList<State>() : new
ArrayList<State>(listStateReference);
00183      }
```

Hace referencia a [EvolutionStrategies\(\)](#), [metaheuristics.generators.GeneratorType.EvolutionStrategies](#), [metaheuristics.strategy.Strategy.getListStateReference\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [listStateReference](#), [metaheuristics.generators.RandomSearch](#) y [metaheuristics.strategy.Strategy.mapGenerators](#).

Referenciado por [EvolutionStrategies\(\)](#).

Gráfico de llamadas de esta función:

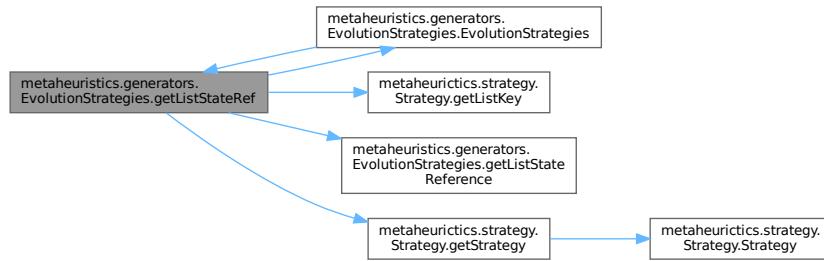


Gráfico de llamadas a esta función:



### 8.22.3.6 getListStateReference()

List< [State](#) > metaheuristics.generators.EvolutionStrategies.getListStateReference () [inline]

[getListStateReference](#) - method to get the list of reference states.

**Devuelve**

returns the list of reference states.

Definición en la línea 189 del archivo [EvolutionStrategies.java](#).

```

00189
00190     return (listStateReference == null) ? new ArrayList<State>() : new
00191     ArrayList<State>(listStateReference);
  
```

Hace referencia a [listStateReference](#).

Referenciado por [getListStateRef\(\)](#).

Gráfico de llamadas a esta función:



### 8.22.3.7 getReference()

```
State metaheuristics.generators.EvolutionStrategies.getReference () [inline]
```

getReference - method to get the current reference state.

Devuelve

returns the current reference state.

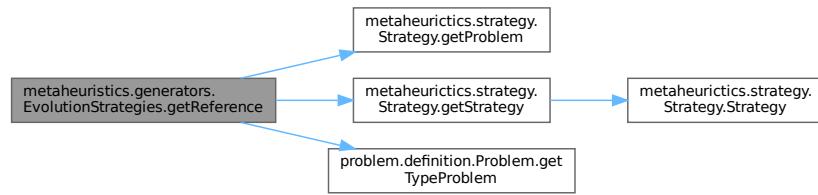
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 101 del archivo [EvolutionStrategies.java](#).

```
00101         {
00102             if (listStateReference == null || listStateReference.isEmpty()) {
00103                 return null;
00104             }
00105             stateReferenceES = listStateReference.get(0);
00106             if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00107                 for (int i = 1; i < listStateReference.size(); i++) {
00108                     if(stateReferenceES.getEvaluation().get(0) <
00109                         listStateReference.get(i).getEvaluation().get(0))
00110                         stateReferenceES = listStateReference.get(i);
00111                 }
00112             else{
00113                 for (int i = 1; i < listStateReference.size(); i++) {
00114                     if(stateReferenceES.getEvaluation().get(0) >
00115                         listStateReference.get(i).getEvaluation().get(0))
00116                         stateReferenceES = listStateReference.get(i);
00117                 }
00118             }
00119         }
00120     }
```

Hace referencia a [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Problem.getTypeProblem\(\)](#), [listStateReference](#), [problem.definition.Problem.ProblemType.Maximizar](#) y [stateReferenceES](#).

Gráfico de llamadas de esta función:



### 8.22.3.8 getReferenceList()

```
List< State > metaheuristics.generators.EvolutionStrategies.getReferenceList () [inline]
```

getReferenceList - method to get the list of reference states.

**Devuelve**

returns the list of reference states.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 222 del archivo [EvolutionStrategies.java](#).

```
00222           {
00223             List<State> ReferenceList = new ArrayList<State>();
00224             for (int i = 0; i < listStateReference.size(); i++) {
00225               State value = listStateReference.get(i);
00226               ReferenceList.add(value);
00227             }
00228             return new ArrayList<State>(ReferenceList);
00229 }
```

Hace referencia a [listStateReference](#).

**8.22.3.9 getSonList()**

```
List< State > metaheuristics.generators.EvolutionStrategies.getSonList () [inline]
```

getSonList - method to get the list of son states.

**Devuelve**

returns the list of son states.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 236 del archivo [EvolutionStrategies.java](#).

```
00236           {
00237             // TODO Auto-generated method stub
00238             return null;
00239 }
```

**8.22.3.10 getTrace()**

```
float[] metaheuristics.generators.EvolutionStrategies.getTrace () [inline]
```

getTrace - method to get the trace.

**Devuelve**

returns the trace.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 297 del archivo [EvolutionStrategies.java](#).

```
00297           {
00298             // TODO Auto-generated method stub
00299             return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00300               this.listTrace.length);
00300 }
```

### 8.22.3.11 getType()

```
GeneratorType metaheuristics.generators.EvolutionStrategies.getType () [inline]
```

getType - method to get the generator type.

Devuelve

returns the generator type.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 134 del archivo [EvolutionStrategies.java](#).

```
00134             {
00135     return this.generatorType;
00136 }
```

Hace referencia a [generatorType](#).

### 8.22.3.12 getTypeGenerator()

```
GeneratorType metaheuristics.generators.EvolutionStrategies.getTypeGenerator () [inline]
```

getTypeGenerator - method to get the generator type.

Devuelve

returns the generator type.

Definición en la línea 205 del archivo [EvolutionStrategies.java](#).

```
00205             {
00206     return generatorType;
00207 }
```

Hace referencia a [generatorType](#).

### 8.22.3.13 getWeight()

```
float metaheuristics.generators.EvolutionStrategies.getWeight () [inline]
```

getWeight - method to get the weight.

Devuelve

returns the weight.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 257 del archivo [EvolutionStrategies.java](#).

```
00257             {
00258     // TODO Auto-generated method stub
00259     return this.weight;
00260 }
```

**Parámetros**

<i>stateInitialRef</i>	<input type="button" value=""/>
------------------------	---------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 143 del archivo [EvolutionStrategies.java](#).

```
00143           {
00144             this.stateReferenceES = (stateInitialRef == null) ? null : new State(stateInitialRef);
00145 }
```

**8.22.3.15 setListStateReference()**

```
void metaheuristics.generators.EvolutionStrategies.setListStateReference (
    List< State > listStateReference) [inline]
```

setListStateReference - method to set the list of reference states.

**Parámetros**

<i>listStateReference</i>	<input type="button" value=""/>
---------------------------	---------------------------------

Definición en la línea 197 del archivo [EvolutionStrategies.java](#).

```
00197           {
00198             this.listStateReference = (listStateReference == null) ? new ArrayList<State>() : new
00199               ArrayList<State>(listStateReference);
00200 }
```

Hace referencia a [listStateReference](#).

**8.22.3.16 setStateRef()**

```
void metaheuristics.generators.EvolutionStrategies.setStateRef (
    State stateRef) [inline]
```

setStateRef - method to set the current reference state.

**Parámetros**

<i>stateRef</i>	<input type="button" value=""/>
-----------------	---------------------------------

Definición en la línea 125 del archivo [EvolutionStrategies.java](#).

```
00125           {
00126             this.stateReferenceES = (stateRef == null) ? null : new State(stateRef);
00127 }
```

**8.22.3.17 setTypeGenerator()**

```
void metaheuristics.generators.EvolutionStrategies.setTypeGenerator (
    GeneratorType generatorType) [inline]
```

setTypeGenerator - method to set the generator type.

Generado por Doxygen

**Parámetros**

<i>generatorType</i>	<input type="button" value=""/>
----------------------	---------------------------------

Definición en la línea 213 del archivo [EvolutionStrategies.java](#).

```
00213         this.generatorType = generatorType;
00214     }
00215 }
```

Hace referencia a [generatorType](#).

**8.22.3.18 setWeight()**

```
void metaheuristics.generators.EvolutionStrategies.setWeight (
    float weight) [inline]
```

`setWeight` - method to set the weight.

**Parámetros**

<i>weight</i>	<input type="button" value=""/>
---------------	---------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 267 del archivo [EvolutionStrategies.java](#).

```
00267         {
00268     // TODO Auto-generated method stub
00269     this.weight = weight;
00270 }
```

Hace referencia a [weight](#).

**8.22.3.19 updateReference()**

```
void metaheuristics.generators.EvolutionStrategies.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

`updateReference` - method to update the reference state.

**Parámetros**

<i>stateCandidate</i>	<input type="button" value=""/>
<i>countIterationsCurrent</i>	<input type="button" value=""/>

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 153 del archivo [EvolutionStrategies.java](#).

```
00153 {
```

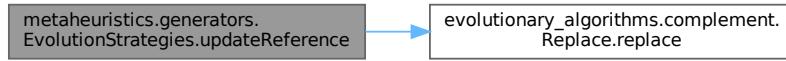
```

00154     iffreplace = new FactoryReplace();
00155     Replace replace = iffreplace.createReplace(replaceType);
00156     listStateReference = replace.replace(stateCandidate, listStateReference);
00157
00158 }

```

Hace referencia a [iffreplace](#), [listStateReference](#), [evolutionary\\_algorithms.complement.Replace.replace\(\)](#) y [replaceType](#).

Gráfico de llamadas de esta función:



## 8.22.4 Documentación de datos miembro

### 8.22.4.1 countRef

```
final int metaheuristics.generators.EvolutionStrategies.countRef = 0 [static]
```

Definición en la línea [41](#) del archivo [EvolutionStrategies.java](#).

Referenciado por [metaheuristics.generators.MultiGenerator.createInstanceGeneratorsBPP\(\)](#), [metaheuristics.generators.RandomSeamlessGenerator.createInstanceGeneratorsBPP\(\)](#), [metaheuristics.strategy.Strategy.update\(\)](#) y [local\\_search.complement.UpdateParameter.updateParameter\(\)](#).

### 8.22.4.2 generatorType

```
GeneratorType metaheuristics.generators.EvolutionStrategies.generatorType [private]
```

Definición en la línea [36](#) del archivo [EvolutionStrategies.java](#).

Referenciado por [getType\(\)](#), [getTypeGenerator\(\)](#) y [setTypeGenerator\(\)](#).

### 8.22.4.3 iffactorymutation

```
IFFactoryMutation metaheuristics.generators.EvolutionStrategies.iffactorymutation [private]
```

Definición en la línea [34](#) del archivo [EvolutionStrategies.java](#).

Referenciado por [generate\(\)](#).

### 8.22.4.4 iffatherselection

```
IFFactoryFatherSelection metaheuristics.generators.EvolutionStrategies.iffatherselection [private]
```

Definición en la línea [33](#) del archivo [EvolutionStrategies.java](#).

Referenciado por [generate\(\)](#).

#### 8.22.4.5 iffreplace

```
IFFactoryReplace metaheuristics.generators.EvolutionStrategies.iffreplace [private]
```

Definición en la línea 35 del archivo [EvolutionStrategies.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.22.4.6 listCountBetterGender

```
int [] metaheuristics.generators.EvolutionStrategies.listCountBetterGender = new int[10] [private]
```

Definición en la línea 47 del archivo [EvolutionStrategies.java](#).

Referenciado por [EvolutionStrategies\(\)](#).

#### 8.22.4.7 listCountGender

```
int [] metaheuristics.generators.EvolutionStrategies.listCountGender = new int[10] [private]
```

Definición en la línea 48 del archivo [EvolutionStrategies.java](#).

Referenciado por [EvolutionStrategies\(\)](#).

#### 8.22.4.8 listStateReference

```
List<State> metaheuristics.generators.EvolutionStrategies.listStateReference = new Array<-->  
List<State>() [private]
```

Definición en la línea 32 del archivo [EvolutionStrategies.java](#).

Referenciado por [generate\(\)](#), [getListStateRef\(\)](#), [getListStateReference\(\)](#), [getReference\(\)](#), [getReferenceList\(\)](#), [setListStateReference\(\)](#) y [updateReference\(\)](#).

#### 8.22.4.9 listTrace

```
float [] metaheuristics.generators.EvolutionStrategies.listTrace = new float[1200000] [private]
```

Definición en la línea 49 del archivo [EvolutionStrategies.java](#).

Referenciado por [EvolutionStrategies\(\)](#).

#### 8.22.4.10 mutationType

```
final MutationType metaheuristics.generators.EvolutionStrategies.mutationType = MutationType.OnePointMutation  
[static]
```

Definición en la línea 38 del archivo [EvolutionStrategies.java](#).

Referenciado por [generate\(\)](#).

#### 8.22.4.11 PM

```
final double metaheuristics.generators.EvolutionStrategies.PM = 0.0 [static]
```

Definición en la línea 37 del archivo [EvolutionStrategies.java](#).

Referenciado por [generate\(\)](#).

#### 8.22.4.12 replaceType

```
final ReplaceType metaheuristics.generators.EvolutionStrategies.replaceType = ReplaceType.GenerationalReplace [static]
```

Definición en la línea 39 del archivo [EvolutionStrategies.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.22.4.13 selectionType

```
final SelectionType metaheuristics.generators.EvolutionStrategies.selectionType = SelectionType.TruncationSelection [static]
```

Definición en la línea 40 del archivo [EvolutionStrategies.java](#).

Referenciado por [generate\(\)](#).

#### 8.22.4.14 stateReferenceES

```
State metaheuristics.generators.EvolutionStrategies.stateReferenceES [private]
```

Definición en la línea 31 del archivo [EvolutionStrategies.java](#).

Referenciado por [getReference\(\)](#).

#### 8.22.4.15 truncation

```
final int metaheuristics.generators.EvolutionStrategies.truncation = 0 [static]
```

Definición en la línea 42 del archivo [EvolutionStrategies.java](#).

Referenciado por [generate\(\)](#).

#### 8.22.4.16 weight

```
float metaheuristics.generators.EvolutionStrategies.weight = 50 [private]
```

Definición en la línea 43 del archivo [EvolutionStrategies.java](#).

Referenciado por [setWeight\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [EvolutionStrategies.java](#)

## 8.23 Referencia de la clase problem.extension.FactoresPonderados

FactoresPonderados.

Diagrama de herencia de problem.extension.FactoresPonderados

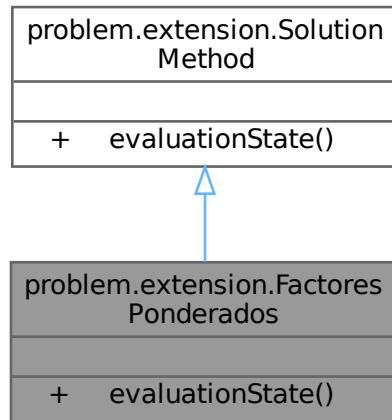
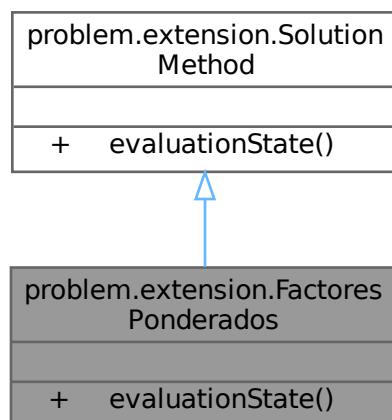


Diagrama de colaboración de problem.extension.FactoresPonderados:



### Métodos públicos

- void `evaluationState (State state)`  
*evaluationState*

## 8.23.1 Descripción detallada

[FactoresPonderados](#).

Implementación de [SolutionMethod](#) que combina funciones objetivo mediante factores ponderados.

Definición en la línea 16 del archivo [FactoresPonderados.java](#).

## 8.23.2 Documentación de funciones miembro

### 8.23.2.1 evaluationState()

```
void problem.extension.FactoresPonderados.evaluationState (
    State state) [inline]
```

**evaluationState**

Evaluá un estado aplicando la suma ponderada de cada función objetivo, teniendo en cuenta si la función es de maximizar o minimizar.

#### Parámetros

<b>state</b>	estado a evaluar; la lista de evaluación del estado se actualiza con el valor calculado.
--------------	--

Reimplementado de [problem.extension.SolutionMethod](#).

Definición en la línea 29 del archivo [FactoresPonderados.java](#).

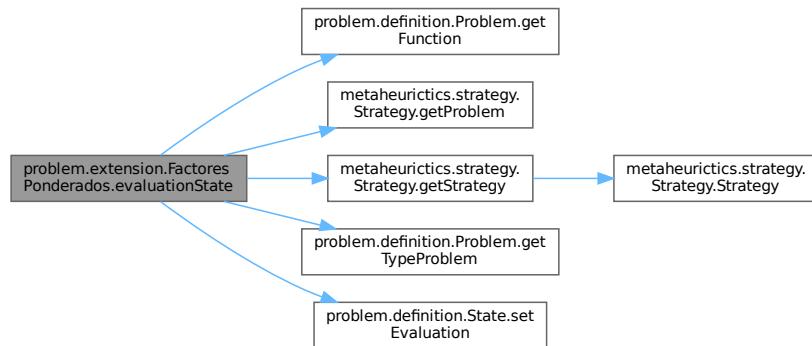
```
00029
00030      // TODO Auto-generated method stub
00031      double eval = 0;
00032      double tempWeight = 0;
00033      ArrayList<Double> evaluation = new
          ArrayList<Double>(Strategy.getStrategy().getProblem().getFunction().size());
00034
00035      for (int i = 0; i < Strategy.getStrategy().getProblem().getFunction().size(); i++) {
00036          tempWeight = 0;
00037          if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)) {
00038              if(Strategy.getStrategy().getProblem().getFunction().get(i).getTypeProblem().equals(ProblemType.Maximizar)) {
00039                  tempWeight =
00040                      Strategy.getStrategy().getProblem().getFunction().get(i).Evaluation(state);
00041                  tempWeight = tempWeight *
00042                      Strategy.getStrategy().getProblem().getFunction().get(i).getWeight();
00043              }
00044              else{
00045                  tempWeight = 1 -
00046                      Strategy.getStrategy().getProblem().getFunction().get(i).Evaluation(state);
00047                  tempWeight = tempWeight *
00048                      Strategy.getStrategy().getProblem().getFunction().get(i).getWeight();
00049              }
00050          }
00051          else{
00052              if(Strategy.getStrategy().getProblem().getFunction().get(i).getTypeProblem().equals(ProblemType.Maximizar)) {
00053                  tempWeight = 1 -
00054                      Strategy.getStrategy().getProblem().getFunction().get(i).Evaluation(state);
00055                  tempWeight = tempWeight *
00056                      Strategy.getStrategy().getProblem().getFunction().get(i).getWeight();
00057              }
00058      }
```

```

00058         eval += tempWeight;
00059     }
00060     evaluation.add(evaluation.size(), eval);
00061     state.setEvaluation(evaluation);
00062 }
00063 }
```

Hace referencia a `problem.definition.Problem.getFunction()`, `metaheuristics.strategy.Strategy.getProblem()`, `metaheuristics.strategy.Strategy.getStrategy()`, `problem.definition.Problem.getTypeProblem()`, `problem.definition.Problem.getProblemType()` y `problem.definition.State.setEvaluation()`.

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [FactoresPonderados.java](#)

## 8.24 Referencia de la clase factory\_method.FactoryAcceptCandidate

[FactoryAcceptCandidate](#) - Interface for creating acceptable candidates.

Diagrama de herencia de `factory_method.FactoryAcceptCandidate`

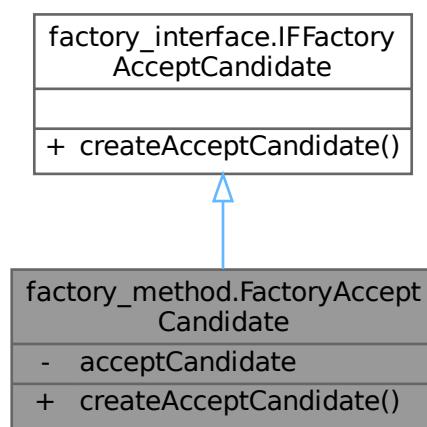
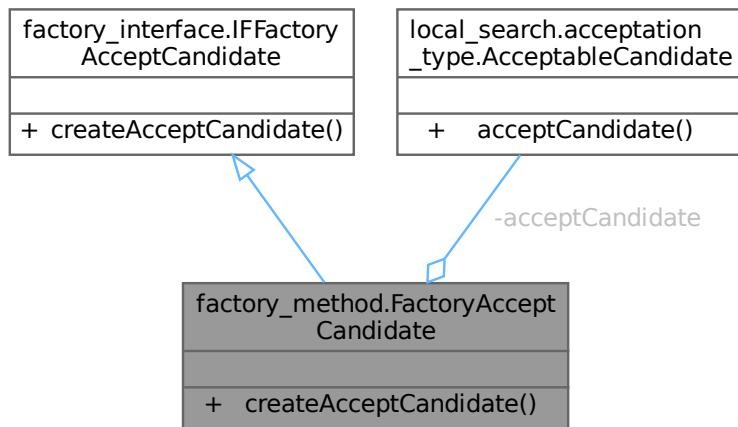


Diagrama de colaboración de factory\_method.FactoryAcceptCandidate:



## Métodos públicos

- `AcceptableCandidate createAcceptCandidate (AcceptType typeacceptation) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`  
`createAcceptCandidate` - Creates an instance of `AcceptableCandidate` based on the provided `AcceptType`.

## Atributos privados

- `AcceptableCandidate acceptCandidate`

### 8.24.1 Descripción detallada

`FactoryAcceptCandidate` - Interface for creating acceptable candidates.

Definición en la línea 21 del archivo `FactoryAcceptCandidate.java`.

### 8.24.2 Documentación de funciones miembro

#### 8.24.2.1 createAcceptCandidate()

```

AcceptableCandidate factory_method.FactoryAcceptCandidate.createAcceptCandidate (
    AcceptType typeacceptation) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
  
```

`createAcceptCandidate` - Creates an instance of `AcceptableCandidate` based on the provided `AcceptType`.

**Parámetros**

<i>typeacceptation</i>	<input type="text"/>
------------------------	----------------------

Devuelve

Implementa [factory\\_interface.IFFactoryAcceptCandidate](#).

Definición en la línea 29 del archivo [FactoryAcceptCandidate.java](#).

```
00029
00030     {
00031         String className = "local_search.acceptation_type." + typeacceptation.toString();
00032         acceptCandidate = (AcceptableCandidate) FactoryLoader.getInstance(className);
00033     }
```

Hace referencia a [acceptCandidate](#) y [factory\\_method.FactoryLoader.getInstance\(\)](#).

Gráfico de llamadas de esta función:



### 8.24.3 Documentación de datos miembro

#### 8.24.3.1 acceptCandidate

```
AcceptableCandidate factory_method.FactoryAcceptCandidate.acceptCandidate [private]
```

Definición en la línea 22 del archivo [FactoryAcceptCandidate.java](#).

Referenciado por [createAcceptCandidate\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [FactoryAcceptCandidate.java](#)

## 8.25 Referencia de la clase factory\_method.FactoryCandidate

[FactoryCandidate](#) - Interface for creating search candidates.

Diagrama de herencia de factory\_method.FactoryCandidate

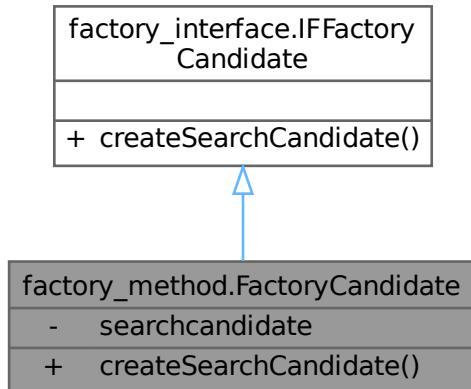
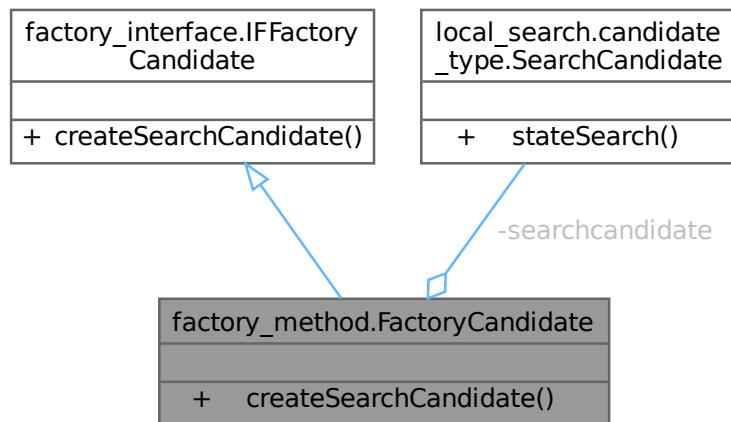


Diagrama de colaboración de `factory_method.FactoryCandidate`:



### Métodos públicos

- `SearchCandidate createSearchCandidate (CandidateType typeCandidate)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`  
`createSearchCandidate - descripción (añade detalles).`

### Atributos privados

- SearchCandidate searchcandidate

#### 8.25.1 Descripción detallada

[FactoryCandidate](#) - Interface for creating search candidates.

Definición en la línea 18 del archivo [FactoryCandidate.java](#).

#### 8.25.2 Documentación de funciones miembro

##### 8.25.2.1 createSearchCandidate()

```
SearchCandidate factory_method.FactoryCandidate.createSearchCandidate (
    CandidateType typeCandidate) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

createSearchCandidate - descripción (añade detalles).

#### Parámetros

<i>typeCandidate</i>	
----------------------	--

Devuelve

Implementa [factory\\_interface.IFFactoryCandidate](#).

Definición en la línea 26 del archivo [FactoryCandidate.java](#).

```
00026
{
00027     String className = "local_search.candidate_type." + typeCandidate.toString();
00028     searchcandidate = (SearchCandidate) FactoryLoader.getInstance(className);
00029     return searchcandidate;
00030 }
```

Hace referencia a [factory\\_method.FactoryLoader.getInstance\(\)](#) y [searchcandidate](#).

Gráfico de llamadas de esta función:



### 8.25.3 Documentación de datos miembro

#### 8.25.3.1 searchcandidate

`SearchCandidate factory_method.FactoryCandidate.searchcandidate [private]`

Definición en la línea 19 del archivo [FactoryCandidate.java](#).

Referenciado por [createSearchCandidate\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [FactoryCandidate.java](#)

## 8.26 Referencia de la clase factory\_method.FactoryCrossover

[FactoryCrossover](#) - Interface for creating crossover strategies.

Diagrama de herencia de factory\_method.FactoryCrossover

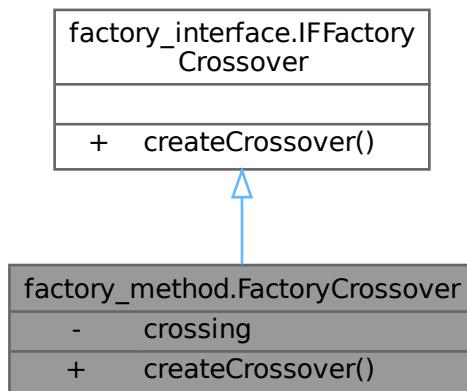
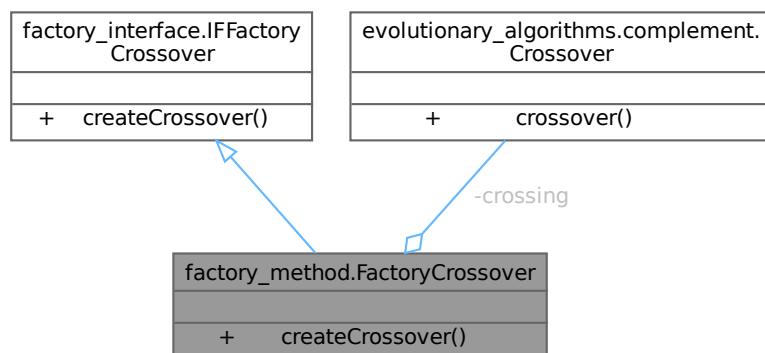


Diagrama de colaboración de `factory_method.FactoryCrossover`:



## Métodos públicos

- `Crossover createCrossover (CrossoverType CrossoverType) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

*createCrossover - Creates an instance of [Crossover](#) based on the provided [CrossoverType](#).*

## Atributos privados

- `Crossover crossing`

### 8.26.1 Descripción detallada

[FactoryCrossover](#) - Interface for creating crossover strategies.

Definición en la línea 16 del archivo [FactoryCrossover.java](#).

### 8.26.2 Documentación de funciones miembro

#### 8.26.2.1 createCrossover()

```
Crossover factory_method.FactoryCrossover.createCrossover (
    CrossoverType CrossoverType) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

*createCrossover - Creates an instance of [Crossover](#) based on the provided [CrossoverType](#).*

#### Parámetros

<code>CrossoverType</code>	
----------------------------	--

#### Devuelve

returns an instance of [Crossover](#)

Implementa [factory\\_interface.IFFactoryCrossover](#).

Definición en la línea 24 del archivo [FactoryCrossover.java](#).

```
00024
00025
00026     String className = "evolutionary_algorithms.complement." + CrossoverType.toString();
00027     crossing = (Crossover) FactoryLoader.getInstance(className);
00028     return crossing;
00029 }
```

Hace referencia a [crossing](#) y [factory\\_method.FactoryLoader.getInstance\(\)](#).

Gráfico de llamadas de esta función:



### 8.26.3 Documentación de datos miembro

#### 8.26.3.1 crossing

`Crossover factory_method.FactoryCrossover.crossing [private]`

Definición en la línea 17 del archivo [FactoryCrossover.java](#).

Referenciado por [createCrossover\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [FactoryCrossover.java](#)

## 8.27 Referencia de la clase factory\_method.FactoryDistribution

[FactoryDistribution](#) - Interface for creating distribution strategies.

Diagrama de herencia de `factory_method.FactoryDistribution`

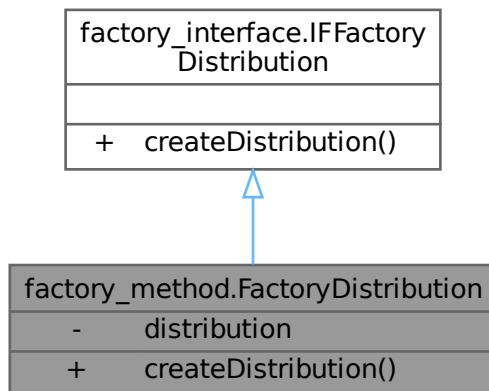
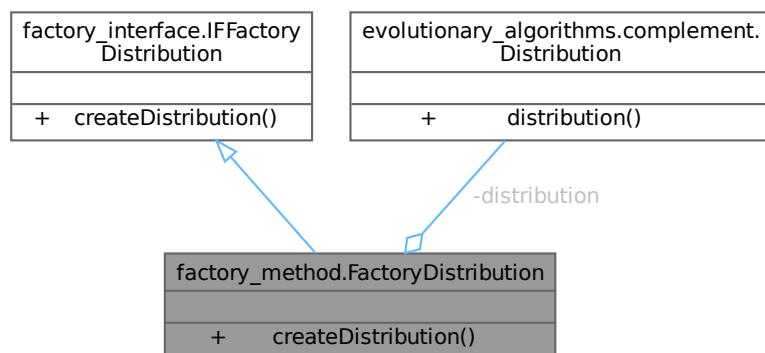


Diagrama de colaboración de `factory_method.FactoryDistribution`:



## Métodos públicos

- `Distribution createDistribution (DistributionType distributiontype) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

`createDistribution` - Creates an instance of `Distribution` based on the provided `DistributionType`.

## Atributos privados

- `Distribution distribution`

### 8.27.1 Descripción detallada

`FactoryDistribution` - Interface for creating distribution strategies.

Definición en la línea 12 del archivo `FactoryDistribution.java`.

### 8.27.2 Documentación de funciones miembro

#### 8.27.2.1 `createDistribution()`

```
Distribution factory_method.FactoryDistribution.createDistribution (
    DistributionType distributiontype) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException [inline]
```

`createDistribution` - Creates an instance of `Distribution` based on the provided `DistributionType`.

## Parámetros

<code>distributiontype</code>	
-------------------------------	--

## Devuelve

returns an instance of `Distribution`

Implementa `factory_interface.IFFactoryDistribution`.

Definición en la línea 20 del archivo `FactoryDistribution.java`.

```
00020
00021
00022     String className = "evolutionary_algorithms.complement." + distributiontype.toString();
00023     distribution = (Distribution) FactoryLoader.getInstance(className);
00024     return distribution;
00025 }
```

Hace referencia a `distribution` y `factory_method.FactoryLoader.getInstance()`.

Gráfico de llamadas de esta función:



### 8.27.3 Documentación de datos miembro

#### 8.27.3.1 distribution

```
Distribution factory_method.FactoryDistribution.distribution [private]
```

Definición en la línea 13 del archivo [FactoryDistribution.java](#).

Referenciado por [createDistribution\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [FactoryDistribution.java](#)

## 8.28 Referencia de la clase factory\_method.FactoryFatherSelection

[FactoryFatherSelection](#) - Interface for creating father selection strategies.

Diagrama de herencia de factory\_method.FactoryFatherSelection

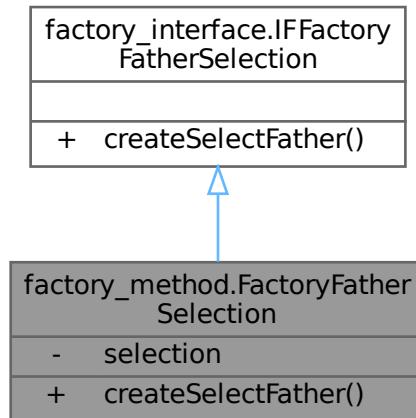
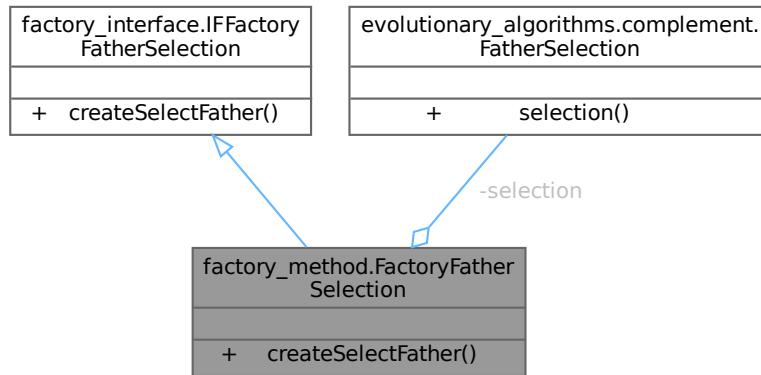


Diagrama de colaboración de factory\_method.FactoryFatherSelection:



## Métodos públicos

- `FatherSelection createSelectFather (SelectionType selectionType) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

*createSelectFather - Creates an instance of `FatherSelection` based on the provided `SelectionType`.*

## Atributos privados

- `FatherSelection selection`

### 8.28.1 Descripción detallada

`FactoryFatherSelection` - Interface for creating father selection strategies.

Definición en la línea 16 del archivo [FactoryFatherSelection.java](#).

### 8.28.2 Documentación de funciones miembro

#### 8.28.2.1 createSelectFather()

```

FatherSelection factory_method.FactoryFatherSelection.createSelectFather (
    SelectionType selectionType) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]

```

*createSelectFather - Creates an instance of `FatherSelection` based on the provided `SelectionType`.*

## Parámetros

<i>selectionType</i>	<input type="text"/>
----------------------	----------------------

## Devuelve

returns an instance of [FatherSelection](#)

Implementa [factory\\_interface.IFFactoryFatherSelection](#).

Definición en la línea 24 del archivo [FactoryFatherSelection.java](#).

```
00024
00025     {
00026         String className = "evolutionary_algorithms.complement." + selectionType.toString();
00027         selection = (FatherSelection) FactoryLoader.getInstance(className);
00028     }
```

Hace referencia a [factory\\_method.FactoryLoader.getInstance\(\)](#) y [selection](#).

Gráfico de llamadas de esta función:



## 8.28.3 Documentación de datos miembro

### 8.28.3.1 selection

`FatherSelection` [factory\\_method.FactoryFatherSelection.selection](#) [private]

Definición en la línea 17 del archivo [FactoryFatherSelection.java](#).

Referenciado por [createSelectFather\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [FactoryFatherSelection.java](#)

## 8.29 Referencia de la clase factory\_method.FactoryGenerator

[FactoryGenerator](#) - Interface for creating generator strategies.

Diagrama de herencia de factory\_method.FactoryGenerator

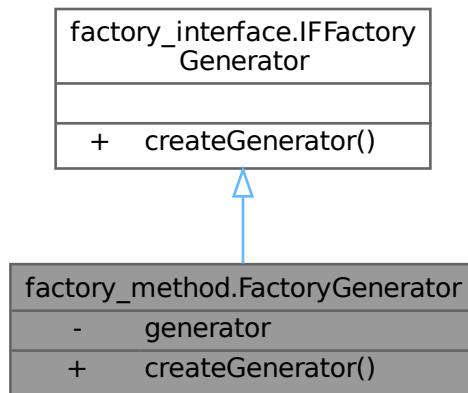
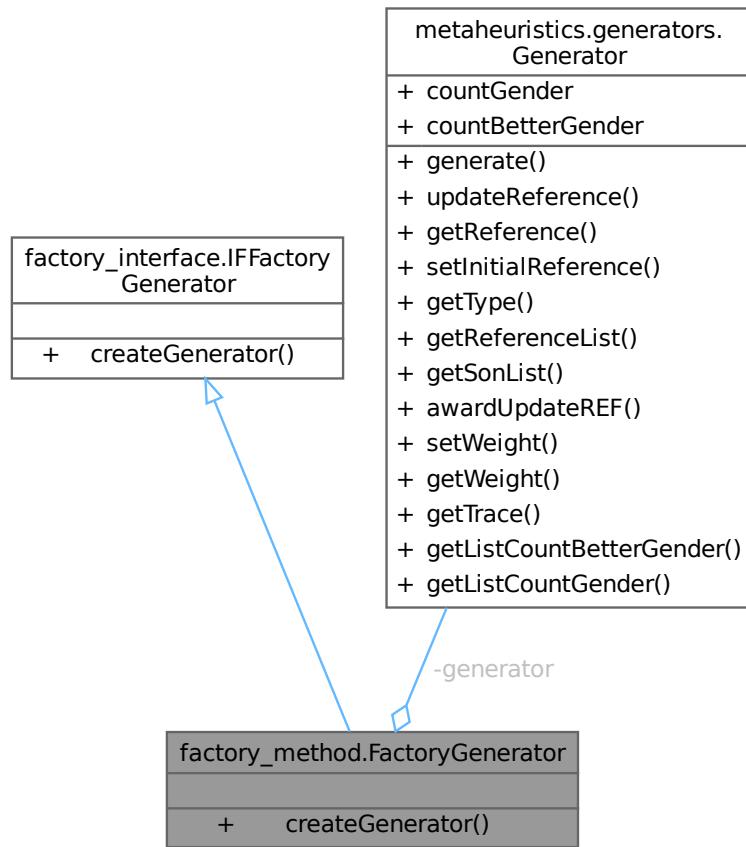


Diagrama de colaboración de factory\_method.FactoryGenerator:



## Métodos públicos

- `Generator createGenerator (GeneratorType generatorType)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*createGenerator - Creates an instance of `Generator` based on the provided `GeneratorType`.*

## Atributos privados

- `Generator generator`

### 8.29.1 Descripción detallada

`FactoryGenerator` - Interface for creating generator strategies.

Definición en la línea 16 del archivo `FactoryGenerator.java`.

## 8.29.2 Documentación de funciones miembro

### 8.29.2.1 createGenerator()

```
Generator factory_method.FactoryGenerator.createGenerator (
    GeneratorType generatorType) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

createGenerator - Creates an instance of [Generator](#) based on the provided [GeneratorType](#).

#### Parámetros

<i>generatorType</i>	
----------------------	--

#### Devuelve

returns an instance of [Generator](#)

Implementa [factory\\_interface.IFFactoryGenerator](#).

Definición en la línea 25 del archivo [FactoryGenerator.java](#).

```
00025
00026     {
00027         String className = "metaheuristics.generators." + generatorType.toString();
00028         generator = (Generator) FactoryLoader.getInstance(className);
00029     }
```

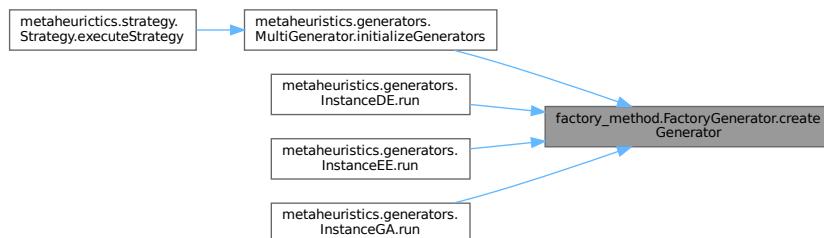
Hace referencia a [generator](#) y [factory\\_method.FactoryLoader.getInstance\(\)](#).

Referenciado por [metaheuristics.generators.MultiGenerator.initializeGenerators\(\)](#), [metaheuristics.generators.InstanceDE.run\(\)](#), [metaheuristics.generators.InstanceEE.run\(\)](#) y [metaheuristics.generators.InstanceGA.run\(\)](#).

Gráfico de llamadas de esta función:



Gráfico de llamadas a esta función:



### 8.29.3 Documentación de datos miembro

#### 8.29.3.1 generator

```
Generator factory_method.FactoryGenerator.generator [private]
```

Definición en la línea 18 del archivo [FactoryGenerator.java](#).

Referenciado por [createGenerator\(\)](#).

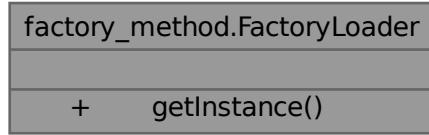
La documentación de esta clase está generada del siguiente archivo:

- [FactoryGenerator.java](#)

## 8.30 Referencia de la clase factory\_method.FactoryLoader

[FactoryLoader](#) - Class responsible for loading factory classes.

Diagrama de colaboración de `factory_method.FactoryLoader`:



### Métodos públicos estáticos

- static Object [getInstance](#) (String className) throws ClassNotFoundException, IllegalArgumentException, SecurityException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException

[FactoryLoader](#) - Carga dinámicamente una clase por su nombre y crea una instancia.

### 8.30.1 Descripción detallada

[FactoryLoader](#) - Class responsible for loading factory classes.

Definición en la línea 8 del archivo [FactoryLoader.java](#).

### 8.30.2 Documentación de funciones miembro

#### 8.30.2.1 [getInstance\(\)](#)

Generado por Doxygen

```
Object factory_method.FactoryLoader.getInstance (
    String className) throws ClassNotFoundException, IllegalArgumentException, SecurityException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException
```

## Parámetros

<i>className</i>	<input type="text"/>
------------------	----------------------

Devuelve

## Excepciones

<i>ClassNotFoundException</i>	
<i>IllegalArgumentException</i>	
<i>SecurityException</i>	
<i>InstantiationException</i>	
<i>IllegalAccessException</i>	
<i>InvocationTargetException</i>	
<i>NoSuchMethodException</i>	

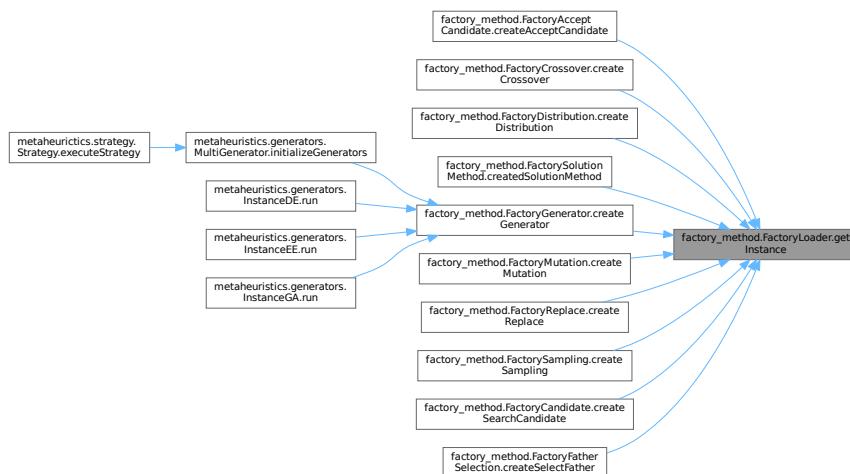
Definición en la línea 21 del archivo [FactoryLoader.java](#).

```

00021
00022     // Let exceptions propagate to caller instead of swallowing them (prevents null dereference)
00023     @SuppressWarnings("rawtypes")
00024     Class c = Class.forName(className);
00025     // use getDeclaredConstructor().newInstance() to avoid deprecated Class.newInstance()
00026     Object o = c.getDeclaredConstructor().newInstance();
00027     return o;
00028 }
```

Referenciado por [factory\\_method.FactoryAcceptCandidate.createAcceptCandidate\(\)](#), [factory\\_method.FactoryCrossover.createCrossover\(\)](#), [factory\\_method.FactoryDistribution.createDistribution\(\)](#), [factory\\_method.FactorySolutionMethod.createdSolutionMethod\(\)](#), [factory\\_method.FactoryGenerator.createGenerator\(\)](#), [factory\\_method.FactoryMutation.createMutation\(\)](#), [factory\\_method.FactoryReplace.createReplace\(\)](#), [factory\\_method.FactorySampling.createSampling\(\)](#), [factory\\_method.FactoryCandidate.createSearchCandidate\(\)](#) y [factory\\_method.FactoryFatherSelection.createSelectFather\(\)](#).

Gráfico de llamadas a esta función:



La documentación de esta clase está generada del siguiente archivo:

- `FactoryLoader.java`

## 8.31 Referencia de la clase `factory_method.FactoryMutation`

[FactoryMutation](#) - Class responsible for creating mutation strategies.

Diagrama de herencia de `factory_method.FactoryMutation`

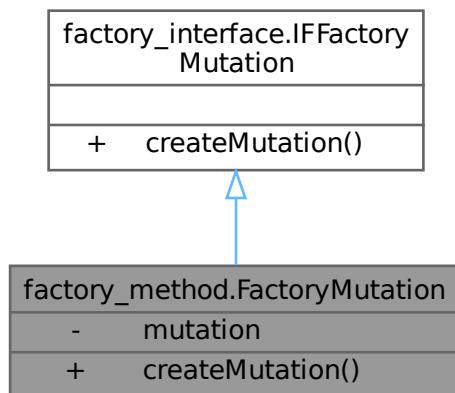
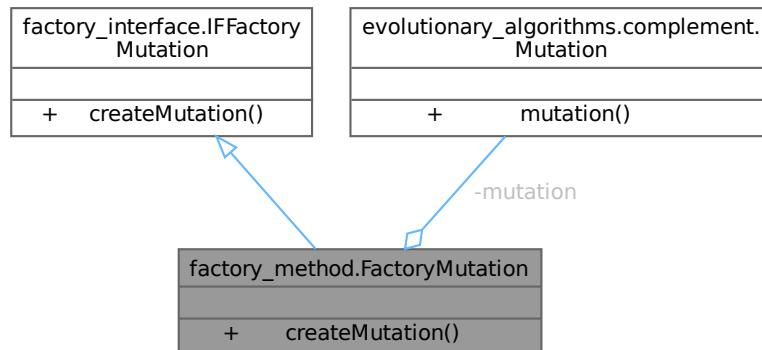


Diagrama de colaboración de `factory_method.FactoryMutation`:



### Métodos públicos

- `Mutation createMutation (MutationType typeMutation)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

`createMutation` - Creates an instance of `Mutation` based on the provided `MutationType`.

### Atributos privados

- Mutation mutation

#### 8.31.1 Descripción detallada

[FactoryMutation](#) - Class responsible for creating mutation strategies.

Definición en la línea 16 del archivo [FactoryMutation.java](#).

#### 8.31.2 Documentación de funciones miembro

##### 8.31.2.1 createMutation()

```
Mutation factory_method.FactoryMutation.createMutation (
    MutationType typeMutation) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

createMutation - Creates an instance of [Mutation](#) based on the provided [MutationType](#).

#### Parámetros

<i>typeMutation</i>	
---------------------	--

#### Devuelve

returns an instance of [Mutation](#)

Implementa [factory\\_interface.IFFactoryMutation](#).

Definición en la línea 24 del archivo [FactoryMutation.java](#).

```
00024 {
00025     String className = "evolutionary_algorithms.complement." + typeMutation.toString();
00026     mutation = (Mutation) FactoryLoader.getInstance(className);
00027     return mutation;
00028 }
00029 }
```

Hace referencia a [factory\\_method.FactoryLoader.getInstance\(\)](#) y [mutation](#).

Gráfico de llamadas de esta función:



### 8.31.3 Documentación de datos miembro

#### 8.31.3.1 mutation

`Mutation` factory\_method.FactoryMutation.mutation [private]

Definición en la línea 17 del archivo [FactoryMutation.java](#).

Referenciado por [createMutation\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [FactoryMutation.java](#)

## 8.32 Referencia de la clase factory\_method.FactoryReplace

[FactoryReplace](#) - Class responsible for creating replacement strategies.

Diagrama de herencia de factory\_method.FactoryReplace

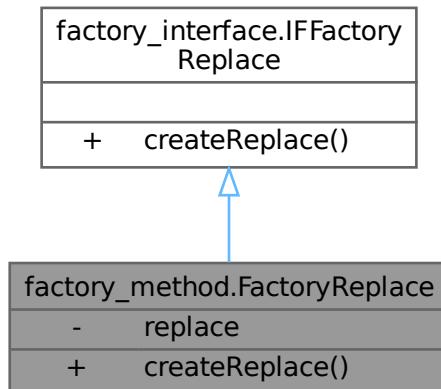
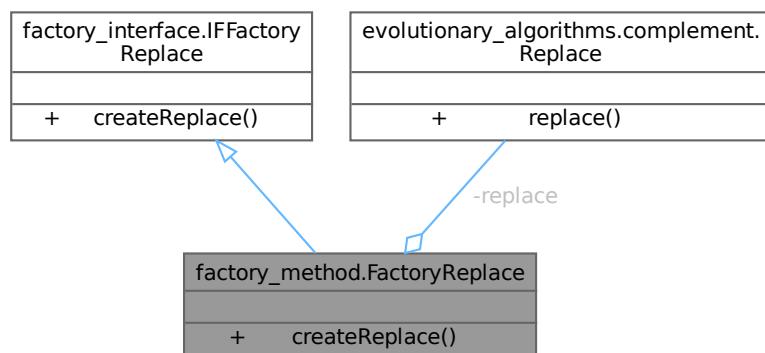


Diagrama de colaboración de factory\_method.FactoryReplace:



### Métodos públicos

- `Replace createReplace (ReplaceType typereplace) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

`createReplace - Creates an instance of Replace based on the provided ReplaceType.`

### Atributos privados

- `Replace replace`

## 8.32.1 Descripción detallada

`FactoryReplace` - Class responsible for creating replacement strategies.

Definición en la línea 17 del archivo [FactoryReplace.java](#).

## 8.32.2 Documentación de funciones miembro

### 8.32.2.1 createReplace()

```
Replace factory_method.FactoryReplace.createReplace (
    ReplaceType typereplace) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException [inline]
```

`createReplace - Creates an instance of Replace based on the provided ReplaceType.`

#### Parámetros

<code>typereplace</code>	<input type="text"/>
--------------------------	----------------------

#### Devuelve

returns an instance of `Replace`

Implementa `factory_interface.IFFactoryReplace`.

Definición en la línea 26 del archivo [FactoryReplace.java](#).

```
00026
{
00027     String className = "evolutionary_algorithms.complement." + typereplace.toString();
00028     replace = (Replace) FactoryLoader.getInstance(className);
00029     return replace;
00030 }
```

Hace referencia a `factory_method.FactoryLoader.getInstance()` y `replace`.

Gráfico de llamadas de esta función:



### 8.32.3 Documentación de datos miembro

#### 8.32.3.1 replace

`Replace factory_method.FactoryReplace.replace [private]`

Definición en la línea 19 del archivo [FactoryReplace.java](#).

Referenciado por [createReplace\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [FactoryReplace.java](#)

## 8.33 Referencia de la clase factory\_method.FactorySampling

[FactorySampling](#) - Class responsible for creating sampling strategies.

Diagrama de herencia de `factory_method.FactorySampling`

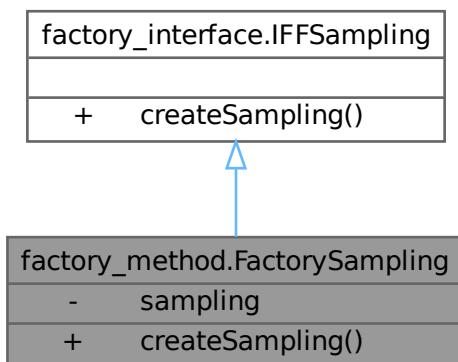
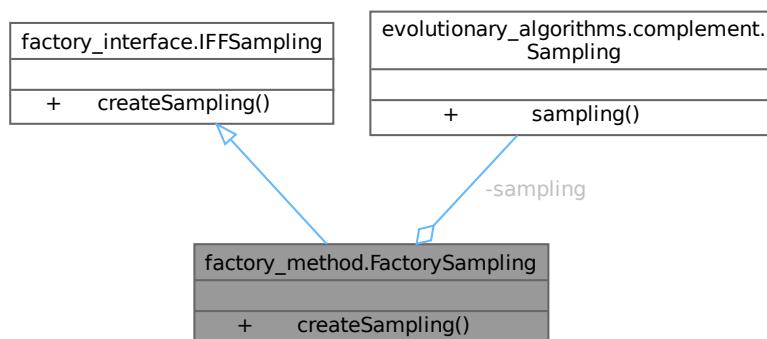


Diagrama de colaboración de `factory_method.FactorySampling`:



## Métodos públicos

- `Sampling createSampling (SamplingType typesampling) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

*createSampling - Creates an instance of Sampling based on the provided SamplingType.*

## Atributos privados

- `Sampling sampling`

### 8.33.1 Descripción detallada

`FactorySampling` - Class responsible for creating sampling strategies.

Definición en la línea 16 del archivo `FactorySampling.java`.

### 8.33.2 Documentación de funciones miembro

#### 8.33.2.1 createSampling()

```
Sampling factory_method.FactorySampling.createSampling (
    SamplingType typesampling) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

*createSampling - Creates an instance of Sampling based on the provided SamplingType.*

## Parámetros

<code>typesampling</code>	<input type="button" value=""/>
---------------------------	---------------------------------

## Devuelve

returns an instance of `Sampling`

Implementa `factory_interface.IFFSampling`.

Definición en la línea 23 del archivo `FactorySampling.java`.

```
00023
00024 {
00025     String className = "evolutionary_algorithms.complement." + typesampling.toString();
00026     sampling = (Sampling) FactoryLoader.getInstance(className);
00027     return sampling;
00028 }
```

Hace referencia a `factory_method.FactoryLoader.getInstance()` y `sampling`.

Gráfico de llamadas de esta función:



### 8.33.3 Documentación de datos miembro

#### 8.33.3.1 sampling

```
Sampling factory_method.FactorySampling.sampling [private]
```

Definición en la línea 17 del archivo [FactorySampling.java](#).

Referenciado por [createSampling\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [FactorySampling.java](#)

## 8.34 Referencia de la clase factory\_method.FactorySolutionMethod

[FactorySolutionMethod](#) - Class responsible for creating solution method strategies.

Diagrama de herencia de `factory_method.FactorySolutionMethod`

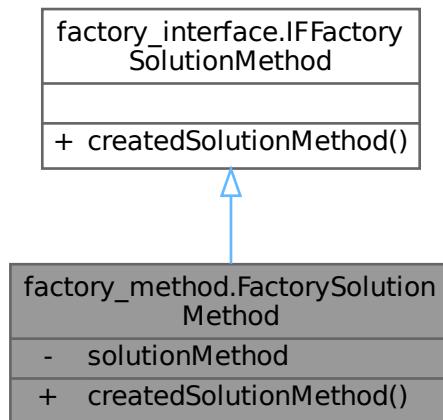
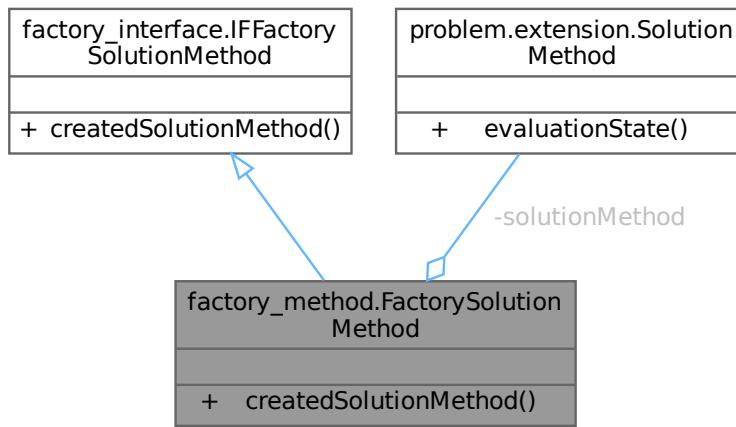


Diagrama de colaboración de factory\_method.FactorySolutionMethod:



## Métodos públicos

- **SolutionMethod createdSolutionMethod (TypeSolutionMethod method)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*createdSolutionMethod - Creates an instance of `SolutionMethod` based on the provided `TypeSolutionMethod`.*

## Atributos privados

- `SolutionMethod solutionMethod`

### 8.34.1 Descripción detallada

`FactorySolutionMethod` - Class responsible for creating solution method strategies.

Definición en la línea 12 del archivo `FactorySolutionMethod.java`.

### 8.34.2 Documentación de funciones miembro

#### 8.34.2.1 createdSolutionMethod()

```

SolutionMethod factory_method.FactorySolutionMethod.createdSolutionMethod (
    TypeSolutionMethod method) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException [inline]
  
```

`createdSolutionMethod` - Creates an instance of `SolutionMethod` based on the provided `TypeSolutionMethod`.

**Parámetros**

<i>method</i>	
---------------	--

**Devuelve**

returns an instance of [SolutionMethod](#)

Implementa [factory\\_interface.IFFactorySolutionMethod](#).

Definición en la línea 22 del archivo [FactorySolutionMethod.java](#).

```
00022
00023     {
00024         String className = "problem.extension." + method.toString();
00025         solutionMethod = (SolutionMethod) FactoryLoader.getInstance(className);
00026     }
00027 }
```

Hace referencia a [factory\\_method.FactoryLoader.getInstance\(\)](#) y [solutionMethod](#).

Gráfico de llamadas de esta función:



### 8.34.3 Documentación de datos miembro

#### 8.34.3.1 solutionMethod

[SolutionMethod](#) `factory_method.FactorySolutionMethod.solutionMethod` [private]

Definición en la línea 14 del archivo [FactorySolutionMethod.java](#).

Referenciado por [createdSolutionMethod\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [FactorySolutionMethod.java](#)

## 8.35 Referencia de la clase evolutionary\_algorithms.complement.FatherSelection

[FatherSelection](#) - selects the best fathers from the population.

Diagrama de herencia de `evolutionary_algorithms.complement.FatherSelection`

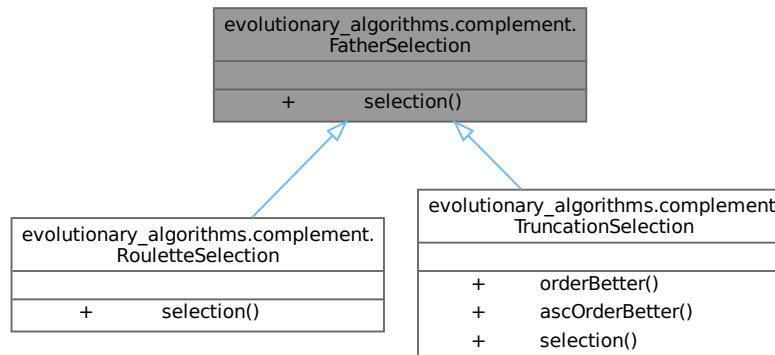
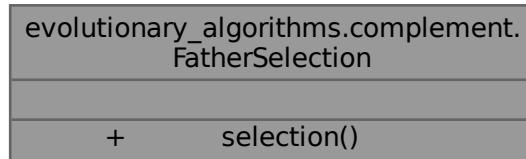


Diagrama de colaboración de `evolutionary_algorithms.complement.FatherSelection`:



### Métodos públicos

- abstract `List< State > selection (List< State > listState, int truncation)`  
`selection` - selects the best fathers from the population.

### 8.35.1 Descripción detallada

[FatherSelection](#) - selects the best fathers from the population.

Definición en la línea 11 del archivo [FatherSelection.java](#).

## 8.35.2 Documentación de funciones miembro

### 8.35.2.1 selection()

```
abstract List< State > evolutionary_algorithms.complement.FatherSelection.selection (
    List< State > listState,
    int truncation) [abstract]
```

selection - selects the best fathers from the population.

#### Parámetros

<i>listState</i>	
<i>truncation</i>	

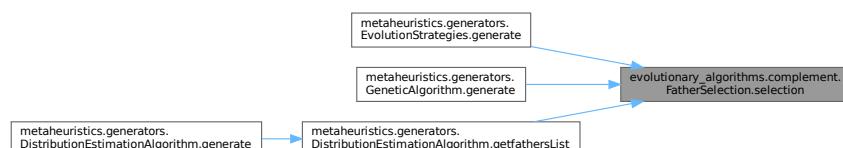
#### Devuelve

returns a list of selected fathers

Reimplementado en [evolutionary\\_algorithms.complement.RouletteSelection](#) y [evolutionary\\_algorithms.complement.TruncationSelection](#).

Referenciado por [metaheuristics.generators.EvolutionStrategies.generate\(\)](#), [metaheuristics.generators.GeneticAlgorithm.generate\(\)](#) y [metaheuristics.generators.DistributionEstimationAlgorithm.getfathersList\(\)](#).

Gráfico de llamadas a esta función:



La documentación de esta clase está generada del siguiente archivo:

- [FatherSelection.java](#)

## 8.36 Referencia de la clase

### [evolutionary\\_algorithms.complement.GenerationalReplace](#)

[GenerationalReplace](#) - replaces the worst individual in the population.

Diagrama de herencia de `evolutionary_algorithms.complement.GenerationalReplace`

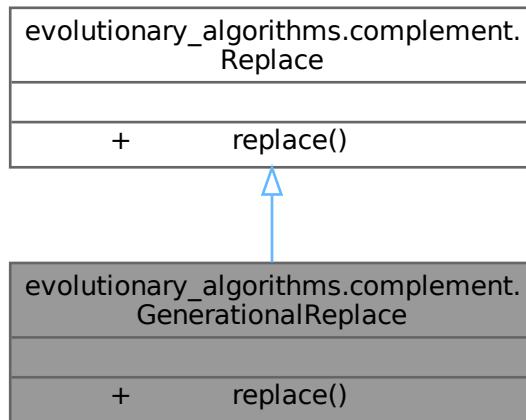
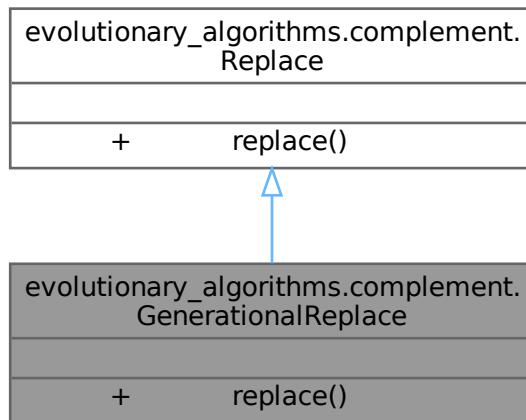


Diagrama de colaboración de `evolutionary_algorithms.complement.GenerationalReplace`:



## Métodos públicos

- `List< State > replace (State stateCandidate, List< State > listState)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*replace - replaces the worst individual in the population.*

### 8.36.1 Descripción detallada

[GenerationalReplace](#) - replaces the worst individual in the population.

Definición en la línea 12 del archivo [GenerationalReplace.java](#).

### 8.36.2 Documentación de funciones miembro

#### 8.36.2.1 replace()

```
List< State > evolutionary_algorithms.complement.GenerationalReplace.replace (
    State stateCandidate,
    List< State > listState) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

replace - replaces the worst individual in the population.

#### Parámetros

<i>stateCandidate</i>	
<i>listState</i>	

#### Devuelve

returns the new population after replacing the worst individual with the stateCandidate

Reimplementado de [evolutionary\\_algorithms.complement.Replace](#).

Definición en la línea 21 del archivo [GenerationalReplace.java](#).

```
00021
00022     {
00023         listState.remove(0);
00024         listState.add(stateCandidate);
00025     }
00026 }
```

La documentación de esta clase está generada del siguiente archivo:

- [GenerationalReplace.java](#)

## 8.37 Referencia de la clase metaheuristics.generators.Generator

[Generator](#) - abstract base class for solution generators in metaheuristic algorithms.

Diagrama de herencia de metaheuristics.generators.Generator

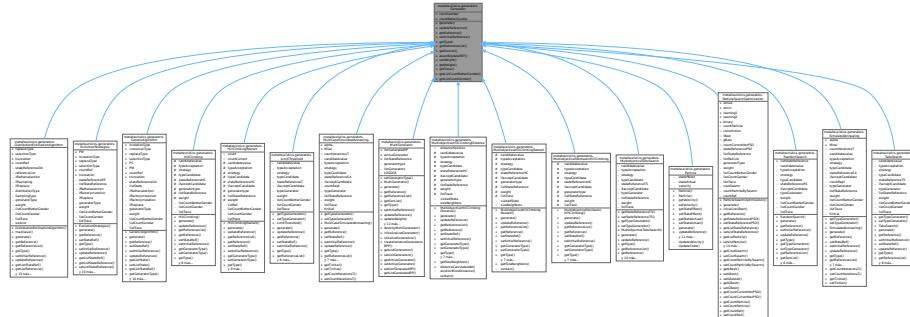


Diagrama de colaboración de metaheuristics.generators.Generator:

metaheuristics.generators. Generator
+ countGender
+ countBetterGender
+ generate()
+ updateReference()
+ getReference()
+ setInitialReference()
+ getType()
+ getReferenceList()
+ getSonList()
+ awardUpdateREF()
+ setWeight()
+ getWeight()
+ getTrace()
+ getListCountBetterGender()
+ getListCountGender()

### Métodos públicos

- abstract [State generate](#) (Integer operatornumber) throws [IllegalArgumentException](#), [SecurityException](#), [ClassNotFoundException](#), [InstantiationException](#), [IllegalAccessException](#), [InvocationTargetException](#), [NoSuchMethodException](#)

- abstract void `updateReference (State stateCandidate, Integer countIterationsCurrent)` throws `IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`
- abstract `State getReference ()`
- abstract void `setInitialReference (State stateInitialRef)`
- abstract `GeneratorType getType ()`
- abstract `List< State > getReferenceList ()`
- abstract `List< State > getSonList ()`
- abstract boolean `awardUpdateREF (State stateCandidate)`
- abstract void `setWeight (float weight)`
- abstract float `getWeight ()`
- abstract float[] `getTrace ()`
- abstract int[] `getListCountBetterGender ()`
- abstract int[] `getListCountGender ()`

### Atributos públicos

- int `countGender`
- int `countBetterGender`

### 8.37.1 Descripción detallada

`Generator` - abstract base class for solution generators in metaheuristic algorithms.

Definición en la línea 13 del archivo `Generator.java`.

### 8.37.2 Documentación de funciones miembro

#### 8.37.2.1 awardUpdateREF()

```
abstract boolean metaheuristics.generators.Generator.awardUpdateREF (
    State stateCandidate) [abstract]
```

Reimplementado en `metaheuristics.generators.DistributionEstimationAlgorithm`, `metaheuristics.generators.EvolutionStrategies`, `metaheuristics.generators.GeneticAlgorithm`, `metaheuristics.generators.HillClimbing`, `metaheuristics.generators.HillClimbingRestart`, `metaheuristics.generators.LimitThreshold`, `metaheuristics.generators.MultiCaseSimulatedAnnealing`, `metaheuristics.generators.Multi`, `metaheuristics.generators.MultiobjectiveHillClimbingDistance`, `metaheuristics.generators.MultiobjectiveHillClimbingRestart`, `metaheuristics.generators.MultiobjectiveStochasticHillClimbing`, `metaheuristics.generators.MultiobjectiveTabuSearch`, `metaheuristics.generators.Particle`, `metaheuristics.generators.ParticleSwarmOptimization`, `metaheuristics.generators.RandomSearch`, `metaheuristics.generators.SimulatedAnnealing` y `metaheuristics.generators.TabuSearch`.

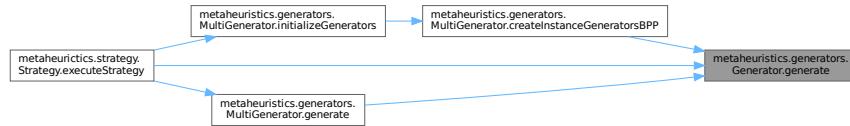
### 8.37.2.2 generate()

```
abstract State metaheuristics.generators.Generator.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [abstract]
```

Reimplementado en [metaheuristics.generators.DistributionEstimationAlgorithm](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm](#), [metaheuristics.generators.HillClimbing](#), [metaheuristics.generators.HillClimbingRestart](#), [metaheuristics.generators.LimitThreshold](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#), [metaheuristics.generators.MultiobjectiveTabuSearch](#), [metaheuristics.generators.Particle](#), [metaheuristics.generators.ParticleSwarmOptimization](#), [metaheuristics.generators.RandomSearch](#), [metaheuristics.generators.SimulatedAnnealing](#) y [metaheuristics.generators.TabuSearch](#).

Referenciado por [metaheuristics.generators.MultiGenerator.createInstanceGeneratorsBPP\(\)](#), [metaheuristics.strategy.Strategy.execute\(\)](#) y [metaheuristics.generators.MultiGenerator.generate\(\)](#).

Gráfico de llamadas a esta función:



### 8.37.2.3 getListCountBetterGender()

```
abstract int[] metaheuristics.generators.Generator.getListCountBetterGender () [abstract]
```

Reimplementado en [metaheuristics.generators.DistributionEstimationAlgorithm](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm](#), [metaheuristics.generators.HillClimbing](#), [metaheuristics.generators.HillClimbingRestart](#), [metaheuristics.generators.LimitThreshold](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#), [metaheuristics.generators.MultiobjectiveTabuSearch](#), [metaheuristics.generators.Particle](#), [metaheuristics.generators.ParticleSwarmOptimization](#), [metaheuristics.generators.RandomSearch](#), [metaheuristics.generators.SimulatedAnnealing](#) y [metaheuristics.generators.TabuSearch](#).

Referenciado por [metaheuristics.strategy.Strategy.updateCountGender\(\)](#).

Gráfico de llamadas a esta función:



### 8.37.2.4 getListCountGender()

```
abstract int[] metaheuristics.generators.Generator.getListCountGender () [abstract]
```

Reimplementado en [metaheuristics.generators.DistributionEstimationAlgorithm](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm](#), [metaheuristics.generators.HillClimbing](#), [metaheuristics.generators.HillClimbingRestart](#), [metaheuristics.generators.LimitThreshold](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#), [metaheuristics.generators.MultiobjectiveTabuSearch](#), [metaheuristics.generators.Particle](#), [metaheuristics.generators.ParticleSwarmOptimization](#), [metaheuristics.generators.RandomSearch](#), [metaheuristics.generators.SimulatedAnnealing](#) y [metaheuristics.generators.TabuSearch](#).

Referenciado por [metaheuristics.strategy.Strategy.updateCountGender\(\)](#).

Gráfico de llamadas a esta función:



### 8.37.2.5 getReference()

```
abstract State metaheuristics.generators.Generator.getReference () [abstract]
```

Reimplementado en [metaheuristics.generators.DistributionEstimationAlgorithm](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm](#), [metaheuristics.generators.HillClimbing](#), [metaheuristics.generators.HillClimbingRestart](#), [metaheuristics.generators.LimitThreshold](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#), [metaheuristics.generators.MultiobjectiveTabuSearch](#), [metaheuristics.generators.Particle](#), [metaheuristics.generators.ParticleSwarmOptimization](#), [metaheuristics.generators.RandomSearch](#), [metaheuristics.generators.SimulatedAnnealing](#) y [metaheuristics.generators.TabuSearch](#).

### 8.37.2.6 getReferenceList()

```
abstract List<State> metaheuristics.generators.Generator.getReferenceList () [abstract]
```

Reimplementado en [metaheuristics.generators.DistributionEstimationAlgorithm](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm](#), [metaheuristics.generators.HillClimbing](#), [metaheuristics.generators.HillClimbingRestart](#), [metaheuristics.generators.LimitThreshold](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#), [metaheuristics.generators.MultiobjectiveTabuSearch](#), [metaheuristics.generators.Particle](#), [metaheuristics.generators.ParticleSwarmOptimization](#), [metaheuristics.generators.RandomSearch](#), [metaheuristics.generators.SimulatedAnnealing](#) y [metaheuristics.generators.TabuSearch](#).

### 8.37.2.7 getSonList()

```
abstract List< State > metaheuristics.generators.Generator.getSonList () [abstract]
```

Reimplementado en [metaheuristics.generators.DistributionEstimationAlgorithm](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm](#), [metaheuristics.generators.HillClimbing](#), [metaheuristics.generators.HillClimbingRestart](#), [metaheuristics.generators.LimitThreshold](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing](#), [metaheuristics.generators.MultiCaseSimulatedAnnealingBPP](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#), [metaheuristics.generators.MultiobjectiveTabuSearch](#), [metaheuristics.generators.Particle](#), [metaheuristics.generators.ParticleSwarmOptimization](#), [metaheuristics.generators.RandomSearch](#), [metaheuristics.generators.SimulatedAnnealing](#) y [metaheuristics.generators.TabuSearch](#).

### 8.37.2.8 getTrace()

```
abstract float[] metaheuristics.generators.Generator.getTrace () [abstract]
```

Reimplementado en [metaheuristics.generators.DistributionEstimationAlgorithm](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm](#), [metaheuristics.generators.HillClimbing](#), [metaheuristics.generators.HillClimbingRestart](#), [metaheuristics.generators.LimitThreshold](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing](#), [metaheuristics.generators.MultiCaseSimulatedAnnealingBPP](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#), [metaheuristics.generators.MultiobjectiveTabuSearch](#), [metaheuristics.generators.Particle](#), [metaheuristics.generators.ParticleSwarmOptimization](#), [metaheuristics.generators.RandomSearch](#), [metaheuristics.generators.SimulatedAnnealing](#) y [metaheuristics.generators.TabuSearch](#).

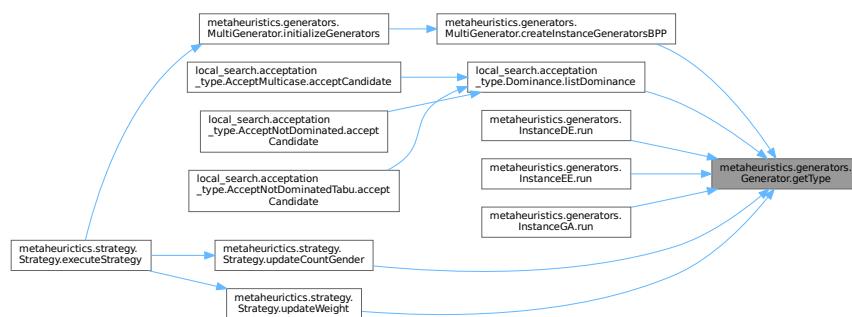
### 8.37.2.9 getType()

```
abstract GeneratorType metaheuristics.generators.Generator.getType () [abstract]
```

Reimplementado en [metaheuristics.generators.DistributionEstimationAlgorithm](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm](#), [metaheuristics.generators.HillClimbing](#), [metaheuristics.generators.HillClimbingRestart](#), [metaheuristics.generators.LimitThreshold](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing](#), [metaheuristics.generators.MultiCaseSimulatedAnnealingBPP](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#), [metaheuristics.generators.MultiobjectiveTabuSearch](#), [metaheuristics.generators.Particle](#), [metaheuristics.generators.ParticleSwarmOptimization](#), [metaheuristics.generators.RandomSearch](#), [metaheuristics.generators.SimulatedAnnealing](#) y [metaheuristics.generators.TabuSearch](#).

Referenciado por [metaheuristics.generators.MultiGenerator.createInstanceGeneratorsBPP\(\)](#), [local\\_search.acceptation\\_type.Dominance.listDominance\(\)](#), [metaheuristics.generators.InstanceDE.run\(\)](#), [metaheuristics.generators.InstanceEE.run\(\)](#), [metaheuristics.generators.InstanceGA.run\(\)](#), [metaheuristics.strategy.Strategy.updateCountGender\(\)](#) y [metaheuristics.strategy.Strategy.updateWeight\(\)](#).

Gráfico de llamadas a esta función:



### 8.37.2.10 `getWeight()`

```
abstract float metaheuristics.generators.Generator.getWeight () [abstract]
```

Reimplementado en [metaheuristics.generators.DistributionEstimationAlgorithm](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm](#), [metaheuristics.generators.HillClimbing](#), [metaheuristics.generators.HillClimbingRestart](#), [metaheuristics.generators.LimitThreshold](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing](#), [metaheuristics.generators.Multi](#)  
[metaheuristics.generators.MultiobjectiveHillClimbingDistance](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart](#),  
[metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#), [metaheuristics.generators.MultiobjectiveTabuSearch](#),  
[metaheuristics.generators.Particle](#), [metaheuristics.generators.ParticleSwarmOptimization](#), [metaheuristics.generators.RandomSearch](#)  
[metaheuristics.generators.SimulatedAnnealing](#) y [metaheuristics.generators.TabuSearch](#).

### 8.37.2.11 `setInitialReference()`

```
abstract void metaheuristics.generators.Generator.setInitialReference (
    State stateInitialRef) [abstract]
```

Reimplementado en [metaheuristics.generators.DistributionEstimationAlgorithm](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm](#), [metaheuristics.generators.HillClimbing](#), [metaheuristics.generators.HillClimbingRestart](#), [metaheuristics.generators.LimitThreshold](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing](#), [metaheuristics.generators.Multi](#)  
[metaheuristics.generators.MultiobjectiveHillClimbingDistance](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart](#),  
[metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#), [metaheuristics.generators.MultiobjectiveTabuSearch](#),  
[metaheuristics.generators.Particle](#), [metaheuristics.generators.ParticleSwarmOptimization](#), [metaheuristics.generators.RandomSearch](#)  
[metaheuristics.generators.SimulatedAnnealing](#) y [metaheuristics.generators.TabuSearch](#).

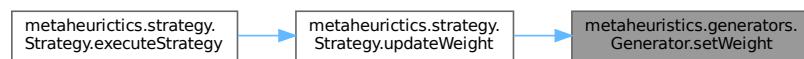
### 8.37.2.12 `setWeight()`

```
abstract void metaheuristics.generators.Generator.setWeight (
    float weight) [abstract]
```

Reimplementado en [metaheuristics.generators.DistributionEstimationAlgorithm](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm](#), [metaheuristics.generators.HillClimbing](#), [metaheuristics.generators.HillClimbingRestart](#), [metaheuristics.generators.LimitThreshold](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing](#), [metaheuristics.generators.Multi](#)  
[metaheuristics.generators.MultiobjectiveHillClimbingDistance](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart](#),  
[metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#), [metaheuristics.generators.MultiobjectiveTabuSearch](#),  
[metaheuristics.generators.Particle](#), [metaheuristics.generators.ParticleSwarmOptimization](#), [metaheuristics.generators.RandomSearch](#)  
[metaheuristics.generators.SimulatedAnnealing](#) y [metaheuristics.generators.TabuSearch](#).

Referenciado por [metaheuristics.strategy.Strategy.updateWeight\(\)](#).

Gráfico de llamadas a esta función:



### 8.37.2.13 updateReference()

```
abstract void metaheuristics.generators.Generator.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [abstract]
```

Reimplementado en [metaheuristics.generators.DistributionEstimationAlgorithm](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm](#), [metaheuristics.generators.HillClimbing](#), [metaheuristics.generators.HillClimbingRestart](#), [metaheuristics.generators.LimitThreshold](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing](#), [metaheuristics.generators.MultiObjectiveHillClimbingDistance](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#), [metaheuristics.generators.MultiobjectiveTabuSearch](#), [metaheuristics.generators.Particle](#), [metaheuristics.generators.ParticleSwarmOptimization](#), [metaheuristics.generators.RandomSearch](#), [metaheuristics.generators.SimulatedAnnealing](#) y [metaheuristics.generators.TabuSearch](#).

## 8.37.3 Documentación de datos miembro

### 8.37.3.1 countBetterGender

```
int metaheuristics.generators.Generator.countBetterGender
```

Definición en la línea 38 del archivo [Generator.java](#).

Referenciado por [metaheuristics.strategy.Strategy.updateCountGender\(\)](#).

### 8.37.3.2 countGender

```
int metaheuristics.generators.Generator.countGender
```

Definición en la línea 37 del archivo [Generator.java](#).

Referenciado por [metaheuristics.strategy.Strategy.updateCountGender\(\)](#).

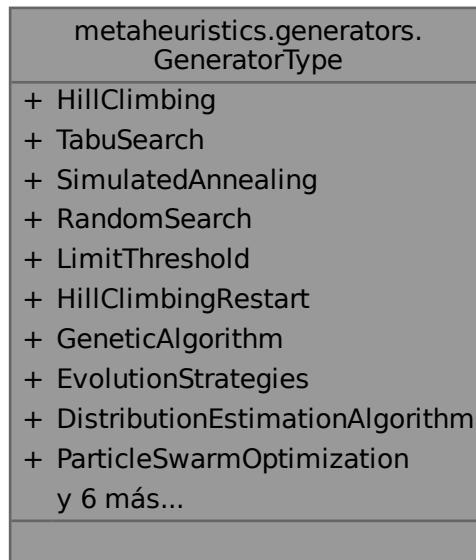
La documentación de esta clase está generada del siguiente archivo:

- [Generator.java](#)

## 8.38 Referencia de la enumeración metaheuristics.generators.GeneratorType

[GeneratorType](#) - enumeration of different types of solution generators used in metaheuristic algorithms.

Diagrama de colaboración de `metaheuristics.generators.GeneratorType`:



### Atributos públicos

- [HillClimbing](#)
- [TabuSearch](#)
- [SimulatedAnnealing](#)
- [RandomSearch](#)
- [LimitThreshold](#)
- [HillClimbingRestart](#)
- [GeneticAlgorithm](#)
- [EvolutionStrategies](#)
- [DistributionEstimationAlgorithm](#)
- [ParticleSwarmOptimization](#)
- [MultiGenerator](#)
- [MultiobjectiveTabuSearch](#)
- [MultiobjectiveStochasticHillClimbing](#)
- [MultiCaseSimulatedAnnealing](#)
- [MultiobjectiveHillClimbingRestart](#)
- [MultiobjectiveHillClimbingDistance](#)

### 8.38.1 Descripción detallada

`GeneratorType` - enumeration of different types of solution generators used in metaheuristic algorithms.

Definición en la línea 6 del archivo `GeneratorType.java`.

### 8.38.2 Documentación de datos miembro

#### 8.38.2.1 DistributionEstimationAlgorithm

`metaheuristics.generators.GeneratorType.DistributionEstimationAlgorithm`

Definición en la línea 8 del archivo `GeneratorType.java`.

Referenciado por `metaheuristics.generators.DistributionEstimationAlgorithm.DistributionEstimationAlgorithm()`, `metaheuristics.generators.DistributionEstimationAlgorithm.getListStateRef()`, `metaheuristics.generators.MultiGenerator.initializeGenerators()`, `metaheuristics.algorithms.complement.ProbabilisticSampling.listState()`, `metaheuristics.generators.InstanceDE.run()`, `metaheuristics.strategy.Strategy.update()`, `local_search.complement.UpdateParameter.updateParameter()` y `metaheuristics.strategy.Strategy.updateRefGenerator()`.

#### 8.38.2.2 EvolutionStrategies

`metaheuristics.generators.GeneratorType.EvolutionStrategies`

Definición en la línea 8 del archivo `GeneratorType.java`.

Referenciado por `metaheuristics.generators.EvolutionStrategies.EvolutionStrategies()`, `metaheuristics.generators.EvolutionStrategies.getListStateRef()`, `metaheuristics.generators.MultiGenerator.initializeGenerators()`, `metaheuristics.generators.InstanceEE.run()`, `metaheuristics.strategy.Strategy.update()`, `local_search.complement.UpdateParameter.updateParameter()` y `metaheuristics.strategy.Strategy.updateRefGenerator()`.

#### 8.38.2.3 GeneticAlgorithm

`metaheuristics.generators.GeneratorType.GeneticAlgorithm`

Definición en la línea 8 del archivo `GeneratorType.java`.

Referenciado por `metaheuristics.generators.GeneticAlgorithm.GeneticAlgorithm()`, `metaheuristics.generators.GeneticAlgorithm.getListStateRef()`, `metaheuristics.generators.MultiGenerator.initializeGenerators()`, `metaheuristics.generators.InstanceGA.run()`, `metaheuristics.strategy.Strategy.update()`, `local_search.complement.UpdateParameter.updateParameter()` y `metaheuristics.strategy.Strategy.updateRefGenerator()`.

#### 8.38.2.4 HillClimbing

`metaheuristics.generators.GeneratorType.HillClimbing`

Definición en la línea 7 del archivo `GeneratorType.java`.

Referenciado por `metaheuristics.generators.HillClimbing.HillClimbing()`, `metaheuristics.generators.HillClimbingRestart.HillClimbingRestart()`, `metaheuristics.generators.MultiGenerator.initializeGenerators()` y `metaheuristics.strategy.Strategy.updateRefGenerator()`.

### 8.38.2.5 HillClimbingRestart

```
metaheuristics.generators.GeneratorType.HillClimbingRestart
```

Definición en la línea 7 del archivo [GeneratorType.java](#).

### 8.38.2.6 LimitThreshold

```
metaheuristics.generators.GeneratorType.LimitThreshold
```

Definición en la línea 7 del archivo [GeneratorType.java](#).

Referenciado por [metaheuristics.generators.MultiGenerator.initializeGenerators\(\)](#) y [metaheuristics.generators.LimitThreshold.LimitThreshold](#).

### 8.38.2.7 MultiCaseSimulatedAnnealing

```
metaheuristics.generators.GeneratorType.MultiCaseSimulatedAnnealing
```

Definición en la línea 10 del archivo [GeneratorType.java](#).

Referenciado por [metaheuristics.generators.MultiCaseSimulatedAnnealing.MultiCaseSimulatedAnnealing\(\)](#).

### 8.38.2.8 MultiGenerator

```
metaheuristics.generators.GeneratorType.MultiGenerator
```

Definición en la línea 9 del archivo [GeneratorType.java](#).

Referenciado por [metaheuristics.strategy.Strategy.executeStrategy\(\)](#), [metaheuristics.generators.MultiGenerator.MultiGenerator\(\)](#), [metaheuristics.generators.MultiGenerator.tournament\(\)](#), [metaheuristics.generators.MultiGenerator.updateAwardImp\(\)](#), [metaheuristics.generators.MultiGenerator.updateAwardSC\(\)](#), [metaheuristics.strategy.Strategy.updateCountGender\(\)](#), [metaheuristics.strategy.Strategy.updateRef\(\)](#) y [metaheuristics.strategy.Strategy.updateWeight\(\)](#).

### 8.38.2.9 MultiobjectiveHillClimbingDistance

```
metaheuristics.generators.GeneratorType.MultiobjectiveHillClimbingDistance
```

Definición en la línea 10 del archivo [GeneratorType.java](#).

Referenciado por [local\\_search.acceptation\\_type.Dominance.listDominance\(\)](#) y [metaheuristics.generators.MultiobjectiveHillClimbingDistance](#).

### 8.38.2.10 MultiobjectiveHillClimbingRestart

```
metaheuristics.generators.GeneratorType.MultiobjectiveHillClimbingRestart
```

Definición en la línea 10 del archivo [GeneratorType.java](#).

Referenciado por [metaheuristics.generators.MultiobjectiveHillClimbingRestart.MultiobjectiveHillClimbingRestart\(\)](#).

### 8.38.2.11 MultiobjectiveStochasticHillClimbing

```
metaheuristics.generators.GeneratorType.MultiobjectiveStochasticHillClimbing
```

Definición en la línea 10 del archivo [GeneratorType.java](#).

Referenciado por [metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#).[MultiobjectiveStochasticHillClimbing\(\)](#).

### 8.38.2.12 MultiobjectiveTabuSearch

```
metaheuristics.generators.GeneratorType.MultiobjectiveTabuSearch
```

Definición en la línea 10 del archivo [GeneratorType.java](#).

Referenciado por [metaheuristics.generators.MultiobjectiveTabuSearch](#).[MultiobjectiveTabuSearch\(\)](#).

### 8.38.2.13 ParticleSwarmOptimization

```
metaheuristics.generators.GeneratorType.ParticleSwarmOptimization
```

Definición en la línea 8 del archivo [GeneratorType.java](#).

Referenciado por [metaheuristics.generators.ParticleSwarmOptimization](#).[getListStateRef\(\)](#), [metaheuristics.generators.ParticleSwarmO](#)  
[metaheuristics.strategy.Strategy.update\(\)](#) y [local\\_search.complement.UpdateParameter](#).[updateParameter\(\)](#).

### 8.38.2.14 RandomSearch

```
metaheuristics.generators.GeneratorType.RandomSearch
```

Definición en la línea 7 del archivo [GeneratorType.java](#).

Referenciado por [metaheuristics.generators.MultiGenerator](#).[initializeGenerators\(\)](#), [metaheuristics.generators.RandomSearch](#).[Random](#)  
y [metaheuristics.strategy.Strategy.updateRefGenerator\(\)](#).

### 8.38.2.15 SimulatedAnnealing

```
metaheuristics.generators.GeneratorType.SimulatedAnnealing
```

Definición en la línea 7 del archivo [GeneratorType.java](#).

Referenciado por [metaheuristics.generators.MultiGenerator](#).[initializeGenerators\(\)](#), [metaheuristics.generators.SimulatedAnnealing](#).[Sim](#)  
y [metaheuristics.strategy.Strategy.updateRefGenerator\(\)](#).

### 8.38.2.16 TabuSearch

```
metaheuristics.generators.GeneratorType.TabuSearch
```

Definición en la línea 7 del archivo [GeneratorType.java](#).

Referenciado por [metaheuristics.generators.MultiGenerator](#).[initializeGenerators\(\)](#), [metaheuristics.generators.TabuSearch](#).[TabuSearch](#)  
y [metaheuristics.strategy.Strategy.updateRefGenerator\(\)](#).

La documentación de esta enumeración está generada del siguiente archivo:

- [GeneratorType.java](#)

## 8.39 Referencia de la clase metaheuristics.generators.GeneticAlgorithm

[GeneticAlgorithm](#) - descripción (añade detalles).

Diagrama de herencia de metaheuristics.generators.GeneticAlgorithm

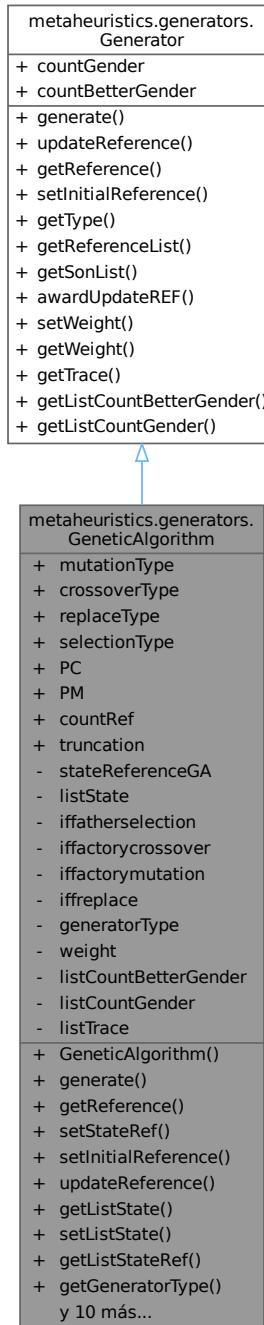
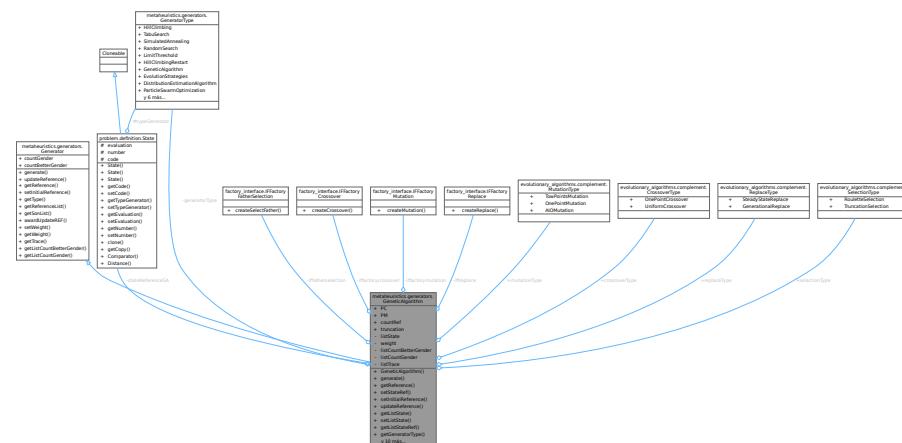


Diagrama de colaboración de metaheuristics.generators.GeneticAlgorithm:



## Métodos públicos

- `GeneticAlgorithm ()`

*GeneticAlgorithm - method to create an instance of GeneticAlgorithm.*

- `State generate (Integer operatornumber)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.*

- `State getReference ()`

*getReference - method to get the current reference state.*

- `void setStateRef (State stateRef)`

*setStateRef - method to set the current reference state.*

- `void setInitialReference (State statelInitialRef)`

*setInitialReference - method to set the initial reference state.*

- `void updateReference (State stateCandidate, Integer countIterationsCurrent)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*updateReference - method to update the current reference state.*

- `List< State > getListState ()`

*getListState - method to get the current list of states.*

- `void setListState (List< State > listState)`

*setListState - method to set the current list of states.*

- `List< State > getListStateRef ()`

*getListStateRef - method to get the current list of reference states.*

- `GeneratorType getGeneratorType ()`

*getGeneratorType - method to get the generator type.*

- `void setGeneratorType (GeneratorType generatorType)`

*setGeneratorType - method to set the generator type.*

- `GeneratorType getType ()`

*getType - method to get the generator type.*

- `List< State > getReferenceList ()`

*getReferenceList - method to get the current list of reference states.*

- `List< State > getSonList ()`

*getSonList - method to get the current list of son states.*

- boolean `awardUpdateREF (State stateCandidate)`  
`awardUpdateREF` - method to award the update of the reference state.
- float `getWeight ()`  
`getWeight` - method to get the weight.
- void `setWeight (float weight)`  
`setWeight` - method to set the weight.
- int[] `getListCountBetterGender ()`  
`getListCountBetterGender` - method to get the list of count better gender.
- int[] `getListCountGender ()`  
`getListCountGender` - method to get the list of count gender.
- float[] `getTrace ()`  
`getTrace` - method to get the trace.

### Atributos públicos estáticos

- static final `MutationType mutationType = MutationType.OnePointMutation`
- static final `CrossoverType crossoverType = CrossoverType.UniformCrossover`
- static final `ReplaceType replaceType = ReplaceType.GenerationalReplace`
- static final `SelectionType selectionType = SelectionType.TruncationSelection`
- static final double `PC = 0.6`
- static final double `PM = 0.01`
- static final int `countRef = 0`
- static final int `truncation = 0`

### Atributos privados

- `State stateReferenceGA`
- List<`State`> `listState = new ArrayList<State>()`
- `IFFactoryFatherSelection iffatherselection`
- `IFFactoryCrossover iffactorycrossover`
- `IFFactoryMutation iffactorymutation`
- `IFFactoryReplace iffreplace`
- `GeneratorType generatorType`
- float `weight`
- int[] `listCountBetterGender = new int[10]`
- int[] `listCountGender = new int[10]`
- float[] `listTrace = new float[1200000]`

### Otros miembros heredados

#### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- int `countGender`
- int `countBetterGender`

### 8.39.1 Descripción detallada

[GeneticAlgorithm](#) - descripción (añade detalles).

Definición en la línea 34 del archivo [GeneticAlgorithm.java](#).

## 8.39.2 Documentación de constructores y destructores

### 8.39.2.1 GeneticAlgorithm()

```
metaheuristics.generators.GeneticAlgorithm.GeneticAlgorithm () [inline]
```

[GeneticAlgorithm](#) - method to create an instance of [GeneticAlgorithm](#).

Definición en la línea 68 del archivo [GeneticAlgorithm.java](#).

```
00068     super();
00069     this.listState = getListStateRef(); // llamada al método que devuelve la lista.
00070     this.generatorType = GeneratorType.GeneticAlgorithm;
00071     this.weight = 50;
00072     listTrace[0] = this.weight;
00073     listCountBetterGender[0] = 0;
00074     listCountGender[0] = 0;
00075 }
00076 }
```

Hace referencia a [metaheuristics.generators.GeneratorType.GeneticAlgorithm](#), [getListStateRef\(\)](#), [listCountBetterGender](#), [listCountGender](#), [listTrace](#) y [weight](#).

Referenciado por [getListStateRef\(\)](#).

Gráfico de llamadas de esta función:

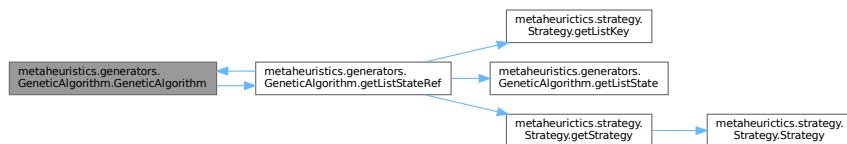
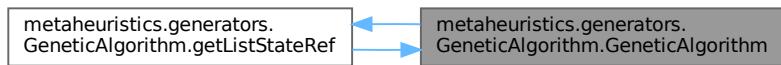


Gráfico de llamadas a esta función:



## 8.39.3 Documentación de funciones miembro

### 8.39.3.1 awardUpdateREF()

```
boolean metaheuristics.generators.GeneticAlgorithm.awardUpdateREF (
    State stateCandidate) [inline]
```

[awardUpdateREF](#) - method to award the update of the reference state.

## Parámetros

<i>stateCandidate</i>	<input type="button" value=""/>
-----------------------	---------------------------------

### Devuelve

returns true if the update is awarded, false otherwise.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 278 del archivo [GeneticAlgorithm.java](#).

```
00278      // TODO Auto-generated method stub
00279      return false;
00280
00281 }
```

### 8.39.3.2 generate()

```
State metaheuristics.generators.GeneticAlgorithm.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.

This RNG is used to decide whether a mutation occurs in the evolutionary algorithm and is not used for security-sensitive purposes. Suppress Sonar security hotspot S2245 for this usage. generate - method to generate a new state.

## Parámetros

<i>operatornumber</i>	<input type="button" value=""/>
-----------------------	---------------------------------

### Devuelve

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 90 del archivo [GeneticAlgorithm.java](#).

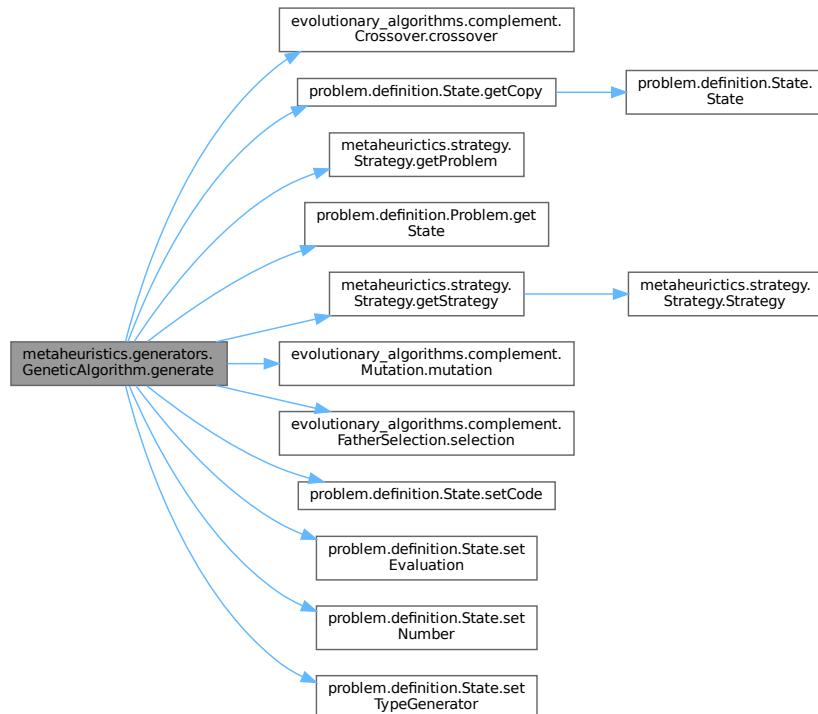
```
00090
00091 {
00092     //*****selection*****
00093
00094     List<State> refList = new ArrayList<State>(this.listState);
00095     iffatherselection = new FactoryFatherSelection();
00096     FatherSelection selection = iffatherselection.createSelectFather(selectionType);
00097     List<State> fathers = selection.selection(refList, truncation);
00098     int pos1 = (fathers.size() > 0) ? ThreadLocalRandom.current().nextInt(fathers.size()) : 0;
00099     int pos2 = (fathers.size() > 0) ? ThreadLocalRandom.current().nextInt(fathers.size()) : 0;
00100
00101     State auxState1 = (State) Strategy.getStrategy().getProblem().getState().getCopy();
00102     auxState1.setCode(new ArrayList<Object>(fathers.get(pos1).getCode()));
00103     auxState1.setEvaluation(fathers.get(pos1).getEvaluation());
00104     auxState1.setNumber(fathers.get(pos1).getNumber());
00105     auxState1.setTypeGenerator(fathers.get(pos1).getTypeGenerator());
00106
00107     State auxState2 = (State) Strategy.getStrategy().getProblem().getState().getCopy();
```

```

00108     auxState2.setCode(new ArrayList<Object>(fathers.get(pos2).getCode()));
00109     auxState2.setEvaluation(fathers.get(pos2).getEvaluation());
00110     auxState2.setNumber(fathers.get(pos2).getNumber());
00111     auxState2.setTypeGenerator(fathers.get(pos2).getTypeGenerator());
00112
00113     //*****cruzamiento*****
00114     iffactorycrossover = new FactoryCrossover();
00115     Crossover crossover = iffactorycrossover.createCrossover(crossoverType);
00116     auxState1 = crossover.crossover(auxState1, auxState2, PC);
00117
00118     //*****mutacion*****
00119     iffactorymutation = new FactoryMutation();
00120     Mutation mutation = iffactorymutation.createMutation(mutationType);
00121     auxState1 = mutation.mutation(auxState1, PM);
00122     //list.add(auxState1);
00123
00124     return auxState1;
00125 }
```

Hace referencia a `evolutionary_algorithms.complement.Crossover.crossover()`, `crossoverType`, `problem.definition.State.getCopy()`, `metaheuristics.strategy.Strategy.getProblem()`, `problem.definition.Problem.getState()`, `metaheuristics.strategy.Strategy.getStrategy()`, `iffactorycrossover`, `iffactorymutation`, `iffatherselection`, `listState`, `evolutionary_algorithms.complement.Mutation.mutation()`, `mutationType`, `PC`, `PM`, `evolutionary_algorithms.complement.FatherSelection.selection()`, `selectionType`, `problem.definition.State.setCode()`, `problem.definition.State.setEvaluation()`, `problem.definition.State.setNumber()`, `problem.definition.State.setTypeGenerator()` y `truncation`.

Gráfico de llamadas de esta función:



### 8.39.3.3 getGeneratorType()

```
GeneratorType metaheuristics.generators.GeneticAlgorithm.getGeneratorType () [inline]
```

`getGeneratorType` - method to get the generator type.

**Devuelve**

returns the generator type.

Definición en la línea 227 del archivo [GeneticAlgorithm.java](#).

```
00227             {
00228     return generatorType;
00229 }
```

Hace referencia a [generatorType](#).

**8.39.3.4 getListCountBetterGender()**

```
int[] metaheuristics.generators.GeneticAlgorithm.getListCountBetterGender () [inline]
```

getListCountBetterGender - method to get the list of count better gender.

**Devuelve**

returns the list of count better gender.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 308 del archivo [GeneticAlgorithm.java](#).

```
00308             {
00309     // TODO Auto-generated method stub
00310     return (this.listCountBetterGender == null) ? new int[0] :
00311         Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
00311 }
```

Hace referencia a [listCountBetterGender](#).

**8.39.3.5 getListCountGender()**

```
int[] metaheuristics.generators.GeneticAlgorithm.getListCountGender () [inline]
```

getListCountGender - method to get the list of count gender.

**Devuelve**

returns the list of count gender.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 318 del archivo [GeneticAlgorithm.java](#).

```
00318             {
00319     // TODO Auto-generated method stub
00320     return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00321         this.listCountGender.length);
00321 }
```

Hace referencia a [listCountGender](#).

### 8.39.3.6 getListState()

```
List< State > metaheuristics.generators.GeneticAlgorithm.getListState () [inline]
```

getListState - method to get the current list of states.

**Devuelve**

returns the current list of states.

Definición en la línea 184 del archivo [GeneticAlgorithm.java](#).

```
00184             {
00185         return (listState == null) ? new ArrayList<State>() : new ArrayList<State>(listState);
00186     }
```

Hace referencia a [listState](#).

Referenciado por [getListStateRef\(\)](#).

Gráfico de llamadas a esta función:



### 8.39.3.7 getListStateRef()

```
List< State > metaheuristics.generators.GeneticAlgorithm.getListStateRef () [inline]
```

getListStateRef - method to get the current list of reference states.

**Devuelve**

returns the current list of reference states.

Definición en la línea 200 del archivo [GeneticAlgorithm.java](#).

```
00200     Boolean found = false;
00201     Boolean found = false;
00202     List<String> key = Strategy.getStrategy().getListKey();
00203     int count = 0;
00204
00205     while((found.equals(false)) && (Strategy.getStrategy().mapGenerators.size() > count)){
00206         if(key.get(count).equals(GeneratorType.GeneticAlgorithm.toString())){
00207             GeneratorType keyGenerator = GeneratorType.valueOf(String.valueOf(key.get(count)));
00208             GeneticAlgorithm generator = (GeneticAlgorithm)
00209                 Strategy.getStrategy().mapGenerators.get(keyGenerator);
00210             if(generator.getListState().isEmpty()){
00211                 listState.addAll(RandomSearch.listStateReference);
00212             }
00213             else{
00214                 listState = generator.getListState();
00215             }
00216             found = true;
00217         }
00218         count++;
00219     }
00220     return (listState == null) ? new ArrayList<State>() : new ArrayList<State>(listState);
00221 }
```

Hace referencia a [metaheuristics.generators.GeneratorType.GeneticAlgorithm](#), [GeneticAlgorithm\(\)](#), [metaheuristics.strategy.Strategy.getListState\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [listState](#), [metaheuristics.generators.RandomSearch.listStateReference](#) y [metaheuristics.strategy.Strategy.mapGenerators](#).

Referenciado por [GeneticAlgorithm\(\)](#).

Gráfico de llamadas de esta función:

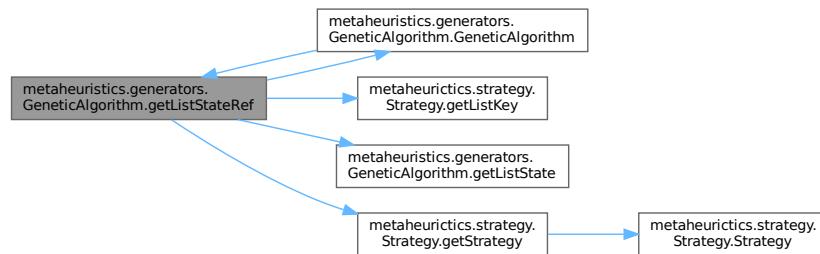


Gráfico de llamadas a esta función:



### 8.39.3.8 getReference()

`State metaheuristics.generators.GeneticAlgorithm.getReference () [inline]`

getReference - method to get the current reference state.

Devuelve

returns the current reference state.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 132 del archivo [GeneticAlgorithm.java](#).

```

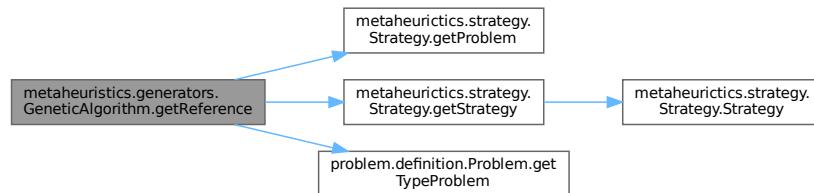
00132
00133     if (listState == null || listState.isEmpty()) {
00134         return null;
00135     }
00136     stateReferenceGA = listState.get(0);
00137     if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00138         for (int i = 1; i < listState.size(); i++) {
00139             if(stateReferenceGA.getEvaluation().get(0) < listState.get(i).getEvaluation().get(0))
00140                 stateReferenceGA = listState.get(i);
00141         }
00142     }
00143     else{
00144         for (int i = 1; i < listState.size(); i++) {
00145             if(stateReferenceGA.getEvaluation().get(0) > listState.get(i).getEvaluation().get(0))
  
```

```

00146         stateReferenceGA = listState.get(i);
00147     }
00148 }
00149 return (stateReferenceGA == null) ? null : new State(stateReferenceGA);
00150 }
```

Hace referencia a `metaheuristics.strategy.Strategy.getProblem()`, `metaheuristics.strategy.Strategy.getStrategy()`, `problem.definition.Problem.getTypeProblem()`, `listState`, `problem.definition.Problem.ProblemType.Maximizar` y `stateReferenceGA`.

Gráfico de llamadas de esta función:



### 8.39.3.9 getReferenceList()

`List< State > metaheuristics.generators.GeneticAlgorithm.getReferenceList () [inline]`

`getReferenceList` - method to get the current list of reference states.

Devuelve

returns the current list of reference states.

Reimplementado de `metaheuristics.generators.Generator`.

Definición en la línea 253 del archivo `GeneticAlgorithm.java`.

```

00253     {
00254         List<State> ReferenceList = new ArrayList<State>();
00255         for (int i = 0; i < listState.size(); i++) {
00256             State value = listState.get(i);
00257             ReferenceList.add(value);
00258         }
00259         return new ArrayList<State>(ReferenceList);
00260     }
```

Hace referencia a `listState`.

### 8.39.3.10 getSonList()

`List< State > metaheuristics.generators.GeneticAlgorithm.getSonList () [inline]`

`getSonList` - method to get the current list of son states.

Devuelve

returns the current list of son states.

Reimplementado de `metaheuristics.generators.Generator`.

Definición en la línea 267 del archivo `GeneticAlgorithm.java`.

```

00267     {
00268         // TODO Auto-generated method stub
00269         return null;
00270     }
```

### 8.39.3.11 getTrace()

```
float[ ] metaheuristics.generators.GeneticAlgorithm.getTrace () [inline]
```

getTrace - method to get the trace.

#### Devuelve

returns the trace.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 328 del archivo [GeneticAlgorithm.java](#).

```
00328      {
00329          // TODO Auto-generated method stub
00330          return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00331              this.listTrace.length);
00331      }
```

Hace referencia a [listTrace](#).

### 8.39.3.12 getType()

```
GeneratorType metaheuristics.generators.GeneticAlgorithm.getType () [inline]
```

getType - method to get the generator type.

#### Devuelve

returns the generator type.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 244 del archivo [GeneticAlgorithm.java](#).

```
00244      {
00245          return this.generatorType;
00246      }
```

Hace referencia a [generatorType](#).

### 8.39.3.13 getWeight()

```
float metaheuristics.generators.GeneticAlgorithm.getWeight () [inline]
```

getWeight - method to get the weight.

#### Devuelve

returns the weight.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 289 del archivo [GeneticAlgorithm.java](#).

```
00289      {
00290          // TODO Auto-generated method stub
00291          return this.weight;
00292      }
```

Generado por Doxygen

Hace referencia a [weight](#).

**Parámetros**

<i>generatorType</i>	<input type="button" value=""/>
----------------------	---------------------------------

Definición en la línea 235 del archivo [GeneticAlgorithm.java](#).

```
00235      this.generatorType = generatorType;
00236
00237 }
```

Hace referencia a [generatorType](#).

**8.39.3.15 setInitialReference()**

```
void metaheuristics.generators.GeneticAlgorithm.setInitialReference (
    State stateInitialRef) [inline]
```

setInitialReference - method to set the initial reference state.

**Parámetros**

<i>stateInitialRef</i>	<input type="button" value=""/>
------------------------	---------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 164 del archivo [GeneticAlgorithm.java](#).

```
00164
00165     this.stateReferenceGA = (stateInitialRef == null) ? null : new State(stateInitialRef);
00166 }
```

**8.39.3.16 setListState()**

```
void metaheuristics.generators.GeneticAlgorithm.setListState (
    List< State > listState) [inline]
```

setListState - method to set the current list of states.

**Parámetros**

<i>listState</i>	<input type="button" value=""/>
------------------	---------------------------------

Definición en la línea 192 del archivo [GeneticAlgorithm.java](#).

```
00192
00193     this.listState = (listState == null) ? new ArrayList<State>() : new
00194         ArrayList<State>(listState);
```

Hace referencia a [listState](#).

**8.39.3.17 setStateRef()**

---

```
void metaheuristics.generators.GeneticAlgorithm.setStateRef (
    Generado por Doxygen State stateRef) [inline]
```

setStateRef - method to set the current reference state.

**Parámetros**

<i>stateRef</i>	<input type="button" value=""/>
-----------------	---------------------------------

Definición en la línea 156 del archivo [GeneticAlgorithm.java](#).

```
00156           {
00157             this.stateReferenceGA = (stateRef == null) ? null : new State(stateRef);
00158           }
```

**8.39.3.18 setWeight()**

```
void metaheuristics.generators.GeneticAlgorithm.setWeight (
    float weight) [inline]
```

`setWeight` - method to set the weight.

**Parámetros**

<i>weight</i>	<input type="button" value=""/>
---------------	---------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 299 del archivo [GeneticAlgorithm.java](#).

```
00299           {
00300             // TODO Auto-generated method stub
00301             this.weight = weight;
00302           }
```

Hace referencia a [weight](#).

**8.39.3.19 updateReference()**

```
void metaheuristics.generators.GeneticAlgorithm.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

`updateReference` - method to update the current reference state.

**Parámetros**

<i>stateCandidate</i>	<input type="button" value=""/>
<i>countIterationsCurrent</i>	<input type="button" value=""/>

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 174 del archivo [GeneticAlgorithm.java](#).

```
00174           {
00175             iffreplace = new FactoryReplace();
00176             Replace replace = iffreplace.createReplace(replaceType);
00177             listState = replace.replace(stateCandidate, listState);
```

```
00178 }
```

Hace referencia a [iffreplace](#), [listState](#), [evolutionary\\_algorithms.complement.Replace.replace\(\)](#) y [replaceType](#).

Gráfico de llamadas de esta función:



## 8.39.4 Documentación de datos miembro

### 8.39.4.1 countRef

```
final int metaheuristics.generators.GeneticAlgorithm.countRef = 0 [static]
```

Definición en la línea 55 del archivo [GeneticAlgorithm.java](#).

Referenciado por [metaheuristics.generators.RandomSearch.generate\(\)](#), [metaheuristics.strategy.Strategy.update\(\)](#) y [local\\_search.complement.UpdateParameter.updateParameter\(\)](#).

### 8.39.4.2 crossoverType

```
final CrossoverType metaheuristics.generators.GeneticAlgorithm.crossoverType = CrossoverType.UniformCrossover [static]
```

Definición en la línea 50 del archivo [GeneticAlgorithm.java](#).

Referenciado por [generate\(\)](#).

### 8.39.4.3 generatorType

```
GeneratorType metaheuristics.generators.GeneticAlgorithm.generatorType [private]
```

Definición en la línea 48 del archivo [GeneticAlgorithm.java](#).

Referenciado por [getGeneratorType\(\)](#), [getType\(\)](#) y [setGeneratorType\(\)](#).

### 8.39.4.4 iffactorycrossover

```
IFFactoryCrossover metaheuristics.generators.GeneticAlgorithm.iffactorycrossover [private]
```

Definición en la línea 39 del archivo [GeneticAlgorithm.java](#).

Referenciado por [generate\(\)](#).

#### 8.39.4.5 iffactorymutation

```
IFFFactoryMutation metaheuristics.generators.GeneticAlgorithm.iffactorymutation [private]
```

Definición en la línea 40 del archivo [GeneticAlgorithm.java](#).

Referenciado por [generate\(\)](#).

#### 8.39.4.6 iffatherselection

```
IFFFactoryFatherSelection metaheuristics.generators.GeneticAlgorithm.iffatherselection [private]
```

Definición en la línea 38 del archivo [GeneticAlgorithm.java](#).

Referenciado por [generate\(\)](#).

#### 8.39.4.7 iffreplace

```
IFFFactoryReplace metaheuristics.generators.GeneticAlgorithm.iffreplace [private]
```

Definición en la línea 41 del archivo [GeneticAlgorithm.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.39.4.8 listCountBetterGender

```
int [] metaheuristics.generators.GeneticAlgorithm.listCountBetterGender = new int[10] [private]
```

Definición en la línea 61 del archivo [GeneticAlgorithm.java](#).

Referenciado por [GeneticAlgorithm\(\)](#) y [getListCountBetterGender\(\)](#).

#### 8.39.4.9 listCountGender

```
int [] metaheuristics.generators.GeneticAlgorithm.listCountGender = new int[10] [private]
```

Definición en la línea 62 del archivo [GeneticAlgorithm.java](#).

Referenciado por [GeneticAlgorithm\(\)](#) y [getListCountGender\(\)](#).

#### 8.39.4.10 listState

```
List<State> metaheuristics.generators.GeneticAlgorithm.listState = new ArrayList<State>() [private]
```

Definición en la línea 37 del archivo [GeneticAlgorithm.java](#).

Referenciado por [generate\(\)](#), [getListState\(\)](#), [getListStateRef\(\)](#), [getReference\(\)](#), [getReferenceList\(\)](#), [setListState\(\)](#) y [updateReference\(\)](#).

**8.39.4.11 listTrace**

```
float [ ] metaheuristics.generators.GeneticAlgorithm.listTrace = new float[1200000] [private]
```

Definición en la línea 63 del archivo [GeneticAlgorithm.java](#).

Referenciado por [GeneticAlgorithm\(\)](#) y [getTrace\(\)](#).

**8.39.4.12 mutationType**

```
final MutationType metaheuristics.generators.GeneticAlgorithm.mutationType = MutationType.OnePointMutation  
[static]
```

Definición en la línea 49 del archivo [GeneticAlgorithm.java](#).

Referenciado por [generate\(\)](#).

**8.39.4.13 PC**

```
final double metaheuristics.generators.GeneticAlgorithm.PC = 0.6 [static]
```

Definición en la línea 53 del archivo [GeneticAlgorithm.java](#).

Referenciado por [generate\(\)](#).

**8.39.4.14 PM**

```
final double metaheuristics.generators.GeneticAlgorithm.PM = 0.01 [static]
```

Definición en la línea 54 del archivo [GeneticAlgorithm.java](#).

Referenciado por [generate\(\)](#).

**8.39.4.15 replaceType**

```
final ReplaceType metaheuristics.generators.GeneticAlgorithm.replaceType = ReplaceType.GenerationalReplace  
[static]
```

Definición en la línea 51 del archivo [GeneticAlgorithm.java](#).

Referenciado por [updateReference\(\)](#).

**8.39.4.16 selectionType**

```
final SelectionType metaheuristics.generators.GeneticAlgorithm.selectionType = SelectionType.TruncationSelecti  
[static]
```

Definición en la línea 52 del archivo [GeneticAlgorithm.java](#).

Referenciado por [generate\(\)](#).

#### 8.39.4.17 stateReferenceGA

```
State metaheuristics.generators.GeneticAlgorithm.stateReferenceGA [private]
```

Definición en la línea 36 del archivo [GeneticAlgorithm.java](#).

Referenciado por [getReference\(\)](#).

#### 8.39.4.18 truncation

```
final int metaheuristics.generators.GeneticAlgorithm.truncation = 0 [static]
```

Definición en la línea 56 del archivo [GeneticAlgorithm.java](#).

Referenciado por [generate\(\)](#).

#### 8.39.4.19 weight

```
float metaheuristics.generators.GeneticAlgorithm.weight [private]
```

Definición en la línea 57 del archivo [GeneticAlgorithm.java](#).

Referenciado por [GeneticAlgorithm\(\)](#), [getWeight\(\)](#) y [setWeight\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [GeneticAlgorithm.java](#)

### 8.40 Referencia de la clase

#### [local\\_search.candidate\\_type.GreaterCandidate](#)

[GreaterCandidate](#) - choose the neighbor with the greatest evaluation.

Diagrama de herencia de [local\\_search.candidate\\_type.GreaterCandidate](#)

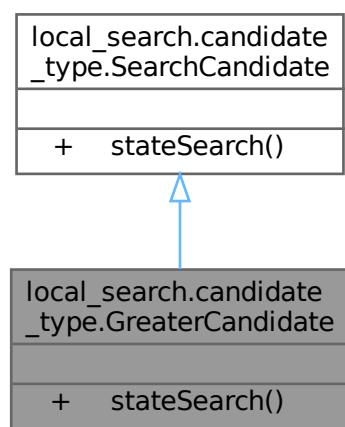
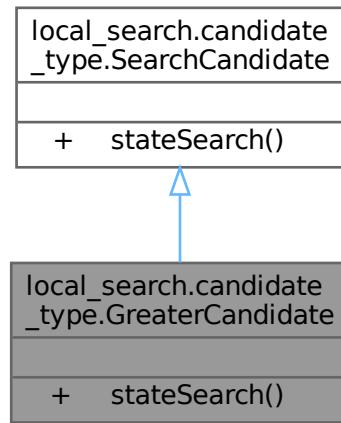


Diagrama de colaboración de local\_search.candidate\_type.GreaterCandidate:



## Métodos públicos

- **State stateSearch (List< State > listNeighborhood)** throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException

*Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.*

### 8.40.1 Descripción detallada

[GreaterCandidate](#) - choose the neighbor with the greatest evaluation.

Scans the provided neighborhood and returns the state with the highest primary objective value. If multiple candidates tie, a random choice among them is returned.

Definición en la línea [21](#) del archivo [GreaterCandidate.java](#).

### 8.40.2 Documentación de funciones miembro

#### 8.40.2.1 stateSearch()

```

State local_search.candidate_type.GreaterCandidate.stateSearch (
    List< State > listNeighborhood) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
  
```

Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.

This RNG is used to pick a random neighbor when multiple equal bests are found and is not security-sensitive. Suppress Sonar S2245. Select the greatest neighbor from the list.

## Parámetros

<i>listNeighborhood</i>	list of neighbor states
-------------------------	-------------------------

## Devuelve

the neighbor with the greatest evaluation (primary objective)

## Excepciones

<i>ReflectiveOperationException</i>	propagated from potential evaluations
-------------------------------------	---------------------------------------

Reimplementado de [local\\_search.candidate\\_type.SearchCandidate](#).

Definición en la línea 37 del archivo [GreaterCandidate.java](#).

```

00037
00038     {
00039         State stateGreater = null;
00040         if(listNeighborhood.size() > 1){
00041             double counter = 0;
00042             double currentCount = listNeighborhood.get(0).getEvaluation().get(0);
00043             for (int i = 1; i < listNeighborhood.size(); i++) {
00044                 counter = listNeighborhood.get(i).getEvaluation().get(0);
00045                 if (counter > currentCount) {
00046                     currentCount = counter;
00047                     stateGreater = listNeighborhood.get(i);
00048                 }
00049             }
00050             counter = 0;
00051             if(stateGreater == null){
00052                 // bound is listNeighborhood.size() - 1 (>=1 since size>1)
00053                 int pos = ThreadLocalRandom.current().nextInt(listNeighborhood.size() - 1);
00054                 stateGreater = listNeighborhood.get(pos);
00055             }
00056             else stateGreater = listNeighborhood.get(0);
00057         }
00058     }

```

La documentación de esta clase está generada del siguiente archivo:

- [GreaterCandidate.java](#)

## 8.41 Referencia de la clase metaheuristics.generators.HillClimbing

[HillClimbing](#) - class that implements the Hill Climbing metaheuristic.

Diagrama de herencia de metaheuristics.generators.HillClimbing

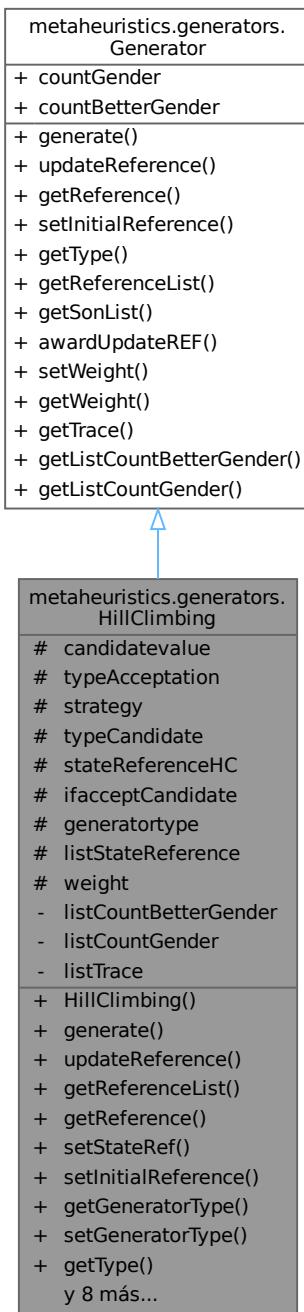
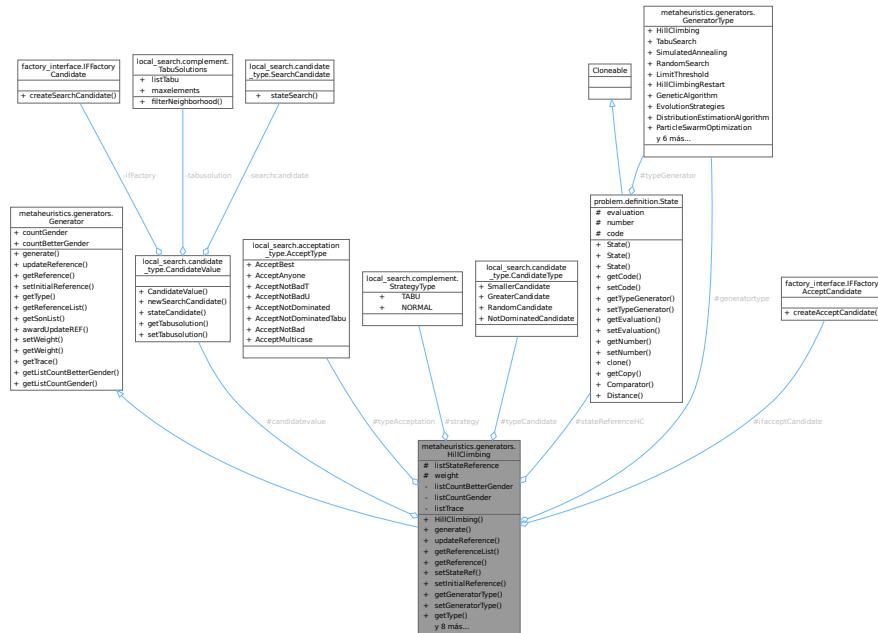


Diagrama de colaboración de metaheuristics.generators.HillClimbing:



## Métodos públicos

- **HillClimbing ()**

*HillClimbing - class that implements the Hill Climbing metaheuristic.*

- **State generate (Integer operatornumber)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*generate - generate a new state based on the current state and the operator number.*

- **void updateReference (State stateCandidate, Integer countIterationsCurrent)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*updateReference - update the reference state if the candidate state is accepted.*

- **List< State > getReferenceList ()**

*getReferenceList - get the list of reference states.*

- **State getReference ()**

*getReference - get the reference state.*

- **void setStateRef (State stateRef)**

*setStateRef - set the reference state.*

- **void setInitialReference (State statelinitialRef)**

*setInitialReference - set the initial reference state.*

- **GeneratorType getGeneratorType ()**

*getGeneratorType - get the generator type.*

- **void setGeneratorType (GeneratorType generatortype)**

*setGeneratorType - set the generator type.*

- **GeneratorType getType ()**

*getType - get the generator type.*

- **List< State > getSonList ()**

*getSonList - get the list of son states.*

- void [setTypeCandidate \(CandidateType typeCandidate\)](#)  
*setTypeCandidate - set the type of candidate.*
- boolean [awardUpdateREF \(State stateCandidate\)](#)  
*awardUpdateREF - award the update of the reference state if the candidate state is accepted.*
- float [getWeight \(\)](#)  
*getWeight - get the weight of the candidate.*
- void [setWeight \(float weight\)](#)  
*setWeight - set the weight of the candidate.*
- int[] [getListCountBetterGender \(\)](#)  
*getListCountBetterGender - get the list of counts of better candidates by gender.*
- int[] [getListCountGender \(\)](#)  
*getListCountGender - get the list of counts of candidates by gender.*
- float[] [getTrace \(\)](#)  
*getTrace - get the trace of the candidate.*

### Atributos protegidos

- CandidateValue [candidatevalue](#)
- AcceptType [typeAcceptation](#)
- StrategyType [strategy](#)
- CandidateType [typeCandidate](#)
- State [stateReferenceHC](#)
- IFFactoryAcceptCandidate [ifacceptCandidate](#)
- GeneratorType [generatortype](#)
- List< State > [listStateReference](#) = new ArrayList<State>()
- float [weight](#)

### Atributos privados

- int[] [listCountBetterGender](#) = new int[10]
- int[] [listCountGender](#) = new int[10]
- float[] [listTrace](#) = new float[1200000]

### Otros miembros heredados

#### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- int [countGender](#)
- int [countBetterGender](#)

### 8.41.1 Descripción detallada

[HillClimbing](#) - class that implements the Hill Climbing metaheuristic.

Definición en la línea 27 del archivo [HillClimbing.java](#).

## 8.41.2 Documentación de constructores y destructores

### 8.41.2.1 HillClimbing()

```
metaheuristics.generators.HillClimbing.HillClimbing () [inline]
```

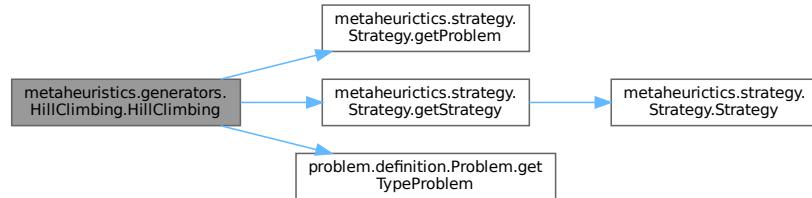
HillClimbing - class that implements the Hill Climbing metaheuristic.

Definición en la línea 48 del archivo [HillClimbing.java](#).

```
00048         {
00049             super();
00050             this.typeAcceptation = AcceptType.AcceptBest;
00051             this.strategy = StrategyType.NORMAL;
00052             if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)) {
00053                 this.typeCandidate = CandidateType.GreaterCandidate;
00054             }
00055             else{
00056                 this.typeCandidate = CandidateType.SmallerCandidate;
00057             }
00058             this.candidatevalue = new CandidateValue();
00059             this.generatortype = GeneratorType.HillClimbing;
00060             this.weight = 50;
00061             listTrace[0] = this.weight;
00062             listCountBetterGender[0] = 0;
00063             listCountGender[0] = 0;
00064 }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptType.AcceptBest](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Problem.getTypeProblem\(\)](#), [local\\_search.candidate\\_type.Candidate](#), [metaheuristics.generators.GeneratorType.HillClimbing](#), [listCountBetterGender](#), [listCountGender](#), [listTrace](#), [problem.definition.Problem.ProblemType.Maximizar](#), [local\\_search.complement.StrategyType.NORMAL](#) y [local\\_search.candidate\\_type](#)

Gráfico de llamadas de esta función:



## 8.41.3 Documentación de funciones miembro

### 8.41.3.1 awardUpdateREF()

```
boolean metaheuristics.generators.HillClimbing.awardUpdateREF (
    State stateCandidate) [inline]
```

awardUpdateREF - award the update of the reference state if the candidate state is accepted.

#### Parámetros

<i>stateCandidate</i>	
-----------------------	--

**Devuelve**

returns true if the update is awarded, false otherwise.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 183 del archivo [HillClimbing.java](#).

```
00183         // TODO Auto-generated method stub
00184         return false;
00185     }
00186 }
```

**8.41.3.2 generate()**

```
State metaheuristics.generators.HillClimbing.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

generate - generate a new state based on the current state and the operator number.

**Parámetros**

<i>operatornumber</i>	the operator number to use for generating the new state
-----------------------	---

**Devuelve**

the newly generated state

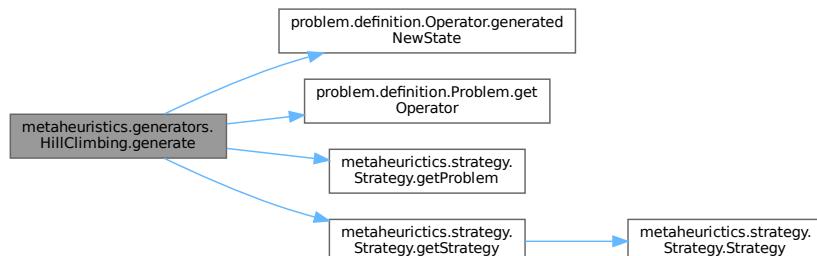
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 72 del archivo [HillClimbing.java](#).

```
00072 {
00073     List<State> neighborhood =
00074         Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC, operatornumber);
00075     State statecandidate = candidatevalue.stateCandidate(stateReferenceHC, typeCandidate,
00076         strategy, operatornumber, neighborhood);
00077 }
00078 return statecandidate;
```

Hace referencia a [candidatevalue](#), [problem.definition.Operator.generatedNewState\(\)](#), [problem.definition.Problem.getOperator\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [stateReferenceHC](#), [strategy](#) y [typeCandidate](#).

Gráfico de llamadas de esta función:



### 8.41.3.3 getGeneratorType()

```
GeneratorType metaheuristics.generators.HillClimbing.getGeneratorType () [inline]
```

getGeneratorType - get the generator type.

Devuelve

the generator type

Definición en la línea 138 del archivo [HillClimbing.java](#).

```
00138             {
00139         return generatortype;
00140     }
```

Hace referencia a [generatortype](#).

### 8.41.3.4 getListCountBetterGender()

```
int[] metaheuristics.generators.HillClimbing.getListCountBetterGender () [inline]
```

getListCountBetterGender - get the list of counts of better candidates by gender.

Devuelve

the list of counts of better candidates by gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 213 del archivo [HillClimbing.java](#).

```
00213             {
00214         // TODO Auto-generated method stub
00215         return (this.listCountBetterGender == null) ? new int[0] :
00216             java.util.Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
00217     }
```

### 8.41.3.5 getListCountGender()

```
int[] metaheuristics.generators.HillClimbing.getListCountGender () [inline]
```

getListCountGender - get the list of counts of candidates by gender.

Devuelve

the list of counts of candidates by gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 223 del archivo [HillClimbing.java](#).

```
00223             {
00224         // TODO Auto-generated method stub
00225         return (this.listCountGender == null) ? new int[0] :
00226             java.util.Arrays.copyOf(this.listCountGender, this.listCountGender.length);
00227     }
```

### 8.41.3.6 getReference()

```
State metaheuristics.generators.HillClimbing.getReference () [inline]
```

getReference - get the reference state.

Devuelve

the reference state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 113 del archivo [HillClimbing.java](#).

```
00113             {
00114         return (stateReferenceHC == null) ? null : new State(stateReferenceHC);
00115     }
```

Hace referencia a [stateReferenceHC](#).

### 8.41.3.7 getReferenceList()

```
List< State > metaheuristics.generators.HillClimbing.getReferenceList () [inline]
```

getReferenceList - get the list of reference states.

Devuelve

the list of reference states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 98 del archivo [HillClimbing.java](#).

```
00098             {
00099         if (stateReferenceHC != null) {
00100             // keep internal list updated but do not expose it directly
00101             if (listStateReference.isEmpty() || listStateReference.get(listStateReference.size() - 1)
00102                 != stateReferenceHC) {
00103                 listStateReference.add(stateReferenceHC);
00104             }
00105         }
00106     }
```

Hace referencia a [listStateReference](#) y [stateReferenceHC](#).

### 8.41.3.8 getSonList()

```
List< State > metaheuristics.generators.HillClimbing.getSonList () [inline]
```

getSonList - get the list of son states.

Devuelve

the list of son states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 164 del archivo [HillClimbing.java](#).

```
00164             {
00165         // TODO Auto-generated method stub
00166         return null;
00167     }
```

#### 8.41.3.9 getTrace()

```
float[ ] metaheuristics.generators.HillClimbing.getTrace () [inline]
```

getTrace - get the trace of the candidate.

**Devuelve**

the trace of the candidate

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 233 del archivo [HillClimbing.java](#).

```
00233           {
00234           // TODO Auto-generated method stub
00235           return (this.listTrace == null) ? new float[0] : java.util.Arrays.copyOf(this.listTrace,
00236           this.listTrace.length);
00236       }
```

#### 8.41.3.10 getType()

```
GeneratorType metaheuristics.generators.HillClimbing.getType () [inline]
```

getType - get the generator type.

**Devuelve**

the generator type

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 155 del archivo [HillClimbing.java](#).

```
00155           {
00156           return this.generatortype;
00157       }
```

#### 8.41.3.11 getWeight()

```
float metaheuristics.generators.HillClimbing.getWeight () [inline]
```

getWeight - get the weight of the candidate.

**Devuelve**

the weight of the candidate

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 193 del archivo [HillClimbing.java](#).

```
00193           {
00194           // TODO Auto-generated method stub
00195           return 0;
00196       }
```

#### 8.41.3.12 setGeneratorType()

```
void metaheuristics.generators.HillClimbing.setGeneratorType (
```

**Parámetros**

generatorotype	<input type="button" value=""/>
----------------	---------------------------------

Definición en la línea 146 del archivo [HillClimbing.java](#).

```
00146          this.generatorotype = generatorotype;
00147
00148 }
```

Hace referencia a [generatorotype](#).

**8.41.3.13 setInitialReference()**

```
void metaheuristics.generators.HillClimbing.setInitialReference (
    State stateInitialRef) [inline]
```

setInitialReference - set the initial reference state.

**Parámetros**

stateInitialRef	<input type="button" value=""/>
-----------------	---------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 130 del archivo [HillClimbing.java](#).

```
00130
00131     this.stateReferenceHC = (stateInitialRef == null) ? null : new State(stateInitialRef);
00132 }
```

**8.41.3.14 setStateRef()**

```
void metaheuristics.generators.HillClimbing.setStateRef (
    State stateRef) [inline]
```

setStateRef - set the reference state.

**Parámetros**

stateRef	<input type="button" value=""/>
----------	---------------------------------

Definición en la línea 121 del archivo [HillClimbing.java](#).

```
00121
00122     this.stateReferenceHC = (stateRef == null) ? null : new State(stateRef);
00123 }
```

**8.41.3.15 setTypeCandidate()**

```
void metaheuristics.generators.HillClimbing.setTypeCandidate (
    CandidateType typeCandidate) [inline]
```

setTypeCandidate - set the type of candidate.

Generado por Doxygen

## Parámetros

<code>typeCandidate</code>	<input type="button" value=""/>
----------------------------	---------------------------------

Definición en la línea 173 del archivo [HillClimbing.java](#).

```
00173         this.typeCandidate = typeCandidate;
00174     }
00175 }
```

Hace referencia a [typeCandidate](#).

### 8.41.3.16 setWeight()

```
void metaheuristics.generators.HillClimbing.setWeight (
    float weight) [inline]
```

`setWeight` - set the weight of the candidate.

## Parámetros

<code>weight</code>	<input type="button" value=""/>
---------------------	---------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 203 del archivo [HillClimbing.java](#).

```
00203         {
00204     // TODO Auto-generated method stub
00205
00206 }
```

Hace referencia a [weight](#).

### 8.41.3.17 updateReference()

```
void metaheuristics.generators.HillClimbing.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

`updateReference` - update the reference state if the candidate state is accepted.

## Parámetros

<code>stateCandidate</code>	<input type="button" value=""/>
<code>countIterationsCurrent</code>	<input type="button" value=""/>

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 85 del archivo [HillClimbing.java](#).

```
00085 {
```

```

00086     ifacceptCandidate = new FactoryAcceptCandidate();
00087     AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00088     Boolean accept = candidate.acceptCandidate(stateReferenceHC, stateCandidate);
00089     if(accept.equals(true))
00090         stateReferenceHC = stateCandidate;
00091 }

```

Hace referencia a `local_search.acceptation_type.AcceptableCandidate.acceptCandidate()`, `ifacceptCandidate`, `stateReferenceHC` y `typeAcceptation`.

Gráfico de llamadas de esta función:



#### 8.41.4 Documentación de datos miembro

##### 8.41.4.1 candidatevalue

`CandidateValue` metaheuristics.generators.HillClimbing.candidatevalue [protected]

Definición en la línea 29 del archivo [HillClimbing.java](#).

Referenciado por [generate\(\)](#).

##### 8.41.4.2 generatortype

`GeneratorType` metaheuristics.generators.HillClimbing.generatortype [protected]

Definición en la línea 35 del archivo [HillClimbing.java](#).

Referenciado por [getGeneratorType\(\)](#) y [setGeneratorType\(\)](#).

##### 8.41.4.3 ifacceptCandidate

`IIFactoryAcceptCandidate` metaheuristics.generators.HillClimbing.ifacceptCandidate [protected]

Definición en la línea 34 del archivo [HillClimbing.java](#).

Referenciado por [updateReference\(\)](#).

##### 8.41.4.4 listCountBetterGender

`int []` metaheuristics.generators.HillClimbing.listCountBetterGender = new int[10] [private]

Definición en la línea 41 del archivo [HillClimbing.java](#).

Referenciado por [HillClimbing\(\)](#).

#### 8.41.4.5 listCountGender

```
int [] metaheuristics.generators.HillClimbing.listCountGender = new int[10] [private]
```

Definición en la línea 42 del archivo [HillClimbing.java](#).

Referenciado por [HillClimbing\(\)](#).

#### 8.41.4.6 listStateReference

```
List<State> metaheuristics.generators.HillClimbing.listStateReference = new ArrayList<State>()  
[protected]
```

Definición en la línea 36 del archivo [HillClimbing.java](#).

Referenciado por [getReferenceList\(\)](#).

#### 8.41.4.7 listTrace

```
float [] metaheuristics.generators.HillClimbing.listTrace = new float[1200000] [private]
```

Definición en la línea 43 del archivo [HillClimbing.java](#).

Referenciado por [HillClimbing\(\)](#).

#### 8.41.4.8 stateReferenceHC

```
State metaheuristics.generators.HillClimbing.stateReferenceHC [protected]
```

Definición en la línea 33 del archivo [HillClimbing.java](#).

Referenciado por [generate\(\)](#), [getReference\(\)](#), [getReferenceList\(\)](#) y [updateReference\(\)](#).

#### 8.41.4.9 strategy

```
StrategyType metaheuristics.generators.HillClimbing.strategy [protected]
```

Definición en la línea 31 del archivo [HillClimbing.java](#).

Referenciado por [generate\(\)](#).

#### 8.41.4.10 typeAcceptation

```
AcceptType metaheuristics.generators.HillClimbing.typeAcceptation [protected]
```

Definición en la línea 30 del archivo [HillClimbing.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.41.4.11 typeCandidate

`CandidateType` metaheuristics.generators.HillClimbing.typeCandidate [protected]

Definición en la línea 32 del archivo [HillClimbing.java](#).

Referenciado por [generate\(\)](#) y [setTypeCandidate\(\)](#).

#### 8.41.4.12 weight

`float` metaheuristics.generators.HillClimbing.weight [protected]

Definición en la línea 37 del archivo [HillClimbing.java](#).

Referenciado por [setWeight\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [HillClimbing.java](#)

## 8.42 Referencia de la clase metaheuristics.generators.HillClimbingRestart

[HillClimbingRestart](#) - class that implements the Hill Climbing Restart metaheuristic.

## Diagrama de herencia de metaheuristics.generators.HillClimbingRestart

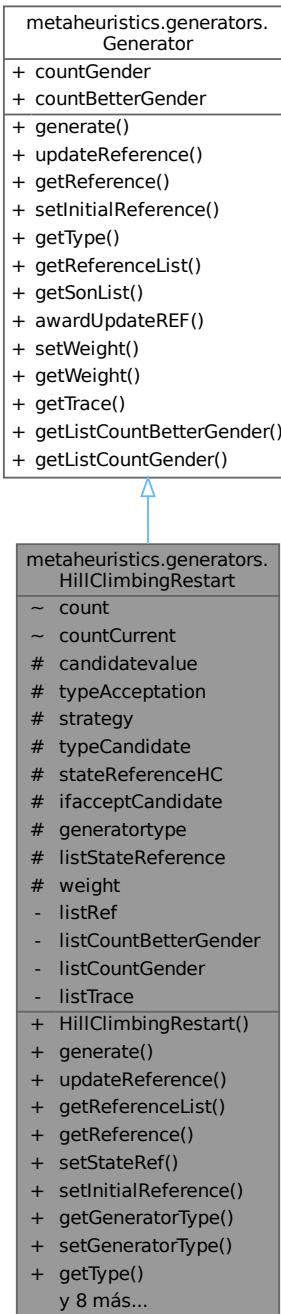
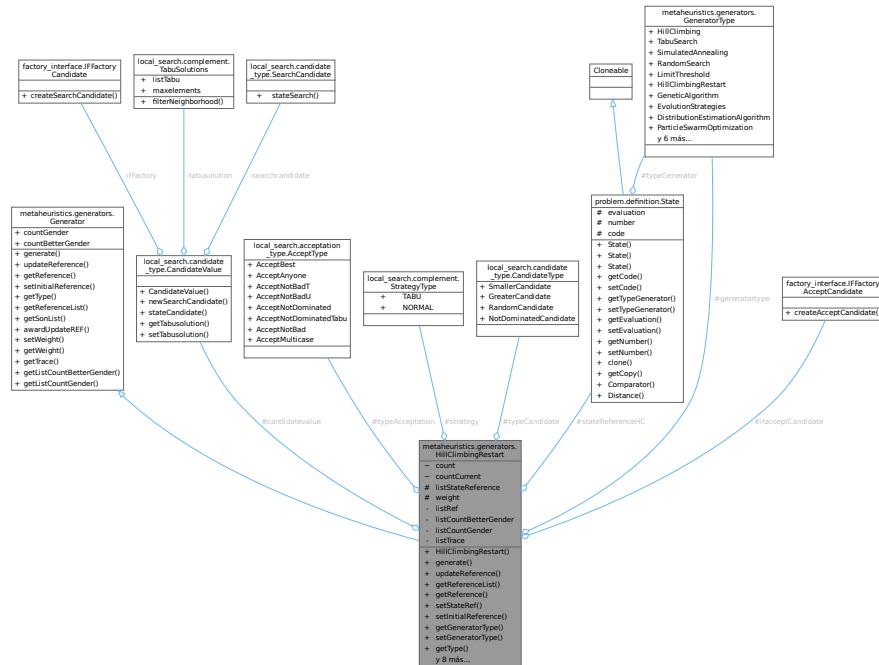


Diagrama de colaboración de metaheuristics.generators.HillClimbingRestart:



## Métodos públicos

- **HillClimbingRestart ()**

*HillClimbingRestart - class that implements the Hill Climbing Restart metaheuristic.*

- **State generate (Integer operatornumber)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*generate - generate a new state based on the current state and the operator number.*

- **void updateReference (State stateCandidate, Integer countIterationsCurrent)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

- **List< State > getReferenceList ()**

*getReferenceList - get the list of reference states.*

- **State getReference ()**

*getReference - get the reference state.*

- **void setStateRef (State stateRef)**

*setStateRef - set the reference state.*

- **void setInitialReference (State statelInitialRef)**

*setInitialReference - set the initial reference state.*

- **GeneratorType getGeneratorType ()**

*getGeneratorType - get the generator type.*

- **void setGeneratorType (GeneratorType generatortype)**

*setGeneratorType - set the generator type.*

- **GeneratorType getType ()**

*getType - get the generator type.*

- **List< State > getSonList ()**

*getSonList - get the list of son states.*

- void [setTypeCandidate \(CandidateType typeCandidate\)](#)  
*setTypeCandidate - set the candidate type.*
- boolean [awardUpdateREF \(State stateCandidate\)](#)  
*awardUpdateREF - award the update of the reference state.*
- float [getWeight \(\)](#)  
*getWeight - get the weight of the state.*
- void [setWeight \(float weight\)](#)  
*setWeight - set the weight of the state.*
- int[] [getListCountBetterGender \(\)](#)  
*getListCountBetterGender - get the list of count better gender.*
- int[] [getListCountGender \(\)](#)  
*getListCountGender - get the list of count gender.*
- float[] [getTrace \(\)](#)  
*getTrace - get the trace of the state.*

### Atributos protegidos

- CandidateValue [candidatevalue](#)
- AcceptType [typeAcceptation](#)
- StrategyType [strategy](#)
- CandidateType [typeCandidate](#)
- State [stateReferenceHC](#)
- IFFactoryAcceptCandidate [ifacceptCandidate](#)
- GeneratorType [generatortype](#)
- List< State > [listStateReference](#) = new ArrayList<State>()
- float [weight](#)

### Atributos privados

- List< State > [listRef](#) = new ArrayList<State>()
- int[] [listCountBetterGender](#) = new int[10]
- int[] [listCountGender](#) = new int[10]
- float[] [listTrace](#) = new float[1200000]

### Otros miembros heredados

#### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- int [countGender](#)
- int [countBetterGender](#)

### 8.42.1 Descripción detallada

[HillClimbingRestart](#) - class that implements the Hill Climbing Restart metaheuristic.

Definición en la línea 23 del archivo [HillClimbingRestart.java](#).

## 8.42.2 Documentación de constructores y destructores

### 8.42.2.1 HillClimbingRestart()

```
metaheuristics.generators.HillClimbingRestart.HillClimbingRestart () [inline]
```

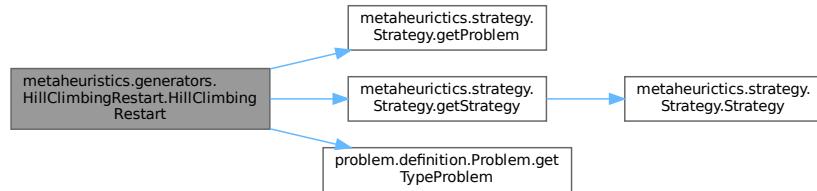
[HillClimbingRestart](#) - class that implements the Hill Climbing Restart metaheuristic.

Definición en la línea 47 del archivo [HillClimbingRestart.java](#).

```
00047         super();
00048         countCurrent = count;
00049         this.typeAcceptation = AcceptType.AcceptBest;
00050         this.strategy = StrategyType.NORMAL;
00051         if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)) {
00052             this.typeCandidate = CandidateType.GreaterCandidate;
00053         }
00054         else{
00055             this.typeCandidate = CandidateType.SmallerCandidate;
00056         }
00057         this.candidatevalue = new CandidateValue();
00058         this.generatortype = GeneratorType.HillClimbing;
00059         this.weight = 50;
00060         listTrace[0] = this.weight;
00061         listCountBetterGender[0] = 0;
00062         listCountGender[0] = 0;
00063     }
00064 }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptType.AcceptBest](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Problem.getTypeProblem\(\)](#), [local\\_search.candidate\\_type.Candidate](#), [metaheuristics.generators.GeneratorType.HillClimbing](#), [listCountBetterGender](#), [listCountGender](#), [listTrace](#), [problem.definition.ProblemType.Maximizar](#), [local\\_search.complement.StrategyType.NORMAL](#) y [local\\_search.candidate\\_type](#)

Gráfico de llamadas de esta función:



## 8.42.3 Documentación de funciones miembro

### 8.42.3.1 awardUpdateREF()

```
boolean metaheuristics.generators.HillClimbingRestart.awardUpdateREF (
    State stateCandidate) [inline]
```

`awardUpdateREF` - award the update of the reference state.

#### Parámetros

<i>stateCandidate</i>	
-----------------------	--

**Devuelve**

returns true if the update is awarded, false otherwise.

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 192 del archivo [HillClimbingRestart.java](#).

```
00192
00193     // TODO Auto-generated method stub
00194     return false;
00195 }
```

**8.42.3.2 generate()**

```
State metaheuristics.generators.HillClimbingRestart.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

generate - generate a new state based on the current state and the operator number.

**Parámetros**

<i>operatornumber</i>	the operator number to use for generating the new state
-----------------------	---

**Devuelve**

the newly generated state

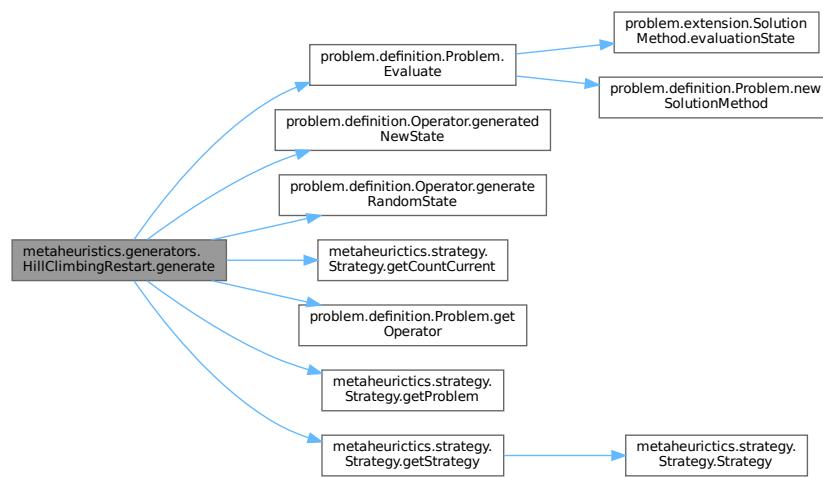
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 73 del archivo [HillClimbingRestart.java](#).

```
00073
00074     {
00075         State statecandidate;
00076         if(count == Strategy.getStrategy().getCountCurrent()){
00077             State stateR = new State(stateReferenceHC);
00078             listRef.add(stateR);
00079             stateReferenceHC =
00080                 Strategy.getStrategy().getProblem().getOperator().generateRandomState(1).get(0);
00081             Strategy.getStrategy().getProblem().Evaluate(stateReferenceHC);
00082             count = count + countCurrent;
00083         }
00084         List<State> neighborhood =
00085             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC, operatornumber);
00086         statecandidate = candidatevalue.stateCandidate(stateReferenceHC, typeCandidate, strategy,
00087             operatornumber, neighborhood);
00088         return statecandidate;
00089     }
```

Hace referencia a [candidatevalue](#), [problem.definition.Problem.Evaluate\(\)](#), [problem.definition.Operator.generatedNewState\(\)](#), [problem.definition.Operator.generateRandomState\(\)](#), [metaheuristics.strategy.Strategy.getCountCurrent\(\)](#), [problem.definition.Problem](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [listRef](#), [stateReferenceHC](#), [strategy](#) y [typeCandidate](#).

Gráfico de llamadas de esta función:



#### 8.42.3.3 getGeneratorType()

`GeneratorType metaheuristics.generators.HillClimbingRestart.getGeneratorType () [inline]`

getGeneratorType - get the generator type.

Devuelve

the generator type

Definición en la línea 147 del archivo [HillClimbingRestart.java](#).

```
00147           {  
00148     return generatortype;  
00149 }
```

Hace referencia a [generatortype](#).

#### 8.42.3.4 getListCountBetterGender()

`int[] metaheuristics.generators.HillClimbingRestart.getListCountBetterGender () [inline]`

getListCountBetterGender - get the list of count better gender.

Devuelve

the list of count better gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 225 del archivo [HillClimbingRestart.java](#).

```
00225           {  
00226     // TODO Auto-generated method stub  
00227     return (this.listCountBetterGender == null) ? new int[0] :  
        java.util.Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);  
00228 }
```

### 8.42.3.5 getListCountGender()

```
int[] metaheuristics.generators.HillClimbingRestart.getListCountGender () [inline]
```

getListCountGender - get the list of count gender.

Devuelve

the list of count gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 235 del archivo [HillClimbingRestart.java](#).

```
00235      {
00236          // TODO Auto-generated method stub
00237          return (this.listCountGender == null) ? new int[0] :
00238              java.util.Arrays.copyOf(this.listCountGender, this.listCountGender.length);
00239      }
```

### 8.42.3.6 getReference()

```
State metaheuristics.generators.HillClimbingRestart.getReference () [inline]
```

getReference - get the reference state.

Devuelve

the reference state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 122 del archivo [HillClimbingRestart.java](#).

```
00122      {
00123          return (stateReferenceHC == null) ? null : new State(stateReferenceHC);
00124      }
```

Hace referencia a [stateReferenceHC](#).

### 8.42.3.7 getReferenceList()

```
List< State > metaheuristics.generators.HillClimbingRestart.getReferenceList () [inline]
```

getReferenceList - get the list of reference states.

Devuelve

the list of reference states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 108 del archivo [HillClimbingRestart.java](#).

```
00108      {
00109          if (stateReferenceHC != null) {
00110              if (listStateReference.isEmpty() || listStateReference.get(listStateReference.size() - 1)
00111                  != stateReferenceHC) {
00112                  listStateReference.add(stateReferenceHC);
00113              }
00114          }
00115      }
```

Hace referencia a [listStateReference](#) y [stateReferenceHC](#).

### 8.42.3.8 getSonList()

```
List< State > metaheuristics.generators.HillClimbingRestart.getSonList () [inline]
```

getSonList - get the list of son states.

Devuelve

the list of son states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 173 del archivo [HillClimbingRestart.java](#).

```
00173                               {  
00174         // TODO Auto-generated method stub  
00175         return null;  
00176     }
```

### 8.42.3.9 getTrace()

```
float[] metaheuristics.generators.HillClimbingRestart.getTrace () [inline]
```

getTrace - get the trace of the state.

Devuelve

the trace of the state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 245 del archivo [HillClimbingRestart.java](#).

```
00245                               {  
00246         // TODO Auto-generated method stub  
00247         return (this.listTrace == null) ? new float[0] : java.util.Arrays.copyOf(this.listTrace,  
00248             this.listTrace.length);  
00248     }
```

### 8.42.3.10 getType()

```
GeneratorType metaheuristics.generators.HillClimbingRestart.getType () [inline]
```

getType - get the generator type.

Devuelve

the generator type

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 164 del archivo [HillClimbingRestart.java](#).

```
00164                               {  
00165         return this.generatortype;  
00166     }
```

#### 8.42.3.11 `getWeight()`

```
float metaheuristics.generators.HillClimbingRestart.getWeight () [inline]
```

`getWeight` - get the weight of the state.

##### Devuelve

the weight of the state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 203 del archivo [HillClimbingRestart.java](#).

```
00203             {
00204         // TODO Auto-generated method stub
00205         return 0;
00206     }
```

#### 8.42.3.12 `setGeneratorType()`

```
void metaheuristics.generators.HillClimbingRestart.setGeneratorType (
    GeneratorType generatortype) [inline]
```

`setGeneratorType` - set the generator type.

##### Parámetros

<code>generatortype</code>	<input type="button" value=""/>
----------------------------	---------------------------------

Definición en la línea 155 del archivo [HillClimbingRestart.java](#).

```
00155             {
00156         this.generatortype = generatortype;
00157     }
```

Hace referencia a [generatortype](#).

#### 8.42.3.13 `setInitialReference()`

```
void metaheuristics.generators.HillClimbingRestart.setInitialReference (
    State stateInitialRef) [inline]
```

`setInitialReference` - set the initial reference state.

##### Parámetros

<code>stateInitialRef</code>	<input type="button" value=""/>
------------------------------	---------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 139 del archivo [HillClimbingRestart.java](#).

```
00139             {
00140         this.stateReferenceHC = (stateInitialRef == null) ? null : new State(stateInitialRef);
00141     }
```

#### 8.42.3.14 `setStateRef()`

**Parámetros**

<i>stateRef</i>	<input type="text"/>
-----------------	----------------------

Definición en la línea 130 del archivo [HillClimbingRestart.java](#).

```
00130             {
00131         this.stateReferenceHC = (stateRef == null) ? null : new State(stateRef);
00132     }
```

**8.42.3.15 setTypeCandidate()**

```
void metaheuristics.generators.HillClimbingRestart.setTypeCandidate (
    CandidateType typeCandidate) [inline]
```

`setTypeCandidate` - set the candidate type.

**Parámetros**

<i>typeCandidate</i>	<input type="text"/>
----------------------	----------------------

Definición en la línea 182 del archivo [HillClimbingRestart.java](#).

```
00182             {
00183         this.typeCandidate = typeCandidate;
00184     }
```

Hace referencia a [typeCandidate](#).

**8.42.3.16 setWeight()**

```
void metaheuristics.generators.HillClimbingRestart.setWeight (
    float weight) [inline]
```

`setWeight` - set the weight of the state.

**Parámetros**

<i>weight</i>	<input type="text"/>
---------------	----------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 213 del archivo [HillClimbingRestart.java](#).

```
00213             {
00214         // TODO Auto-generated method stub
00215     }
00216 }
```

Hace referencia a [weight](#).

#### 8.42.3.17 updateReference()

```
void metaheuristics.generators.HillClimbingRestart.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

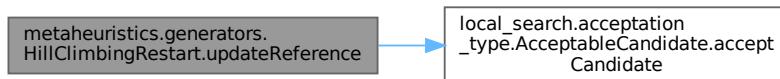
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 88 del archivo [HillClimbingRestart.java](#).

```
00092             {
00093         // TODO Auto-generated method stub
00094         ifacceptCandidate = new FactoryAcceptCandidate();
00095         AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00096         Boolean accept = candidate.acceptCandidate(stateReferenceHC, stateCandidate);
00097         if(accept.equals(true))
00098             stateReferenceHC = stateCandidate;
00099     }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptableCandidate.acceptCandidate\(\)](#), [ifacceptCandidate](#), [stateReferenceHC](#) y [typeAcceptation](#).

Gráfico de llamadas de esta función:



### 8.42.4 Documentación de datos miembro

#### 8.42.4.1 candidatevalue

[CandidateValue](#) metaheuristics.generators.HillClimbingRestart.candidatevalue [protected]

Definición en la línea 28 del archivo [HillClimbingRestart.java](#).

Referenciado por [generate\(\)](#).

#### 8.42.4.2 generatortype

[GeneratorType](#) metaheuristics.generators.HillClimbingRestart.generatortype [protected]

Definición en la línea 34 del archivo [HillClimbingRestart.java](#).

Referenciado por [getGeneratorType\(\)](#) y [setGeneratorType\(\)](#).

#### 8.42.4.3 ifacceptCandidate

```
IFFactoryAcceptCandidate metaheuristics.generators.HillClimbingRestart.ifacceptCandidate [protected]
```

Definición en la línea 33 del archivo [HillClimbingRestart.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.42.4.4 listCountBetterGender

```
int [] metaheuristics.generators.HillClimbingRestart.listCountBetterGender = new int[10] [private]
```

Definición en la línea 40 del archivo [HillClimbingRestart.java](#).

Referenciado por [HillClimbingRestart\(\)](#).

#### 8.42.4.5 listCountGender

```
int [] metaheuristics.generators.HillClimbingRestart.listCountGender = new int[10] [private]
```

Definición en la línea 41 del archivo [HillClimbingRestart.java](#).

Referenciado por [HillClimbingRestart\(\)](#).

#### 8.42.4.6 listRef

```
List<State> metaheuristics.generators.HillClimbingRestart.listRef = new ArrayList<State>()  
[private]
```

Definición en la línea 27 del archivo [HillClimbingRestart.java](#).

Referenciado por [generate\(\)](#).

#### 8.42.4.7 listStateReference

```
List<State> metaheuristics.generators.HillClimbingRestart.listStateReference = new Array←  
List<State>() [protected]
```

Definición en la línea 35 del archivo [HillClimbingRestart.java](#).

Referenciado por [getReferenceList\(\)](#).

#### 8.42.4.8 listTrace

```
float [] metaheuristics.generators.HillClimbingRestart.listTrace = new float[1200000] [private]
```

Definición en la línea 42 del archivo [HillClimbingRestart.java](#).

Referenciado por [HillClimbingRestart\(\)](#).

#### 8.42.4.9 stateReferenceHC

```
State metaheuristics.generators.HillClimbingRestart.stateReferenceHC [protected]
```

Definición en la línea 32 del archivo [HillClimbingRestart.java](#).

Referenciado por [generate\(\)](#), [getReference\(\)](#), [getReferenceList\(\)](#) y [updateReference\(\)](#).

#### 8.42.4.10 strategy

```
StrategyType metaheuristics.generators.HillClimbingRestart.strategy [protected]
```

Definición en la línea 30 del archivo [HillClimbingRestart.java](#).

Referenciado por [generate\(\)](#).

#### 8.42.4.11 typeAcceptation

```
AcceptType metaheuristics.generators.HillClimbingRestart.typeAcceptation [protected]
```

Definición en la línea 29 del archivo [HillClimbingRestart.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.42.4.12 typeCandidate

```
CandidateType metaheuristics.generators.HillClimbingRestart.typeCandidate [protected]
```

Definición en la línea 31 del archivo [HillClimbingRestart.java](#).

Referenciado por [generate\(\)](#) y [setTypeCandidate\(\)](#).

#### 8.42.4.13 weight

```
float metaheuristics.generators.HillClimbingRestart.weight [protected]
```

Definición en la línea 36 del archivo [HillClimbingRestart.java](#).

Referenciado por [setWeight\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [HillClimbingRestart.java](#)

## 8.43 Referencia de la interface factory\_interface.IFFactoryAcceptCandidate

[IFFactoryAcceptCandidate](#) - Interface for creating acceptable candidates.

Diagrama de herencia de factory\_interface.IFFactoryAcceptCandidate

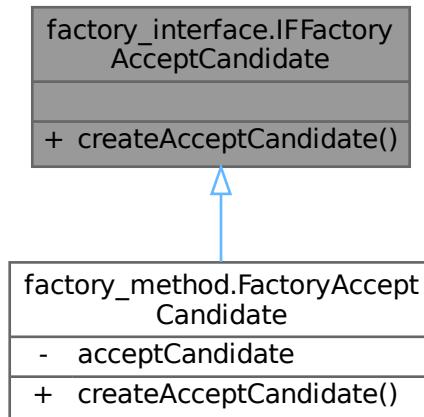
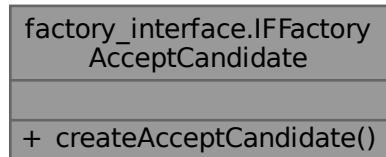


Diagrama de colaboración de factory\_interface.IFFactoryAcceptCandidate:



### Métodos públicos

- `AcceptableCandidate createAcceptCandidate (AcceptType typeacceptation) throws IllegalArgument←Exception, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

#### 8.43.1 Descripción detallada

[IFFactoryAcceptCandidate](#) - Interface for creating acceptable candidates.

Definición en la línea 15 del archivo [IFFactoryAcceptCandidate.java](#).

## 8.43.2 Documentación de funciones miembro

### 8.43.2.1 createAcceptCandidate()

```
AcceptableCandidate factory_interface.IFFactoryAcceptCandidate.createAcceptCandidate (
    AcceptType typeacceptation) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException
```

Implementado en [factory\\_method.FactoryAcceptCandidate](#).

La documentación de esta interface está generada del siguiente archivo:

- [IFFactoryAcceptCandidate.java](#)

## 8.44 Referencia de la interface factory\_interface.IFFactoryCandidate

[IFFactoryCandidate](#) - Interface for creating search candidates.

Diagrama de herencia de `factory_interface.IFFactoryCandidate`

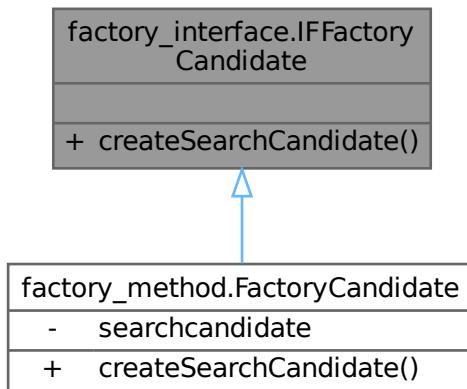
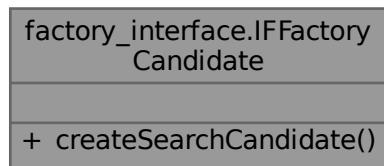


Diagrama de colaboración de `factory_interface.IFFactoryCandidate`:



## Métodos públicos

- `SearchCandidate createSearchCandidate (CandidateType typeCandidate) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

### 8.44.1 Descripción detallada

[IFFactoryCandidate](#) - Interface for creating search candidates.

Definición en la línea 18 del archivo [IFFactoryCandidate.java](#).

### 8.44.2 Documentación de funciones miembro

#### 8.44.2.1 createSearchCandidate()

```
SearchCandidate factory_interface.IFFactoryCandidate.createSearchCandidate (
    CandidateType typeCandidate) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException
```

Implementado en [factory\\_method.FactoryCandidate](#).

La documentación de esta interfaz está generada del siguiente archivo:

- [IFFactoryCandidate.java](#)

## 8.45 Referencia de la interface factory\_interface.IFFactoryCrossover

[IFFactoryCrossover](#) - Interface for creating crossover operators.

Diagrama de herencia de `factory_interface.IFFactoryCrossover`

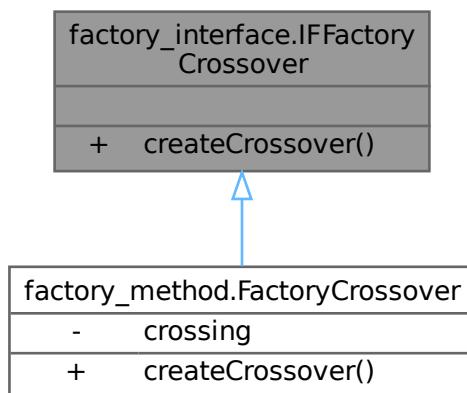
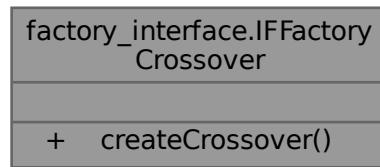


Diagrama de colaboración de factory\_interface.IFFactoryCrossover:



## Métodos públicos

- `Crossover createCrossover (CrossoverType CrossoverType) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

### 8.45.1 Descripción detallada

[IFFactoryCrossover](#) - Interface for creating crossover operators.

Definición en la línea 14 del archivo [IFFactoryCrossover.java](#).

### 8.45.2 Documentación de funciones miembro

#### 8.45.2.1 createCrossover()

```
Crossover factory_interface.IFFactoryCrossover.createCrossover (
    CrossoverType CrossoverType) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException
```

Implementado en [factory\\_method.FactoryCrossover](#).

La documentación de esta interface está generada del siguiente archivo:

- [IFFactoryCrossover.java](#)

## 8.46 Referencia de la interface factory\_interface.IFFactoryDistribution

[IFFactoryDistribution](#) - Interface for creating distribution strategies.

Diagrama de herencia de factory\_interface.IFFactoryDistribution

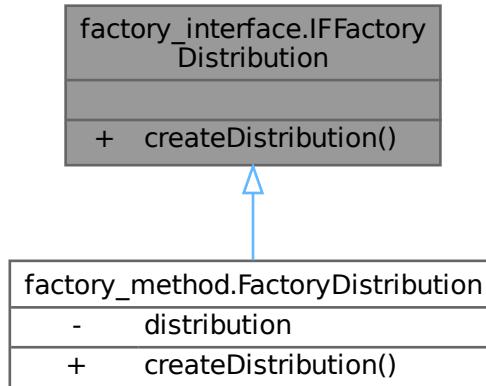
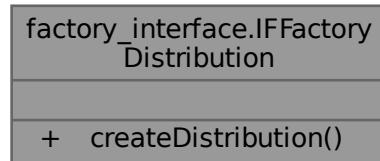


Diagrama de colaboración de factory\_interface.IFFactoryDistribution:



### Métodos públicos

- `Distribution createDistribution (DistributionType typedistribution)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

### 8.46.1 Descripción detallada

[IFFactoryDistribution](#) - Interface for creating distribution strategies.

Definición en la línea 14 del archivo [IFFactoryDistribution.java](#).

## 8.46.2 Documentación de funciones miembro

### 8.46.2.1 createDistribution()

```
Distribution factory_interface.IFFactoryDistribution.createDistribution (
    DistributionType typedistribution) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException
```

Implementado en [factory\\_method.FactoryDistribution](#).

La documentación de esta interfaz está generada del siguiente archivo:

- [IFFactoryDistribution.java](#)

## 8.47 Referencia de la interface [factory\\_interface.IFFactoryFatherSelection](#)

[IFFactoryFatherSelection](#) - Interface for creating father selection strategies.

Diagrama de herencia de `factory_interface.IFFactoryFatherSelection`

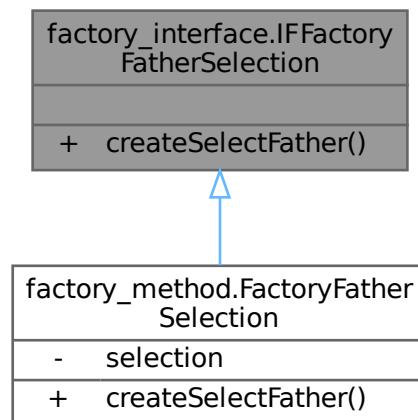
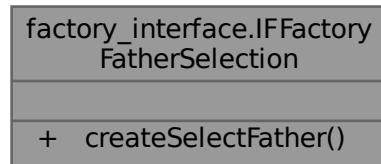


Diagrama de colaboración de `factory_interface.IFFactoryFatherSelection`:



## Métodos públicos

- `FatherSelection createSelectFather (SelectionType selectionType) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

### 8.47.1 Descripción detallada

[IFFactoryFatherSelection](#) - Interface for creating father selection strategies.

Definición en la línea 14 del archivo [IFFactoryFatherSelection.java](#).

### 8.47.2 Documentación de funciones miembro

#### 8.47.2.1 createSelectFather()

```
FatherSelection factory_interface.IFFactoryFatherSelection.createSelectFather (
    SelectionType selectionType) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException
```

Implementado en [factory\\_method.FactoryFatherSelection](#).

La documentación de esta interfaz está generada del siguiente archivo:

- [IFFactoryFatherSelection.java](#)

## 8.48 Referencia de la interface factory\_interface.IFFactoryGenerator

[IFFactoryGenerator](#) - Interface for creating generator instances.

Diagrama de herencia de `factory_interface.IFFactoryGenerator`

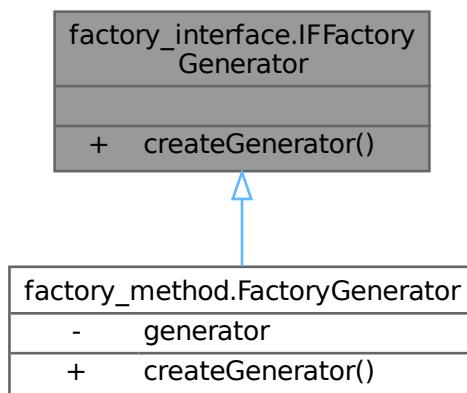
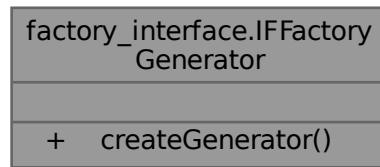


Diagrama de colaboración de factory\_interface.IFFactoryGenerator:



## Métodos públicos

- `Generator createGenerator (GeneratorType Generatortype) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

### 8.48.1 Descripción detallada

[IFFactoryGenerator](#) - Interface for creating generator instances.

Definición en la línea 11 del archivo [IFFactoryGenerator.java](#).

### 8.48.2 Documentación de funciones miembro

#### 8.48.2.1 createGenerator()

```
Generator factory_interface.IFFactoryGenerator.createGenerator (
    GeneratorType Generatortype) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException
```

Implementado en [factory\\_method.FactoryGenerator](#).

La documentación de esta interface está generada del siguiente archivo:

- [IFFactoryGenerator.java](#)

## 8.49 Referencia de la interface factory\_interface.IFFactoryMutation

[IFFactoryMutation](#) - Interface for creating mutation strategies.

Diagrama de herencia de factory\_interface.IFFactoryMutation

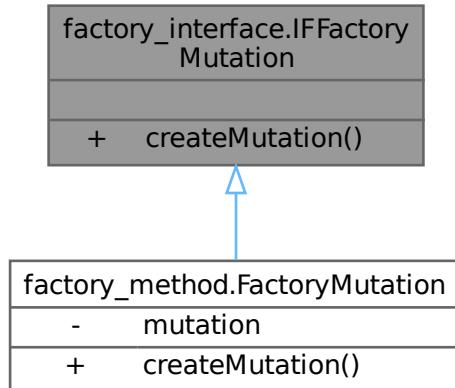
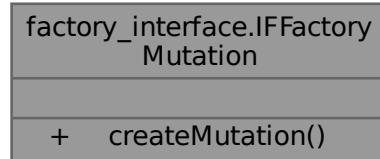


Diagrama de colaboración de factory\_interface.IFFactoryMutation:



### Métodos públicos

- **Mutation createMutation (MutationType typeMutation)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

#### 8.49.1 Descripción detallada

[IFFactoryMutation](#) - Interface for creating mutation strategies.

Definición en la línea 14 del archivo [IFFactoryMutation.java](#).

## 8.49.2 Documentación de funciones miembro

### 8.49.2.1 createMutation()

```
Mutation factory_interface.IFFactoryMutation.createMutation (
    MutationType typeMutation) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException
```

Implementado en [factory\\_method.FactoryMutation](#).

La documentación de esta interfaz está generada del siguiente archivo:

- [IFFactoryMutation.java](#)

## 8.50 Referencia de la interface factory\_interface.IFFactoryReplace

[IFFactoryReplace](#) - Interface for creating replacement strategies.

Diagrama de herencia de `factory_interface.IFFactoryReplace`

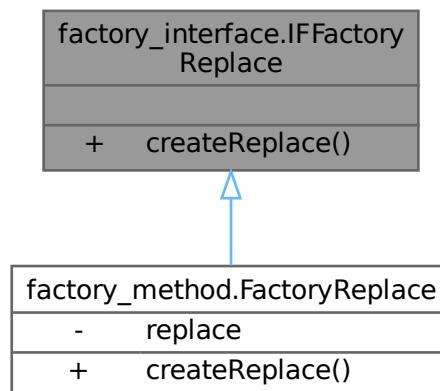
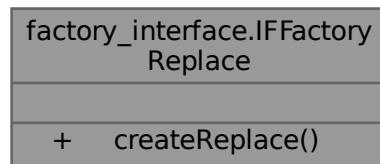


Diagrama de colaboración de `factory_interface.IFFactoryReplace`:



## Métodos públicos

- `Replace createReplace (ReplaceType typereplace) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

### 8.50.1 Descripción detallada

[IFFactoryReplace](#) - Interface for creating replacement strategies.

Definición en la línea 14 del archivo [IFFactoryReplace.java](#).

### 8.50.2 Documentación de funciones miembro

#### 8.50.2.1 createReplace()

```
Replace factory_interface.IFFactoryReplace.createReplace (
    ReplaceType typereplace) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException
```

Implementado en [factory\\_method.FactoryReplace](#).

La documentación de esta interfaz está generada del siguiente archivo:

- [IFFactoryReplace.java](#)

## 8.51 Referencia de la interface factory\_interface.IFFactorySolutionMethod

[IFFactorySolutionMethod](#) - Interface for creating solution methods.

Diagrama de herencia de `factory_interface.IFFactorySolutionMethod`

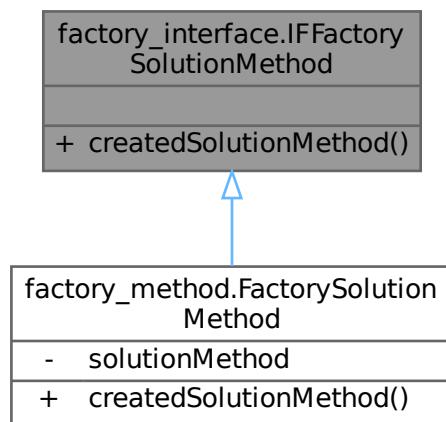
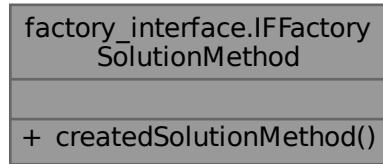


Diagrama de colaboración de factory\_interface.IFFactorySolutionMethod:



## Métodos públicos

- `SolutionMethod createdSolutionMethod (TypeSolutionMethod method) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

### 8.51.1 Descripción detallada

[IFFactorySolutionMethod](#) - Interface for creating solution methods.

Definición en la línea 11 del archivo [IFFactorySolutionMethod.java](#).

### 8.51.2 Documentación de funciones miembro

#### 8.51.2.1 `createdSolutionMethod()`

```
SolutionMethod factory_interface.IFFactorySolutionMethod.createdSolutionMethod (
    TypeSolutionMethod method) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException
```

Implementado en [factory\\_method.FactorySolutionMethod](#).

La documentación de esta interface está generada del siguiente archivo:

- [IFFactorySolutionMethod.java](#)

## 8.52 Referencia de la interface factory\_interface.IFFSampling

IFFSampling - Interface for creating sampling strategies.

Diagrama de herencia de factory\_interface.IFFSampling

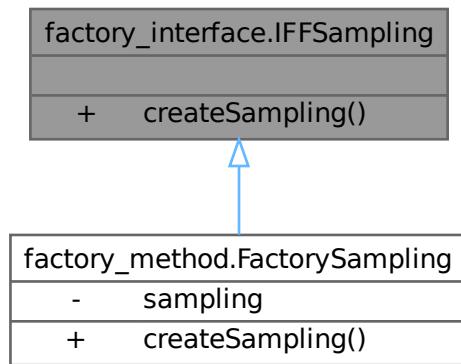


Diagrama de colaboración de factory\_interface.IFFSampling:



### Métodos públicos

- `Sampling createSampling (SamplingType typesampling) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

#### 8.52.1 Descripción detallada

IFFSampling - Interface for creating sampling strategies.

Definición en la línea 14 del archivo `IFFSampling.java`.

## 8.52.2 Documentación de funciones miembro

### 8.52.2.1 createSampling()

```
Sampling factory_interface.IFFSampling.createSampling (
    SamplingType typesampling) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException
```

Implementado en [factory\\_method.FactorySampling](#).

La documentación de esta interface está generada del siguiente archivo:

- [IFFSampling.java](#)

## 8.53 Referencia de la clase metaheuristics.generators.InstanceDE

[InstanceDE](#) - class that implements the Runnable interface to create an instance of the Distribution Estimation Algorithm generator.

Diagrama de herencia de [metaheuristics.generators.InstanceDE](#)

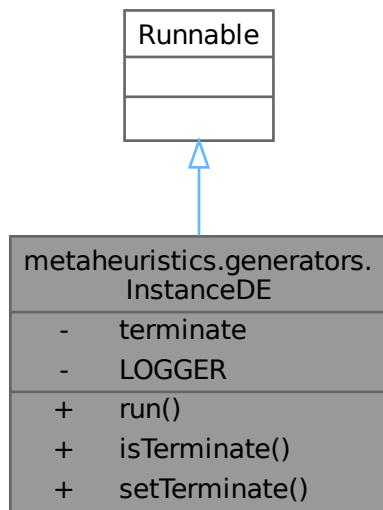
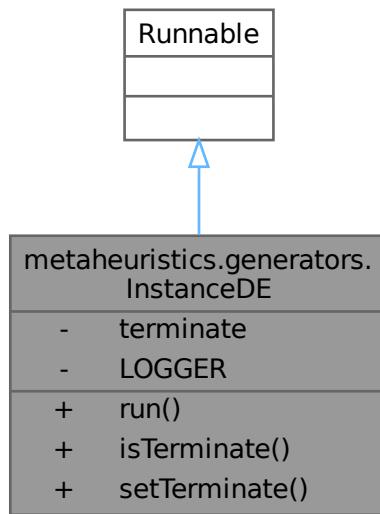


Diagrama de colaboración de metaheuristics.generators.InstanceDE:



### Métodos públicos

- void `run ()`  
*run - create an instance of the Distribution Estimation Algorithm generator.*
- boolean `isTerminate ()`  
*isTerminate - check if the instance is terminated.*
- void `setTerminate (boolean terminate)`  
*setTerminate - set the termination status of the instance.*

### Atributos privados

- boolean `terminate = false`

### Atributos estáticos privados

- static final Logger `LOGGER = Logger.getLogger(InstanceDE.class.getName())`

## 8.53.1 Descripción detallada

`InstanceDE` - class that implements the `Runnable` interface to create an instance of the Distribution Estimation Algorithm generator.

Definición en la línea 12 del archivo [InstanceDE.java](#).

## 8.53.2 Documentación de funciones miembro

### 8.53.2.1 isTerminate()

```
boolean metaheuristics.generators.InstanceDE.isTerminate () [inline]
```

isTerminate - check if the instance is terminated.

Devuelve

true if terminated, false otherwise

Definición en la línea 47 del archivo [InstanceDE.java](#).

```
00047         {
00048             return terminate;
00049         }
```

Hace referencia a [terminate](#).

### 8.53.2.2 run()

```
void metaheuristics.generators.InstanceDE.run () [inline]
```

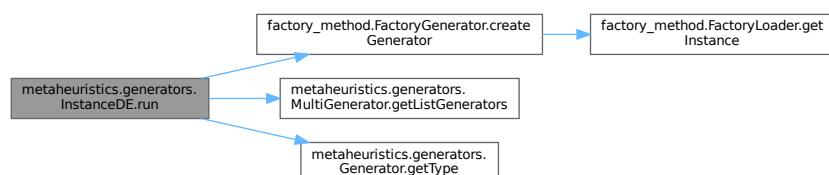
run - create an instance of the Distribution Estimation Algorithm generator.

Definición en la línea 21 del archivo [InstanceDE.java](#).

```
00021         {
00022             FactoryGenerator ifFactoryGenerator = new FactoryGenerator();
00023             Generator generatorDE = null;
00024             try {
00025                 generatorDE =
00026                     ifFactoryGenerator.createGenerator(GeneratorType.DistributionEstimationAlgorithm);
00027             } catch (Exception e) {
00028                 // Log exception instead of printing stack trace so debug info is available
00029                 // but the code is safe for production.
00030                 LOGGER.log(Level.SEVERE, "Failed to create DistributionEstimationAlgorithm generator", e);
00031             }
00032             boolean find = false;
00033             int i = 0;
00034             while (find == false) {
00035                 if(MultiGenerator.getListGenerators()[i].getType().equals(GeneratorType.DistributionEstimationAlgorithm)) {
00036                     MultiGenerator.getListGenerators()[i] = generatorDE;
00037                     find = true;
00038                 } else i++;
00039             }
00040             terminate = true;
00041         }
```

Hace referencia a [factory\\_method.FactoryGenerator.createGenerator\(\)](#), [metaheuristics.generators.GeneratorType.DistributionEstimationAlgorithm](#), [metaheuristics.generators.MultiGenerator.getListGenerators\(\)](#), [metaheuristics.generators.Generator.getType\(\)](#), [LOGGER](#) y [terminate](#).

Gráfico de llamadas de esta función:



### 8.53.2.3 setTerminate()

```
void metaheuristics.generators.InstanceDE.setTerminate (
    boolean terminate) [inline]
```

setTerminate - set the termination status of the instance.

#### Parámetros

<i>terminate</i>	true to terminate, false to continue
------------------	--------------------------------------

Definición en la línea 55 del archivo [InstanceDE.java](#).

```
00055
00056     this.terminate = terminate;
00057 }
```

Hace referencia a [terminate](#).

### 8.53.3 Documentación de datos miembro

#### 8.53.3.1 LOGGER

```
final Logger metaheuristics.generators.InstanceDE.LOGGER = Logger.getLogger(InstanceDE.←
    class.getName()) [static], [private]
```

Definición en la línea 14 del archivo [InstanceDE.java](#).

Referenciado por [run\(\)](#).

#### 8.53.3.2 terminate

```
boolean metaheuristics.generators.InstanceDE.terminate = false [private]
```

Definición en la línea 16 del archivo [InstanceDE.java](#).

Referenciado por [isTerminate\(\)](#), [run\(\)](#) y [setTerminate\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [InstanceDE.java](#)

## 8.54 Referencia de la clase metaheuristics.generators.InstanceEE

[InstanceEE](#) - class that implements the Runnable interface to create an instance of.

Diagrama de herencia de `metaheuristics.generators.InstanceEE`

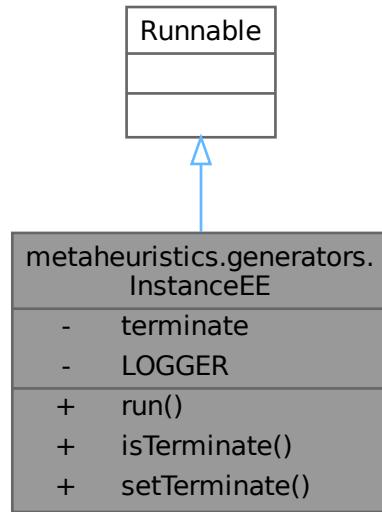
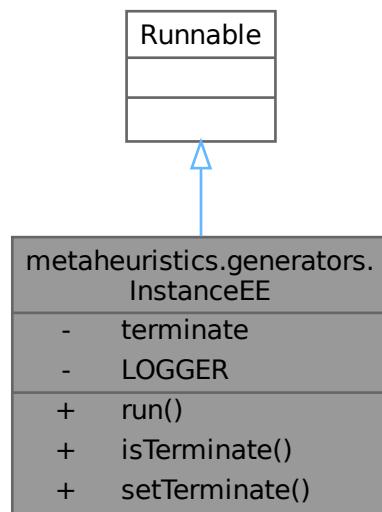


Diagrama de colaboración de `metaheuristics.generators.InstanceEE`:



## Métodos públicos

- void **run** ()  
*run - create an instance of the Evolution Strategies generator.*
- boolean **isTerminate** ()  
*isTerminate - check if the instance is terminated.*
- void **setTerminate** (boolean **terminate**)  
*setTerminate - set the termination status of the instance.*

## Atributos privados

- boolean **terminate** = false

## Atributos estáticos privados

- static final Logger **LOGGER** = Logger.getLogger(InstanceEE.class.getName())

## 8.54.1 Descripción detallada

[InstanceEE](#) - class that implements the Runnable interface to create an instance of.

Definición en la línea 10 del archivo [InstanceEE.java](#).

## 8.54.2 Documentación de funciones miembro

### 8.54.2.1 **isTerminate()**

boolean metaheuristics.generators.InstanceEE.isTerminate () [inline]

isTerminate - check if the instance is terminated.

Devuelve

true if terminated, false otherwise

Definición en la línea 44 del archivo [InstanceEE.java](#).

```
00044
00045     return terminate;
00046 }
```

Hace referencia a [terminate](#).

### 8.54.2.2 run()

```
void metaheuristics.generators.InstanceEE.run () [inline]
```

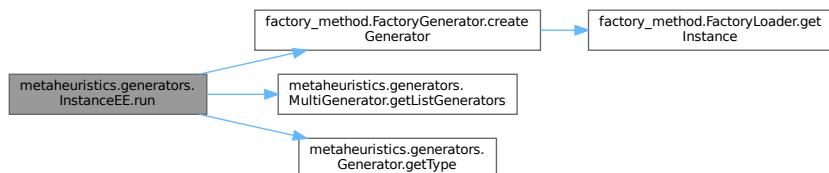
run - create an instance of the Evolution Strategies generator.

Definición en la línea 19 del archivo [InstanceEE.java](#).

```
00019         {
00020             FactoryGenerator iffFactoryGenerator = new FactoryGenerator();
00021             Generator generatorEE = null;
00022             try {
00023                 generatorEE = iffFactoryGenerator.createGenerator(GeneratorType.EvolutionStrategies);
00024             } catch (Exception e) {
00025                 // Log exception instead of printing stack trace to avoid debug output in production
00026                 LOGGER.log(Level.SEVERE, "Failed to create EvolutionStrategies generator", e);
00027             }
00028             boolean find = false;
00029             int i = 0;
00030             while (find == false) {
00031
00032                 if(MultiGenerator.getListGenerators()[i].getType().equals(GeneratorType.EvolutionStrategies)){
00033                     MultiGenerator.getListGenerators()[i] = generatorEE;
00034                     find = true;
00035                 } else i++;
00036             }
00037             terminate = true;
00038         }
```

Hace referencia a [factory\\_method.FactoryGenerator.createGenerator\(\)](#), [metaheuristics.generators.GeneratorType.EvolutionStrategies](#), [metaheuristics.generators.MultiGenerator.getListGenerators\(\)](#), [metaheuristics.generators.Generator.getType\(\)](#), [LOGGER](#) y [terminate](#).

Gráfico de llamadas de esta función:



### 8.54.2.3 setTerminate()

```
void metaheuristics.generators.InstanceEE.setTerminate (
    boolean terminate) [inline]
```

setTerminate - set the termination status of the instance.

#### Parámetros

<code>terminate</code>	true to terminate, false to continue
------------------------	--------------------------------------

Definición en la línea 52 del archivo [InstanceEE.java](#).

```
00052
00053     this.terminate = terminate;
00054 }
```

Hace referencia a [terminate](#).

### 8.54.3 Documentación de datos miembro

#### 8.54.3.1 LOGGER

```
final Logger metaheuristics.generators.InstanceEE.LOGGER = Logger.getLogger(InstanceEE.class.getName()) [static], [private]
```

Definición en la línea 12 del archivo [InstanceEE.java](#).

Referenciado por [run\(\)](#).

#### 8.54.3.2 terminate

```
boolean metaheuristics.generators.InstanceEE.terminate = false [private]
```

Definición en la línea 14 del archivo [InstanceEE.java](#).

Referenciado por [isTerminate\(\)](#), [run\(\)](#) y [setTerminate\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [InstanceEE.java](#)

## 8.55 Referencia de la clase metaheuristics.generators.InstanceGA

[InstanceGA](#) - class that implements the Runnable interface to create an instance of the Genetic Algorithm generator.

Diagrama de herencia de metaheuristics.generators.InstanceGA

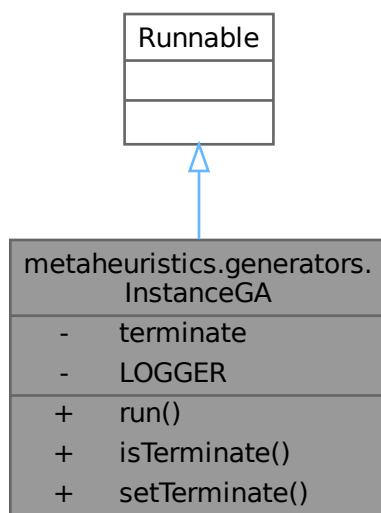
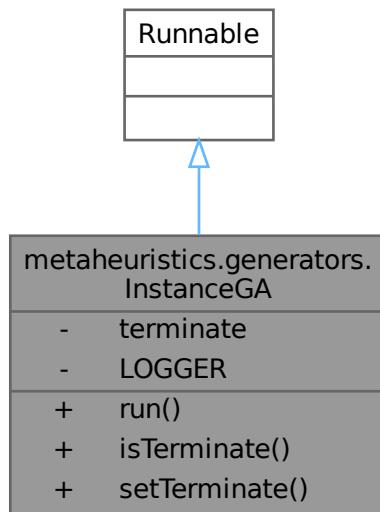


Diagrama de colaboración de metaheuristics.generators.InstanceGA:



## Métodos públicos

- void `run ()`  
*run - create an instance of the Genetic Algorithm generator.*
- boolean `isTerminate ()`  
*isTerminate - check if the instance is terminated.*
- void `setTerminate (boolean terminate)`  
*setTerminate - determine the termination status of the instance.*

## Atributos privados

- boolean `terminate = false`

## Atributos estáticos privados

- static final Logger `LOGGER = Logger.getLogger(InstanceGA.class.getName())`

### 8.55.1 Descripción detallada

`InstanceGA` - class that implements the `Runnable` interface to create an instance of the Genetic Algorithm generator.

Definición en la línea 12 del archivo `InstanceGA.java`.

## 8.55.2 Documentación de funciones miembro

### 8.55.2.1 isTerminate()

```
boolean metaheuristics.generators.InstanceGA.isTerminate () [inline]
```

isTerminate - check if the instance is terminated.

Devuelve

true if terminated, false otherwise

Definición en la línea 45 del archivo [InstanceGA.java](#).

```
00045         {
00046             return terminate;
00047         }
```

Hace referencia a [terminate](#).

### 8.55.2.2 run()

```
void metaheuristics.generators.InstanceGA.run () [inline]
```

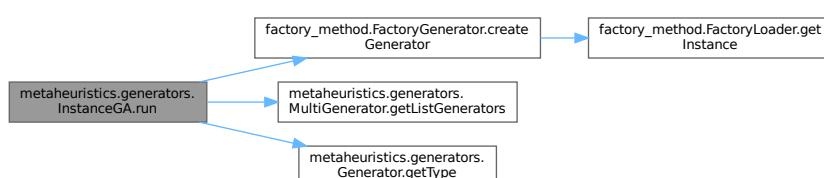
run - create an instance of the Genetic Algorithm generator.

Definición en la línea 21 del archivo [InstanceGA.java](#).

```
00021         {
00022             FactoryGenerator iffFactoryGenerator = new FactoryGenerator();
00023             Generator generatorGA = null;
00024             try {
00025                 generatorGA = iffFactoryGenerator.createGenerator(GeneratorType.GeneticAlgorithm);
00026             } catch (Exception e) {
00027                 LOGGER.log(Level.SEVERE, "Failed to create GeneticAlgorithm generator", e);
00028             }
00029             boolean find = false;
00030             int i = 0;
00031             while (find == false) {
00032
00033                 if(MultiGenerator.getListGenerators()[i].getType().equals(GeneratorType.GeneticAlgorithm)) {
00034                     MultiGenerator.getListGenerators()[i] = generatorGA;
00035                     find = true;
00036                 }
00037             }
00038             terminate = true;
00039         }
```

Hace referencia a [factory\\_method.FactoryGenerator.createGenerator\(\)](#), [metaheuristics.generators.GeneratorType.GeneticAlgorithm](#), [metaheuristics.generators.MultiGenerator.getListGenerators\(\)](#), [metaheuristics.generators.Generator.getType\(\)](#), [LOGGER](#) y [terminate](#).

Gráfico de llamadas de esta función:



## Parámetros

<i>terminate</i>	true to terminate, false to continue
------------------	--------------------------------------

Definición en la línea 53 del archivo [InstanceGA.java](#).

```
00053          {  
00054      this.terminate = terminate;  
00055 }
```

Hace referencia a [terminate](#).

### 8.55.3 Documentación de datos miembro

#### 8.55.3.1 LOGGER

```
final Logger metaheuristics.generators.InstanceGA.LOGGER = Logger.getLogger(InstanceGA.  
class.getName()) [static], [private]
```

Definición en la línea 14 del archivo [InstanceGA.java](#).

Referenciado por [run\(\)](#).

#### 8.55.3.2 terminate

```
boolean metaheuristics.generators.InstanceGA.terminate = false [private]
```

Definición en la línea 16 del archivo [InstanceGA.java](#).

Referenciado por [isTerminate\(\)](#), [run\(\)](#) y [setTerminate\(\)](#).

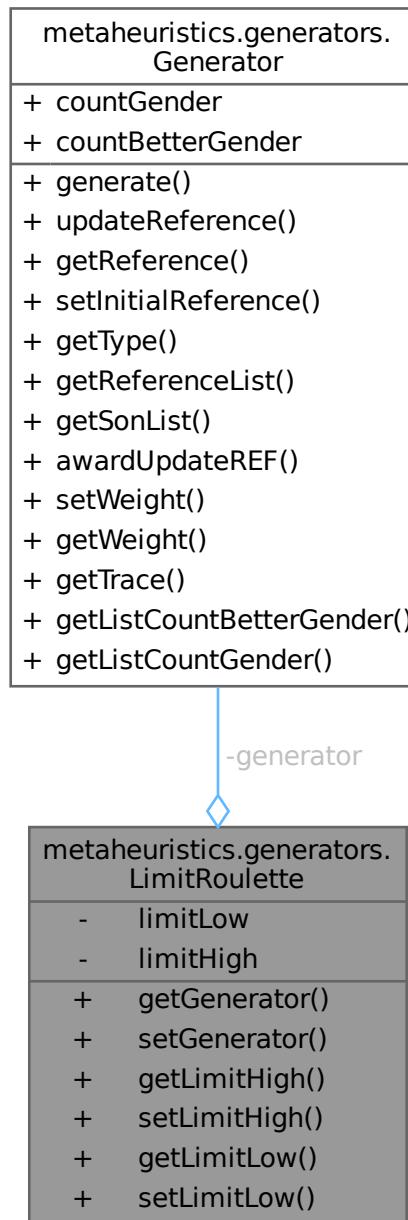
La documentación de esta clase está generada del siguiente archivo:

- [InstanceGA.java](#)

## 8.56 Referencia de la clase metaheuristics.generators.LimitRoulette

[LimitRoulette](#) - class that implements the Limit Roulette structure.

Diagrama de colaboración de metaheuristics.generators.LimitRoulette:



## Métodos públicos

- **Generator getGenerator ()**  
`getGenerator` - get the generator associated with the Limit Roulette.
- **void setGenerator (Generator generator)**  
`setGenerator` - set the generator associated with the Limit Roulette.
- **float getLimitHigh ()**  
`getLimitHigh` - get the upper limit of the Limit Roulette.

- void [setLimitHigh](#) (float [limitHigh](#))
 

*setLimitHigh - set the upper limit of the Limit Roulette.*
- float [getLimitLow](#) ()
 

*getLimitLow - get the lower limit of the Limit Roulette.*
- void [setLimitLow](#) (float [limitLow](#))
 

*setLimitLow - set the lower limit of the Limit Roulette.*

### Atributos privados

- float [limitLow](#)
- float [limitHigh](#)
- [Generator generator](#)

## 8.56.1 Descripción detallada

[LimitRoulette](#) - class that implements the Limit Roulette structure.

Definición en la línea 6 del archivo [LimitRoulette.java](#).

## 8.56.2 Documentación de funciones miembro

### 8.56.2.1 [getGenerator\(\)](#)

```
Generator metaheuristics.generators.LimitRoulette.getGenerator () [inline]
```

getGenerator - get the generator associated with the Limit Roulette.

Devuelve

the generator

Definición en la línea 16 del archivo [LimitRoulette.java](#).

```
00016
00017     return generator;
00018 }
```

Hace referencia a [generator](#).

### 8.56.2.2 [getLimitHigh\(\)](#)

```
float metaheuristics.generators.LimitRoulette.getLimitHigh () [inline]
```

getLimitHigh - get the upper limit of the Limit Roulette.

Devuelve

the upper limit

Definición en la línea 30 del archivo [LimitRoulette.java](#).

```
00030
00031     return limitHigh;
00032 }
```

Hace referencia a [limitHigh](#).

### 8.56.2.3 getLimitLow()

```
float metaheuristics.generators.LimitRoulette.getLimitLow () [inline]
```

getLimitLow - get the lower limit of the Limit Roulette.

Devuelve

the lower limit

Definición en la línea 44 del archivo [LimitRoulette.java](#).

```
00044 {  
00045     return limitLow;  
00046 }
```

Hace referencia a [limitLow](#).

### 8.56.2.4 setGenerator()

```
void metaheuristics.generators.LimitRoulette.setGenerator (  
    Generator generator) [inline]
```

setGenerator - set the generator associated with the Limit Roulette.

#### Parámetros

<i>generator</i>	the generator to set
------------------	----------------------

Definición en la línea 23 del archivo [LimitRoulette.java](#).

```
00023 {  
00024     this.generator = generator;  
00025 }
```

Hace referencia a [generator](#).

Referenciado por [metaheuristics.generators.MultiGenerator.roulette\(\)](#).

Gráfico de llamadas a esta función:



### 8.56.2.5 setLimitHigh()

```
void metaheuristics.generators.LimitRoulette.setLimitHigh (  
    float limitHigh) [inline]
```

setLimitHigh - set the upper limit of the Limit Roulette.

## Parámetros

<i>limitHigh</i>	the upper limit to set
------------------	------------------------

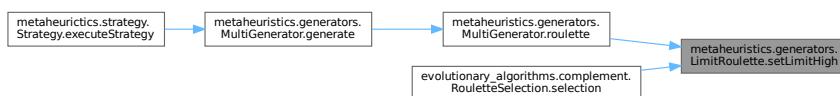
Definición en la línea 37 del archivo [LimitRoulette.java](#).

```
00037          {
00038      this.limitHigh = limitHigh;
00039 }
```

Hace referencia a [limitHigh](#).

Referenciado por [metaheuristics.generators.MultiGenerator.roulette\(\)](#) y [evolutionary\\_algorithms.complement.RouletteSelection.select\(\)](#)

Gráfico de llamadas a esta función:



## 8.56.2.6 setLimitLow()

```
void metaheuristics.generators.LimitRoulette.setLimitLow (
    float limitLow) [inline]
```

**setLimitLow** - set the lower limit of the Limit Roulette.

## Parámetros

<i>limitLow</i>	the lower limit to set
-----------------	------------------------

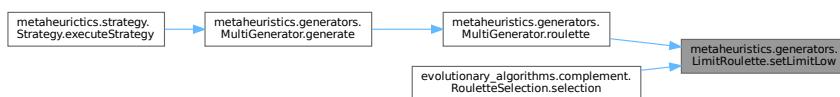
Definición en la línea 51 del archivo [LimitRoulette.java](#).

```
00051          {
00052      this.limitLow = limitLow;
00053 }
```

Hace referencia a [limitLow](#).

Referenciado por [metaheuristics.generators.MultiGenerator.roulette\(\)](#) y [evolutionary\\_algorithms.complement.RouletteSelection.select\(\)](#)

Gráfico de llamadas a esta función:



### 8.56.3 Documentación de datos miembro

#### 8.56.3.1 generator

```
Generator metaheuristics.generators.LimitRoulette.generator [private]
```

Definición en la línea 10 del archivo [LimitRoulette.java](#).

Referenciado por [getGenerator\(\)](#) y [setGenerator\(\)](#).

#### 8.56.3.2 limitHigh

```
float metaheuristics.generators.LimitRoulette.limitHigh [private]
```

Definición en la línea 9 del archivo [LimitRoulette.java](#).

Referenciado por [getLimitHigh\(\)](#) y [setLimitHigh\(\)](#).

#### 8.56.3.3 limitLow

```
float metaheuristics.generators.LimitRoulette.limitLow [private]
```

Definición en la línea 8 del archivo [LimitRoulette.java](#).

Referenciado por [getLimitLow\(\)](#) y [setLimitLow\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [LimitRoulette.java](#)

## 8.57 Referencia de la clase metaheuristics.generators.LimitThreshold

[LimitThreshold](#) - class that implements the Limit Threshold metaheuristic.

## Diagrama de herencia de metaheuristics.generators.LimitThreshold

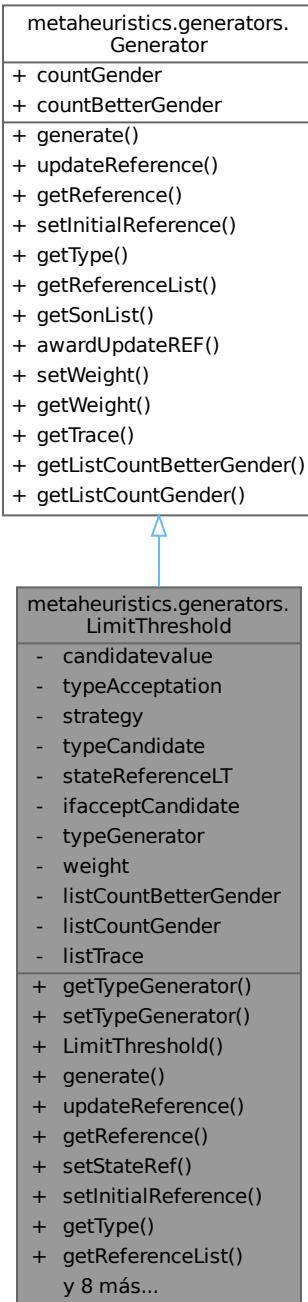
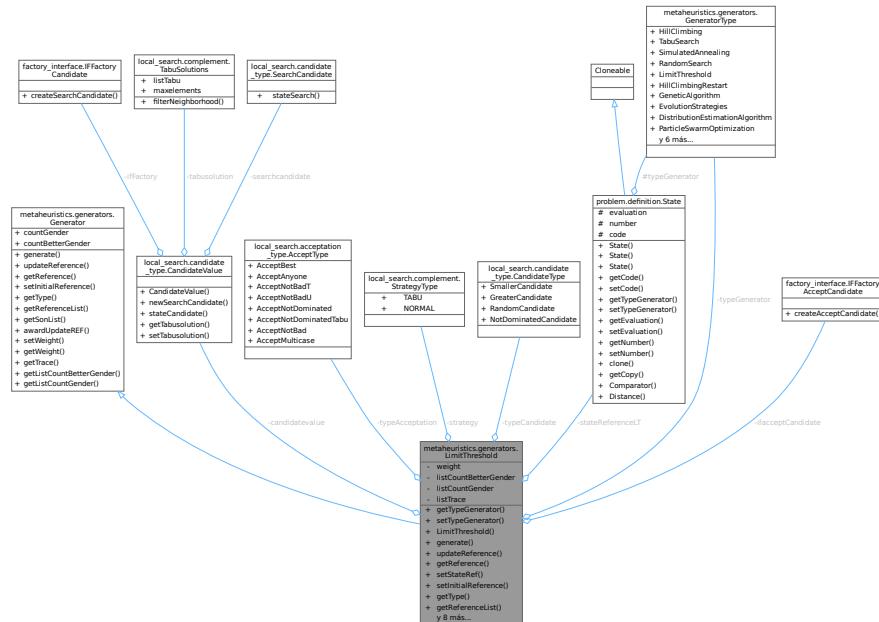


Diagrama de colaboración de metaheuristics.generators.LimitThreshold:



## Métodos públicos

- **GeneratorType getTypeGenerator ()**  
*getTypeGenerator - get the type of the generator.*
- **void setTypeGenerator (GeneratorType typeGenerator)**  
*setTypeGenerator - set the type of the generator.*
- **LimitThreshold ()**  
*LimitThreshold - class that implements the Limit Threshold metaheuristic.*
- **State generate (Integer operatornumber)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`  
*generate - generate a new state based on the operator number.*
- **void updateReference (State stateCandidate, Integer countIterationsCurrent)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`  
*updateReference - update the reference state based on the candidate state.*
- **State getReference ()**  
*getReference - get the reference state.*
- **void setStateRef (State stateRef)**  
*setStateRef - set the reference state.*
- **void setInitialReference (State statelInitialRef)**  
*setInitialReference - set the initial reference state.*
- **GeneratorType getType ()**  
*getType - get the type of the generator.*
- **List< State > getReferenceList ()**  
*getReferenceList - get the list of reference states.*
- **List< State > getSonList ()**  
*getSonList - get the list of son states.*

- void [setTypeCandidate \(CandidateType typeCandidate\)](#)  
*setTypeCandidate - set the type of candidate.*
- boolean [awardUpdateREF \(State stateCandidate\)](#)  
*awardUpdateREF - award the update of the reference state.*
- float [getWeight \(\)](#)  
*getWeight - get the weight of the generator.*
- void [setWeight \(float weight\)](#)  
*setWeight - set the weight of the generator.*
- int[] [getListCountBetterGender \(\)](#)  
*getListCountBetterGender - get the list of count better gender.*
- int[] [getListCountGender \(\)](#)  
*getListCountGender - get the list of count gender.*
- float[] [getTrace \(\)](#)  
*getTrace - get the trace of the generator.*

### Atributos privados

- CandidateValue [candidatevalue](#)
- AcceptType [typeAcceptation](#)
- StrategyType [strategy](#)
- CandidateType [typeCandidate](#)
- State [stateReferenceLT](#)
- IFFactoryAcceptCandidate [ifacceptCandidate](#)
- GeneratorType [typeGenerator](#)
- float [weight](#)
- int[] [listCountBetterGender = new int\[10\]](#)
- int[] [listCountGender = new int\[10\]](#)
- float[] [listTrace = new float\[1200000\]](#)

### Otros miembros heredados

#### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- int [countGender](#)
- int [countBetterGender](#)

### 8.57.1 Descripción detallada

[LimitThreshold](#) - class that implements the Limit Threshold metaheuristic.

Definición en la línea 28 del archivo [LimitThreshold.java](#).

## 8.57.2 Documentación de constructores y destructores

### 8.57.2.1 LimitThreshold()

```
metaheuristics.generators.LimitThreshold.LimitThreshold () [inline]
```

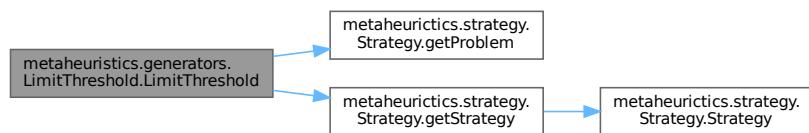
[LimitThreshold](#) - class that implements the Limit Threshold metaheuristic.

Definición en la línea 64 del archivo [LimitThreshold.java](#).

```
00064         super();
00065         this.typeAcceptation = AcceptType.AcceptNotBadU;
00066         this.strategy = StrategyType.NORMAL;
00067
00068
00069
00070         Problem problem = Strategy.getStrategy().getProblem();
00071
00072         if(problem.getTypeProblem().equals(ProblemType.Maximizar)) {
00073             this.typeCandidate = CandidateType.GreaterCandidate;
00074         }
00075         else{
00076             this.typeCandidate = CandidateType.SmallerCandidate;
00077         }
00078
00079         this.candidatevalue = new CandidateValue();
00080         this.typeGenerator = GeneratorType.LimitThreshold;
00081         this.weight = (float) 50.0;
00082         listTrace[0] = weight;
00083         listCountBetterGender[0] = 0;
00084         listCountGender[0] = 0;
00085
00086     }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptType.AcceptNotBadU](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [local\\_search.candidate\\_type.CandidateType.GreaterCandidate](#), [metaheuristics.generators.GeneratorType.LimitThreshold](#), [listCountBetterGender](#), [listCountGender](#), [listTrace](#), [problem.definition.ProblemType.Maximizar](#), [local\\_search.complement.StrategyType.NORMAL](#), [local\\_search.candidate\\_type](#) y [weight](#).

Gráfico de llamadas de esta función:



## 8.57.3 Documentación de funciones miembro

### 8.57.3.1 awardUpdateREF()

```
boolean metaheuristics.generators.LimitThreshold.awardUpdateREF (
    State stateCandidate) [inline]
```

[awardUpdateREF](#) - award the update of the reference state.

## Parámetros

<i>stateCandidate</i>	
-----------------------	--

### Devuelve

return true if the reference state was updated, false otherwise

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 184 del archivo [LimitThreshold.java](#).

```
00184
00185     // TODO Auto-generated method stub
00186     return false;
00187 }
```

## 8.57.3.2 generate()

```
State metaheuristics.generators.LimitThreshold.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

generate - generate a new state based on the operator number.

## Parámetros

<i>operatornumber</i>	the operator number to use for generating the new state
-----------------------	---

### Devuelve

the generated state

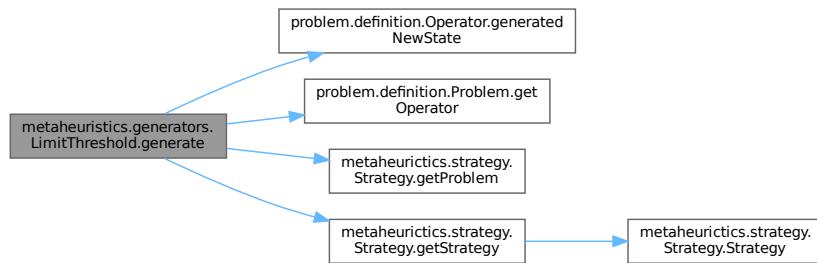
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 93 del archivo [LimitThreshold.java](#).

```
00093
00094     {
00095         List<State> neighborhood =
00096             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceLT, operatornumber);
00097         State statecandidate = candidatevalue.stateCandidate(stateReferenceLT, typeCandidate,
00098             strategy, operatornumber, neighborhood);
00099         return statecandidate;
00100     }
```

Hace referencia a [candidatevalue](#), [problem.definition.Operator.generatedNewState\(\)](#), [problem.definition.Problem.getOperator\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [stateReferenceLT](#), [strategy](#) y [typeCandidate](#).

Gráfico de llamadas de esta función:



### 8.57.3.3 getListCountBetterGender()

```
int[] metaheuristics.generators.LimitThreshold.getListCountBetterGender () [inline]
```

getListCountBetterGender - get the list of count better gender.

**Devuelve**

the list of count better gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 214 del archivo [LimitThreshold.java](#).

```
00214
00215      // TODO Auto-generated method stub
00216      return (this.listCountBetterGender == null) ? new int[0] :
00217          Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
```

### 8.57.3.4 getListCountGender()

```
int[] metaheuristics.generators.LimitThreshold.getListCountGender () [inline]
```

getListCountGender - get the list of count gender.

**Devuelve**

the list of count gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 224 del archivo [LimitThreshold.java](#).

```
00224
00225      // TODO Auto-generated method stub
00226      return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00227          this.listCountGender.length);
```

### 8.57.3.5 `getReference()`

```
State metaheuristics.generators.LimitThreshold.getReference () [inline]
```

getReference - get the reference state.

Devuelve

the reference state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 120 del archivo [LimitThreshold.java](#).

```
00120           {
00121             return stateReferenceLT;
00122 }
```

Hace referencia a [stateReferenceLT](#).

### 8.57.3.6 `getReferenceList()`

```
List< State > metaheuristics.generators.LimitThreshold.getReferenceList () [inline]
```

getReferenceList - get the list of reference states.

Devuelve

the list of reference states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 156 del archivo [LimitThreshold.java](#).

```
00156           {
00157             return null;
00158 }
```

### 8.57.3.7 `getSonList()`

```
List< State > metaheuristics.generators.LimitThreshold.getSonList () [inline]
```

getSonList - get the list of son states.

Devuelve

the list of son states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 165 del archivo [LimitThreshold.java](#).

```
00165           {
00166             // TODO Auto-generated method stub
00167             return null;
00168 }
```

### 8.57.3.8 getTrace()

```
float[ ] metaheuristics.generators.LimitThreshold.getTrace () [inline]
```

getTrace - get the trace of the generator.

Devuelve

the trace of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 234 del archivo [LimitThreshold.java](#).

```
00234          {
00235      // TODO Auto-generated method stub
00236      return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00237      this.listTrace.length);
}
```

### 8.57.3.9 getType()

```
GeneratorType metaheuristics.generators.LimitThreshold.getType () [inline]
```

getType - get the type of the generator.

Devuelve

the type of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 147 del archivo [LimitThreshold.java](#).

```
00147          {
00148      return this.typeGenerator;
00149 }
```

### 8.57.3.10 getTypeGenerator()

```
GeneratorType metaheuristics.generators.LimitThreshold.getTypeGenerator () [inline]
```

getTypeGenerator - get the type of the generator.

Devuelve

the type of the generator

Definición en la línea 49 del archivo [LimitThreshold.java](#).

```
00049          {
00050      return typeGenerator;
00051 }
```

Hace referencia a [typeGenerator](#).

### 8.57.3.11 getWeight()

```
float metaheuristics.generators.LimitThreshold.getWeight () [inline]
```

getWeight - get the weight of the generator.

#### Devuelve

the weight of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 193 del archivo [LimitThreshold.java](#).

```
00193             {
00194         // TODO Auto-generated method stub
00195         return this.weight;
00196     }
```

### 8.57.3.12 setInitialReference()

```
void metaheuristics.generators.LimitThreshold.setInitialReference (
    State stateInitialRef) [inline]
```

setInitialReference - set the initial reference state.

#### Parámetros

<i>stateInitialRef</i>	
------------------------	--

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 137 del archivo [LimitThreshold.java](#).

```
00137             {
00138         this.stateReferenceLT = stateInitialRef;
00139     }
```

### 8.57.3.13 setStateRef()

```
void metaheuristics.generators.LimitThreshold.setStateRef (
    State stateRef) [inline]
```

setStateRef - set the reference state.

#### Parámetros

<i>stateRef</i>	the reference state to set
-----------------	----------------------------

Definición en la línea 128 del archivo [LimitThreshold.java](#).

```
00128             {
00129         this.stateReferenceLT = stateRef;
00130     }
```

### 8.57.3.14 setTypeCandidate()

```
void metaheuristics.generators.LimitThreshold.setTypeCandidate (
```

**Parámetros**

<i>typeCandidate</i>	the type of candidate to set
----------------------	------------------------------

Definición en la línea 174 del archivo [LimitThreshold.java](#).

```
00174
00175     this.typeCandidate = typeCandidate;
00176 }
```

Hace referencia a [typeCandidate](#).

**8.57.3.15 setTypeGenerator()**

```
void metaheuristics.generators.LimitThreshold.setTypeGenerator (
    GeneratorType typeGenerator) [inline]
```

`setTypeGenerator` - set the type of the generator.

**Parámetros**

<i>typeGenerator</i>	the type of the generator to set
----------------------	----------------------------------

Definición en la línea 57 del archivo [LimitThreshold.java](#).

```
00057
00058     this.typeGenerator = typeGenerator;
00059 }
```

Hace referencia a [typeGenerator](#).

**8.57.3.16 setWeight()**

```
void metaheuristics.generators.LimitThreshold.setWeight (
    float weight) [inline]
```

`setWeight` - set the weight of the generator.

**Parámetros**

<i>weight</i>	
---------------	--

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 203 del archivo [LimitThreshold.java](#).

```
00203
00204     // TODO Auto-generated method stub
00205     this.weight = weight;
00206 }
```

Hace referencia a [weight](#).

---

**8.57.3.17 updateReference()**

Generado por Doxygen

```
void metaheuristics.generators.LimitThreshold.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
```

## Parámetros

<i>stateCandidate</i>	
<i>countIterationsCurrent</i>	

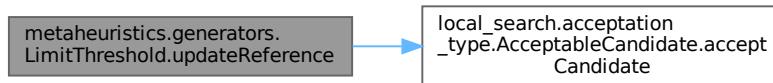
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 105 del archivo [LimitThreshold.java](#).

```
00105
00106     ifacceptCandidate = new FactoryAcceptCandidate();
00107     AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00108     Boolean accept = candidate.acceptCandidate(stateReferenceLT , stateCandidate);
00109     if(accept.equals(true)){
00110         stateReferenceLT = stateCandidate;
00111     }
00112 }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptableCandidate.acceptCandidate\(\)](#), [ifacceptCandidate](#), [stateReferenceLT](#) y [typeAcceptation](#).

Gráfico de llamadas de esta función:



## 8.57.4 Documentación de datos miembro

### 8.57.4.1 candidatevalue

`CandidateValue` metaheuristics.generators.LimitThreshold.candidatevalue [private]

Definición en la línea 30 del archivo [LimitThreshold.java](#).

Referenciado por [generate\(\)](#).

### 8.57.4.2 ifacceptCandidate

`IFFactoryAcceptCandidate` metaheuristics.generators.LimitThreshold.ifacceptCandidate [private]

Definición en la línea 35 del archivo [LimitThreshold.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.57.4.3 listCountBetterGender

```
int [] metaheuristics.generators.LimitThreshold.listCountBetterGender = new int[10] [private]
```

Definición en la línea 41 del archivo [LimitThreshold.java](#).

Referenciado por [LimitThreshold\(\)](#).

#### 8.57.4.4 listCountGender

```
int [] metaheuristics.generators.LimitThreshold.listCountGender = new int[10] [private]
```

Definición en la línea 42 del archivo [LimitThreshold.java](#).

Referenciado por [LimitThreshold\(\)](#).

#### 8.57.4.5 listTrace

```
float [] metaheuristics.generators.LimitThreshold.listTrace = new float[1200000] [private]
```

Definición en la línea 43 del archivo [LimitThreshold.java](#).

Referenciado por [LimitThreshold\(\)](#).

#### 8.57.4.6 stateReferenceLT

```
State metaheuristics.generators.LimitThreshold.stateReferenceLT [private]
```

Definición en la línea 34 del archivo [LimitThreshold.java](#).

Referenciado por [generate\(\)](#), [getReference\(\)](#) y [updateReference\(\)](#).

#### 8.57.4.7 strategy

```
StrategyType metaheuristics.generators.LimitThreshold.strategy [private]
```

Definición en la línea 32 del archivo [LimitThreshold.java](#).

Referenciado por [generate\(\)](#).

#### 8.57.4.8 typeAcceptation

```
AcceptType metaheuristics.generators.LimitThreshold.typeAcceptation [private]
```

Definición en la línea 31 del archivo [LimitThreshold.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.57.4.9 typeCandidate

`CandidateType` metaheuristics.generators.LimitThreshold.typeCandidate [private]

Definición en la línea 33 del archivo [LimitThreshold.java](#).

Referenciado por [generate\(\)](#) y [setTypeCandidate\(\)](#).

#### 8.57.4.10 typeGenerator

`GeneratorType` metaheuristics.generators.LimitThreshold.typeGenerator [private]

Definición en la línea 36 del archivo [LimitThreshold.java](#).

Referenciado por [getTypeGenerator\(\)](#) y [setTypeGenerator\(\)](#).

#### 8.57.4.11 weight

`float` metaheuristics.generators.LimitThreshold.weight [private]

Definición en la línea 37 del archivo [LimitThreshold.java](#).

Referenciado por [LimitThreshold\(\)](#) y [setWeight\(\)](#).

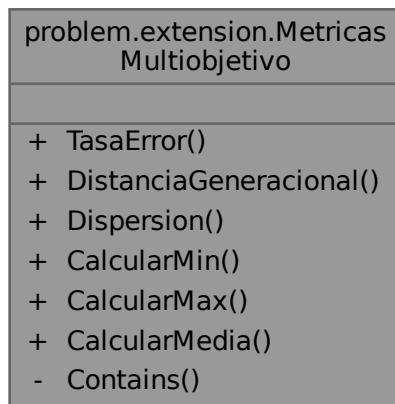
La documentación de esta clase está generada del siguiente archivo:

- [LimitThreshold.java](#)

### 8.58 Referencia de la clase problem.extension.MetricasMultiobjetivo

[MetricasMultiobjetivo](#).

Diagrama de colaboración de `problem.extension.MetricasMultiobjetivo`:



## Métodos públicos

- double **TasaError** (List< State > solutionsFPcurrent, List< State > solutionsFPtrue) throws BiffException, IOException  
*Calcular el porcentaje de soluciones del frente actual que no pertenecen al frente de Pareto verdadero.*
- double **DistanciaGeneracional** (List< State > solutionsFPcurrent, List< State > solutionsFPtrue) throws BiffException, IOException  
*DistanciaGeneracional.*
- double **Dispersion** (ArrayList< State > solutions) throws BiffException, IOException  
*Dispersion.*
- double **CalcularMin** (ArrayList< Double > allMetrics)  
*Calcular el valor mínimo de una lista de métricas.*
- double **CalcularMax** (ArrayList< Double > allMetrics)  
*Calcular el valor máximo de una lista de métricas.*
- double **CalcularMedia** (ArrayList< Double > allMetrics)  
*Calcular la media aritmética de una lista de métricas.*

## Métodos privados

- boolean **Contains** (State solA, List< State > solutions)  
*Comprueba si una solución (por su vector de evaluación) está en una lista.*

### 8.58.1 Descripción detallada

#### MetricasMultiobjetivo.

Conjunto de métricas para evaluar frentes de Pareto en problemas multiobjetivo (error, distancia generacional, dispersión, etc.).

Definición en la línea 18 del archivo [MetricasMultiobjetivo.java](#).

### 8.58.2 Documentación de funciones miembro

#### 8.58.2.1 CalcularMax()

```
double problem.extension.MetricasMultiobjetivo.CalcularMax (
    ArrayList< Double > allMetrics) [inline]
```

Calcular el valor máximo de una lista de métricas.

#### Parámetros

<i>allMetrics</i>	lista de métricas
-------------------	-------------------

#### Devuelve

valor máximo

Definición en la línea 169 del archivo [MetricasMultiobjetivo.java](#).

```
00169
00170     double max = 0;
00171     for (Iterator<Double> iter = allMetrics.iterator(); iter.hasNext();) {
00172         double element = (Double) iter.next();
00173         if(element > max){
00174             max = element;
00175         }
00176     }
00177     return max;
00178 }
```

### 8.58.2.2 CalcularMedia()

```
double problem.extension.MetricasMultiobjetivo.CalcularMedia (
    ArrayList< Double > allMetrics) [inline]
```

Calcular la media aritmética de una lista de métricas.

#### Parámetros

<code>allMetrics</code>	lista de métricas
-------------------------	-------------------

#### Devuelve

media

Definición en la línea 185 del archivo [MetricasMultiobjetivo.java](#).

```
00185         double sum = 0;
00186         for (Iterator<Double> iter = allMetrics.iterator(); iter.hasNext();) {
00187             double element = (Double) iter.next();
00188             sum = sum + element;
00189         }
00190         double media = sum/allMetrics.size();
00191         return media;
00192     }
00193 }
```

### 8.58.2.3 CalcularMin()

```
double problem.extension.MetricasMultiobjetivo.CalcularMin (
    ArrayList< Double > allMetrics) [inline]
```

Calcular el valor mínimo de una lista de métricas.

#### Parámetros

<code>allMetrics</code>	lista de métricas
-------------------------	-------------------

#### Devuelve

valor mínimo

Definición en la línea 152 del archivo [MetricasMultiobjetivo.java](#).

```
00152         double min = 1000;
00153         for (Iterator<Double> iter = allMetrics.iterator(); iter.hasNext();) {
00154             double element = (Double) iter.next();
00155             if(element < min){
00156                 min = element;
00157             }
00158         }
00159         return min;
00160     }
00161 }
```

### 8.58.2.4 Contains()

---

```
boolean problem.extension.MetricasMultiobjetivo.Contains (
    State sola,
    List< State > solutions) [inline], [private]
```

## Parámetros

<i>solA</i>	solución a comprobar
<i>solutions</i>	lista de soluciones

## Devuelve

true si se encuentra una evaluación idéntica

Definición en la línea 135 del archivo [MetricasMultiobjetivo.java](#).

```
00135
00136     int i = 0;
00137     boolean result = false;
00138     while(i<solutions.size()&& result==false){
00139         if(solutions.get(i).getEvaluation().equals(solA.getEvaluation()))
00140             result=true;
00141         else
00142             i++;
00143     }
00144     return result;
00145 }
```

Hace referencia a [problem.definition.State.getEvaluation\(\)](#).

Referenciado por [TasaError\(\)](#).

Gráfico de llamadas de esta función:



Gráfico de llamadas a esta función:



## 8.58.2.5 Dispersion()

```
double problem.extension.MetricasMultiobjetivo.Dispersion (
    ArrayList< State > solutions) throws BiffException, IOException [inline]
```

Dispersion.

## Parámetros

<i>solutions</i>	lista de estados que conforman el frente
------------------	--

## Devuelve

medida de dispersión

Definición en la línea 88 del archivo [MetricasMultiobjetivo.java](#).

```

00088
00089      //Soluciones obtenidas con la ejecución del algoritmo X
00090      LinkedList<Float> distancias = new LinkedList<Float>();
00091      float distancia = 0;
00092      float min = 1000;
00093      for (Iterator<State> iter = solutions.iterator(); iter.hasNext();) {
00094          State solutionVO = (State) iter.next();
00095          min = 1000;
00096          for (Iterator<State> iterator = solutions.iterator(); iterator.hasNext();) {
00097              State solVO = (State) iterator.next();
00098              for (int i = 0; i < solutionVO.getEvaluation().size(); i++) {
00099                  if(!solutionVO.getEvaluation().equals(solVO.getEvaluation())){
00100                      distancia += (solutionVO.getEvaluation().get(i)-
solVO.getEvaluation().get(i));
00101                  }
00102                  if(distancia < min){
00103                      min = distancia;
00104                  }
00105              }
00106              distancias.add(Float.valueOf(min));
00107          }
00108          //Calculando las media de las distancias
00109          float sum = 0;
00110          for (Iterator<Float> iter = distancias.iterator(); iter.hasNext();) {
00111              Float dist = (Float) iter.next();
00112              sum += dist;
00113          }
00114          float media = sum/distancias.size();
00115          float sumDistancias = 0;
00116          for (Iterator<Float> iter = distancias.iterator(); iter.hasNext();) {
00117              Float dist = (Float) iter.next();
00118              sumDistancias += Math.pow((media - dist),2);
00119          }
00120          //Calculando la dispersion
00121          double dispersion = 0;
00122          if(solutions.size() > 1){
00123              dispersion = Math.sqrt((1.0/(solutions.size()-1))*sumDistancias);
00124          }
00125          //System.out.println(dispersion);
00126          return dispersion;
00127      }
  
```

Hace referencia a [problem.definition.State.getEvaluation\(\)](#).

Gráfico de llamadas de esta función:



### 8.58.2.6 DistanciaGeneracional()

```
double problem.extension.MetricasMultiobjetivo.DistanciaGeneracional (
    List< State > solutionsFPcurrent,
    List< State > solutionsFPtrue) throws BiffException, IOException [inline]
```

DistanciaGeneracional.

Calcula la distancia media entre los miembros del frente actual y sus puntos más cercanos en el frente verdadero.

#### Parámetros

<i>solutionsFPcurrent</i>	frente actual
<i>solutionsFPtrue</i>	frente verdadero

Devuelve

distancia agregada normalizada

Definición en la línea 53 del archivo [MetricasMultiobjetivo.java](#).

```
00053
{
00054     float min = 1000;
00055     float distancia = 0;
00056     float distanciaGeneracional = 0;
00057     for (int i = 0; i < solutionsFPcurrent.size(); i++) {
00058         State solutionVO = solutionsFPcurrent.get(i);
00059         //Calculando la distancia euclideana entre solutionVO y el miembro más cercano del frente
        de Pareto verdadero
00060         min = 1000;
00061         for (int j = 0; j < solutionsFPtrue.size(); j++) {
00062             for (int j2 = 0; j2 < solutionVO.getEvaluation().size(); j2++) {
00063                 State solutionFPV = solutionsFPtrue.get(j);
00064                 //porq elevar la distancia al cuadrado
00065                 distancia += (solutionVO.getEvaluation().get(j2) -
solutionFPV.getEvaluation().get(j2)) *
                    (solutionVO.getEvaluation().get(j2) -
solutionFPV.getEvaluation().get(j2)); //ceros si el argumento es el cero, 1.0 si el argumento es mayor
que el cero, -1.0 si el argumento está menos del cero
00066             }
00067             if(distancia < min){
00068                 min = distancia;
00069             }
00070         }
00071     }
00072     distanciaGeneracional += min;
00073 }
00074 double total = Math.sqrt(distanciaGeneracional)/solutionsFPcurrent.size();
00075 //System.out.println(total);
00076 return total;
00077 }
```

Hace referencia a [problem.definition.State.getEvaluation\(\)](#).

Gráfico de llamadas de esta función:



### 8.58.2.7 TasaError()

```
double problem.extension.MetricasMultiobjetivo.TasaError (
    List< State > solutionsFPcurrent,
    List< State > solutionsFPtrue) throws BiffException, IOException [inline]
```

Calcular el porcentaje de soluciones del frente actual que no pertenecen al frente de Pareto verdadero.

#### Parámetros

<code>solutionsFPcurrent</code>	frente de Pareto obtenido por el algoritmo
<code>solutionsFPtrue</code>	frente de Pareto de referencia (verdadero)

#### Devuelve

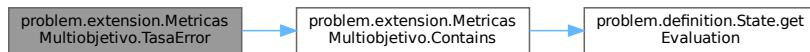
proporción de soluciones incorrectas (0..1)

Definición en la línea 29 del archivo [MetricasMultiobjetivo.java](#).

```
00029
00030     {
00031         float tasaError = 0;
00032         for (int i = 0; i < solutionsFPcurrent.size() ; i++) { // frente de pareto actual
00033             State solutionVO = solutionsFPcurrent.get(i);
00034             if(!Contains(solutionVO, solutionsFPtrue)){ // no esta en el frente de pareto verdadero
00035                 tasaError++;
00036             }
00037         double total = tasaError/solutionsFPcurrent.size();
00038         //System.out.println(solutionsFP.size() + "/" + solutions.size() + "/" + total);
00039         return total;
00040     }
```

Hace referencia a [Contains\(\)](#).

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [MetricasMultiobjetivo.java](#)

## 8.59 Referencia de la clase [metaheuristics.generators.MultiCaseSimulatedAnnealing](#)

[MultiCaseSimulatedAnnealing](#) - class that implements the Multi-Case Simulated Annealing metaheuristic.

Diagrama de herencia de metaheuristics.generators.MultiCaseSimulatedAnnealing

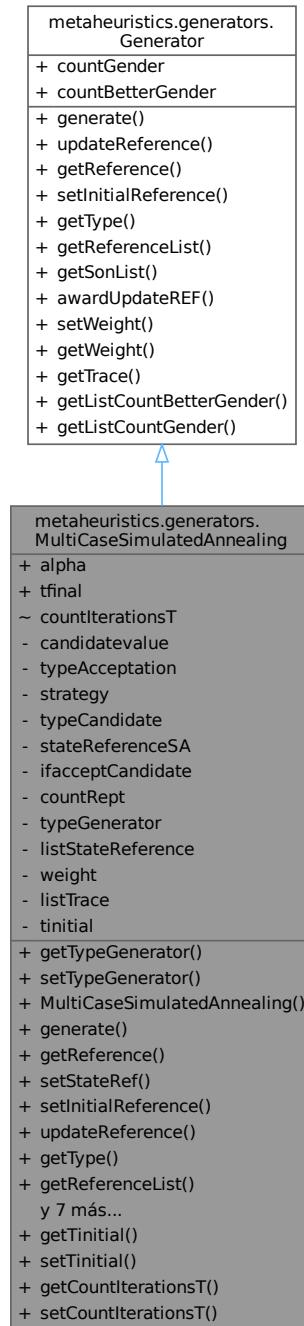
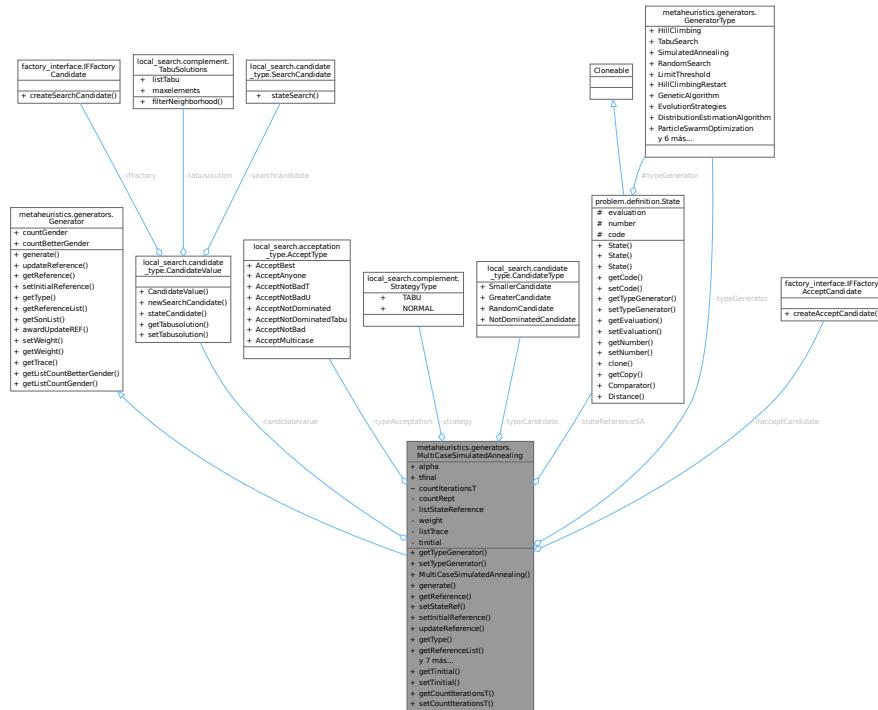


Diagrama de colaboración de metaheuristics.generators.MultiCaseSimulatedAnnealing:



## Métodos públicos

- **GeneratorType getTypeGenerator ()**  
*getTypeGenerator - get the type of the generator.*
  - void **setTypeGenerator (GeneratorType typeGenerator)**  
*setTypeGenerator - set the type of the generator.*
  - **MultiCaseSimulatedAnnealing ()**  
*MultiCaseSimulatedAnnealing - class that implements the Multi-Case Simulated Annealing metaheuristic.*
  - **State generate (Integer operatornumber)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`  
*generate - generate a new state based on the operator number.*
  - **State getReference ()**  
*getReference - get the reference state.*
  - void **setStateRef (State stateRef)**  
*setStateRef - set the reference state.*
  - void **setInitialReference (State stateInitialRef)**  
*setInitialReference - set the initial reference state.*
  - void **updateReference (State stateCandidate, Integer countIterationsCurrent)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`  
*updateReference - update the reference state based on the candidate state.*
  - **GeneratorType getType ()**  
*getType - get the type of the generator.*
  - List< **State > getReferenceList ()**  
*getReferenceList - get the list of reference states.*

- List< [State](#) > [getSonList](#) ()
 

*getSonList - get the list of son states.*
- boolean [awardUpdateREF](#) ([State](#) stateCandidate)
 

*awardUpdateREF - award the update of the reference state.*
- float [getWeight](#) ()
 

*getWeight - get the weight of the generator.*
- void [setWeight](#) (float [weight](#))
 

*setWeight - set the weight of the generator.*
- int[] [getListCountBetterGender](#) ()
 

*getListCountBetterGender - get the list of counts for better gender.*
- int[] [getListCountGender](#) ()
 

*getListCountGender - get the list of counts for gender.*
- float[] [getTrace](#) ()
 

*getTrace - get the trace of the generator.*

### Métodos públicos estáticos

- static Double [getTinitial](#) ()
- static void [setTinitial](#) (Double t)
 

*setTinitial - set the initial temperature.*
- static int [getCountIterationsT](#) ()
 

*getCountIterationsT - get the count of iterations for temperature update.*
- static void [setCountIterationsT](#) (int c)
 

*setCountIterationsT - set the count of iterations for temperature update.*

### Atributos públicos estáticos

- static final double [alpha](#) = 0.93
- static final Double [tfinal](#) = 41.66

### Atributos privados

- [CandidateValue](#) candidatevalue
- [AcceptType](#) typeAcceptation
- [StrategyType](#) strategy
- [CandidateType](#) typeCandidate
- [State](#) stateReferenceSA
- [IFFactoryAcceptCandidate](#) ifacceptCandidate
- int [countRept](#)
- [GeneratorType](#) typeGenerator
- List< [State](#) > [listStateReference](#) = new ArrayList<[State](#)>()
- float [weight](#)
- List< Float > [listTrace](#) = new ArrayList<Float>()

### Atributos estáticos privados

- static Double [tinitial](#) = 250.0

## Otros miembros heredados

### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- int [countGender](#)
- int [countBetterGender](#)

## 8.59.1 Descripción detallada

[MultiCaseSimulatedAnnealing](#) - class that implements the Multi-Case Simulated Annealing metaheuristic.

Definición en la línea 22 del archivo [MultiCaseSimulatedAnnealing.java](#).

## 8.59.2 Documentación de constructores y destructores

### 8.59.2.1 [MultiCaseSimulatedAnnealing\(\)](#)

```
metaheuristics.generators.MultiCaseSimulatedAnnealing.MultiCaseSimulatedAnnealing () [inline]
```

[MultiCaseSimulatedAnnealing](#) - class that implements the Multi-Case Simulated Annealing metaheuristic.

Definición en la línea 73 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00073     super();
00074     this.typeAcceptation = AcceptType.AcceptMulticase;
00075     this.strategy = StrategyType.NORMAL;
00076     this.typeCandidate = CandidateType.RandomCandidate;
00077     this.candidatevalue = new CandidateValue();
00078     this.typeGenerator = GeneratorType.MultiCaseSimulatedAnnealing;
00079     this.weight = 50;
00080     listTrace.add(weight);
00081 }
00082 }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptType.AcceptMulticase](#), [listTrace](#), [metaheuristics.generators.GeneratorType.M](#), [local\\_search.complement.StrategyType.NORMAL](#), [local\\_search.candidate\\_type.CandidateType.RandomCandidate](#) y [weight](#).

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas a esta función:



## 8.59.3 Documentación de funciones miembro

### 8.59.3.1 [awardUpdateREF\(\)](#)

```
boolean metaheuristics.generators.MultiCaseSimulatedAnnealing.awardUpdateREF (
    State stateCandidate) [inline]
```

## Parámetros

<i>stateCandidate</i>	
-----------------------	--

### Devuelve

return true if the reference state was updated, false otherwise

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 197 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00197
00198     // TODO Auto-generated method stub
00199     return false;
00200 }
```

### 8.59.3.2 generate()

```
State metaheuristics.generators.MultiCaseSimulatedAnnealing.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

generate - generate a new state based on the operator number.

## Parámetros

<i>operatornumber</i>	the operator number to use for generating the new state
-----------------------	---

### Devuelve

the generated state

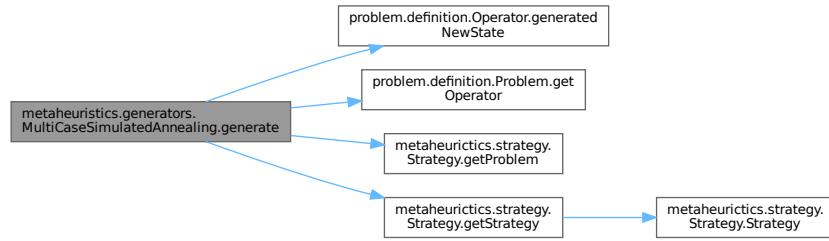
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 90 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00090
00091     {
00092         Problem problem = Strategy.getStrategy().getProblem();
00093         List<State> neighborhood = problem.getOperator().generatedNewState(stateReferenceSA,
00094             operatornumber);
00093         State statecandidate = candidatevalue.stateCandidate(stateReferenceSA, typeCandidate,
00094             strategy, operatornumber, neighborhood);
00094         return statecandidate;
00095     }
```

Hace referencia a [candidatevalue](#), [problem.definition.Operator.generatedNewState\(\)](#), [problem.definition.Problem.getOperator\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [stateReferenceSA](#), [strategy](#) y [typeCandidate](#).

Gráfico de llamadas de esta función:



### 8.59.3.3 getCountIterationsT()

```
int metaheuristics.generators.MultiCaseSimulatedAnnealing.getCountIterationsT () [inline],  
[static]
```

getCountIterationsT - get the count of iterations for temperature update.

Devuelve

the count of iterations for temperature update

Definición en la línea 150 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00150
00151      return countIterationsT;
00152 }
```

### 8.59.3.4 getListCountBetterGender()

```
int[] metaheuristics.generators.MultiCaseSimulatedAnnealing.getListCountBetterGender () [inline]
```

getListCountBetterGender - get the list of counts for better gender.

Devuelve

the list of counts for better gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 228 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00228
00229      // TODO Auto-generated method stub
00230      return new int[0];
00231 }
```

### 8.59.3.5 getListCountGender()

```
int[] metaheuristics.generators.MultiCaseSimulatedAnnealing.getListCountGender () [inline]
getListCountGender - get the list of counts for gender.
```

Devuelve

the list of counts for gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 238 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00238      {
00239          // TODO Auto-generated method stub
00240          return new int[0];
00241      }
```

### 8.59.3.6 getReference()

```
State metaheuristics.generators.MultiCaseSimulatedAnnealing.getReference () [inline]
getReference - get the reference state.
```

Devuelve

the reference state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 102 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00102      {
00103          return stateReferenceSA;
00104      }
```

Hace referencia a [stateReferenceSA](#).

### 8.59.3.7 getReferenceList()

```
List< State > metaheuristics.generators.MultiCaseSimulatedAnnealing.getReferenceList () [inline]
getReferenceList - get the list of reference states.
```

Devuelve

the list of reference states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 176 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00176      {
00177          listStateReference.add(stateReferenceSA.clone ());
00178          return listStateReference;
00179      }
```

Hace referencia a [listStateReference](#) y [stateReferenceSA](#).

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.59.3.8 getSonList()

```
List< State > metaheuristics.generators.MultiCaseSimulatedAnnealing.getSonList () [inline]
getSonList - get the list of son states.
```

Devuelve

the list of son states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 186 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00186                               {
00187           // TODO Auto-generated method stub
00188           return null;
00189       }
```

### 8.59.3.9 getTinitial()

```
Double metaheuristics.generators.MultiCaseSimulatedAnnealing.getTinitial () [inline], [static]
```

Definición en la línea 59 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00059                               {
00060           return tinitial;
00061       }
```

Hace referencia a [tinitial](#).

Referenciado por [local\\_search.acceptation\\_type.AcceptMulticase.acceptCandidate\(\)](#).

Gráfico de llamadas a esta función:



### 8.59.3.10 getTrace()

```
float[] metaheuristics.generators.MultiCaseSimulatedAnnealing.getTrace () [inline]
```

getTrace - get the trace of the generator.

Devuelve

the trace of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 248 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00248                               {
00249           // TODO Auto-generated method stub
00250           if \(this.listTrace == null\) return new float\[0\];
00251           float[] arr = new float\[this.listTrace.size\(\)\];
00252           for \(int i = 0; i < this.listTrace.size\(\); i++\) {
00253               Float v = this.listTrace.get\(i\);
00254               arr\[i\] = \(v == null\) ? 0f : v.floatValue\(\);
00255           }
00256           return arr;
00257       }
```

Hace referencia a [listTrace](#).

### 8.59.3.11 getType()

```
GeneratorType metaheuristics.generators.MultiCaseSimulatedAnnealing.getType () [inline]
```

getType - get the type of the generator.

Devuelve

the type of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 167 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00167             {
00168     return this.typeGenerator;
00169 }
```

### 8.59.3.12 getTypeGenerator()

```
GeneratorType metaheuristics.generators.MultiCaseSimulatedAnnealing.getTypeGenerator () [inline]
```

getTypeGenerator - get the type of the generator.

Devuelve

the type of the generator

Definición en la línea 46 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00046             {
00047     return typeGenerator;
00048 }
```

Hace referencia a [typeGenerator](#).

### 8.59.3.13 getWeight()

```
float metaheuristics.generators.MultiCaseSimulatedAnnealing.getWeight () [inline]
```

getWeight - get the weight of the generator.

Devuelve

the weight of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 208 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00208             {
00209     // TODO Auto-generated method stub
00210     return 0;
00211 }
```

---

### 8.59.3.14 setCountIterationsT()

Generado por Doxygen

```
void metaheuristics.generators.MultiCaseSimulatedAnnealing.setCountIterationsT (
    int c) [inline], [static]
```

## Parámetros

<code>c</code>	the count of iterations to set
----------------	--------------------------------

Definición en la línea 158 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00158             {
00159         countIterationsT = c;
00160     }
```

### 8.59.3.15 setInitialReference()

```
void metaheuristics.generators.MultiCaseSimulatedAnnealing.setInitialReference (
    State stateInitialRef) [inline]
```

`setInitialReference` - set the initial reference state.

## Parámetros

<code>stateInitialRef</code>	the initial reference state to set
------------------------------	------------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 119 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00119             {
00120         this.stateReferenceSA = stateInitialRef;
00121     }
```

### 8.59.3.16 setStateRef()

```
void metaheuristics.generators.MultiCaseSimulatedAnnealing.setStateRef (
    State stateRef) [inline]
```

`setStateRef` - set the reference state.

## Parámetros

<code>stateRef</code>	the reference state to set
-----------------------	----------------------------

Definición en la línea 110 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00110             {
00111         this.stateReferenceSA = stateRef;
00112     }
```

### 8.59.3.17 setTinitial()

```
void metaheuristics.generators.MultiCaseSimulatedAnnealing.setTinitial (
    Double t) [inline], [static]
```

`setTinitial` - set the initial temperature.

## Parámetros

<i>t</i>	the initial temperature to set
----------	--------------------------------

Definición en la línea 66 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00066          {
00067      tinitial = t;
00068 }
```

Hace referencia a [tinitial](#).

### 8.59.3.18 setTypeGenerator()

```
void metaheuristics.generators.MultiCaseSimulatedAnnealing.setTypeGenerator (
    GeneratorType typeGenerator) [inline]
```

setTypeGenerator - set the type of the generator.

## Parámetros

<i>typeGenerator</i>	the type of the generator to set
----------------------	----------------------------------

Definición en la línea 54 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00054          {
00055      this.typeGenerator = typeGenerator;
00056 }
```

Hace referencia a [typeGenerator](#).

### 8.59.3.19 setWeight()

```
void metaheuristics.generators.MultiCaseSimulatedAnnealing.setWeight (
    float weight) [inline]
```

setWeight - set the weight of the generator.

## Parámetros

<i>weight</i>	the weight to set
---------------	-------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 218 del archivo [MultiCaseSimulatedAnnealing.java](#).

```
00218          {
00219      // TODO Auto-generated method stub
00220
00221 }
```

Hace referencia a [weight](#).

### 8.59.3.20 updateReference()

Generado por Doxygen

```
void metaheuristics.generators.MultiCaseSimulatedAnnealing.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
```

## Parámetros

<i>stateCandidate</i>	
<i>countIterationsCurrent</i>	

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 129 del archivo [MultiCaseSimulatedAnnealing.java](#).

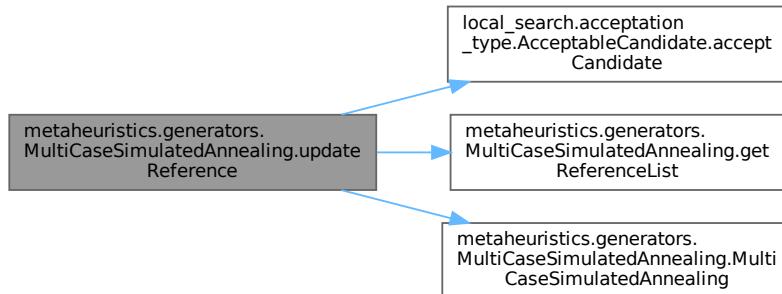
```

00129
00130     countRept = MultiCaseSimulatedAnnealing.getCountIterationsT();
00131     ifacceptCandidate = new FactoryAcceptCandidate();
00132     AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00133     Boolean accept = candidate.acceptCandidate(stateReferenceSA, stateCandidate);
00134     if(accept.equals(true))
00135         stateReferenceSA = stateCandidate.clone();
00136     if(countIterationsCurrent.equals(MultiCaseSimulatedAnnealing.getCountIterationsT())){
00137         // update via accessors to avoid writing static fields inside an instance method
00138         MultiCaseSimulatedAnnealing.setTinitial(MultiCaseSimulatedAnnealing.getTinitial() *
00139             alpha);
00140         //Variante Fast MOSA
00141         System.out.println("La T:" + MultiCaseSimulatedAnnealing.getTinitial());
00142         MultiCaseSimulatedAnnealing.setCountIterationsT(MultiCaseSimulatedAnnealing.getCountIterationsT() +
00143             countRept);
00144         System.out.println("La Cant es: " + MultiCaseSimulatedAnnealing.getCountIterationsT());
00145         getReferenceList();
00146     }

```

Hace referencia a [local\\_search.acceptation\\_type.AcceptableCandidate.acceptCandidate\(\)](#), [alpha](#), [countRept](#), [getReferenceList\(\)](#), [ifacceptCandidate](#), [MultiCaseSimulatedAnnealing\(\)](#), [stateReferenceSA](#) y [typeAcceptation](#).

Gráfico de llamadas de esta función:



## 8.59.4 Documentación de datos miembro

### 8.59.4.1 alpha

```
final double metaheuristics.generators.MultiCaseSimulatedAnnealing.alpha = 0.93 [static]
```

Definición en la línea 31 del archivo [MultiCaseSimulatedAnnealing.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.59.4.2 candidatevalue

```
CandidateValue metaheuristics.generators.MultiCaseSimulatedAnnealing.candidatevalue [private]
```

Definición en la línea 24 del archivo [MultiCaseSimulatedAnnealing.java](#).

Referenciado por [generate\(\)](#).

#### 8.59.4.3 countRept

```
int metaheuristics.generators.MultiCaseSimulatedAnnealing.countRept [private]
```

Definición en la línea 36 del archivo [MultiCaseSimulatedAnnealing.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.59.4.4 ifacceptCandidate

```
IFFactoryAcceptCandidate metaheuristics.generators.MultiCaseSimulatedAnnealing.ifacceptCandidate [private]
```

Definición en la línea 29 del archivo [MultiCaseSimulatedAnnealing.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.59.4.5 listStateReference

```
List<State> metaheuristics.generators.MultiCaseSimulatedAnnealing.listStateReference = new ArrayList<State>() [private]
```

Definición en la línea 38 del archivo [MultiCaseSimulatedAnnealing.java](#).

Referenciado por [getReferenceList\(\)](#).

#### 8.59.4.6 listTrace

```
List<Float> metaheuristics.generators.MultiCaseSimulatedAnnealing.listTrace = new ArrayList<Float>() [private]
```

Definición en la línea 40 del archivo [MultiCaseSimulatedAnnealing.java](#).

Referenciado por [getTrace\(\)](#) y [MultiCaseSimulatedAnnealing\(\)](#).

#### 8.59.4.7 stateReferenceSA

```
State metaheuristics.generators.MultiCaseSimulatedAnnealing.stateReferenceSA [private]
```

Definición en la línea 28 del archivo [MultiCaseSimulatedAnnealing.java](#).

Referenciado por [generate\(\)](#), [getReference\(\)](#), [getReferenceList\(\)](#) y [updateReference\(\)](#).

#### 8.59.4.8 strategy

```
StrategyType metaheuristics.generators.MultiCaseSimulatedAnnealing.strategy [private]
```

Definición en la línea 26 del archivo [MultiCaseSimulatedAnnealing.java](#).

Referenciado por [generate\(\)](#).

#### 8.59.4.9 tfinal

```
final Double metaheuristics.generators.MultiCaseSimulatedAnnealing.tfinal = 41.66 [static]
```

Definición en la línea 34 del archivo [MultiCaseSimulatedAnnealing.java](#).

#### 8.59.4.10 tinitial

```
Double metaheuristics.generators.MultiCaseSimulatedAnnealing.tinitial = 250.0 [static], [private]
```

Definición en la línea 33 del archivo [MultiCaseSimulatedAnnealing.java](#).

Referenciado por [getTinitial\(\)](#) y [setTinitial\(\)](#).

#### 8.59.4.11 typeAcceptation

```
AcceptType metaheuristics.generators.MultiCaseSimulatedAnnealing.typeAcceptation [private]
```

Definición en la línea 25 del archivo [MultiCaseSimulatedAnnealing.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.59.4.12 typeCandidate

```
CandidateType metaheuristics.generators.MultiCaseSimulatedAnnealing.typeCandidate [private]
```

Definición en la línea 27 del archivo [MultiCaseSimulatedAnnealing.java](#).

Referenciado por [generate\(\)](#).

#### 8.59.4.13 typeGenerator

```
GeneratorType metaheuristics.generators.MultiCaseSimulatedAnnealing.typeGenerator [private]
```

Definición en la línea 37 del archivo [MultiCaseSimulatedAnnealing.java](#).

Referenciado por [getTypeGenerator\(\)](#) y [setTypeGenerator\(\)](#).

#### 8.59.4.14 weight

```
float metaheuristics.generators.MultiCaseSimulatedAnnealing.weight [private]
```

Definición en la línea 39 del archivo [MultiCaseSimulatedAnnealing.java](#).

Referenciado por [MultiCaseSimulatedAnnealing\(\)](#) y [setWeight\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [MultiCaseSimulatedAnnealing.java](#)

## 8.60 Referencia de la clase metaheuristics.generators.MultiGenerator

[MultiGenerator](#) - class that implements the Multi-Generator metaheuristic.

Diagrama de herencia de metaheuristics.generators.MultiGenerator

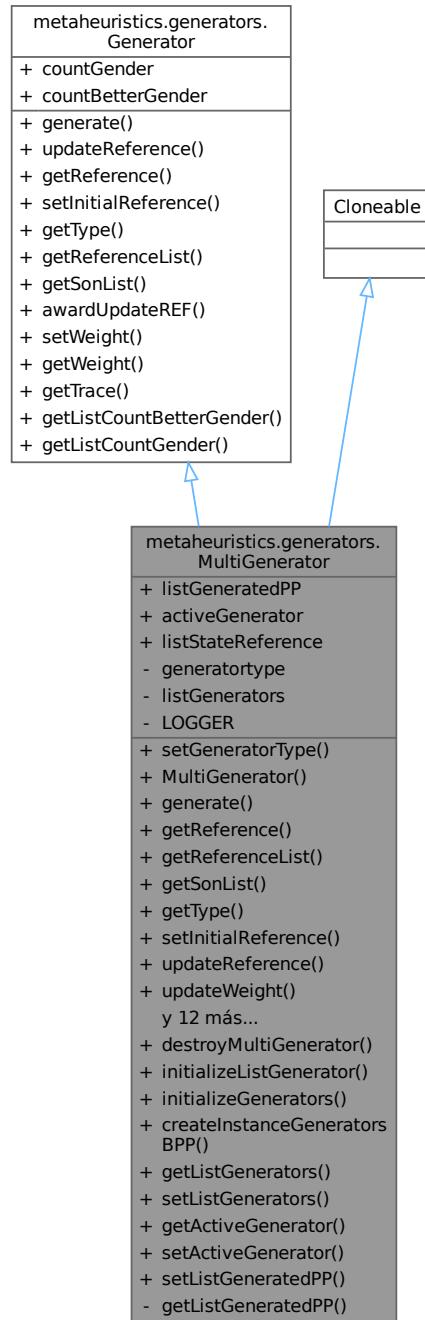
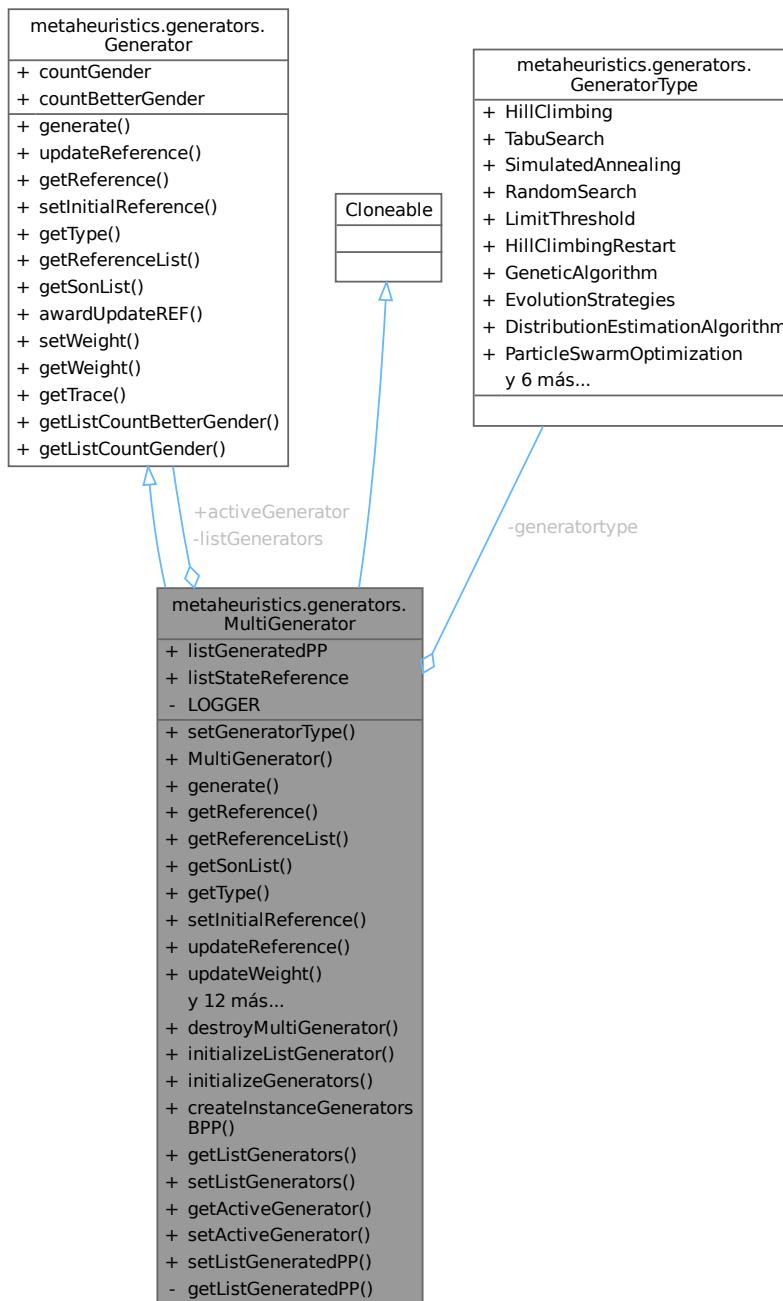


Diagrama de colaboración de metaheuristics.generators.MultiGenerator:



## Métodos públicos

- void [setGeneratorType \(GeneratorType generatorType\)](#)  
`setGeneratorType` - set the type of the generator.
- [MultiGenerator \(\)](#)  
`MultiGenerator` - default constructor.

- `State generate (Integer operatornumber)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`  
*generate - generate a new state using the `MultiGenerator`.*
- `State getReference ()`  
*getReference - get the reference state.*
- `List< State > getReferenceList ()`  
*getReferenceList - get the list of reference states.*
- `List< State > getSonList ()`  
*getSonList - get the list of son states.*
- `GeneratorType getType ()`  
*getType - get the type of the generator.*
- `void setInitialReference (State statelInitialRef)`  
*setInitialReference - set the initial reference state.*
- `void updateReference (State stateCandidate, Integer countIterationsCurrent)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`
- `void updateWeight (State stateCandidate)`  
*updateWeight - modify the weight of the generator.*
- `boolean searchState (State stateCandidate)`  
*searchState - search for a state in the reference list.*
- `float getWeight ()`  
*getWeight - get the weight of the generator.*
- `Generator roulette ()`  
*roulette - select a generator using roulette wheel selection.*
- `boolean awardUpdateREF (State stateCandidate)`  
*awardUpdateREF - update the reference of the generator.*
- `void updateAwardSC ()`  
*updateAwardSC - update the award for the selected generator.*
- `void updateAwardImp ()`  
*updateAwardImp - update the award for the selected generator.*
- `void setWeight (float weight)`  
*setWeight - set the weight of the generator.*
- `float[] getTrace ()`  
*getTrace - get the trace of the generator.*
- `void tournament (State stateCandidate, Integer countIterationsCurrent)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`  
*tournament - update the tournament for the selected generator.*
- `Object clone ()`  
*clone - create a copy of the generator.*
- `int[] getListCountBetterGender ()`  
*getListCountBetterGender - get the list of counts for better gender.*
- `int[] getListCountGender ()`  
*getListCountGender - get the list of counts for gender.*

## Métodos públicos estáticos

- static void [destroyMultiGenerator \(\)](#)  
*destroyMultiGenerator - destroy the MultiGenerator instance.*
- static void [initializeListGenerator \(\)](#) throws [IllegalArgumentException](#), [SecurityException](#), [ClassNotFoundException](#), [InstantiationException](#), [IllegalAccessException](#), [InvocationTargetException](#), [NoSuchMethodException](#)  
*initializeListGenerator - initialize the list of generators.*
- static void [initializeGenerators \(\)](#) throws [IllegalArgumentException](#), [SecurityException](#), [ClassNotFoundException](#), [InstantiationException](#), [IllegalAccessException](#), [InvocationTargetException](#), [NoSuchMethodException](#)  
*initializeGenerators - initialize the list of generators.*
- static void [createInstanceGeneratorsBPP \(\)](#)  
*createInstanceGeneratorsBPP - create instances of generators for the BPP problem.*
- static [Generator\[\] getListGenerators \(\)](#)  
*getListGenerators - get the list of generators.*
- static void [setListGenerators \(Generator\[\] listGenerators\)](#)  
*setListGenerators - set the list of generators.*
- static [Generator getActiveGenerator \(\)](#)  
*getActiveGenerator - get the active generator.*
- static void [setActiveGenerator \(Generator activeGenerator\)](#)  
*setActiveGenerator - set the active generator.*
- static void [setListGeneratedPP \(List< State > newListGeneratedPP\)](#)  
*setListGeneratedPP - set the list of generated states for the PP problem.*

## Atributos públicos estáticos

- static final List< State > [listGeneratedPP](#) = Collections.synchronizedList(new ArrayList<State>())
- static volatile [Generator activeGenerator](#)
- static final List< State > [listStateReference](#) = Collections.synchronizedList(new ArrayList<State>())

## Métodos privados estáticos

- static List< State > [getListGeneratedPP \(\)](#)  
*getListGeneratedPP - get the list of generated states for the PP problem.*

## Atributos privados

- [GeneratorType generatortype](#)

## Atributos estáticos privados

- static [Generator\[\] listGenerators](#) = new Generator[GeneratorType.values().length]
- static final Logger [LOGGER](#) = Logger.getLogger(MultiGenerator.class.getName())

## Otros miembros heredados

### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- int [countGender](#)
- int [countBetterGender](#)

### 8.60.1 Descripción detallada

[MultiGenerator](#) - class that implements the Multi-Generator metaheuristic.

Definición en la línea 22 del archivo [MultiGenerator.java](#).

### 8.60.2 Documentación de constructores y destructores

#### 8.60.2.1 MultiGenerator()

```
metaheuristics.generators.MultiGenerator.MultiGenerator () [inline]
```

[MultiGenerator](#) - default constructor.

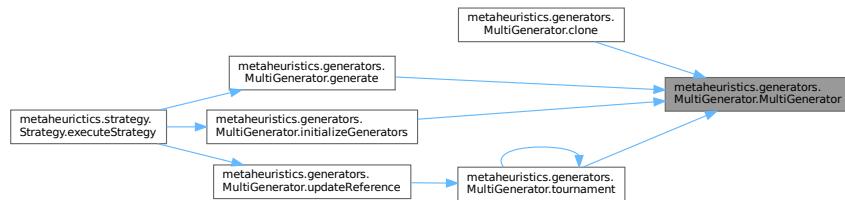
Definición en la línea 42 del archivo [MultiGenerator.java](#).

```
00042             {
00043         super();
00044         this.generatortype = GeneratorType.MultiGenerator;
00045     }
```

Hace referencia a [metaheuristics.generators.GeneratorType.MultiGenerator](#).

Referenciado por [clone\(\)](#), [generate\(\)](#), [initializeGenerators\(\)](#) y [tournament\(\)](#).

Gráfico de llamadas a esta función:



### 8.60.3 Documentación de funciones miembro

#### 8.60.3.1 awardUpdateREF()

```
boolean metaheuristics.generators.MultiGenerator.awardUpdateREF (
    State stateCandidate) [inline]
```

[awardUpdateREF](#) - update the reference of the generator.

#### Parámetros

<i>stateCandidate</i>	<input type="text"/>
-----------------------	----------------------

#### Devuelve

return true if the reference was updated, false otherwise

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 384 del archivo [MultiGenerator.java](#).

```
00384         // TODO Auto-generated method stub
00385         return false;
00386     }
00387 }
```

### 8.60.3.2 clone()

```
Object metaheuristics.generators.MultiGenerator.clone () [inline]
```

clone - create a copy of the generator.

Devuelve

a clone of the generator

Definición en la línea 468 del archivo [MultiGenerator.java](#).

```
00468         {
00469             try{
00470                 return super.clone();
00471             } catch (CloneNotSupportedException e) {
00472                 return new MultiGenerator();
00473             }
00474         }
```

Hace referencia a [MultiGenerator\(\)](#).

Gráfico de llamadas de esta función:



### 8.60.3.3 createInstanceGeneratorsBPP()

```
void metaheuristics.generators.MultiGenerator.createInstanceGeneratorsBPP () [inline], [static]
```

`createInstanceGeneratorsBPP` - create instances of generators for the BPP problem.

Definición en la línea 129 del archivo [MultiGenerator.java](#).

```
00129
00130     Generator generator = new RandomSearch();
00131
00132     int j = 0;
00133     while (j < EvolutionStrategies.countRef) {
00134         State stateCandidate;
00135         try {
00136             stateCandidate = generator.generate(1);
00137             Strategy.getStrategy().getProblem().Evaluate(stateCandidate);
00138             stateCandidate.setNumber(j);
00139             stateCandidate.setTypeGenerator(generator.getType());
00140             listGeneratedPP.add(stateCandidate);
00141         } catch (Exception e) {
00142             // Log exception instead of printing stack trace to avoid debug output in production
00143             LOGGER.log(Level.SEVERE, "Failed to create state candidate in
00144             createInstanceGeneratorsBPP", e);
00144         }
00145         j++;
00146     }
00147 }
```

Hace referencia a [metaheuristics.generators.EvolutionStrategies.countRef](#), [problem.definition.Problem.Evaluate\(\)](#), [metaheuristics.generators.Generator.generate\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getType\(\)](#)

`metaheuristics.generators.Generator.getType()`, `listGeneratedPP`, `LOGGER`, `problem.definition.State.setNumber()` y `problem.definition.State.setTypeGenerator()`.

Referenciado por [initializeGenerators\(\)](#).

Gráfico de llamadas de esta función:

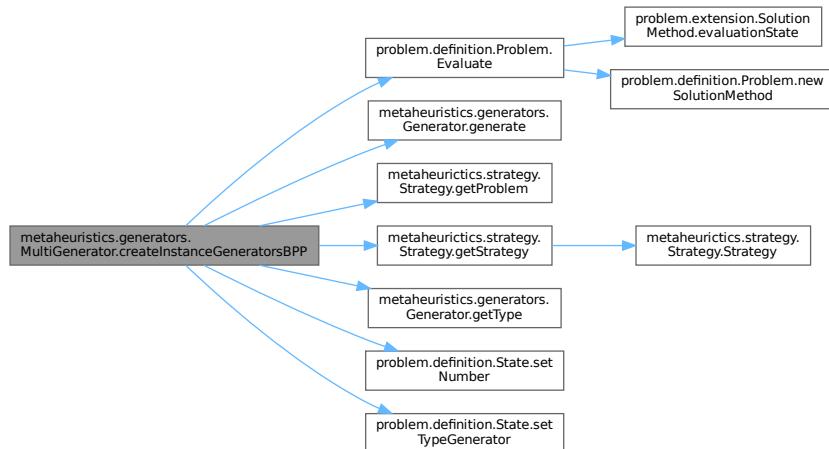


Gráfico de llamadas a esta función:



#### 8.60.3.4 `destroyMultiGenerator()`

`void metaheuristics.generators.MultiGenerator.destroyMultiGenerator () [inline], [static]`

`destroyMultiGenerator` - destroy the [MultiGenerator](#) instance.

Definición en la línea 50 del archivo [MultiGenerator.java](#).

```

00050
00051     listGeneratedPP.clear();
00052     //listGenerators.clear();
00053     listStateReference.clear();
00054     activeGenerator = null;
00055     listGenerators = null;
00056 }
```

Hace referencia a `activeGenerator`, `listGeneratedPP`, `listGenerators` y `listStateReference`.

#### 8.60.3.5 `generate()`

```

State metaheuristics.generators.MultiGenerator.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Classe
    NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
    NoSuchMethodException [inline]
```

`generate` - generate a new state using the [MultiGenerator](#).

## Parámetros

<i>operatornumber</i>	<input type="text"/>
-----------------------	----------------------

Devuelve

## Excepciones

<i>IllegalArgumentException</i>	
<i>SecurityException</i>	
<i>ClassNotFoundException</i>	
<i>InstantiationException</i>	
<i>IllegalAccessException</i>	
<i>InvocationTargetException</i>	
<i>NoSuchMethodException</i>	

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 213 del archivo [MultiGenerator.java](#).

```
00217      {
00218      // TODO Auto-generated method stub
00219      Strategy.getStrategy().generator = roulette();
00220      MultiGenerator.setActiveGenerator(Strategy.getStrategy().generator);
00221      activeGenerator.countGender++;
00222      State state = Strategy.getStrategy().generator.generate(1);
00223      return state;
00224 }
```

Hace referencia a [activeGenerator](#), [metaheuristics.generators.Generator.generate\(\)](#), [metaheuristics.strategy.Strategy.generator](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [MultiGenerator\(\)](#) y [roulette\(\)](#).

Referenciado por [metaheuristics.strategy.Strategy.executeStrategy\(\)](#).

Gráfico de llamadas de esta función:

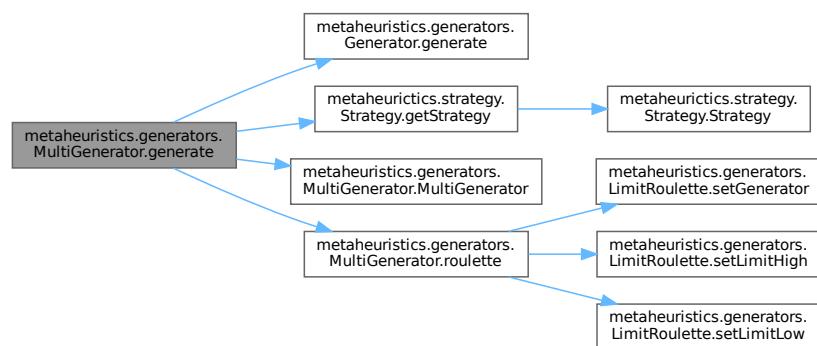


Gráfico de llamadas a esta función:



#### 8.60.3.6 `getActiveGenerator()`

`Generator` `metaheuristics.generators.MultiGenerator.getActiveGenerator ()` [inline], [static]

`getActiveGenerator` - get the active generator.

Devuelve

the active generator

Definición en la línea 177 del archivo [MultiGenerator.java](#).

```

00177
00178     return activeGenerator;
00179 }

```

Hace referencia a [activeGenerator](#).

#### 8.60.3.7 `getListCountBetterGender()`

`int[]` `metaheuristics.generators.MultiGenerator.getListCountBetterGender ()` [inline]

`getListCountBetterGender` - get the list of counts for better gender.

Devuelve

the list of counts for better gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 481 del archivo [MultiGenerator.java](#).

```

00481
00482     // TODO Auto-generated method stub
00483     // MultiGenerator does not maintain these arrays itself; return empty array to avoid nulls
00484     return new int[0];
00485 }

```

### 8.60.3.8 getListCountGender()

```
int[] metaheuristics.generators.MultiGenerator.getListCountGender () [inline]
```

getListCountGender - get the list of counts for gender.

Devuelve

the list of counts for gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 492 del archivo [MultiGenerator.java](#).

```
00492 {  
00493     // TODO Auto-generated method stub  
00494     // MultiGenerator does not maintain these arrays itself; return empty array to avoid nulls  
00495     return new int[0];  
00496 }
```

### 8.60.3.9 getListGeneratedPP()

```
List< State > metaheuristics.generators.MultiGenerator.getListGeneratedPP () [inline], [static],  
[private]
```

getListGeneratedPP - get the list of generated states for the PP problem.

Devuelve

the list of generated states

Definición en la línea 153 del archivo [MultiGenerator.java](#).

```
00153 {  
00154     return listGeneratedPP;  
00155 }
```

Hace referencia a [listGeneratedPP](#).

### 8.60.3.10 getListGenerators()

```
Generator[] metaheuristics.generators.MultiGenerator.getListGenerators () [inline], [static]
```

getListGenerators - get the list of generators.

Devuelve

the list of generators

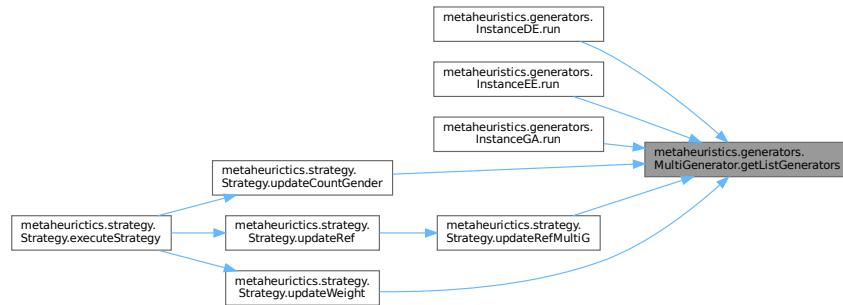
Definición en la línea 161 del archivo [MultiGenerator.java](#).

```
00161          {
00162      // return a defensive copy to avoid exposing internal static array
00163      return (listGenerators == null) ? null : Arrays.copyOf(listGenerators, listGenerators.length);
00164 }
```

Hace referencia a [listGenerators](#).

Referenciado por [metaheuristics.generators.InstanceDE.run\(\)](#), [metaheuristics.generators.InstanceEE.run\(\)](#), [metaheuristics.generators.InstanceGA.run\(\)](#), [metaheuristics.strategy.Strategy.updateCountGender\(\)](#), [metaheuristics.strategy.Strategy](#) y [metaheuristics.strategy.Strategy.updateWeight\(\)](#).

Gráfico de llamadas a esta función:



### 8.60.3.11 getReference()

```
State metaheuristics.generators.MultiGenerator.getReference () [inline]
```

`getReference` - get the reference state.

Devuelve

the reference state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 231 del archivo [MultiGenerator.java](#).

```
00231          {
00232      // TODO Auto-generated method stub
00233      return null;
00234 }
```

### 8.60.3.12 getReferenceList()

```
List< State > metaheuristics.generators.MultiGenerator.getReferenceList () [inline]
```

getReferenceList - get the list of reference states.

Devuelve

the list of reference states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 241 del archivo [MultiGenerator.java](#).

```
00241                               {  
00242         // TODO Auto-generated method stub  
00243         return listStateReference;  
00244     }
```

Hace referencia a [listStateReference](#).

Referenciado por [metaheuristics.strategy.Strategy.executeStrategy\(\)](#).

Gráfico de llamadas a esta función:



### 8.60.3.13 getSonList()

```
List< State > metaheuristics.generators.MultiGenerator.getSonList () [inline]
```

getSonList - get the list of son states.

Devuelve

the list of son states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 251 del archivo [MultiGenerator.java](#).

```
00251                               {  
00252         // TODO Auto-generated method stub  
00253         return null;  
00254     }
```

### 8.60.3.14 getTrace()

```
float[ ] metaheuristics.generators.MultiGenerator.getTrace () [inline]
```

getTrace - get the trace of the generator.

**Devuelve**

the trace array

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 443 del archivo [MultiGenerator.java](#).

```
00443           {
00444             // TODO Auto-generated method stub
00445             // MultiGenerator does not maintain a single trace array; return empty array to avoid nulls
00446             return new float[0];
00447 }
```

### 8.60.3.15 getType()

```
GeneratorType metaheuristics.generators.MultiGenerator.getType () [inline]
```

getType - get the type of the generator.

**Devuelve**

the type of the generator

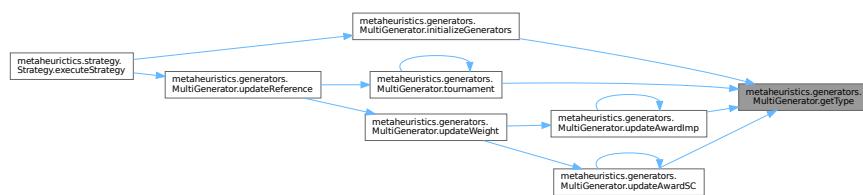
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 261 del archivo [MultiGenerator.java](#).

```
00261           {
00262             return this.generatortype;
00263 }
```

Referenciado por [initializeGenerators\(\)](#), [tournament\(\)](#), [updateAwardImp\(\)](#) y [updateAwardSC\(\)](#).

Gráfico de llamadas a esta función:



### 8.60.3.16 getWeight()

```
float metaheuristics.generators.MultiGenerator.getWeight () [inline]
```

getWeight - get the weight of the generator.

**Devuelve**

the weight of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 328 del archivo [MultiGenerator.java](#).

```
00328         {
00329             // TODO Auto-generated method stub
00330             return 0;
00331         }
```

### 8.60.3.17 initializeGenerators()

```
void metaheuristics.generators.MultiGenerator.initializeGenerators () throws IllegalArgumentException<-
Exception, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException<-
Exception, InvocationTargetException, NoSuchMethodException [inline], [static]
```

initializeGenerators - initialize the generators.

#### Excepciones

<i>IllegalArgumentException</i>	
<i>SecurityException</i>	
<i>ClassNotFoundException</i>	
<i>InstantiationException</i>	
<i>IllegalAccessException</i>	
<i>InvocationTargetException</i>	
<i>NoSuchMethodException</i>	

Definición en la línea 90 del archivo [MultiGenerator.java](#).

```
00090     {
00091         initializeListGenerator();
00092         State stateREF = new State(Strategy.getStrategy().getProblem().getState());
00093         listStateReference.add(stateREF);
00094         for (int i = 0; i < listGenerators.length; i++) {
00095             if ((listGenerators[i].getType().equals(GeneratorType.HillClimbing)) ||
00096                 (listGenerators[i].getType().equals(GeneratorType.RandomSearch)) ||
00097                 (listGenerators[i].getType().equals(GeneratorType.TabuSearch)) ||
00098                 (listGenerators[i].getType().equals(GeneratorType.SimulatedAnnealing)) ||
00099                 (listGenerators[i].getType().equals(GeneratorType.LimitThreshold))) {
00100                 listGenerators[i].setInitialReference(stateREF);
00101             }
00102             createInstanceGeneratorsBPP();
00103             Strategy.getStrategy().listStates = MultiGenerator.getListGeneratedPP();
00104             FactoryGenerator iffFactoryGeneratorEE = new FactoryGenerator();
00105             Generator generatorEE =
iffFactoryGeneratorEE.createGenerator(GeneratorType.EvolutionStrategies);
```

```

00105
00106     FactoryGenerator iffFactoryGeneratorGA = new FactoryGenerator();
00107     Generator generatorGA = iffFactoryGeneratorGA.createGenerator(GeneratorType.GeneticAlgorithm);
00108
00109     FactoryGenerator iffFactoryGeneratorEDA = new FactoryGenerator();
00110     Generator generatorEDA =
00111         iffFactoryGeneratorEDA.createGenerator(GeneratorType.DistributionEstimationAlgorithm);
00112
00113     for (int i = 0; i < MultiGenerator.getListGenerators().length; i++) {
00114
00115         if(MultiGenerator.getListGenerators()[i].getType().equals(GeneratorType.EvolutionStrategies)){
00116             MultiGenerator.getListGenerators()[i] = generatorEE;
00117
00118         }
00119
00120         if(MultiGenerator.getListGenerators()[i].getType().equals(GeneratorType.GeneticAlgorithm)){
00121             MultiGenerator.getListGenerators()[i] = generatorGA;
00122
00123         }
00124     }

```

Hace referencia a `factory_method.FactoryGenerator.createGenerator()`, `createInstanceGeneratorsBPP()`, `metaheuristics.generators.GeneratorType.DistributionEstimationAlgorithm`, `metaheuristics.generators.GeneratorType.EvolutionStrategies`, `metaheuristics.generators.GeneratorType.GeneticAlgorithm`, `metaheuristics.strategy.Strategy.getProblem()`, `problem.definition.Problem.getState()`, `metaheuristics.strategy.Strategy.getStrategy()`, `getType()`, `metaheuristics.generators.Generator.initializeListGenerator()`, `metaheuristics.generators.GeneratorType.LimitThreshold`, `listGenerators`, `listStateReference`, `metaheuristics.strategy.Strategy.listStates`, `MultiGenerator()`, `metaheuristics.generators.GeneratorType.RandomSearch`, `metaheuristics.generators.GeneratorType.SimulatedAnnealing` y `metaheuristics.generators.GeneratorType.TabuSearch`.

Referenciado por [metaheuristics.strategy.Strategy.executeStrategy\(\)](#).

Gráfico de llamadas de esta función:

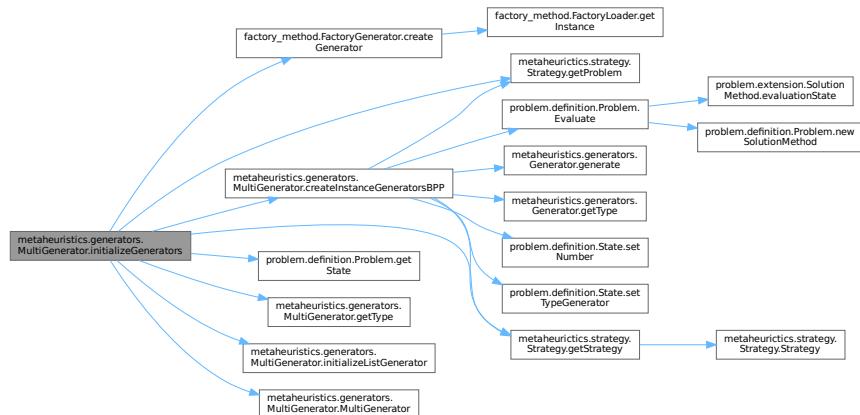


Gráfico de llamadas a esta función:



### 8.60.3.18 initializeListGenerator()

```
void metaheuristics.generators.MultiGenerator.initializeListGenerator () throws IllegalArgument-
Exception, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccess-
Exception, InvocationTargetException, NoSuchMethodException [inline], [static]
```

initializeListGenerator - initialize the list of generators.

#### Excepciones

<i>IllegalArgumentException</i>	
<i>SecurityException</i>	
<i>ClassNotFoundException</i>	
<i>InstantiationException</i>	
<i>IllegalAccessException</i>	
<i>InvocationTargetException</i>	
<i>NoSuchMethodException</i>	

Definición en la línea 68 del archivo [MultiGenerator.java](#).

```
00068
{
00069     listGenerators = new Generator[4];
00070     Generator generator1 = new HillClimbing();
00071     Generator generator2 = new EvolutionStrategies();
00072     Generator generator3 = new LimitThreshold();
00073     Generator generator4 = new GeneticAlgorithm();
00074     listGenerators[0] = generator1;
00075     listGenerators[1] = generator2;
00076     listGenerators[2] = generator3;
00077     listGenerators[3] = generator4;
00078 }
```

Hace referencia a [listGenerators](#).

Referenciado por [initializeGenerators\(\)](#).

Gráfico de llamadas a esta función:



### 8.60.3.19 roulette()

```
Generator metaheuristics.generators.MultiGenerator.roulette () [inline]
```

roulette - select a generator using roulette wheel selection.

Devuelve

the selected generator

Definición en la línea 337 del archivo [MultiGenerator.java](#).

```

00337      {
00338          float totalWeight = 0;
00339          for (int i = 0; i < listGenerators.length; i++) {
00340              totalWeight = listGenerators[i].getWeight() + totalWeight;
00341          }
00342          List<Float> listProb = new ArrayList<Float>();
00343          for (int i = 0; i < listGenerators.length; i++) {
00344              float probF = listGenerators[i].getWeight() / totalWeight;
00345              listProb.add(probF);
00346          }
00347          List<LimitRoulette> listLimit = new ArrayList<LimitRoulette>();
00348          float limitHigh = 0;
00349          float limitLow = 0;
00350          for (int i = 0; i < listProb.size(); i++) {
00351              LimitRoulette limitRoulette = new LimitRoulette();
00352              limitHigh = listProb.get(i) + limitHigh;
00353              limitRoulette.setLimitHigh(limitHigh);
00354              limitRoulette.setLimitLow(limitLow);
00355              limitLow = limitHigh;
00356              limitRoulette.setGenerator(listGenerators[i]);
00357              listLimit.add(limitRoulette);
00358          }
00359          // Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.
00360          // This RNG chooses a generator according to roulette weights and is not
00361          // used for security-sensitive purposes. Suppress Sonar hotspot S2245.
00362          @SuppressWarnings("squid:S2245")
00363          float numbAleatory = ThreadLocalRandom.current().nextFloat();
00364          boolean find = false;
00365          int i = 0;
00366          while ((find == false) && (i < listLimit.size())){
00367              if((listLimit.get(i).getLimitLow() <= numbAleatory) && (numbAleatory <=
00368                  listLimit.get(i).getLimitHigh())){
00369                  find = true;
00370              } else i++;
00371          }
00372          if (find) {
00373              return listLimit.get(i).getGenerator();
00374          }
00375          else return listLimit.get(listLimit.size() - 1).getGenerator();
00376      }

```

Hace referencia a [listGenerators](#), [metaheuristics.generators.LimitRoulette.setGenerator\(\)](#), [metaheuristics.generators.LimitRoulette.setLimitHigh\(\)](#) y [metaheuristics.generators.LimitRoulette.setLimitLow\(\)](#).

Referenciado por [generate\(\)](#).

Gráfico de llamadas de esta función:

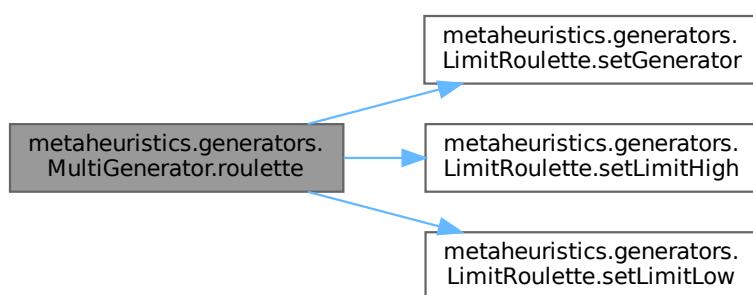


Gráfico de llamadas a esta función:



### 8.60.3.20 searchState()

```
boolean metaheuristics.generators.MultiGenerator.searchState (
    State stateCandidate) [inline]
```

searchState - search for a state in the reference list.

#### Parámetros

<i>stateCandidate</i>	<input type="text"/>
-----------------------	----------------------

#### Devuelve

return true if the state is found, false otherwise

Definición en la línea 302 del archivo [MultiGenerator.java](#).

```

00302
00303     if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00304         if(stateCandidate.getEvaluation().get(0) >
00305             Strategy.getStrategy().getBestState().getEvaluation().get(0)){
00306                 if(stateCandidate.getEvaluation().get(0) >
00307                     Strategy.getStrategy().getBestState().getEvaluation().get(0))
00308                     activeGenerator.countBetterGender++;
00309                 return true;
00310             }
00311         else {
00312             if(stateCandidate.getEvaluation().get(0) <
00313                 Strategy.getStrategy().getBestState().getEvaluation().get(0)){
00314                 if(stateCandidate.getEvaluation().get(0) <
00315                     Strategy.getStrategy().getBestState().getEvaluation().get(0))
00316                     activeGenerator.countBetterGender++;
00317                 return true;
00318             }
00319         }
00320     }
  
```

Hace referencia a [activeGenerator](#), [metaheuristics.strategy.Strategy.getBestState\(\)](#), [problem.definition.State.getEvaluation\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Problem.getTypeProblem\(\)](#) y [problem.definition.Problem.ProblemType.Maximizar](#).

Referenciado por [updateWeight\(\)](#).

Gráfico de llamadas de esta función:

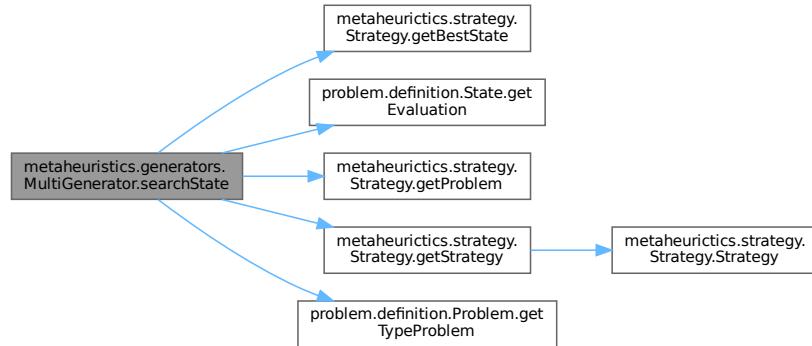


Gráfico de llamadas a esta función:



### 8.60.3.21 setActiveGenerator()

```
void metaheuristics.generators.MultiGenerator.setActiveGenerator (
    Generator activeGenerator) [inline], [static]
```

setActiveGenerator - set the active generator.

#### Parámetros

<i>activeGenerator</i>	the active generator to set
------------------------	-----------------------------

Definición en la línea 184 del archivo [MultiGenerator.java](#).

```
00184
00185     MultiGenerator.activeGenerator = activeGenerator;
00186 }
```

Hace referencia a [activeGenerator](#).

### 8.60.3.22 setGeneratorType()

```
void metaheuristics.generators.MultiGenerator.setGeneratorType (
    GeneratorType generatortype) [inline]
```

setGeneratorType - set the type of the generator.

## Parámetros

<i>generatorotype</i>	the type of the generator to set
-----------------------	----------------------------------

Definición en la línea 35 del archivo [MultiGenerator.java](#).

```
00035
00036     this.generatorotype = generatorotype;
00037 }
```

Hace referencia a [generatorotype](#).

### 8.60.3.23 setInitialReference()

```
void metaheuristics.generators.MultiGenerator.setInitialReference (
    State stateInitialRef) [inline]
```

setInitialReference - set the initial reference state.

## Parámetros

<i>stateInitialRef</i>	the initial reference state
------------------------	-----------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 270 del archivo [MultiGenerator.java](#).

```
00270
00271     // TODO Auto-generated method stub
00272
00273 }
```

### 8.60.3.24 setListGeneratedPP()

```
void metaheuristics.generators.MultiGenerator.setListGeneratedPP (
    List< State > newListGeneratedPP) [inline], [static]
```

setListGeneratedPP - set the list of generated states for the PP problem.

## Parámetros

<i>newListGeneratedPP</i>	the new list of generated states
---------------------------	----------------------------------

Definición en la línea 191 del archivo [MultiGenerator.java](#).

```
00191
00192     synchronized (listGeneratedPP) {
00193         listGeneratedPP.clear();
00194         if (newListGeneratedPP != null) {
00195             listGeneratedPP.addAll(newListGeneratedPP);
00196         }
00197     }
00198 }
```

Hace referencia a [listGeneratedPP](#).

---

### 8.60.3.25 setListGenerators()

```
void metaheuristics.generators.MultiGenerator.setListGenerators (
    Generator[] listGenerators) [inline], [static]
```

## Parámetros

<i>listGenerators</i>	the list of generators to set
-----------------------	-------------------------------

Definición en la línea 169 del archivo [MultiGenerator.java](#).

```
00169
00170      // store a defensive copy to avoid keeping a reference to caller's mutable array
00171      MultiGenerator.listGenerators = (listGenerators == null) ? null :
00172          Arrays.copyOf(listGenerators, listGenerators.length);
00173 }
```

Hace referencia a [listGenerators](#).

### 8.60.3.26 setWeight()

```
void metaheuristics.generators.MultiGenerator.setWeight (
    float weight) [inline]
```

`setWeight` - set the weight of the generator.

## Parámetros

<i>weight</i>	
---------------	--

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 434 del archivo [MultiGenerator.java](#).

```
00434
00435      // TODO Auto-generated method stub
00436
00437 }
```

### 8.60.3.27 tournament()

```
void metaheuristics.generators.MultiGenerator.tournament (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

`tournament` - update the tournament for the selected generator.

## Parámetros

<i>stateCandidate</i>	
<i>countIterationsCurrent</i>	

Definición en la línea 455 del archivo [MultiGenerator.java](#).

```
00455
00456      State stateTem = new State(stateCandidate);
00457      for (int i = 0; i < MultiGenerator.getListGenerators().length; i++) {
00458          if (!listGenerators[i].getType().equals(GeneratorType.MultiGenerator))
```

```

00459             MultiGenerator.getListGenerators() [i].updateReference(stateTem,
00460                 countIterationsCurrent);
00461         }

```

Hace referencia a [getType\(\)](#), [listGenerators](#), [metaheuristics.generators.GeneratorType.MultiGenerator](#), [MultiGenerator\(\)](#) y [tournament\(\)](#).

Referenciado por [tournament\(\)](#) y [updateReference\(\)](#).

Gráfico de llamadas de esta función:

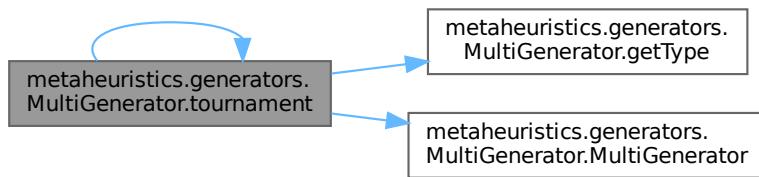


Gráfico de llamadas a esta función:



### 8.60.3.28 updateAwardImp()

```
void metaheuristics.generators.MultiGenerator.updateAwardImp () [inline]
```

[updateAwardImp](#) - update the award for the selected generator.

Definición en la línea 413 del archivo [MultiGenerator.java](#).

```

00413             {
00414         float weightLast = activeGenerator.getWeight();
00415         float weightUpdate = (float) (weightLast * (1 - 0.1));
00416         activeGenerator.setWeight(weightUpdate);
00417         for (int i = 0; i < listGenerators.length; i++) {
00418             if(listGenerators[i].equals(activeGenerator))
00419                 activeGenerator.getTrace() [Strategy.getStrategy().getCountCurrent()] = weightUpdate;
00420             else{
00421                 if(!listGenerators[i].getType().equals(GeneratorType.MultiGenerator)){
00422                     float trace = listGenerators[i].getWeight();
00423                     listGenerators[i].getTrace() [Strategy.getStrategy().getCountCurrent()] = trace;
00424                 }
00425             }
00426         }
00427     }

```

Hace referencia a [activeGenerator](#), [metaheuristics.strategy.Strategy.getCountCurrent\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [getType\(\)](#), [listGenerators](#), [metaheuristics.generators.GeneratorType.MultiGenerator](#) y [updateAwardImp\(\)](#).

Referenciado por [updateAwardImp\(\)](#) y [updateWeight\(\)](#).

Gráfico de llamadas de esta función:

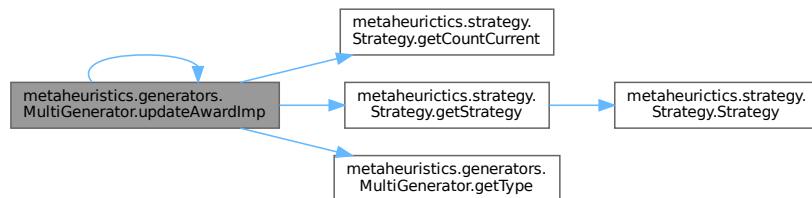


Gráfico de llamadas a esta función:



### 8.60.3.29 updateAwardSC()

`void metaheuristics.generators.MultiGenerator.updateAwardSC () [inline]`

`updateAwardSC` - update the award for the selected generator.

Definición en la línea 393 del archivo [MultiGenerator.java](#).

```

00393         {
00394             float weightLast = activeGenerator.getWeight();
00395             float weightUpdate = (float) (weightLast * (1 - 0.1) + 10);
00396             activeGenerator.setWeight(weightUpdate);
00397             for (int i = 0; i < listGenerators.length; i++) {
00398                 if(listGenerators[i].equals(activeGenerator))
00399                     activeGenerator.getTrace()[Strategy.getStrategy().getCountCurrent()] = weightUpdate;
00400                 else{
00401                     if(!listGenerators[i].getType().equals(GeneratorType.MultiGenerator)){
00402                         float trace = listGenerators[i].getWeight();
00403                         listGenerators[i].getTrace() [Strategy.getStrategy().getCountCurrent()] = trace;
00404                     }
00405                 }
00406             }
00407         }
  
```

Hace referencia a `activeGenerator`, `metaheuristics.strategy.Strategy.getCountCurrent()`, `metaheuristics.strategy.Strategy.getStrategy()`, `get_type()`, `listGenerators`, `metaheuristics.generators.GeneratorType.MultiGenerator` y `updateAwardSC()`.

Referenciado por [updateAwardSC\(\)](#) y [updateWeight\(\)](#).

Gráfico de llamadas de esta función:

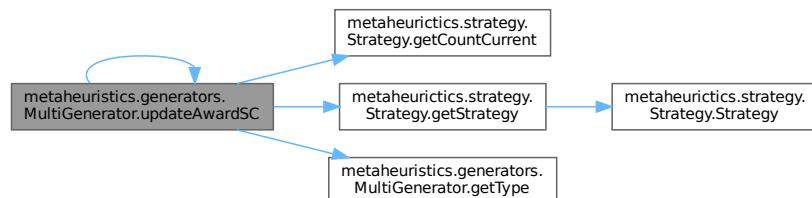


Gráfico de llamadas a esta función:



#### 8.60.3.30 updateReference()

```

void metaheuristics.generators.MultiGenerator.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
  
```

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 276 del archivo [MultiGenerator.java](#).

```

00280      {
00281          // TODO Auto-generated method stub
00282          updateWeight(stateCandidate);
00283          tournament(stateCandidate, countIterationsCurrent);
00284      }
  
```

Hace referencia a [tournament\(\)](#) y [updateWeight\(\)](#).

Referenciado por [metaheuristics.strategy.Strategy.executeStrategy\(\)](#).

Gráfico de llamadas de esta función:

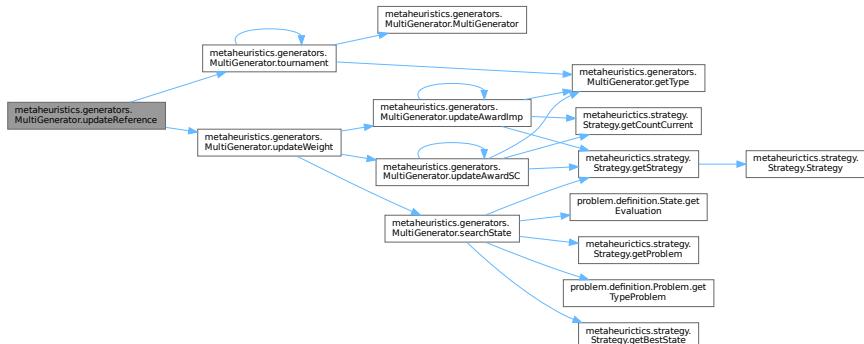


Gráfico de llamadas a esta función:



### 8.60.3.31 updateWeight()

```
void metaheuristics.generators.MultiGenerator.updateWeight (
    State stateCandidate) [inline]
```

updateWeight - modify the weight of the generator.

#### Parámetros

<i>stateCandidate</i>	
-----------------------	--

Definición en la línea 290 del archivo [MultiGenerator.java](#).

```
00290             {
00291     boolean search = searchState(stateCandidate); //premio por calidad.
00292     if(search == false)
00293         updateAwardImp();
00294     else
00295         updateAwardSC();
```

Hace referencia a [searchState\(\)](#), [updateAwardImp\(\)](#) y [updateAwardSC\(\)](#).

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas de esta función:

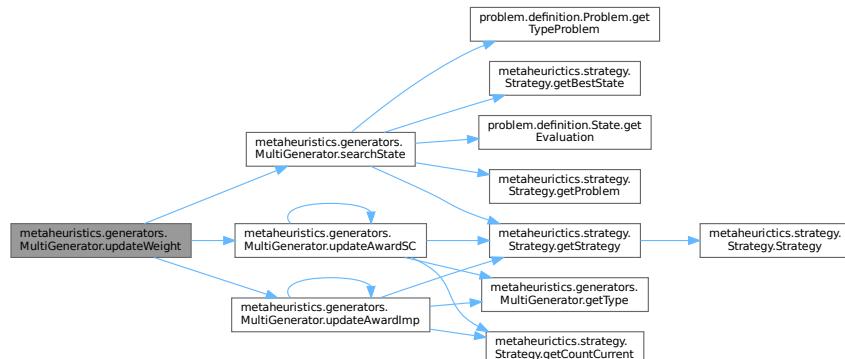


Gráfico de llamadas a esta función:



## 8.60.4 Documentación de datos miembro

### 8.60.4.1 activeGenerator

```
volatile Generator metaheuristics.generators.MultiGenerator.activeGenerator [static]
```

Definición en la línea 27 del archivo [MultiGenerator.java](#).

Referenciado por [destroyMultiGenerator\(\)](#), [generate\(\)](#), [getActiveGenerator\(\)](#), [searchState\(\)](#), [setActiveGenerator\(\)](#), [updateAwardImp\(\)](#) y [updateAwardSC\(\)](#).

#### 8.60.4.2 generatortype

```
GeneratorType metaheuristics.generators.MultiGenerator.generatortype [private]
```

Definición en la línea 24 del archivo [MultiGenerator.java](#).

Referenciado por [setGeneratorType\(\)](#).

#### 8.60.4.3 listGeneratedPP

```
final List<State> metaheuristics.generators.MultiGenerator.listGeneratedPP = Collections.←
synchronizedList(new ArrayList<State>()) [static]
```

Definición en la línea 26 del archivo [MultiGenerator.java](#).

Referenciado por [createInstanceGeneratorsBPP\(\)](#), [destroyMultiGenerator\(\)](#), [metaheuristics.strategy.Strategy.executeStrategy\(\)](#), [getListGeneratedPP\(\)](#) y [setListGeneratedPP\(\)](#).

#### 8.60.4.4 listGenerators

```
Generator [] metaheuristics.generators.MultiGenerator.listGenerators = new Generator[Generator.←
Type.values().length] [static], [private]
```

Definición en la línea 25 del archivo [MultiGenerator.java](#).

Referenciado por [destroyMultiGenerator\(\)](#), [getListGenerators\(\)](#), [initializeGenerators\(\)](#), [initializeListGenerator\(\)](#), [roulette\(\)](#), [setListGenerators\(\)](#), [tournament\(\)](#), [updateAwardImp\(\)](#) y [updateAwardSC\(\)](#).

#### 8.60.4.5 listStateReference

```
final List<State> metaheuristics.generators.MultiGenerator.listStateReference = Collections.←
synchronizedList(new ArrayList<State>()) [static]
```

Definición en la línea 28 del archivo [MultiGenerator.java](#).

Referenciado por [destroyMultiGenerator\(\)](#), [getReferenceList\(\)](#), [initializeGenerators\(\)](#) y [metaheuristics.strategy.Strategy.updateRef\(\)](#).

#### 8.60.4.6 LOGGER

```
final Logger metaheuristics.generators.MultiGenerator.LOGGER = Logger.getLogger(MultiGenerator.←
class.getName()) [static], [private]
```

Definición en la línea 29 del archivo [MultiGenerator.java](#).

Referenciado por [createInstanceGeneratorsBPP\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [MultiGenerator.java](#)

## 8.61 Referencia de la clase

### **metaheuristics.generators.MultiobjectiveHillClimbingDistance**

[MultiobjectiveHillClimbingDistance](#) - class that implements the Multiobjective Hill Climbing Distance metaheuristic.

Diagrama de herencia de `metaheuristics.generators.MultiobjectiveHillClimbingDistance`

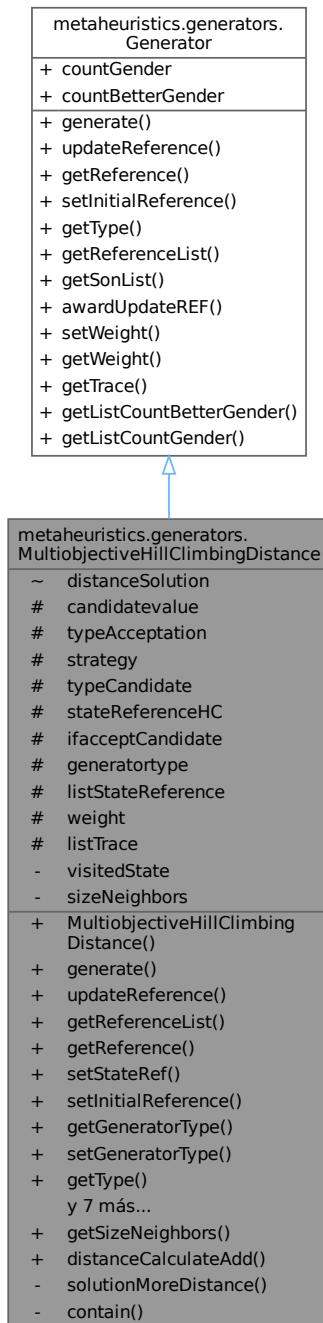
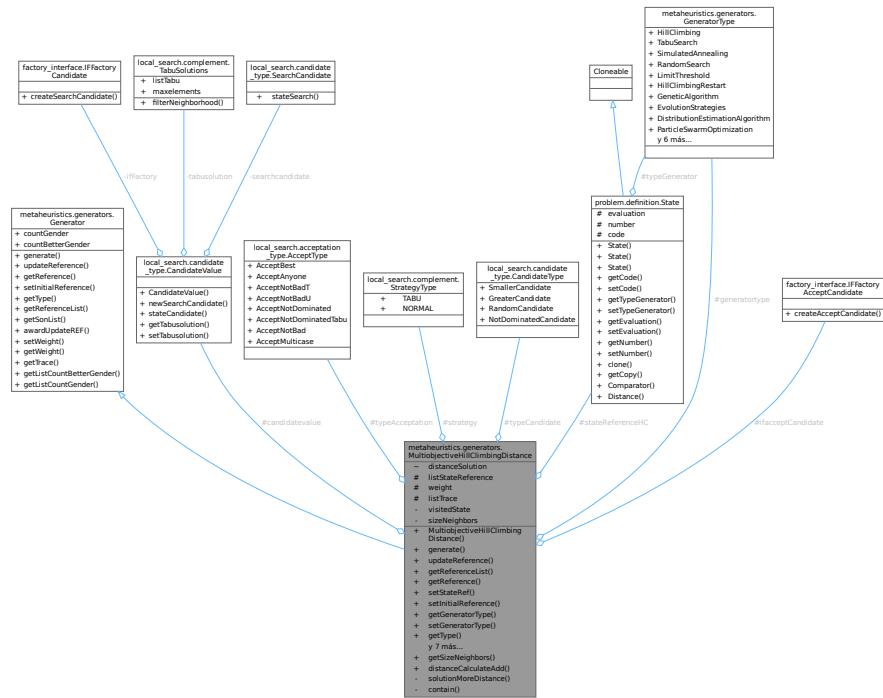


Diagrama de colaboración de metaheuristics.generators.MultiobjectiveHillClimbingDistance:



## Métodos públicos

- **MultiobjectiveHillClimbingDistance ()**  
*MultiobjectiveHillClimbingDistance - class that implements the Multiobjective Hill Climbing Distance metaheuristic.*
- **State generate (Integer operatornumber)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`  
*generate - generate a new state based on the operator number.*
- void **updateReference (State stateCandidate, Integer countIterationsCurrent)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`  
*updateReference - update the reference state and the list of visited states.*
- List< State > **getReferenceList ()**  
*getReferenceList - get the list of reference states.*
- **State getReference ()**  
*getReference - get the reference state.*
- void **setStateRef (State stateRef)**  
*setStateRef - set the reference state.*
- void **setInitialReference (State statelInitialRef)**  
*setInitialReference - set the initial reference state.*
- **GeneratorType getGeneratorType ()**  
*getGeneratorType - get the generator type.*
- void **setGeneratorType (GeneratorType generatortype)**  
*setGeneratorType - set the generator type.*
- **GeneratorType getType ()**  
*getType - get the generator type.*
- List< State > **getSonList ()**

- `getSonList` - get the list of child states.
- boolean `awardUpdateREF` (`State` stateCandidate)
  - awardUpdateREF* - check if the candidate state should update the reference state.
- float `getWeight` ()
  - getWeight* - get the weight of the generator.
- void `setWeight` (float `weight`)
  - setWeight* - set the weight of the generator.
- int[] `getListCountBetterGender` ()
  - getListCountBetterGender* - get the list of counts for better gender.
- int[] `getListCountGender` ()
  - getListCountGender* - get the list of counts for gender.
- float[] `getTrace` ()
  - getTrace* - get the trace of the generator.

## Métodos públicos estáticos

- static int `getSizeNeighbors` ()
  - Accessor for sizeNeighbors.*
- static List< Double > `distanceCalculateAdd` (List< `State` > solution)

## Atributos protegidos

- `CandidateValue` candidatevalue
- `AcceptType` typeAcceptation
- `StrategyType` strategy
- `CandidateType` typeCandidate
- `State` stateReferenceHC
- `IFFactoryAcceptCandidate` ifacceptCandidate
- `GeneratorType` generatortype
- List< `State` > `listStateReference` = new ArrayList<`State`>()
- float `weight`
- List< Float > `listTrace` = new ArrayList<Float>()

## Métodos privados

- `State` `solutionMoreDistance` (List< `State` > state, List< Double > distanceSolution)
  - solutionMoreDistance* - find the solution with the maximum distance.
- boolean `contain` (`State` state)
  - contain* - check if the state is contained in the visited states.

## Atributos privados

- List< `State` > `visitedState` = new ArrayList<`State`>()

## Atributos estáticos privados

- static final int `sizeNeighbors` = 10

## Otros miembros heredados

### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- int [countGender](#)
- int [countBetterGender](#)

## 8.61.1 Descripción detallada

[MultiobjectiveHillClimbingDistance](#) - class that implements the Multiobjective Hill Climbing Distance metaheuristic.

Definición en la línea 22 del archivo [MultiobjectiveHillClimbingDistance.java](#).

## 8.61.2 Documentación de constructores y destructores

### 8.61.2.1 [MultiobjectiveHillClimbingDistance\(\)](#)

```
metaheuristics.generators.MultiobjectiveHillClimbingDistance.MultiobjectiveHillClimbingDistance () [inline]
```

[MultiobjectiveHillClimbingDistance](#) - class that implements the Multiobjective Hill Climbing Distance metaheuristic.

Definición en la línea 50 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00050
00051     super();
00052     this.typeAcceptation = AcceptType.AcceptNotDominated;
00053     this.strategy = StrategyType.NORMAL;
00054     this.typeCandidate = CandidateType.NotDominatedCandidate;
00055     this.candidatevalue = new CandidateValue\(\);
00056     this.generatortype = GeneratorType.MultiobjectiveHillClimbingDistance;
00057     this.stateReferenceHC = new State\(\);
00058     this.weight = 50;
00059     listTrace.add(weight);
00060 }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptType.AcceptNotDominated](#), [listTrace](#), [metaheuristics.generators.GeneratorType](#), [local\\_search.complement.StrategyType.NORMAL](#), [local\\_search.candidate\\_type.CandidateType.NotDominatedCandidate](#) y [weight](#).

## 8.61.3 Documentación de funciones miembro

### 8.61.3.1 [awardUpdateREF\(\)](#)

```
boolean metaheuristics.generators.MultiobjectiveHillClimbingDistance.awardUpdateREF (
    State stateCandidate) [inline]
```

[awardUpdateREF](#) - check if the candidate state should update the reference state.

#### Parámetros

<code>stateCandidate</code>	<input type="button" value=""/>
-----------------------------	---------------------------------

#### Devuelve

return true if the reference state should be updated, false otherwise

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 289 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00289
00290     // TODO Auto-generated method stub
00291     return false;
00292 }
```

### 8.61.3.2 contain()

```
boolean metaheuristics.generators.MultiobjectiveHillClimbingDistance.contains (
    State state) [inline], [private]
```

contains - check if the state is contained in the visited states.

#### Parámetros

<i>state</i>	
--------------	--

#### Devuelve

return true if the state is contained, false otherwise

Definición en la línea 272 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00272         boolean found = false;
00273         for (Iterator<State> iter = visitedState.iterator(); iter.hasNext();) {
00274             State element = (State) iter.next();
00275             if(element.Comparator(state)) {
00276                 found = true;
00277             }
00278         }
00279     }
00280     return found;
00281 }
```

Hace referencia a [problem.definition.State.Comparator\(\)](#) y [visitedState](#).

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas de esta función:

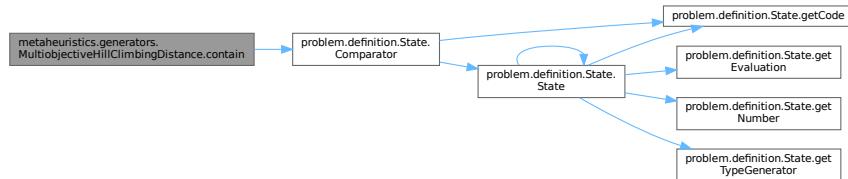


Gráfico de llamadas a esta función:



### 8.61.3.3 distanceCalculateAdd()

```
List< Double > metaheuristics.generators.MultiobjectiveHillClimbingDistance.distanceCalculateAdd (
    List< State > solution) [inline], [static]
```

Definición en la línea 234 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00234
00235     State[] solutions = solution.toArray(new State[solution.size()]);
00236     Double distance = 0.0;
00237     List<Double>listDist=new ArrayList<Double>();
00238     State lastSolution = solution.get(solution.size()-1);
00239     //Actualizando las distancias de todos los elementos excepto el nuevo insertando
00240     for (int k = 0; k < solutions.length-1; k++) {
00241         State solA = solutions[k];
00242         distance = solA.Distance(lastSolution);
00243         listDist.add(distanceSolution.get(k)+distance);
00244         distanceSolution.set(k, distanceSolution.get(k) + distance);
00245     }
00246     distance = 0.0;
00247     //Calculando la distancia del *ultimo elemento (elemento insertado) respecto al resto de los
00248     elementos
00249     if (solutions.length==1) {
00250         return distanceSolution;
00251     }else {
00252         for (int l = 0; l < solutions.length-1; l++) {
00253             State solB = solutions[l];
00254             distance += lastSolution.Distance(solB);
00255         }
00256         listDist.add(distance);
00257         distanceSolution.add(distance);
00258         distanceSolution=listDist;
00259     }
00260     return distanceSolution;
00261 }
00262 }
```

Hace referencia a [problem.definition.State.Distance\(\)](#).

Referenciado por [local\\_search.acceptation\\_type.Dominance.listDominance\(\)](#).

Gráfico de llamadas de esta función:

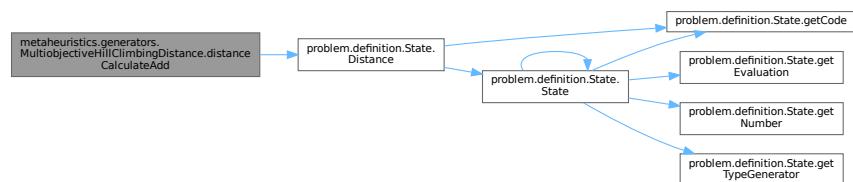


Gráfico de llamadas a esta función:



#### 8.61.3.4 generate()

```
State metaheuristics.generators.MultiobjectiveHillClimbingDistance.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

generate - generate a new state based on the operator number.

#### Parámetros

<code>operatornumber</code>	the operator number to use for generating the state
-----------------------------	---

#### Devuelve

the generated state

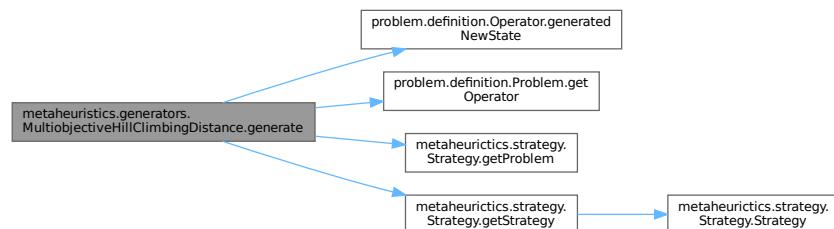
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 68 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00068
{
00069     List<State> neighborhood =
Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC, operatornumber);
00070     State statecandidate = candidatevalue.stateCandidate(stateReferenceHC, typeCandidate,
strategy, operatornumber, neighborhood);
00071     return statecandidate;
00072 }
```

Hace referencia a [candidatevalue](#), [problem.definition.Operator.generatedNewState\(\)](#), [problem.definition.Problem.getOperator\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [stateReferenceHC](#), [strategy](#) y [typeCandidate](#).

Gráfico de llamadas de esta función:



#### 8.61.3.5 getGeneratorType()

```
GeneratorType metaheuristics.generators.MultiobjectiveHillClimbingDistance.getGeneratorType ()
[inline]
```

getGeneratorType - get the generator type.

#### Devuelve

the generator type

Definición en la línea 203 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00203
00204     return generatortype;
00205 }
```

Hace referencia a [generatortype](#).

### 8.61.3.6 getListCountBetterGender()

```
int[] metaheuristics.generators.MultiobjectiveHillClimbingDistance.getListCountBetterGender ()  
[inline]
```

getListCountBetterGender - get the list of counts for better gender.

Devuelve

the list of counts for better gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 320 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00320           {  
00321             // This generator doesn't maintain listCount arrays; return empty array to avoid nulls  
00322             return new int[0];  
00323         }
```

### 8.61.3.7 getListCountGender()

```
int[] metaheuristics.generators.MultiobjectiveHillClimbingDistance.getListCountGender () [inline]
```

getListCountGender - get the list of counts for gender.

Devuelve

the list of counts for gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 330 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00330           {  
00331             // This generator doesn't maintain listCount arrays; return empty array to avoid nulls  
00332             return new int[0];  
00333         }
```

### 8.61.3.8 getReference()

```
State metaheuristics.generators.MultiobjectiveHillClimbingDistance.getReference () [inline]
```

getReference - get the reference state.

Devuelve

the reference state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 178 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00178           {  
00179             return stateReferenceHC;  
00180         }
```

Hace referencia a [stateReferenceHC](#).

### 8.61.3.9 getReferenceList()

```
List< State > metaheuristics.generators.MultiobjectiveHillClimbingDistance.getReferenceList ()  
[inline]
```

getReferenceList - get the list of reference states.

Devuelve

the list of reference states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 168 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00168         {  
00169             listStateReference.add(stateReferenceHC.clone());  
00170             return listStateReference;  
00171         }
```

Hace referencia a [listStateReference](#) y [stateReferenceHC](#).

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.61.3.10 getSizeNeighbors()

```
int metaheuristics.generators.MultiobjectiveHillClimbingDistance.getSizeNeighbors () [inline],  
[static]
```

Accesor for sizeNeighbors.

Definición en la línea 40 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00040         {  
00041             return sizeNeighbors;  
00042         }
```

Hace referencia a [sizeNeighbors](#).

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.61.3.11 getSonList()

```
List< State > metaheuristics.generators.MultiobjectiveHillClimbingDistance.getSonList ()  
[inline]
```

getSonList - get the list of child states.

Devuelve

the list of child states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 229 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00229           {  
00230             // TODO Auto-generated method stub  
00231             return null;  
00232         }
```

### 8.61.3.12 getTrace()

```
float[] metaheuristics.generators.MultiobjectiveHillClimbingDistance.getTrace () [inline]
```

getTrace - get the trace of the generator.

Devuelve

the trace of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 340 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00340           {  
00341             // TODO Auto-generated method stub  
00342             if (this.listTrace == null) return new float[0];  
00343             float[] arr = new float[this.listTrace.size()];  
00344             for (int i = 0; i < this.listTrace.size(); i++) {  
00345               Float v = this.listTrace.get(i);  
00346               arr[i] = (v == null) ? 0f : v.floatValue();  
00347             }  
00348             return arr;  
00349         }
```

### 8.61.3.13 getType()

```
GeneratorType metaheuristics.generators.MultiobjectiveHillClimbingDistance.getType () [inline]
```

getType - get the generator type.

Devuelve

the generator type

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 220 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00220           {  
00221             return this.generatortype;  
00222         }
```

### 8.61.3.14 getWeight()

```
float metaheuristics.generators.MultiobjectiveHillClimbingDistance.getWeight () [inline]
```

getWeight - get the weight of the generator.

#### Devuelve

the weight of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 300 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00300          {
00301      // TODO Auto-generated method stub
00302      return 0;
00303 }
```

### 8.61.3.15 setGeneratorType()

```
void metaheuristics.generators.MultiobjectiveHillClimbingDistance.setGeneratorType (
    GeneratorType generatortype) [inline]
```

setGeneratorType - set the generator type.

#### Parámetros

<i>generatortype</i>	the generator type to set
----------------------	---------------------------

Definición en la línea 211 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00211          {
00212      this.generatortype = generatortype;
00213 }
```

Hace referencia a [generatortype](#).

### 8.61.3.16 setInitialReference()

```
void metaheuristics.generators.MultiobjectiveHillClimbingDistance.setInitialReference (
    State stateInitialRef) [inline]
```

setInitialReference - set the initial reference state.

#### Parámetros

<i>stateInitialRef</i>	the initial reference state to set
------------------------	------------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 195 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00195          {
00196      this.stateReferenceHC = stateInitialRef;
00197 }
```

### 8.61.3.17 setStateRef()

## Parámetros

<i>stateRef</i>	the reference state to set
-----------------	----------------------------

Definición en la línea 186 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00186         {
00187             this.stateReferenceHC = stateRef;
00188         }
```

### 8.61.3.18 setWeight()

```
void metaheuristics.generators.MultiobjectiveHillClimbingDistance.setWeight (
    float weight) [inline]
```

*setWeight* - set the weight of the generator.

## Parámetros

<i>weight</i>	
---------------	--

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 310 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```
00310         {
00311             // TODO Auto-generated method stub
00312
00313     }
```

Hace referencia a [weight](#).

### 8.61.3.19 solutionMoreDistance()

```
State metaheuristics.generators.MultiobjectiveHillClimbingDistance.solutionMoreDistance (
    List< State > state,
    List< Double > distanceSolution) [inline], [private]
```

*solutionMoreDistance* - find the solution with the maximum distance.

## Parámetros

<i>state</i>	the list of states
<i>distanceSolution</i>	the list of distances corresponding to the states

**Devuelve**

the state with the maximum distance

Definición en la línea 145 del archivo [MultiobjectiveHillClimbingDistance.java](#).

```

00145
00146     Double max = (double) -1;
00147     int pos = -1;
00148     Double[] distance = distanceSolution.toArray(new Double[distanceSolution.size()]);
00149     State[] solutions = state.toArray(new State[state.size()]);
00150     for (int i = 0; i < distance.length; i++) {
00151         Double dist = distance[i];
00152         if(dist > max){
00153             max = dist;
00154             pos = i;
00155         }
00156     }
00157     if(pos != -1)
00158         return solutions[pos];
00159     else
00160         return null;
00161 }
```

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.61.3.20 updateReference()

```

void metaheuristics.generators.MultiobjectiveHillClimbingDistance.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

**updateReference** - update the reference state and the list of visited states.

#### Parámetros

<i>stateCandidate</i>	
<i>countIterationsCurrent</i>	

#### Excepciones

<i>IllegalArgumentException</i>	
<i>SecurityException</i>	
<i>ClassNotFoundException</i>	
<i>InstantiationException</i>	

<i>IllegalAccessException</i>	
<i>InvocationTargetException</i>	
<i>NoSuchMethodException</i>	

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 87 del archivo [MultiobjectiveHillClimbingDistance.java](#).

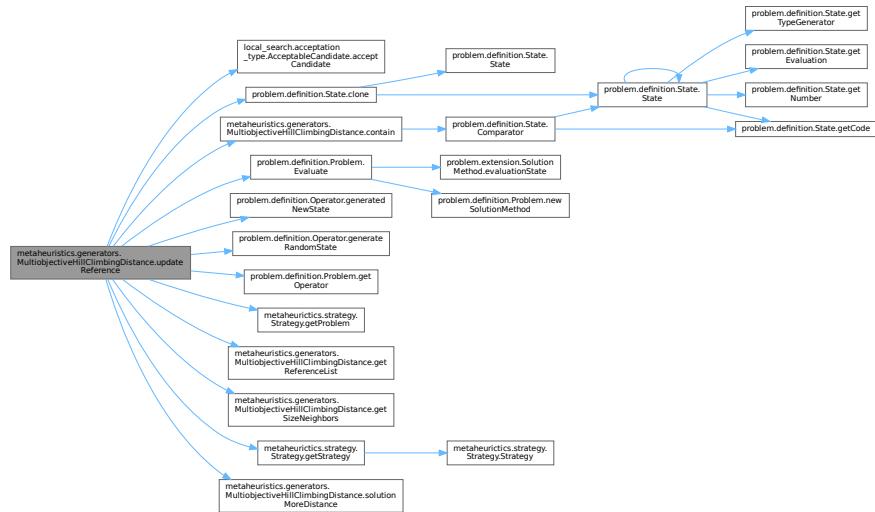
```

00088
00089     //Agregando la primera soluci n a la lista de soluciones no dominadas
00090     if(Strategy.getStrategy().listRefPoblacFinal.size() == 0){
00091         Strategy.getStrategy().listRefPoblacFinal.add(stateReferenceHC.clone());
00092         distanceSolution.add(Double.valueOf(0.0));
00093     }
00094     ifacceptCandidate = new FactoryAcceptCandidate();
00095     AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00096     State lastState =
00097         Strategy.getStrategy().listRefPoblacFinal.get(Strategy.getStrategy().listRefPoblacFinal.size()-1);
00098     List<State> neighborhood =
00099         Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC,
00100         getSizeNeighbors());
00101     int i = 0;
00102
00103     Boolean accept = candidate.acceptCandidate(lastState, stateCandidate.clone());
00104     if(accept.equals(true)){
00105         stateReferenceHC = stateCandidate.clone();
00106         visitedState = new ArrayList<State>();
00107     }
00108     else{
00109         boolean stop = false;
00110         while (i < neighborhood.size() && stop==false) {
00111             if (contain(neighborhood.get(i))==false) {
00112                 stateReferenceHC = solutionMoreDistance(Strategy.getStrategy().listRefPoblacFinal,
00113                 distanceSolution);
00114                 visitedState.add(stateReferenceHC);
00115                 stop=true;
00116                 lastState=stateReferenceHC.clone();
00117             }
00118             i++;
00119         }
00120         int coutrestart=0;
00121         while (stop == false && coutrestart < getSizeNeighbors() && accept==false) {
00122             stateCandidate =
00123                 Strategy.getStrategy().getProblem().generateRandomState(1).get(0);
00124             if (contain(stateCandidate)==false) {
00125                 Strategy.getStrategy().getProblem().Evaluate(stateCandidate);
00126                 visitedState.add(stateCandidate);
00127                 stop=true;
00128                 coutrestart++;
00129                 accept = candidate.acceptCandidate(lastState, stateCandidate.clone());
00130             }
00131             if(accept.equals(true)){
00132                 stateReferenceHC = stateCandidate.clone();
00133                 visitedState = new ArrayList<State>();
00134             }
00135         }
00136         getReferenceList();
00137     }
}

```

Hace referencia a [local\\_search.acceptation\\_type.AcceptableCandidate.acceptCandidate\(\)](#), [problem.definition.State.clone\(\)](#), [contain\(\)](#), [problem.definition.Problem.Evaluate\(\)](#), [problem.definition.Operator.generatedNewState\(\)](#), [problem.definition.Operator.generate\(\)](#), [problem.definition.Problem.getOperator\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [getReferenceList\(\)](#), [getSizeNeighbors\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [ifacceptCandidate](#), [metaheuristics.strategy.Strategy.listRefPoblacFinal\(\)](#), [solutionMoreDistance\(\)](#), [stateReferenceHC](#), [typeAcceptation](#) y [visitedState](#).

Gráfico de llamadas de esta función:



#### 8.61.4 Documentación de datos miembro

#### 8.61.4.1 candidatevalue

**CandidateValue** metaheuristics.generators.MultiobjectiveHillClimbingDistance.candidatevalue  
[protected]

Definición en la línea 24 del archivo [MultiobjectiveHillClimbingDistance.java](#).

Referenciado por [generate\(\)](#).

#### 8.61.4.2 generatortype

**GeneratorType** metaheuristics.generators.MultiobjectiveHillClimbingDistance.generatortype [protected]

Definición en la línea 30 del archivo [MultiobjectiveHillClimbingDistance.java](#).

Referenciado por `getGeneratorType()` y `setGeneratorType()`.

#### 8.61.4.3 ifacceptCandidate

```
IFFFactoryAcceptCandidate metaheuristics.generators.MultiobjectiveHillClimbingDistance.ifaccept←  
Candidate [protected]
```

Definición en la línea 29 del archivo [MultiobjectiveHillClimbingDistance.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.61.4.4 listStateReference

```
List<State> metaheuristics.generators.MultiobjectiveHillClimbingDistance.listStateReference =  
new ArrayList<State>() [protected]
```

Definición en la línea 31 del archivo [MultiobjectiveHillClimbingDistance.java](#).

Referenciado por [getReferenceList\(\)](#).

#### 8.61.4.5 listTrace

```
List<Float> metaheuristics.generators.MultiobjectiveHillClimbingDistance.listTrace = new  
ArrayList<Float>() [protected]
```

Definición en la línea 33 del archivo [MultiobjectiveHillClimbingDistance.java](#).

Referenciado por [MultiobjectiveHillClimbingDistance\(\)](#).

#### 8.61.4.6 sizeNeighbors

```
final int metaheuristics.generators.MultiobjectiveHillClimbingDistance.sizeNeighbors = 10  
[static], [private]
```

Definición en la línea 35 del archivo [MultiobjectiveHillClimbingDistance.java](#).

Referenciado por [getSizeNeighbors\(\)](#).

#### 8.61.4.7 stateReferenceHC

```
State metaheuristics.generators.MultiobjectiveHillClimbingDistance.stateReferenceHC [protected]
```

Definición en la línea 28 del archivo [MultiobjectiveHillClimbingDistance.java](#).

Referenciado por [generate\(\)](#), [getReference\(\)](#), [getReferenceList\(\)](#) y [updateReference\(\)](#).

#### 8.61.4.8 strategy

```
StrategyType metaheuristics.generators.MultiobjectiveHillClimbingDistance.strategy [protected]
```

Definición en la línea 26 del archivo [MultiobjectiveHillClimbingDistance.java](#).

Referenciado por [generate\(\)](#).

#### 8.61.4.9 typeAcceptation

```
AcceptType metaheuristics.generators.MultiobjectiveHillClimbingDistance.typeAcceptation [protected]
```

Definición en la línea 25 del archivo [MultiobjectiveHillClimbingDistance.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.61.4.10 typeCandidate

```
CandidateType metaheuristics.generators.MultiobjectiveHillClimbingDistance.typeCandidate [protected]
```

Definición en la línea 27 del archivo [MultiobjectiveHillClimbingDistance.java](#).

Referenciado por [generate\(\)](#).

#### 8.61.4.11 visitedState

```
List<State> metaheuristics.generators.MultiobjectiveHillClimbingDistance.visitedState = new  
ArrayList<State>() [private]
```

Definición en la línea 34 del archivo [MultiobjectiveHillClimbingDistance.java](#).

Referenciado por [contain\(\)](#) y [updateReference\(\)](#).

#### 8.61.4.12 weight

```
float metaheuristics.generators.MultiobjectiveHillClimbingDistance.weight [protected]
```

Definición en la línea 32 del archivo [MultiobjectiveHillClimbingDistance.java](#).

Referenciado por [MultiobjectiveHillClimbingDistance\(\)](#) y [setWeight\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [MultiobjectiveHillClimbingDistance.java](#)

## 8.62 Referencia de la clase

### **metaheuristics.generators.MultiobjectiveHillClimbingRestart**

[MultiobjectiveHillClimbingRestart](#) - class that implements the Multiobjective Hill Climbing with Restart metaheuristic.

Diagrama de herencia de metaheuristics.generators.MultiobjectiveHillClimbingRestart

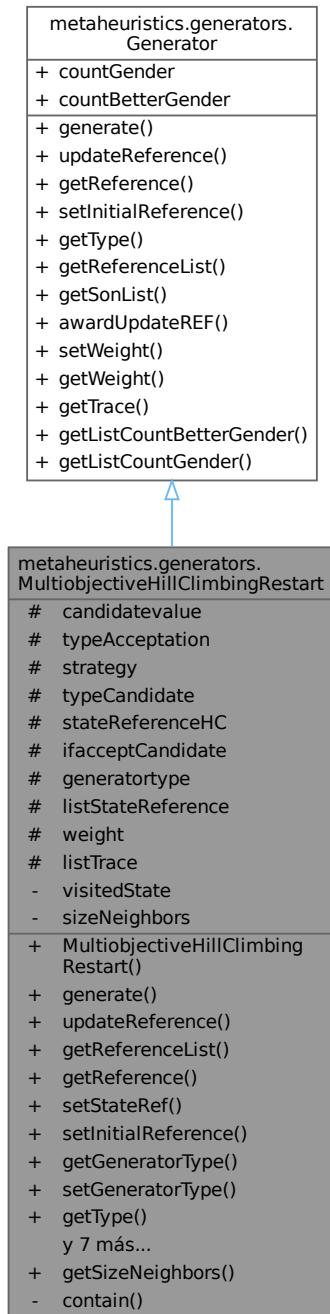
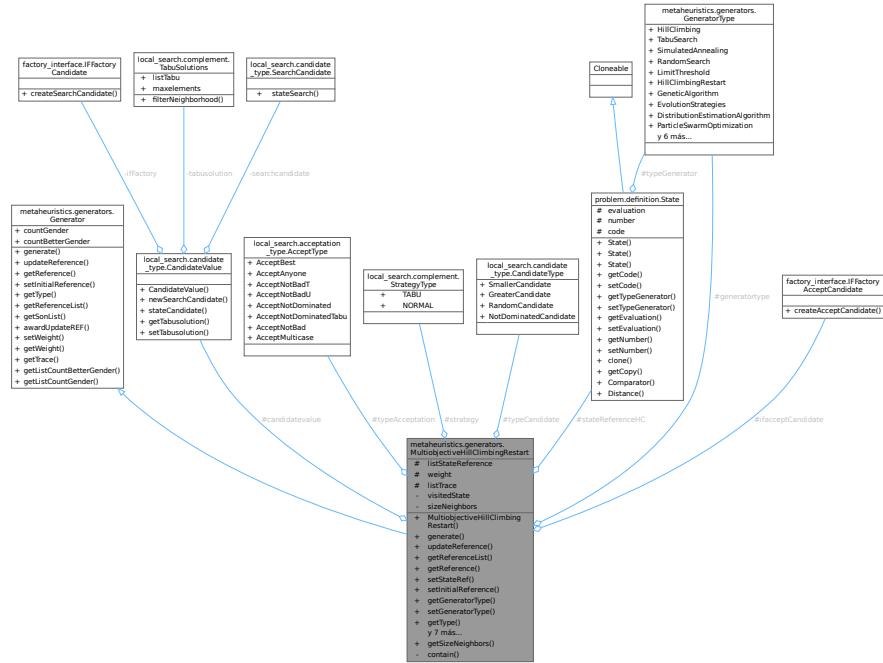


Diagrama de colaboración de metaheuristics.generators.MultiobjectiveHillClimbingRestart:



## Métodos públicos

- **MultiobjectiveHillClimbingRestart ()**  
*MultiobjectiveHillClimbingRestart - class that implements the Multiobjective Hill Climbing with Restart metaheuristic.*
  - **State generate (Integer operatornumber)** throws **IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException**  
*generate - generate a new state based on the operator number.*
  - void **updateReference (State stateCandidate, Integer countIterationsCurrent)** throws **IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException**  
*updateReference - update the reference state and the list of visited states.*
  - List< **State > getReferenceList ()**  
*getReferenceList - get the list of reference states.*
  - **State getReference ()**  
*getReference - get the reference state.*
  - void **setStateRef (State stateRef)**  
*setStateRef - set the reference state.*
  - void **setInitialReference (State stateInitialRef)**  
*setInitialReference - set the initial reference state.*
  - **GeneratorType getGeneratorType ()**  
*getGeneratorType - get the generator type.*
  - void **setGeneratorType (GeneratorType generatortype)**  
*setGeneratorType - set the generator type.*
  - **GeneratorType getType ()**  
*getType - get the generator type.*
  - List< **State > getSonList ()**

- `getSonList` - get the list of child states.
- boolean `awardUpdateREF (State stateCandidate)`  
*awardUpdateREF - award the update of the reference state.*
- float `getWeight ()`  
*getWeight - get the weight of the generator.*
- void `setWeight (float weight)`  
*setWeight - set the weight of the generator.*
- float[] `getTrace ()`  
*getTrace - get the trace of the generator.*
- int[] `getListCountBetterGender ()`  
*getListCountBetterGender - get the list of count of better solutions by gender.*
- int[] `getListCountGender ()`  
*getListCountGender - get the list of count of solutions by gender.*

### Métodos públicos estáticos

- static int `getSizeNeighbors ()`  
*Accessor for sizeNeighbors.*

### Atributos protegidos

- CandidateValue `candidatevalue`
- AcceptType `typeAcceptation`
- StrategyType `strategy`
- CandidateType `typeCandidate`
- State `stateReferenceHC`
- IFFactoryAcceptCandidate `ifacceptCandidate`
- GeneratorType `generatortype`
- List< State > `listStateReference` = new ArrayList<State>()
- float `weight`
- List< Float > `listTrace` = new ArrayList<Float>()

### Métodos privados

- boolean `contain (State state)`  
*contain - check if the state is contained in the visited states.*

### Atributos privados

- List< State > `visitedState` = new ArrayList<State>()

### Atributos estáticos privados

- static final int `sizeNeighbors` = 10

## Otros miembros heredados

### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- int `countGender`
- int `countBetterGender`

## 8.62.1 Descripción detallada

[MultiobjectiveHillClimbingRestart](#) - class that implements the Multiobjective Hill Climbing with Restart metaheuristic.

Definición en la línea 26 del archivo [MultiobjectiveHillClimbingRestart.java](#).

## 8.62.2 Documentación de constructores y destructores

### 8.62.2.1 [MultiobjectiveHillClimbingRestart\(\)](#)

```
metaheuristics.generators.MultiobjectiveHillClimbingRestart.MultiobjectiveHillClimbingRestart
() [inline]
```

[MultiobjectiveHillClimbingRestart](#) - class that implements the Multiobjective Hill Climbing with Restart metaheuristic.

Definición en la línea 52 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00052
00053     super();
00054     this.typeAcceptation = AcceptType.AcceptNotDominated;
00055     this.strategy = StrategyType.NORMAL;
00056     //Problem problem = Strategy.getStrategy().getProblem();
00057     this.typeCandidate = CandidateType.NotDominatedCandidate;
00058     this.candidatevalue = new CandidateValue();
00059     this.generatortype = GeneratorType.MultiobjectiveHillClimbingRestart;
00060     this.stateReferenceHC = new State();
00061     this.weight = 50;
00062     listTrace.add(weight);
00063 }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptType.AcceptNotDominated](#), [listTrace](#), [metaheuristics.generators.GeneratorType](#), [local\\_search.complement.StrategyType.NORMAL](#), [local\\_search.candidate\\_type.CandidateType.NotDominatedCandidate](#) y [weight](#).

## 8.62.3 Documentación de funciones miembro

### 8.62.3.1 [awardUpdateREF\(\)](#)

```
boolean metaheuristics.generators.MultiobjectiveHillClimbingRestart.awardUpdateREF (
    State stateCandidate) [inline]
```

[awardUpdateREF](#) - award the update of the reference state.

#### Parámetros

<code>stateCandidate</code>	
-----------------------------	--

#### Devuelve

return true if the reference state was updated, false otherwise

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 228 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00228
00229     // TODO Auto-generated method stub
00230     return false;
00231 }
```

### 8.62.3.2 contain()

```
boolean metaheuristics.generators.MultiobjectiveHillClimbingRestart.contains (
    State state) [inline], [private]
```

contains - check if the state is contained in the visited states.

#### Parámetros

state	
-------	--

#### Devuelve

return true if the state is contained, false otherwise

Definición en la línea 211 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00211         boolean found = false;
00212         for (Iterator<State> iter = visitedState.iterator(); iter.hasNext();) {
00213             State element = (State) iter.next();
00214             if(element.Comparator(state)==true){
00215                 found = true;
00216             }
00217         }
00218     }
00219     return found;
00220 }
```

Hace referencia a [problem.definition.State.Comparator\(\)](#) y [visitedState](#).

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas de esta función:

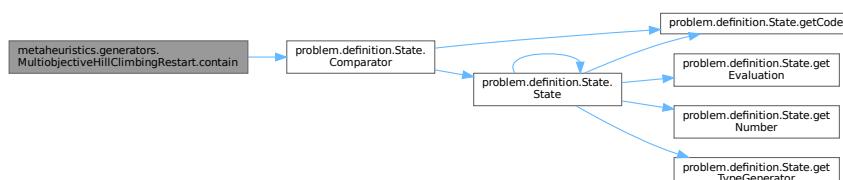


Gráfico de llamadas a esta función:



### 8.62.3.3 generate()

Generado por Doxygen

```
State metaheuristics.generators.MultiobjectiveHillClimbingRestart.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

## Parámetros

<i>operatornumber</i>	the operator number to use for generating the state
-----------------------	---

Devuelve

the generated state

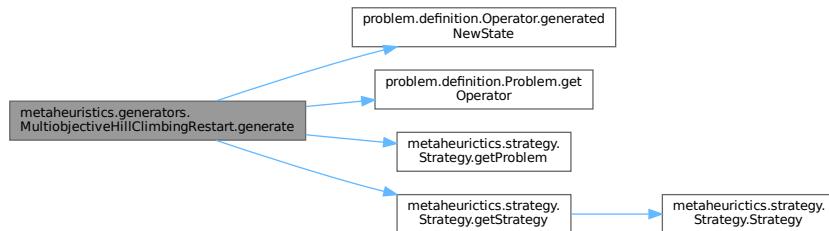
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 71 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00071
00072     {
00073         List<State> neighborhood =
00074             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC, operatornumber);
00075         State statecandidate = candidatevalue.stateCandidate(stateReferenceHC, typeCandidate,
00076             strategy, operatornumber, neighborhood);
00077         return statecandidate;
00078     }
```

Hace referencia a [candidatevalue](#), [problem.definition.Operator.generatedNewState\(\)](#), [problem.definition.Problem.getOperator\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [stateReferenceHC](#), [strategy](#) y [typeCandidate](#).

Gráfico de llamadas de esta función:



### 8.62.3.4 getGeneratorType()

```
GeneratorType metaheuristics.generators.MultiobjectiveHillClimbingRestart.getGeneratorType ()  
[inline]
```

getGeneratorType - get the generator type.

Devuelve

the generator type

Definición en la línea 175 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00175
00176     {
00177         return generatortype;
00178     }
```

Hace referencia a [generatortype](#).

### 8.62.3.5 getListCountBetterGender()

```
int[] metaheuristics.generators.MultiobjectiveHillClimbingRestart.getListCountBetterGender ()  
[inline]
```

getListCountBetterGender - get the list of count of better solutions by gender.

Devuelve

the list of count of better solutions by gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 275 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00275           {  
00276             // This generator doesn't maintain listCount arrays; return empty array to avoid nulls  
00277             return new int[0];  
00278         }
```

### 8.62.3.6 getListCountGender()

```
int[] metaheuristics.generators.MultiobjectiveHillClimbingRestart.getListCountGender () [inline]
```

getListCountGender - get the list of count of solutions by gender.

Devuelve

the list of count of solutions by gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 285 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00285           {  
00286             // This generator doesn't maintain listCount arrays; return empty array to avoid nulls  
00287             return new int[0];  
00288         }
```

### 8.62.3.7 getReference()

```
State metaheuristics.generators.MultiobjectiveHillClimbingRestart.getReference () [inline]
```

getReference - get the reference state.

Devuelve

the reference state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 150 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00150           {  
00151             return stateReferenceHC;  
00152         }
```

Hace referencia a [stateReferenceHC](#).

### 8.62.3.8 getReferenceList()

```
List< State > metaheuristics.generators.MultiobjectiveHillClimbingRestart.getReferenceList ()  
[inline]
```

getReferenceList - get the list of reference states.

Devuelve

the list of reference states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 140 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00140         {  
00141             listStateReference.add(stateReferenceHC.clone());  
00142             return listStateReference;  
00143     }
```

Hace referencia a [listStateReference](#) y [stateReferenceHC](#).

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.62.3.9 getSizeNeighbors()

```
int metaheuristics.generators.MultiobjectiveHillClimbingRestart.getSizeNeighbors () [inline],  
[static]
```

Accesor for sizeNeighbors.

Definición en la línea 44 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00044         {  
00045             return sizeNeighbors;  
00046     }
```

Hace referencia a [sizeNeighbors](#).

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.62.3.10 getSonList()

```
List< State > metaheuristics.generators.MultiobjectiveHillClimbingRestart.getSonList () [inline]  
getSonList - get the list of child states.
```

Devuelve

the list of child states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 201 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00201 {  
00202     // TODO Auto-generated method stub  
00203     return null;  
00204 }
```

### 8.62.3.11 getTrace()

```
float[] metaheuristics.generators.MultiobjectiveHillClimbingRestart.getTrace () [inline]  
getTrace - get the trace of the generator.
```

Devuelve

the trace of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 259 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00259 {  
00260     // TODO Auto-generated method stub  
00261     if (this.listTrace == null) return new float[0];  
00262     float[] arr = new float[this.listTrace.size()];  
00263     for (int i = 0; i < this.listTrace.size(); i++) {  
00264         Float v = this.listTrace.get(i);  
00265         arr[i] = (v == null) ? 0f : v.floatValue();  
00266     }  
00267     return arr;  
00268 }
```

### 8.62.3.12 getType()

```
GeneratorType metaheuristics.generators.MultiobjectiveHillClimbingRestart.getType () [inline]  
getType - get the generator type.
```

Devuelve

the generator type

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 192 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00192 {  
00193     return this.generatortype;  
00194 }
```

### 8.62.3.13 getWeight()

```
float metaheuristics.generators.MultiobjectiveHillClimbingRestart.getWeight () [inline]
```

getWeight - get the weight of the generator.

#### Devuelve

the weight of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 239 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00239          {
00240      // TODO Auto-generated method stub
00241      return 0;
00242 }
```

### 8.62.3.14 setGeneratorType()

```
void metaheuristics.generators.MultiobjectiveHillClimbingRestart.setGeneratorType (
    GeneratorType generatortype) [inline]
```

setGeneratorType - set the generator type.

#### Parámetros

<i>generatortype</i>	the generator type to set
----------------------	---------------------------

Definición en la línea 183 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00183          {
00184      this.generatortype = generatortype;
00185 }
```

Hace referencia a [generatortype](#).

### 8.62.3.15 setInitialReference()

```
void metaheuristics.generators.MultiobjectiveHillClimbingRestart.setInitialReference (
    State stateInitialRef) [inline]
```

setInitialReference - set the initial reference state.

#### Parámetros

<i>stateInitialRef</i>	the initial reference state to set
------------------------	------------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 167 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00167          {
00168      this.stateReferenceHC = stateInitialRef;
00169 }
```

### 8.62.3.16 setStateRef()

## Parámetros

<code>stateRef</code>	the reference state to set
-----------------------	----------------------------

Definición en la línea 158 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00158           {
00159             this.stateReferenceHC = stateRef;
00160           }
```

### 8.62.3.17 setWeight()

```
void metaheuristics.generators.MultiobjectiveHillClimbingRestart.setWeight (
    float weight) [inline]
```

`setWeight` - set the weight of the generator.

## Parámetros

<code>weight</code>	the weight to set
---------------------	-------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 249 del archivo [MultiobjectiveHillClimbingRestart.java](#).

```
00249           {
00250             // TODO Auto-generated method stub
00251           }
00252 }
```

Hace referencia a `weight`.

### 8.62.3.18 updateReference()

```
void metaheuristics.generators.MultiobjectiveHillClimbingRestart.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

`updateReference` - update the reference state and the list of visited states.

## Parámetros

<code>stateCandidate</code>	
<code>countIterationsCurrent</code>	

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 83 del archivo [MultiobjectiveHillClimbingRestart.java](#).

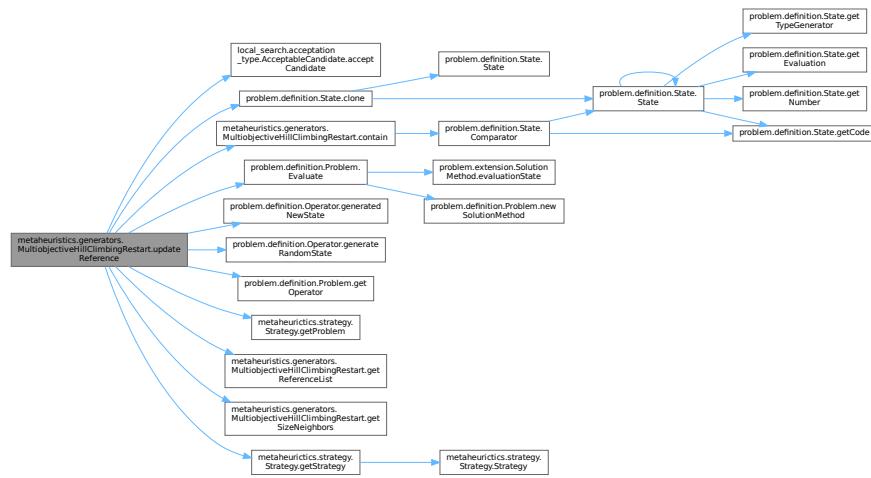
```
00083 {
00084   //Agregando la primera solución a la lista de soluciones no dominadas
00085   if(Strategy.getStrategy().listRefPoblacFinal.size() == 0) {
```

```

00087         Strategy.getStrategy().listRefPoblacFinal.add(stateReferenceHC.clone());
00088     }
00089
00090     ifacceptCandidate = new FactoryAcceptCandidate();
00091     AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00092     State lastState =
00093         Strategy.getStrategy().listRefPoblacFinal.get(Strategy.getStrategy().listRefPoblacFinal.size()-1);
00094     List<State> neighborhood =
00095         Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC,
00096         getSizeNeighbors());
00097     int i= 0;
00098
00099     Boolean accept = candidate.acceptCandidate(lastState, stateCandidate.clone());
00100
00101     if(accept.equals(true)){
00102         stateReferenceHC = stateCandidate.clone();
00103         visitedState = new ArrayList<State>();
00104         //tomar xc q pertenesca a la vecindad de xa
00105     }
00106     else{
00107         boolean stop = false;
00108         while (i < neighborhood.size()&& stop==false) {
00109             if (contain(neighborhood.get(i))==false) {
00110                 stateCandidate = neighborhood.get(i);
00111                 Strategy.getStrategy().getProblem().Evaluate(stateCandidate);
00112                 visitedState.add(stateCandidate);
00113                 accept = candidate.acceptCandidate(lastState, stateCandidate.clone());
00114                 stop=true;
00115             }
00116             i++;
00117         }
00118         while (stop == false) {
00119             stateCandidate =
00120                 Strategy.getStrategy().getProblem().getOperator().generateRandomState(1).get(0);
00121             if (contain(stateCandidate)==false) {
00122                 Strategy.getStrategy().getProblem().Evaluate(stateCandidate);
00123                 stop=true;
00124                 accept = candidate.acceptCandidate(lastState, stateCandidate.clone());
00125             }
00126         }
00127         if(accept.equals(true)){
00128             stateReferenceHC = stateCandidate.clone();
00129             visitedState = new ArrayList<State>();
00130             //tomar xc q pertenesca a la vecindad de xa
00131         }
00132     }
00133     getReferenceList();
00134 }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptableCandidate.acceptCandidate\(\)](#), [problem.definition.State.clone\(\)](#), [contain\(\)](#), [problem.definition.Problem.Evaluate\(\)](#), [problem.definition.Operator.generatedNewState\(\)](#), [problem.definition.Operator.generateRandomState\(\)](#), [problem.definition.Problem.getOperator\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [getReferenceList\(\)](#), [getSizeNeighbors\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [ifacceptCandidate](#), [metaheuristics.strategy.Strategy.listRefPoblacFinal\(\)](#), [stateReferenceHC](#), [typeAcceptation](#) y [visitedState](#).

Gráfico de llamadas de esta función:



## 8.62.4 Documentación de datos miembro

### 8.62.4.1 candidatevalue

`CandidateValue` `metaheuristics.generators.MultiobjectiveHillClimbingRestart.candidatevalue` [protected]

Definición en la línea 28 del archivo [MultiobjectiveHillClimbingRestart.java](#).

Referenciado por [generate\(\)](#).

### 8.62.4.2 generatortype

`GeneratorType` `metaheuristics.generators.MultiobjectiveHillClimbingRestart.generatortype` [protected]

Definición en la línea 34 del archivo [MultiobjectiveHillClimbingRestart.java](#).

Referenciado por [getGeneratorType\(\)](#) y [setGeneratorType\(\)](#).

### 8.62.4.3 ifacceptCandidate

`IFFactoryAcceptCandidate` `metaheuristics.generators.MultiobjectiveHillClimbingRestart.ifacceptCandidate` [protected]

Definición en la línea 33 del archivo [MultiobjectiveHillClimbingRestart.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.62.4.4 listStateReference

```
List<State> metaheuristics.generators.MultiobjectiveHillClimbingRestart.listStateReference =  
new ArrayList<State>() [protected]
```

Definición en la línea 35 del archivo [MultiobjectiveHillClimbingRestart.java](#).

Referenciado por [getReferenceList\(\)](#).

#### 8.62.4.5 listTrace

```
List<Float> metaheuristics.generators.MultiobjectiveHillClimbingRestart.listTrace = new Array←  
List<Float>() [protected]
```

Definición en la línea 37 del archivo [MultiobjectiveHillClimbingRestart.java](#).

Referenciado por [MultiobjectiveHillClimbingRestart\(\)](#).

#### 8.62.4.6 sizeNeighbors

```
final int metaheuristics.generators.MultiobjectiveHillClimbingRestart.sizeNeighbors = 10 [static],  
[private]
```

Definición en la línea 39 del archivo [MultiobjectiveHillClimbingRestart.java](#).

Referenciado por [getSizeNeighbors\(\)](#).

#### 8.62.4.7 stateReferenceHC

```
State metaheuristics.generators.MultiobjectiveHillClimbingRestart.stateReferenceHC [protected]
```

Definición en la línea 32 del archivo [MultiobjectiveHillClimbingRestart.java](#).

Referenciado por [generate\(\)](#), [getReference\(\)](#), [getReferenceList\(\)](#) y [updateReference\(\)](#).

#### 8.62.4.8 strategy

```
StrategyType metaheuristics.generators.MultiobjectiveHillClimbingRestart.strategy [protected]
```

Definición en la línea 30 del archivo [MultiobjectiveHillClimbingRestart.java](#).

Referenciado por [generate\(\)](#).

#### 8.62.4.9 typeAcceptation

```
AcceptType metaheuristics.generators.MultiobjectiveHillClimbingRestart.typeAcceptation [protected]
```

Definición en la línea 29 del archivo [MultiobjectiveHillClimbingRestart.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.62.4.10 typeCandidate

```
CandidateType metaheuristics.generators.MultiobjectiveHillClimbingRestart.typeCandidate [protected]
```

Definición en la línea 31 del archivo [MultiobjectiveHillClimbingRestart.java](#).

Referenciado por [generate\(\)](#).

#### 8.62.4.11 visitedState

```
List<State> metaheuristics.generators.MultiobjectiveHillClimbingRestart.visitedState = new  
ArrayList<State>() [private]
```

Definición en la línea 38 del archivo [MultiobjectiveHillClimbingRestart.java](#).

Referenciado por [contain\(\)](#) y [updateReference\(\)](#).

#### 8.62.4.12 weight

```
float metaheuristics.generators.MultiobjectiveHillClimbingRestart.weight [protected]
```

Definición en la línea 36 del archivo [MultiobjectiveHillClimbingRestart.java](#).

Referenciado por [MultiobjectiveHillClimbingRestart\(\)](#) y [setWeight\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [MultiobjectiveHillClimbingRestart.java](#)

## 8.63 Referencia de la clase **metaheuristics.generators.MultiobjectiveStochasticHillClimbing**

[MultiobjectiveStochasticHillClimbing](#) - class that implements the Multiobjective Stochastic Hill Climbing metaheuristic.

Diagrama de herencia de metaheuristics.generators.MultiobjectiveStochasticHillClimbing

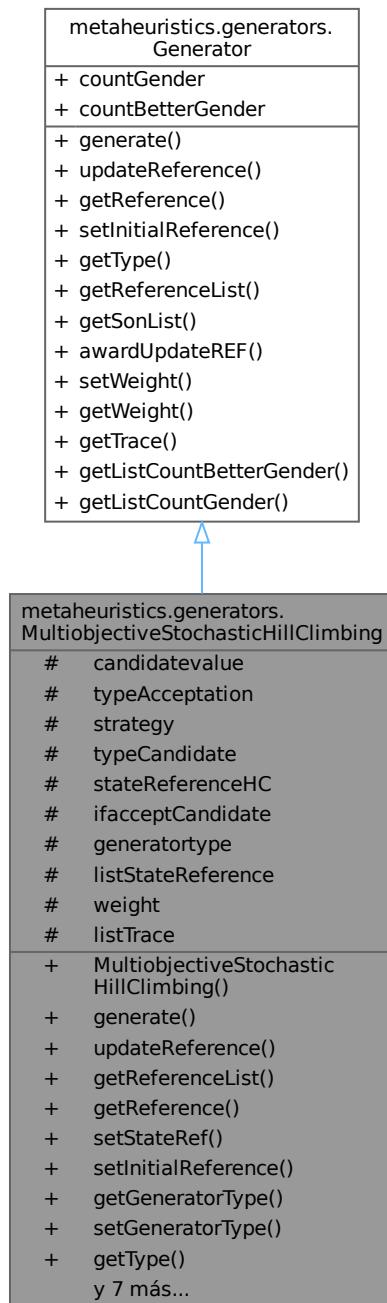
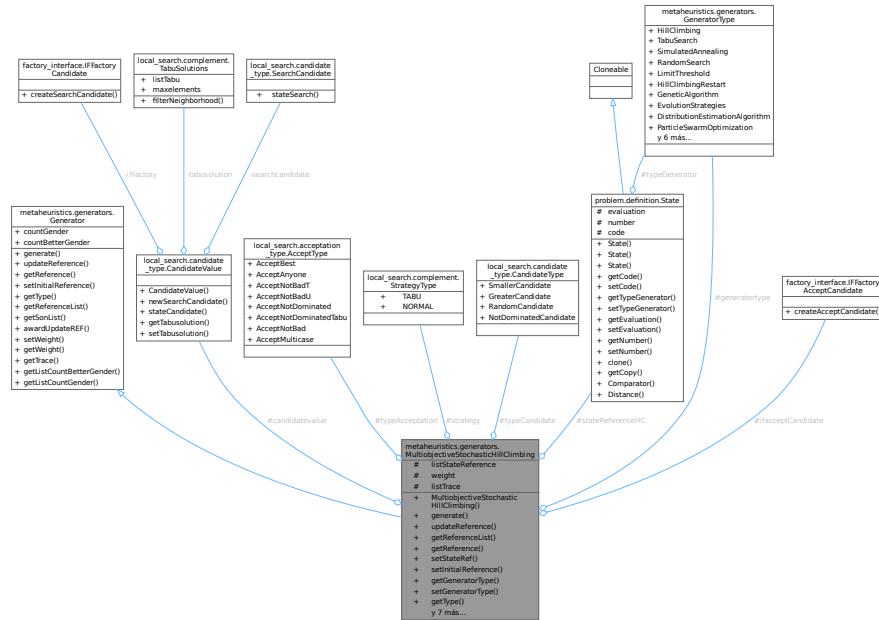


Diagrama de colaboración de metaheuristics.generators.MultiobjectiveStochasticHillClimbing:



## Métodos públicos

- **MultiobjectiveStochasticHillClimbing ()**

*MultiobjectiveStochasticHillClimbing - class that implements the Multiobjective Stochastic Hill Climbing metaheuristic.*

- **State generate (Integer operatornumber)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*generate - generate a new state based on the operator number.*

- **void updateReference (State stateCandidate, Integer countIterationsCurrent)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*updateReference - update the reference state and the list of visited states.*

- **List< State > getReferenceList ()**

*getReferenceList - get the list of reference states.*

- **State getReference ()**

*getReference - get the reference state.*

- **void setStateRef (State stateRef)**

*setStateRef - set the reference state.*

- **void setInitialReference (State statelInitialRef)**

*setInitialReference - set the initial reference state.*

- **GeneratorType getGeneratorType ()**

*getGeneratorType - get the generator type.*

- **void setGeneratorType (GeneratorType generatorType)**

*setGeneratorType - set the generator type.*

- **GeneratorType getType ()**

*getType - get the generator type.*

- **List< State > getSonList ()**

*getSonList - get the list of child states.*

- **boolean awardUpdateREF (State stateCandidate)**

- `float getWeight ()`  
*getWeight - get the weight of the generator.*
- `void setWeight (float weight)`  
*setWeight - set the weight of the generator.*
- `float[] getTrace ()`  
*getTrace - get the trace of the generator.*
- `int[] getListCountBetterGender ()`  
*getListCountBetterGender - get the list of counts of better gender solutions.*
- `int[] getListCountGender ()`  
*getListCountGender - get the list of counts of gender solutions.*

### Atributos protegidos

- `CandidateValue candidatevalue`
- `AcceptType typeAcceptation`
- `StrategyType strategy`
- `CandidateType typeCandidate`
- `State stateReferenceHC`
- `IFFactoryAcceptCandidate ifacceptCandidate`
- `GeneratorType generatortype`
- `List< State > listStateReference = new ArrayList<State>()`
- `float weight`
- `List< Float > listTrace = new ArrayList<Float>()`

### Otros miembros heredados

#### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- `int countGender`
- `int countBetterGender`

### 8.63.1 Descripción detallada

`MultiobjectiveStochasticHillClimbing` - class that implements the Multiobjective Stochastic Hill Climbing metaheuristic.

Definición en la línea 21 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

## 8.63.2 Documentación de constructores y destructores

### 8.63.2.1 MultiobjectiveStochasticHillClimbing()

```
metaheuristics.generators.MultiobjectiveStochasticHillClimbing.MultiobjectiveStochasticHillClimbing () [inline]
```

[MultiobjectiveStochasticHillClimbing](#) - class that implements the Multiobjective Stochastic Hill Climbing metaheuristic.

Definición en la línea 37 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00037         super();  
00038         this.typeAcceptation = AcceptType.AcceptNotDominated;  
00039         this.strategy = StrategyType.NORMAL;  
00040         this.typeCandidate = CandidateType.NotDominatedCandidate;  
00041         this.candidatevalue = new CandidateValue();  
00042         this.generatortype = GeneratorType.MultiobjectiveStochasticHillClimbing;  
00043         this.stateReferenceHC = new State();  
00044         this.weight = 50;  
00045         listTrace.add(weight);  
00046     }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptType.AcceptNotDominated](#), [listTrace](#), [metaheuristics.generators.GeneratorType](#), [local\\_search.complement.StrategyType.NORMAL](#), [local\\_search.candidate\\_type.CandidateType.NotDominatedCandidate](#) y [weight](#).

## 8.63.3 Documentación de funciones miembro

### 8.63.3.1 awardUpdateREF()

```
boolean metaheuristics.generators.MultiobjectiveStochasticHillClimbing.awardUpdateREF (  
    State stateCandidate) [inline]
```

[awardUpdateREF](#) - award the update of the reference state.

#### Parámetros

<a href="#">stateCandidate</a>	
--------------------------------	--

#### Devuelve

return true if the reference state was updated, false otherwise

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 153 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00153         // TODO Auto-generated method stub  
00154         return false;  
00155     }
```

### 8.63.3.2 generate()

---

```
State metaheuristics.generators.MultiobjectivestochasticHillClimbing.generate (  
    Generado por Doxygen  
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←  
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,  
NoSuchMethodException [inline]
```

## Parámetros

<i>operatornumber</i>	the operator number to use for generating the state
-----------------------	---

Devuelve

the generated state

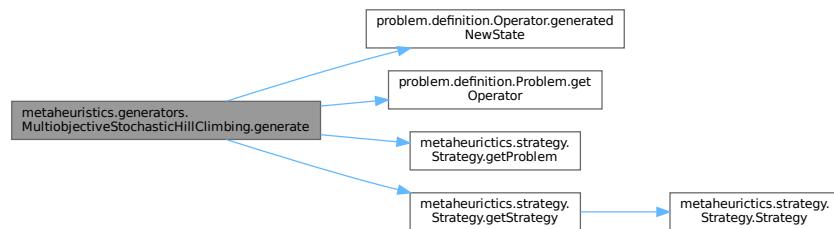
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 55 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00055
00056     {
00057         List<State> neighborhood =
00058             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC, operatornumber);
00059         State statecandidate = candidatevalue.stateCandidate(stateReferenceHC, typeCandidate,
00060             strategy, operatornumber, neighborhood);
00061         return statecandidate;
00062     }
```

Hace referencia a [candidatevalue](#), [problem.definition.Operator.generatedNewState\(\)](#), [problem.definition.Problem.getOperator\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [stateReferenceHC](#), [strategy](#) y [typeCandidate](#).

Gráfico de llamadas de esta función:



### 8.63.3.3 getGeneratorType()

```
GeneratorType metaheuristics.generators.MultiobjectiveStochasticHillClimbing.getGeneratorType()
() [inline]
```

getGeneratorType - get the generator type.

Devuelve

the generator type

Definición en la línea 116 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00116
00117     {
00118         return generatortype;
00119     }
```

Hace referencia a [generatortype](#).

#### 8.63.3.4 getListCountBetterGender()

```
int[] metaheuristics.generators.MultiobjectiveStochasticHillClimbing.getListCountBetterGender()
() [inline]
```

getListCountBetterGender - get the list of counts of better gender solutions.

Devuelve

the list of counts of better gender solutions

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 193 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00193                               {
00194         // TODO Auto-generated method stub
00195         return new int[0];
00196     }
```

#### 8.63.3.5 getListCountGender()

```
int[] metaheuristics.generators.MultiobjectiveStochasticHillClimbing.getListCountGender ()
[inline]
```

getListCountGender - get the list of counts of gender solutions.

Devuelve

the list of counts of gender solutions

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 203 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00203                               {
00204         // TODO Auto-generated method stub
00205         return new int[0];
00206     }
```

#### 8.63.3.6 getReference()

```
State metaheuristics.generators.MultiobjectiveStochasticHillClimbing.getReference () [inline]
```

getReference - get the reference state.

Devuelve

the reference state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 91 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00091                               {
00092         return stateReferenceHC;
00093     }
```

Hace referencia a [stateReferenceHC](#).

### 8.63.3.7 getReferenceList()

```
List< State > metaheuristics.generators.MultiobjectiveStochasticHillClimbing.getReferenceList()
() [inline]
```

getReferenceList - get the list of reference states.

Devuelve

the list of reference states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 81 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00081      {
00082          listStateReference.add( stateReferenceHC.clone() );
00083          return listStateReference;
00084      }
```

Hace referencia a [listStateReference](#) y [stateReferenceHC](#).

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.63.3.8 getSonList()

```
List< State > metaheuristics.generators.MultiobjectiveStochasticHillClimbing.getSonList () [inline]
```

getSonList - get the list of child states.

Devuelve

the list of child states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 142 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00142      {
00143          // TODO Auto-generated method stub
00144          return null;
00145      }
```

### 8.63.3.9 getTrace()

```
float[ ] metaheuristics.generators.MultiobjectiveStochasticHillClimbing.getTrace () [inline]
```

getTrace - get the trace of the generator.

Devuelve

the trace of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 183 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00183           {
00184     // TODO Auto-generated method stub
00185     return new float[0];
00186 }
```

### 8.63.3.10 getType()

```
GeneratorType metaheuristics.generators.MultiobjectiveStochasticHillClimbing.getType () [inline]
```

getType - get the generator type.

Devuelve

the generator type

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 133 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00133           {
00134     return this.generatortype;
00135 }
```

### 8.63.3.11 getWeight()

```
float metaheuristics.generators.MultiobjectiveStochasticHillClimbing.getWeight () [inline]
```

getWeight - get the weight of the generator.

Devuelve

the weight of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 163 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00163           {
00164     // TODO Auto-generated method stub
00165     return 0;
00166 }
```

---

### 8.63.3.12 setGeneratorType()

```
void metaheuristics.generators.MultiobjectiveStochasticHillClimbing.setGeneratorType (
    GeneratorType generatortype) [inline]
```

---

## Parámetros

<code>generatortype</code>	the generator type to set
----------------------------	---------------------------

Definición en la línea 124 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00124
00125     this.generatortype = generatortype;
00126 }
```

Hace referencia a [generatortype](#).

### 8.63.3.13 `setInitialReference()`

```
void metaheuristics.generators.MultiobjectiveStochasticHillClimbing.setInitialReference (
    State stateInitialRef) [inline]
```

`setInitialReference` - set the initial reference state.

## Parámetros

<code>stateInitialRef</code>	the initial reference state to set
------------------------------	------------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 108 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00108
00109     this.stateReferenceHC = stateInitialRef;
00110 }
```

### 8.63.3.14 `setStateRef()`

```
void metaheuristics.generators.MultiobjectiveStochasticHillClimbing.setStateRef (
    State stateRef) [inline]
```

`setStateRef` - set the reference state.

## Parámetros

<code>stateRef</code>	the reference state to set
-----------------------	----------------------------

Definición en la línea 99 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00099
00100     this.stateReferenceHC = stateRef;
00101 }
```

### 8.63.3.15 `setWeight()`

```
void metaheuristics.generators.MultiobjectiveStochasticHillClimbing.setWeight (
    float weight) [inline]
```

`setWeight` - set the weight of the generator.

## Parámetros

<i>weight</i>	the weight to set
---------------	-------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 173 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00173           {
00174     // TODO Auto-generated method stub
00175
00176 }
```

Hace referencia a [weight](#).

### 8.63.3.16 updateReference()

```
void metaheuristics.generators.MultiobjectiveStochasticHillClimbing.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

`updateReference` - update the reference state and the list of visited states.

## Parámetros

<i>stateCandidate</i>	
<i>countIterationsCurrent</i>	

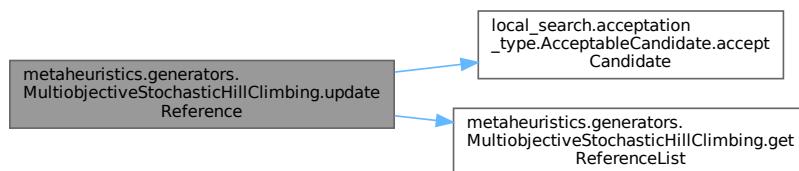
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 67 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

```
00067 {
00068     ifacceptCandidate = new FactoryAcceptCandidate();
00069     AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00070     Boolean accept = candidate.acceptCandidate(stateReferenceHC, stateCandidate);
00071     if(accept.equals(true))
00072         stateReferenceHC = stateCandidate.clone();
00073     getReferenceList();
00074 }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptableCandidate.acceptCandidate\(\)](#), [getReferenceList\(\)](#), [ifacceptCandidate](#), [stateReferenceHC](#) y [typeAcceptation](#).

Gráfico de llamadas de esta función:



## 8.63.4 Documentación de datos miembro

### 8.63.4.1 candidatevalue

```
CandidateValue metaheuristics.generators.MultiobjectiveStochasticHillClimbing.candidatevalue  
[protected]
```

Definición en la línea 23 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

Referenciado por [generate\(\)](#).

### 8.63.4.2 generatortype

```
GeneratorType metaheuristics.generators.MultiobjectiveStochasticHillClimbing.generatortype  
[protected]
```

Definición en la línea 29 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

Referenciado por [getGeneratorType\(\)](#) y [setGeneratorType\(\)](#).

### 8.63.4.3 ifacceptCandidate

```
IFFactoryAcceptCandidate metaheuristics.generators.MultiobjectiveStochasticHillClimbing.ifacceptCandidate  
[protected]
```

Definición en la línea 28 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

Referenciado por [updateReference\(\)](#).

### 8.63.4.4 listStateReference

```
List<State> metaheuristics.generators.MultiobjectiveStochasticHillClimbing.listStateReference  
= new ArrayList<State>() [protected]
```

Definición en la línea 30 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

Referenciado por [getReferenceList\(\)](#).

### 8.63.4.5 listTrace

```
List<Float> metaheuristics.generators.MultiobjectiveStochasticHillClimbing.listTrace = new  
ArrayList<Float>() [protected]
```

Definición en la línea 32 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

Referenciado por [MultiobjectiveStochasticHillClimbing\(\)](#).

#### 8.63.4.6 stateReferenceHC

```
State metaheuristics.generators.MultiobjectiveStochasticHillClimbing.stateReferenceHC [protected]
```

Definición en la línea 27 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

Referenciado por [generate\(\)](#), [getReference\(\)](#), [getReferenceList\(\)](#) y [updateReference\(\)](#).

#### 8.63.4.7 strategy

```
StrategyType metaheuristics.generators.MultiobjectiveStochasticHillClimbing.strategy [protected]
```

Definición en la línea 25 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

Referenciado por [generate\(\)](#).

#### 8.63.4.8 typeAcceptation

```
AcceptType metaheuristics.generators.MultiobjectiveStochasticHillClimbing.typeAcceptation [protected]
```

Definición en la línea 24 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.63.4.9 typeCandidate

```
CandidateType metaheuristics.generators.MultiobjectiveStochasticHillClimbing.typeCandidate [protected]
```

Definición en la línea 26 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

Referenciado por [generate\(\)](#).

#### 8.63.4.10 weight

```
float metaheuristics.generators.MultiobjectiveStochasticHillClimbing.weight [protected]
```

Definición en la línea 31 del archivo [MultiobjectiveStochasticHillClimbing.java](#).

Referenciado por [MultiobjectiveStochasticHillClimbing\(\)](#) y [setWeight\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [MultiobjectiveStochasticHillClimbing.java](#)

## 8.64 Referencia de la clase

### metaheuristics.generators.MultiobjectiveTabuSearch

[MultiobjectiveTabuSearch](#) - class that implements the Multiobjective Tabu Search metaheuristic.

Diagrama de herencia de metaheuristics.generators.MultiobjectiveTabuSearch

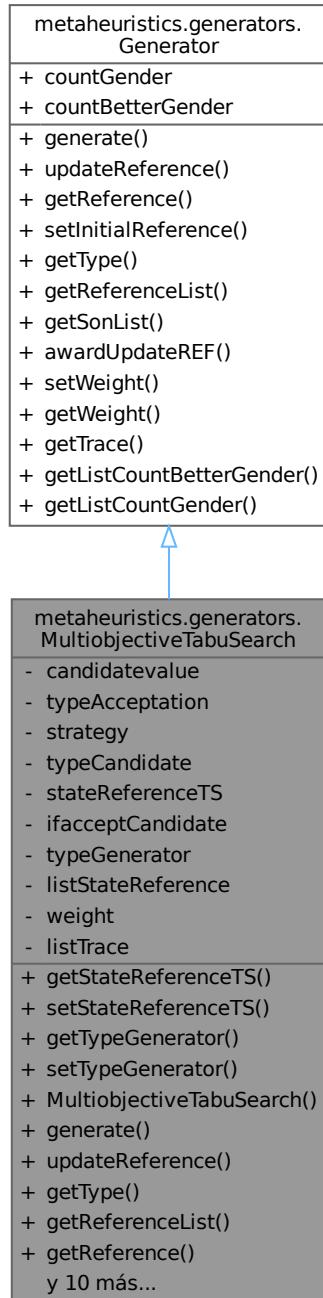
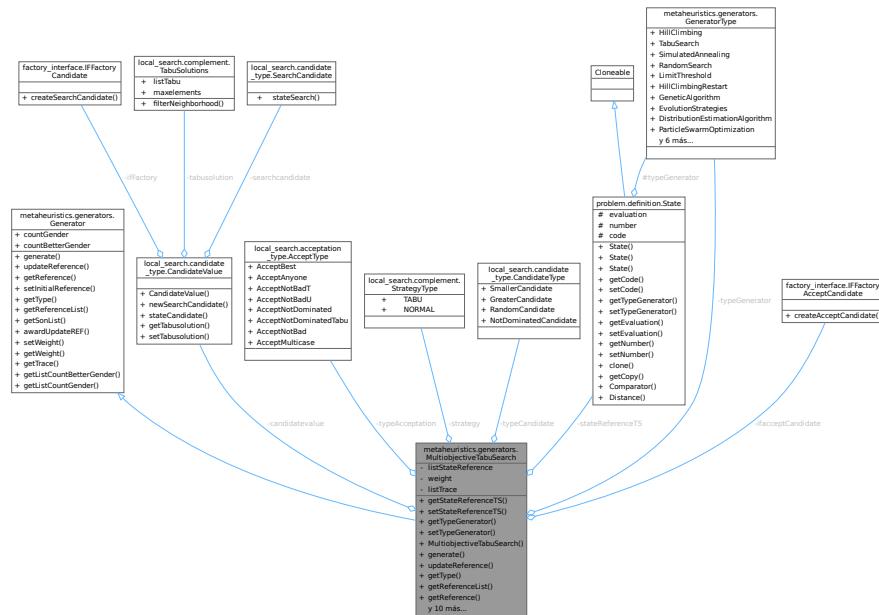


Diagrama de colaboración de metaheuristics.generators.MultiobjectiveTabuSearch:



## Métodos públicos

- **State getStateReferenceTS ()**  
`getStateReferenceTS` - get the reference state for Tabu Search.
- **void setStateReferenceTS (State stateReferenceTS)**  
`setStateReferenceTS` - set the reference state for Tabu Search.
- **GeneratorType getTypeGenerator ()**  
`getTypeGenerator` - get the type of the generator.
- **void setTypeGenerator (GeneratorType typeGenerator)**  
`setTypeGenerator` - set the type of the generator.
- **MultiobjectiveTabuSearch ()**  
`MultiobjectiveTabuSearch` - class that implements the Multiobjective Tabu Search metaheuristic.
- **State generate (Integer operatornumber)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`  
`generate` - generate a new state based on the operator number.
- **void updateReference (State stateCandidate, Integer countIterationsCurrent)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`  
`updateReference` - update the reference state and the list of visited states.
- **GeneratorType getType ()**  
`getType` - get the type of the generator.
- **List< State > getReferenceList ()**  
`getReferenceList` - get the list of reference states.
- **State getReference ()**  
`getReference` - get the reference state.
- **void setInitialReference (State statInitialRef)**  
`setInitialReference` - set the initial reference state.
- **void setStateRef (State stateRef)**

- setStateRef - set the reference state.*
- List< State > [getSonList \(\)](#)  
    *getSonList - get the list of child states.*
- void [setTypeCandidate \(CandidateType typeCandidate\)](#)  
    *setTypeCandidate - set the type of candidate.*
- boolean [awardUpdateREF \(State stateCandidate\)](#)  
    *awardUpdateREF - award the update of the reference state.*
- float [getWeight \(\)](#)  
    *getWeight - get the weight of the solution.*
- void [setWeight \(float weight\)](#)  
    *setWeight - set the weight of the generator.*
- int[] [getListCountBetterGender \(\)](#)  
    *getListCountBetterGender - get the list of counts of better solutions by gender.*
- int[] [getListCountGender \(\)](#)  
    *getListCountGender - get the list of counts of solutions by gender.*
- float[] [getTrace \(\)](#)  
    *getTrace - get the trace of the solution.*

### Atributos privados

- CandidateValue [candidatevalue](#)
- AcceptType [typeAcceptation](#)
- StrategyType [strategy](#)
- CandidateType [typeCandidate](#)
- State [stateReferenceTS](#)
- IFFactoryAcceptCandidate [ifacceptCandidate](#)
- GeneratorType [typeGenerator](#)
- List< State > [listStateReference = new ArrayList<State>\(\)](#)
- float [weight](#)
- List< Float > [listTrace = new ArrayList<Float>\(\)](#)

### Otros miembros heredados

#### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- int [countGender](#)
- int [countBetterGender](#)

### 8.64.1 Descripción detallada

[MultiobjectiveTabuSearch](#) - class that implements the Multiobjective Tabu Search metaheuristic.

Definición en la línea [25](#) del archivo [MultiobjectiveTabuSearch.java](#).

## 8.64.2 Documentación de constructores y destructores

### 8.64.2.1 MultiobjectiveTabuSearch()

```
metaheuristics.generators.MultiobjectiveTabuSearch.MultiobjectiveTabuSearch () [inline]
```

**MultiobjectiveTabuSearch** - class that implements the Multiobjective Tabu Search metaheuristic.

Definición en la línea 73 del archivo [MultiobjectiveTabuSearch.java](#).

```
00073
00074     super();
00075     this.typeAcceptation = AcceptType.AcceptNotDominatedTabu;
00076     this.strategy = StrategyType.TABU;
00077     // Use problem API directly when needed; avoid unused local variable
00078     this.typeCandidate = CandidateType.RandomCandidate;
00079     this.candidatevalue = new CandidateValue();
00080     this.typeGenerator = GeneratorType.MultiobjectiveTabuSearch;
00081     this.weight = 50;
00082     listTrace.add(weight);
00083 }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptType.AcceptNotDominatedTabu](#), [listTrace](#), [metaheuristics.generators.Generator](#), [local\\_search.candidate\\_type.CandidateType.RandomCandidate](#), [local\\_search.complement.StrategyType.TABU](#) y [weight](#).

## 8.64.3 Documentación de funciones miembro

### 8.64.3.1 awardUpdateREF()

```
boolean metaheuristics.generators.MultiobjectiveTabuSearch.awardUpdateREF (
    State stateCandidate) [inline]
```

**awardUpdateREF** - award the update of the reference state.

#### Parámetros

<i>stateCandidate</i>	
-----------------------	--

#### Devuelve

return true if the reference state was updated, false otherwise

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 212 del archivo [MultiobjectiveTabuSearch.java](#).

```
00212
00213     // TODO Auto-generated method stub
00214     return false;
00215 }
```

### 8.64.3.2 generate()

```
State metaheuristics.generators.MultiobjectiveTabuSearch.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
Generado por Doxygen
NoSuchMethodException [inline]
```

**generate** - generate a new state based on the operator number.

## Parámetros

<i>operatornumber</i>	the operator number to use for generating the state
-----------------------	---

## Devuelve

the generated state

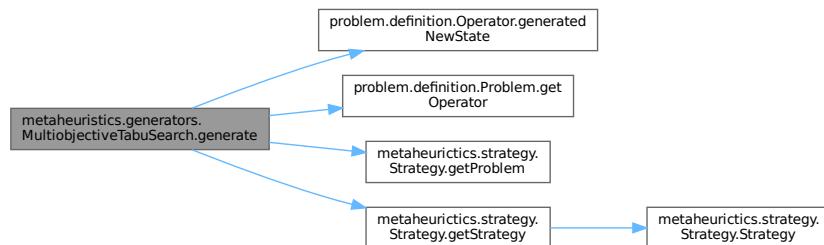
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 91 del archivo [MultiobjectiveTabuSearch.java](#).

```
00091
00092     {
00093         //Devuelve la lista de soluciones no dominadas de todos los vecinos posibles de
00094         //stateReferenceTS que no se encuentran en la lista Tabu
00095         List<State> neighborhood =
00096             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceTS, operatornumber);
00097         //Se escoge uno aleatoriamente como vecino con RandomCandidate
00098         State statecandidate = candidatevalue.stateCandidate(stateReferenceTS, typeCandidate,
00099             strategy, operatornumber, neighborhood);
00100         return statecandidate;
00101     }
```

Hace referencia a [candidatevalue](#), [problem.definition.Operator.generatedNewState\(\)](#), [problem.definition.Problem.getOperator\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [stateReferenceTS](#), [strategy](#) y [typeCandidate](#).

Gráfico de llamadas de esta función:



### 8.64.3.3 getListCountBetterGender()

```
int[] metaheuristics.generators.MultiobjectiveTabuSearch.getListCountBetterGender () [inline]

getListCountBetterGender - get the list of counts of better solutions by gender.
```

## Devuelve

the list of counts of better solutions by gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 242 del archivo [MultiobjectiveTabuSearch.java](#).

```
00242
00243     {
00244         // TODO Auto-generated method stub
00245         // This generator doesn't maintain listCount arrays; return empty array to avoid nulls
00246         return new int[0];
00247     }
```

#### 8.64.3.4 getListCountGender()

```
int[] metaheuristics.generators.MultiobjectiveTabuSearch.getListCountGender () [inline]
getListCountGender - get the list of counts of solutions by gender.
```

Devuelve

the list of counts of solutions by gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 253 del archivo [MultiobjectiveTabuSearch.java](#).

```
00253             {
00254     // TODO Auto-generated method stub
00255     // This generator doesn't maintain listCount arrays; return empty array to avoid nulls
00256     return new int[0];
00257 }
```

#### 8.64.3.5 getReference()

```
State metaheuristics.generators.MultiobjectiveTabuSearch.getReference () [inline]
getReference - get the reference state.
```

Devuelve

the reference state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 167 del archivo [MultiobjectiveTabuSearch.java](#).

```
00167             {
00168     return stateReferenceTS;
00169 }
```

Hace referencia a [stateReferenceTS](#).

#### 8.64.3.6 getReferenceList()

```
List< State > metaheuristics.generators.MultiobjectiveTabuSearch.getReferenceList () [inline]
getReferenceList - get the list of reference states.
```

Devuelve

the list of reference states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 157 del archivo [MultiobjectiveTabuSearch.java](#).

```
00157             {
00158     listStateReference.add(stateReferenceTS);
00159     return listStateReference;
00160 }
```

Hace referencia a [listStateReference](#) y [stateReferenceTS](#).

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.64.3.7 getSonList()

```
List< State > metaheuristics.generators.MultiobjectiveTabuSearch.getSonList () [inline]
```

getSonList - get the list of child states.

**Devuelve**

the list of child states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 193 del archivo [MultiobjectiveTabuSearch.java](#).

```
00193                               {
00194         // TODO Auto-generated method stub
00195         return null;
00196     }
```

### 8.64.3.8 getStateReferenceTS()

```
State metaheuristics.generators.MultiobjectiveTabuSearch.getStateReferenceTS () [inline]
```

getStateReferenceTS - get the reference state for Tabu Search.

**Devuelve**

the reference state for Tabu Search

Definición en la línea 42 del archivo [MultiobjectiveTabuSearch.java](#).

```
00042                               {
00043         return stateReferenceTS;
00044     }
```

Hace referencia a [stateReferenceTS](#).

### 8.64.3.9 getTrace()

```
float[] metaheuristics.generators.MultiobjectiveTabuSearch.getTrace () [inline]
```

getTrace - get the trace of the solution.

**Devuelve**

the trace of the solution

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 264 del archivo [MultiobjectiveTabuSearch.java](#).

```
00264                               {
00265         // TODO Auto-generated method stub
00266         if \(this.listTrace == null\) return new float\[0\];
00267         float[] arr = new float\[this.listTrace.size\(\)\];
00268         for \(int i = 0; i < this.listTrace.size\(\); i++\) {
00269             Float v = this.listTrace.get\(i\);
00270             arr\[i\] = \(v == null\) ? Of : v.floatValue\(\);
00271         }
00272         return arr;
00273     }
```

### 8.64.3.10 getType()

```
GeneratorType metaheuristics.generators.MultiobjectiveTabuSearch.getType () [inline]
```

getType - get the type of the generator.

Devuelve

the type of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 148 del archivo [MultiobjectiveTabuSearch.java](#).

```
00148                     {
00149             return this.typeGenerator;
00150         }
```

### 8.64.3.11 getTypeGenerator()

```
GeneratorType metaheuristics.generators.MultiobjectiveTabuSearch.getTypeGenerator () [inline]
```

getTypeGenerator - get the type of the generator.

Devuelve

the type of the generator

Definición en la línea 58 del archivo [MultiobjectiveTabuSearch.java](#).

```
00058                     {
00059             return typeGenerator;
00060         }
```

Hace referencia a [typeGenerator](#).

### 8.64.3.12 getWeight()

```
float metaheuristics.generators.MultiobjectiveTabuSearch.getWeight () [inline]
```

getWeight - get the weight of the solution.

Devuelve

the weight of the solution

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 222 del archivo [MultiobjectiveTabuSearch.java](#).

```
00222                     {
00223             // TODO Auto-generated method stub
00224             return this.weight;
00225         }
```

---

### 8.64.3.13 setInitialReference()

Generado por Doxygen

```
void metaheuristics.generators.MultiobjectiveTabuSearch.setInitialReference (
    State stateInitialRef) [inline]
```

## Parámetros

<i>stateInitialRef</i>	the initial reference state to set
------------------------	------------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 176 del archivo [MultiobjectiveTabuSearch.java](#).

```
00176          {
00177      this.stateReferenceTS = stateInitialRef;
00178 }
```

### 8.64.3.14 setStateRef()

```
void metaheuristics.generators.MultiobjectiveTabuSearch.setStateRef (
    State stateRef) [inline]
```

setStateRef - set the reference state.

## Parámetros

<i>stateRef</i>	
-----------------	--

Definición en la línea 184 del archivo [MultiobjectiveTabuSearch.java](#).

```
00184          {
00185      this.stateReferenceTS = stateRef;
00186 }
```

### 8.64.3.15 setStateReferenceTS()

```
void metaheuristics.generators.MultiobjectiveTabuSearch.setStateReferenceTS (
    State stateReferenceTS) [inline]
```

setStateReferenceTS - set the reference state for Tabu Search.

## Parámetros

<i>stateReferenceTS</i>	
-------------------------	--

Definición en la línea 50 del archivo [MultiobjectiveTabuSearch.java](#).

```
00050          {
00051      this.stateReferenceTS = stateReferenceTS;
00052 }
```

Hace referencia a [stateReferenceTS](#).

### 8.64.3.16 setTypeCandidate()

```
void metaheuristics.generators.MultiobjectiveTabuSearch.setTypeCandidate (
    CandidateType typeCandidate) [inline]
```

setTypeCandidate - set the type of candidate.

**Parámetros**

<i>typeCandidate</i>	the type of candidate to set
----------------------	------------------------------

Definición en la línea 202 del archivo [MultiobjectiveTabuSearch.java](#).

```
00202
00203     this.typeCandidate = typeCandidate;
00204 }
```

Hace referencia a [typeCandidate](#).

**8.64.3.17 setTypeGenerator()**

```
void metaheuristics.generators.MultiobjectiveTabuSearch.setTypeGenerator (
    GeneratorType typeGenerator) [inline]
```

`setTypeGenerator` - set the type of the generator.

**Parámetros**

<i>typeGenerator</i>	
----------------------	--

Definición en la línea 66 del archivo [MultiobjectiveTabuSearch.java](#).

```
00066
00067     this.typeGenerator = typeGenerator;
00068 }
```

Hace referencia a [typeGenerator](#).

**8.64.3.18 setWeight()**

```
void metaheuristics.generators.MultiobjectiveTabuSearch.setWeight (
    float weight) [inline]
```

`setWeight` - set the weight of the generator.

**Parámetros**

<i>weight</i>	the weight to set
---------------	-------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 232 del archivo [MultiobjectiveTabuSearch.java](#).

```
00232
00233     // TODO Auto-generated method stub
00234     this.weight = weight;
00235 }
```

Hace referencia a [weight](#).

**8.64.3.19 updateReference()**

---

Generado por Doxygen

```
void metaheuristics.generators.MultiobjectiveTabuSearch.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
```

## Parámetros

<i>stateCandidate</i>	
<i>countIterationsCurrent</i>	

Reimplementado de [metaheuristics.generators.Generator](#).

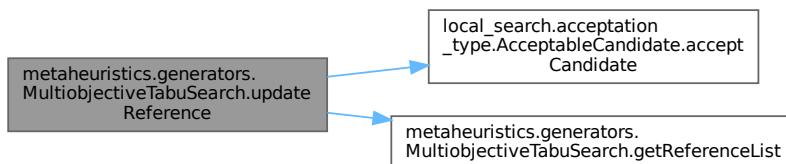
Definición en la línea 105 del archivo [MultiobjectiveTabuSearch.java](#).

```

00105
00106     ifacceptCandidate = new FactoryAcceptCandidate();
00107     AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00108     Boolean accept = candidate.acceptCandidate(stateReferenceTS, stateCandidate);
00109     if(accept.equals(true))
00110         stateReferenceTS = stateCandidate;
00111
00112     if (strategy.equals(StrategyType.TABU) && accept.equals(true)) {
00113         if (TabuSolutions.listTabu.size() < TabuSolutions.maxelements) {
00114             Boolean find = false;
00115             int count = 0;
00116             while ((TabuSolutions.listTabu.size() > count) && (find.equals(false))) {
00117                 if (TabuSolutions.listTabu.get(count).equals(stateCandidate)) {
00118                     find = true;
00119                 }
00120                 count++;
00121             }
00122             if (find.equals(false)) {
00123                 TabuSolutions.listTabu.add(stateCandidate);
00124             }
00125         } else {
00126             TabuSolutions.listTabu.remove(0);
00127             Boolean find = false;
00128             int count = 0;
00129             while (TabuSolutions.listTabu.size() > count && find.equals(false)) {
00130                 if (TabuSolutions.listTabu.get(count).equals(stateCandidate)) {
00131                     find = true;
00132                 }
00133                 count++;
00134             }
00135             if (find.equals(false)) {
00136                 TabuSolutions.listTabu.add(stateCandidate);
00137             }
00138         }
00139     }
00140     getReferenceList();
00141 }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptableCandidate.acceptCandidate\(\)](#), [getReferenceList\(\)](#), [ifacceptCandidate](#), [local\\_search.complement.TabuSolutions.listTabu](#), [local\\_search.complement.TabuSolutions.maxelements](#), [stateReferenceTS](#), [strategy](#), [local\\_search.complement.StrategyType.TABU](#) y [typeAcceptation](#).

Gráfico de llamadas de esta función:



## 8.64.4 Documentación de datos miembro

### 8.64.4.1 candidatevalue

```
CandidateValue metaheuristics.generators.MultiobjectiveTabuSearch.candidatevalue [private]
```

Definición en la línea 27 del archivo [MultiobjectiveTabuSearch.java](#).

Referenciado por [generate\(\)](#).

### 8.64.4.2 ifacceptCandidate

```
IFFactoryAcceptCandidate metaheuristics.generators.MultiobjectiveTabuSearch.ifacceptCandidate [private]
```

Definición en la línea 32 del archivo [MultiobjectiveTabuSearch.java](#).

Referenciado por [updateReference\(\)](#).

### 8.64.4.3 listStateReference

```
List<State> metaheuristics.generators.MultiobjectiveTabuSearch.listStateReference = new ArrayList<State>()  
[private]
```

Definición en la línea 34 del archivo [MultiobjectiveTabuSearch.java](#).

Referenciado por [getReferenceList\(\)](#).

### 8.64.4.4 listTrace

```
List<Float> metaheuristics.generators.MultiobjectiveTabuSearch.listTrace = new ArrayList<Float>()  
[private]
```

Definición en la línea 36 del archivo [MultiobjectiveTabuSearch.java](#).

Referenciado por [MultiobjectiveTabuSearch\(\)](#).

### 8.64.4.5 stateReferenceTS

```
State metaheuristics.generators.MultiobjectiveTabuSearch.stateReferenceTS [private]
```

Definición en la línea 31 del archivo [MultiobjectiveTabuSearch.java](#).

Referenciado por [generate\(\)](#), [getReference\(\)](#), [getReferenceList\(\)](#), [getStateReferenceTS\(\)](#), [setStateReferenceTS\(\)](#) y [updateReference\(\)](#).

#### 8.64.4.6 strategy

```
StrategyType metaheuristics.generators.MultiobjectiveTabuSearch.strategy [private]
```

Definición en la línea 29 del archivo [MultiobjectiveTabuSearch.java](#).

Referenciado por [generate\(\)](#) y [updateReference\(\)](#).

#### 8.64.4.7 typeAcceptation

```
AcceptType metaheuristics.generators.MultiobjectiveTabuSearch.typeAcceptation [private]
```

Definición en la línea 28 del archivo [MultiobjectiveTabuSearch.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.64.4.8 typeCandidate

```
CandidateType metaheuristics.generators.MultiobjectiveTabuSearch.typeCandidate [private]
```

Definición en la línea 30 del archivo [MultiobjectiveTabuSearch.java](#).

Referenciado por [generate\(\)](#) y [setTypeCandidate\(\)](#).

#### 8.64.4.9 typeGenerator

```
GeneratorType metaheuristics.generators.MultiobjectiveTabuSearch.typeGenerator [private]
```

Definición en la línea 33 del archivo [MultiobjectiveTabuSearch.java](#).

Referenciado por [getTypeGenerator\(\)](#) y [setTypeGenerator\(\)](#).

#### 8.64.4.10 weight

```
float metaheuristics.generators.MultiobjectiveTabuSearch.weight [private]
```

Definición en la línea 35 del archivo [MultiobjectiveTabuSearch.java](#).

Referenciado por [MultiobjectiveTabuSearch\(\)](#) y [setWeight\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [MultiobjectiveTabuSearch.java](#)

## 8.65 Referencia de la clase problem.extension.MultiObjetivoPuro

[MultiObjetivoPuro](#).

Diagrama de herencia de problem.extension.MultiObjetivoPuro

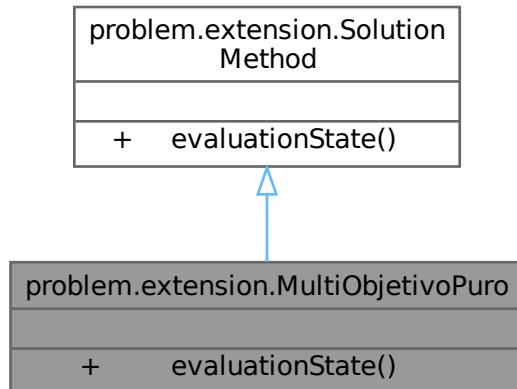
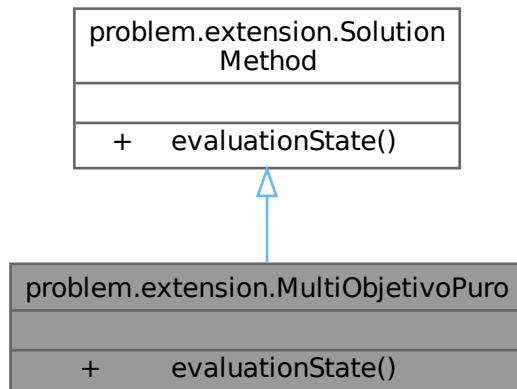


Diagrama de colaboración de problem.extension.MultiObjetivoPuro:



### Métodos públicos

- void [evaluationState \(State state\)](#)

*evaluationState*

## 8.65.1 Descripción detallada

[MultiObjetivoPuro](#).

Método de evaluación multiobjetivo puro: calcula el vector de evaluaciones sin agregación (mantiene cada función objetivo por separado).

Definición en la línea 17 del archivo [MultiObjetivoPuro.java](#).

## 8.65.2 Documentación de funciones miembro

### 8.65.2.1 evaluationState()

```
void problem.extension.MultiObjetivoPuro.evaluationState (
    State state) [inline]
```

evaluationState

Rellena la lista de evaluaciones del estado con los valores correspondientes a cada función objetivo, adaptando según si la función o el problema es de maximizar/minimizar.

#### Parámetros

<code>state</code>	estado a evaluar
--------------------	------------------

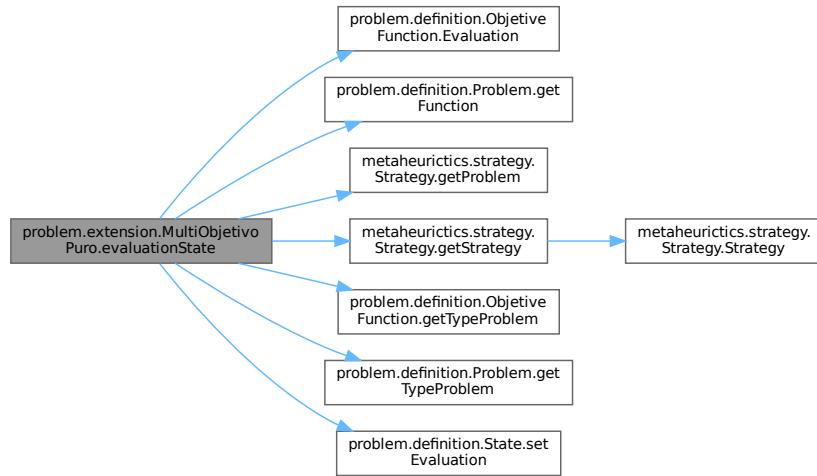
Reimplementado de [problem.extension.SolutionMethod](#).

Definición en la línea 29 del archivo [MultiObjetivoPuro.java](#).

```
00029
00030     // TODO Auto-generated method stub
00031     double tempEval = -1;
00032     ArrayList<Double> evaluation = new
00033         ArrayList<Double>(Strategy.getStrategy().getProblem().getFunction().size());
00034     for (int i = 0; i < Strategy.getStrategy().getProblem().getFunction().size(); i++)
00035     {
00036         ObjetiveFunction objfunction =
00037             (ObjetiveFunction)Strategy.getStrategy().getProblem().getFunction().get(i);
00038         if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar))
00039         {
00040             tempEval = objfunction.Evaluation(state);
00041         }
00042         else{
00043             tempEval = 1-objfunction.Evaluation(state);
00044         }
00045     else{
00046         if(objfunction.getTypeProblem().equals(ProblemType.Maximizar))
00047         {
00048             tempEval = 1-objfunction.Evaluation(state);
00049         }
00050         else{
00051             tempEval = objfunction.Evaluation(state);
00052         }
00053     }
00054     evaluation.add(tempEval);
00055 }
00056 //evaluation.add( (double) -1);
00057 state.setEvaluation(evaluation);
00058 }
```

Hace referencia a [problem.definition.ObjetiveFunction.Evaluation\(\)](#), [problem.definition.Problem.getFunction\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.ObjetiveFunction.get](#), [problem.definition.Problem.getTypeProblem\(\)](#), [problem.definition.Problem.ProblemType.Maximizar](#) y [problem.definition.State.setEvaluate](#)

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [MultiObjetivoPuro.java](#)

## 8.66 Referencia de la clase evolutionary\_algorithms.complement.Mutation

[Mutation](#) - applies a mutation to a given state.

Diagrama de herencia de evolutionary\_algorithms.complement.Mutation

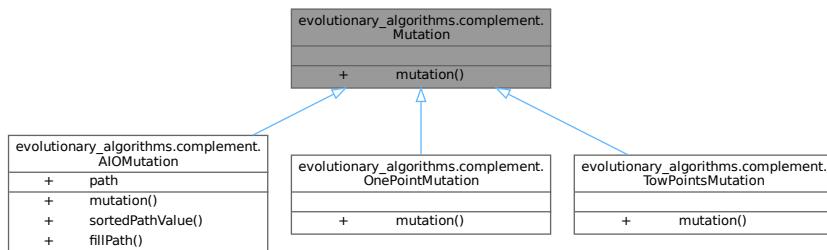
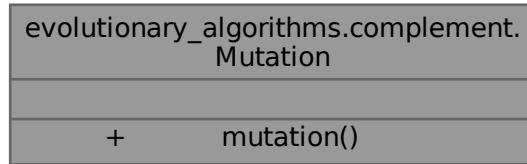


Diagrama de colaboración de evolutionary\_algorithms.complement.Mutation:



## Métodos públicos

- abstract `State mutation (State state, double PM)`  
*mutation - applies the mutation to the state.*

### 8.66.1 Descripción detallada

`Mutation` - applies a mutation to a given state.

Definición en la línea 8 del archivo [Mutation.java](#).

### 8.66.2 Documentación de funciones miembro

#### 8.66.2.1 mutation()

```
abstract State evolutionary_algorithms.complement.Mutation.mutation (
    State state,
    double PM) [abstract]
```

`mutation` - applies the mutation to the state.

#### Parámetros

<code>state</code>	
<code>PM</code>	

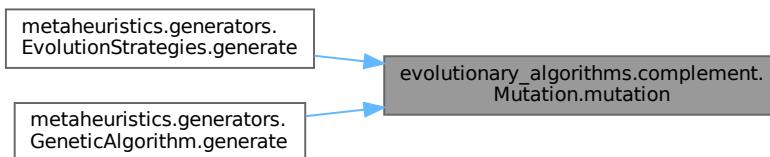
Devuelve

returns the mutated state

Reimplementado en [evolutionary\\_algorithms.complement.AIMutation](#), [evolutionary\\_algorithms.complement.OnePointMutation](#) y [evolutionary\\_algorithms.complement.TwoPointsMutation](#).

Referenciado por [metaheuristics.generators.EvolutionStrategies.generate\(\)](#) y [metaheuristics.generators.GeneticAlgorithm.generate\(\)](#).

Gráfico de llamadas a esta función:



La documentación de esta clase está generada del siguiente archivo:

- [Mutation.java](#)

## 8.67 Referencia de la clase problem\_operators.MutationOperator

[MutationOperator](#) - defines a mutation operator for generating new states.

Diagrama de herencia de problem\_operators.MutationOperator

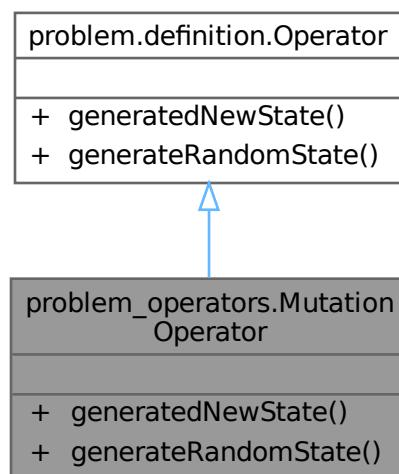
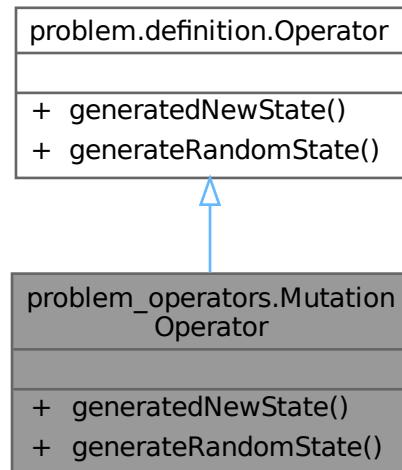


Diagrama de colaboración de problem\_operators.MutationOperator:



## Métodos públicos

- List< [State](#) > [generatedNewState](#) (final [State](#) stateCurrent, final Integer operatornumber)  
*Generate a neighborhood of states by mutating the provided state.*
- List< [State](#) > [generateRandomState](#) (final Integer operatornumber)  
*Generate a list of random states using this mutation operator.*

### 8.67.1 Descripción detallada

[MutationOperator](#) - defines a mutation operator for generating new states.

Definición en la línea 15 del archivo [MutationOperator.java](#).

### 8.67.2 Documentación de funciones miembro

#### 8.67.2.1 [generatedNewState\(\)](#)

```
List< State > problem_operators.MutationOperator.generatedNewState (
    final State stateCurrent,
    final Integer operatornumber) [inline]
```

Generate a neighborhood of states by mutating the provided state.

Contract for overriding implementations:

- Return a non-null List of [State](#) objects (may be empty).
- Each returned [State](#) should be a safe copy (no shared mutable internals).
- The `operatornumber` parameter indicates how many neighbors should be produced.

## Parámetros

<i>stateCurrent</i>	the state to mutate (must not be modified by callers)
<i>operatornumber</i>	number of neighbor states to generate

Devuelve

list of neighboring states, never null

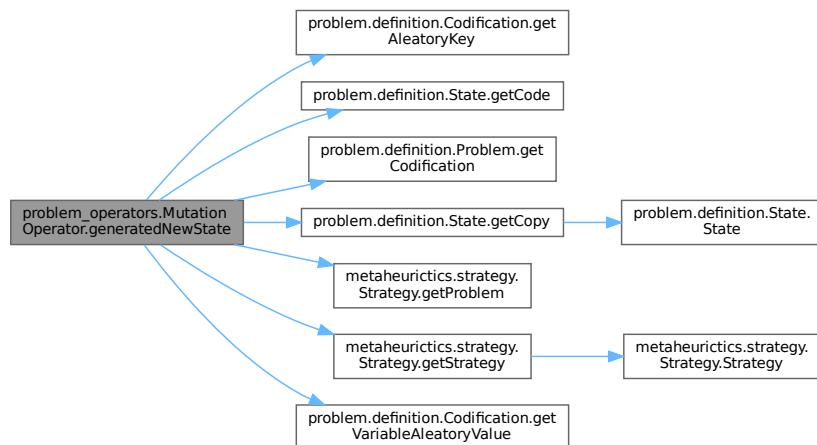
Reimplementado de [problem.definition.Operator](#).

Definición en la línea 32 del archivo [MutationOperator.java](#).

```
00033      {
00034          List<State> listNeighborhood = new ArrayList<State>();
00035          for (int i = 0; i < operatornumber; i++) {
00036              int key = Strategy.getStrategy().getProblem()
00037                  .getCodification()
00038                  .getAleatoryKey();
00039              Object candidate = Strategy.getStrategy().getProblem()
00040                  .getCodification()
00041                  .getVariableAleatoryValue(key);
00042              State state = (State) stateCurrent.getCopy();
00043              state.getCode().set(key, candidate);
00044              listNeighborhood.add(state);
00045          }
00046      return listNeighborhood;
00047 }
```

Hace referencia a [problem.definition.Codification.getAleatoryKey\(\)](#), [problem.definition.State.getCode\(\)](#), [problem.definition.Problem.get](#), [problem.definition.State.getCopy\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#) y [problem.definition.Codification.getVariableAleatoryValue\(\)](#).

Gráfico de llamadas de esta función:



### 8.67.2.2 generateRandomState()

```
List< State > problem_operators.MutationOperator.generateRandomState (
    final Integer operatornumber) [inline]
```

Generate a list of random states using this mutation operator.

Contract for overriding implementations:

- Must return a non-null List of [State](#) objects (may be empty).
- Each returned [State](#) should be a safe copy (no shared mutable internals that could be modified externally).
- The `operatornumber` parameter indicates how many random states should be produced.

Subclasses overriding this method should preserve the contract above.

#### Parámetros

<code>operatornumber</code>	number of random states to generate
-----------------------------	-------------------------------------

#### Devuelve

list of randomly generated states, never null `generateRandomState` - method to generate random states.

#### Parámetros

<code>operatornumber</code>	
-----------------------------	--

#### Devuelve

returns list of random states.

Reimplementado de [problem.definition.Operator](#).

Definición en la línea 70 del archivo [MutationOperator.java](#).

```
00070
00071     // TODO Auto-generated method stub
00072     return null;
00073 }
```

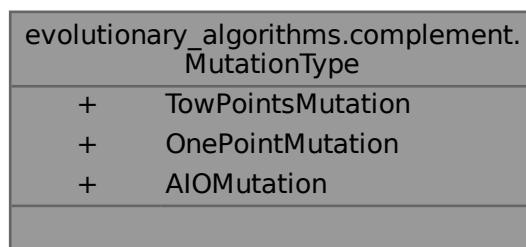
La documentación de esta clase está generada del siguiente archivo:

- [MutationOperator.java](#)

## 8.68 Referencia de la enumeración [evolutionary\\_algorithms.complement.MutationType](#)

[MutationType](#) - descripción (añade detalles).

Diagrama de colaboración de [evolutionary\\_algorithms.complement.MutationType](#):



### Atributos públicos

- [TowPointsMutation](#)
- [OnePointMutation](#)
- [AIOMutation](#)

### 8.68.1 Descripción detallada

[MutationType](#) - descripción (añade detalles).

Definición en la línea 6 del archivo [MutationType.java](#).

### 8.68.2 Documentación de datos miembro

#### 8.68.2.1 AIOMutation

`evolutionary_algorithms.complement.MutationType.AIOMutation`

Definición en la línea 7 del archivo [MutationType.java](#).

#### 8.68.2.2 OnePointMutation

`evolutionary_algorithms.complement.MutationType.OnePointMutation`

Definición en la línea 7 del archivo [MutationType.java](#).

#### 8.68.2.3 TowPointsMutation

`evolutionary_algorithms.complement.MutationType.TowPointsMutation`

Definición en la línea 7 del archivo [MutationType.java](#).

La documentación de esta enumeración está generada del siguiente archivo:

- [MutationType.java](#)

## 8.69 Referencia de la clase

### **local\_search.candidate\_type.NotDominatedCandidate**

**NotDominatedCandidate** - select a non-dominated neighbor for multi-objective.

Diagrama de herencia de local\_search.candidate\_type.NotDominatedCandidate

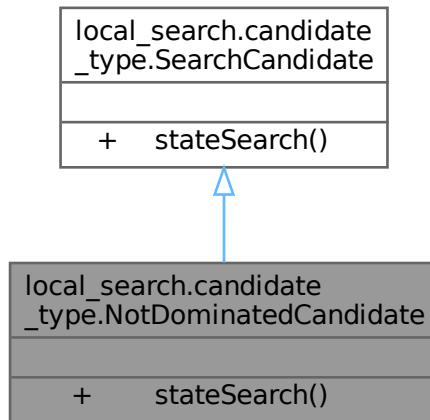
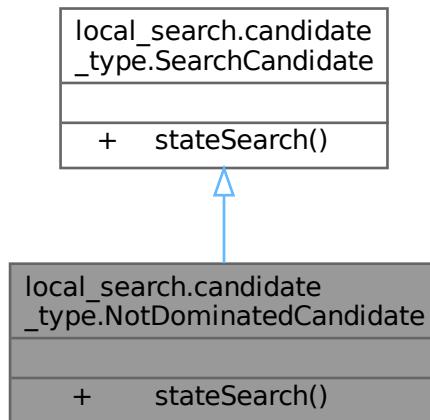


Diagrama de colaboración de local\_search.candidate\_type.NotDominatedCandidate:



#### Métodos públicos

- **State stateSearch (List< State > listNeighborhood)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*Select a non-dominated state from the neighborhood.*

## 8.69.1 Descripción detallada

[NotDominatedCandidate](#) - select a non-dominated neighbor for multi-objective.

Chooses a neighbor that is not dominated by others in the neighborhood using the Domination comparator. Intended for multi-objective problems.

Definición en la línea 17 del archivo [NotDominatedCandidate.java](#).

## 8.69.2 Documentación de funciones miembro

### 8.69.2.1 stateSearch()

```
State local_search.candidate_type.NotDominatedCandidate.stateSearch (
    List< State > listNeighborhood) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

Select a non-dominated state from the neighborhood.

#### Parámetros

<i>listNeighborhood</i>	list of neighbor states
-------------------------	-------------------------

#### Devuelve

a non-dominated [State](#) chosen from the list (or the first if only one)

#### Excepciones

<i>ReflectiveOperationException</i>	when evaluation or reflective calls fail
-------------------------------------	--

Reimplementado de [local\\_search.candidate\\_type.SearchCandidate](#).

Definición en la línea 27 del archivo [NotDominatedCandidate.java](#).

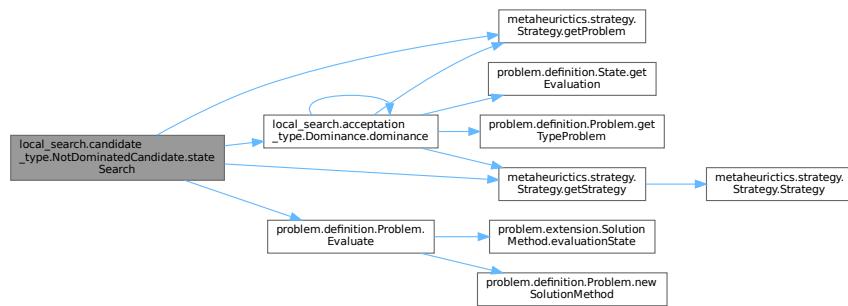
```
00027
{
00028     State state = new State();
00029     State stateA = listNeighborhood.get(0);
00030     boolean stop = false;
00031     if(listNeighborhood.size() == 1){
00032         stop = true;
00033         state = stateA;
00034     }
00035     else {
00036         Strategy.getStrategy().getProblem().Evaluate(stateA);
00037         State stateB;
00038         Dominance dominance = new Dominance();
00039         for (int i = 1; i < listNeighborhood.size(); i++) {
00040             while(stop == false){
00041                 stateB = listNeighborhood.get(i);
00042                 Strategy.getStrategy().getProblem().Evaluate(stateB);
00043                 if(dominance.dominance(stateB, stateA) == true){
00044                     stateA = stateB;
00045                 }else{
00046                     stop = true;
00047                     state = stateA;
00048                 }
00049             }
00050         }
00051     }
00052 }
```

```

00049         }
00050     }
00051 }
00052     return state;
00053 }
```

Hace referencia a [local\\_search.acceptation\\_type.Dominance.dominance\(\)](#), [problem.definition.Problem.Evaluate\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#) y [metaheuristics.strategy.Strategy.getStrategy\(\)](#).

Gráfico de llamadas de esta función:



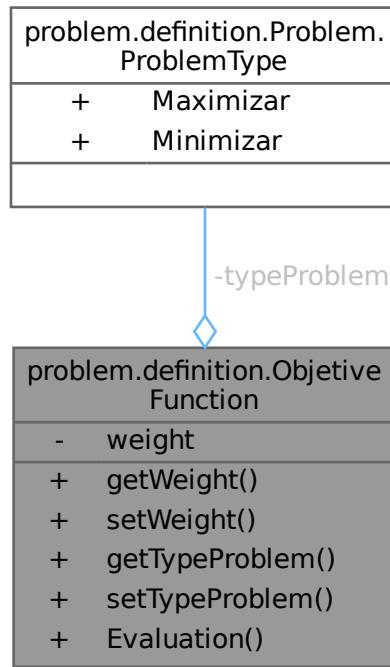
La documentación de esta clase está generada del siguiente archivo:

- [NotDominatedCandidate.java](#)

## 8.70 Referencia de la clase problem.definition.ObjetiveFunction

[ObjetiveFunction](#).

Diagrama de colaboración de problem.definition.ObjectiveFunction:



## Métodos públicos

- float `getWeight ()`  
*Obtener el peso relativo de esta función objetivo.*
- void `setWeight (float weight)`  
*Establecer el peso relativo de la función objetivo.*
- `ProblemType getTypeProblem ()`  
*Obtener si la función es de tipo Maximizar o Minimizar.*
- void `setTypeProblem (ProblemType typeProblem)`  
*Establecer el tipo (Maximizar/Minimizar) de la función objetivo.*
- abstract Double `Evaluation (State state)`  
*Evaluar el estado respecto a esta función objetivo.*

## Atributos privados

- `ProblemType typeProblem`
- float `weight`

### 8.70.1 Descripción detallada

#### ObjectiveFunction.

Representa una función objetivo del problema. Puede ser usada en problemas mono o multi-objetivo y contiene el peso y el tipo (Maximizar/Minimizar).

Definición en la línea 12 del archivo [ObjectiveFunction.java](#).

## 8.70.2 Documentación de funciones miembro

### 8.70.2.1 Evaluation()

```
abstract Double problem.definition.ObjectiveFunction.Evaluation (
    State state) [abstract]
```

Evaluar el estado respecto a esta función objetivo.

#### Parámetros

<code>state</code>	estado a evaluar
--------------------	------------------

#### Devuelve

valor de evaluación (Double)

Referenciado por [problem.extension.MultiObjetivoPuro.evaluationState\(\)](#).

Gráfico de llamadas a esta función:



### 8.70.2.2 getTypeProblem()

```
ProblemType problem.definition.ObjectiveFunction.getTypeProblem () [inline]
```

Obtener si la función es de tipo Maximizar o Minimizar.

#### Devuelve

tipo de problema asociado a la función

Definición en la línea 40 del archivo [ObjectiveFunction.java](#).

```
00040
00041     return typeProblem;
00042 }
```

Hace referencia a [typeProblem](#).

Referenciado por [problem.extension.MultiObjetivoPuro.evaluationState\(\)](#).

Gráfico de llamadas a esta función:



### 8.70.2.3 getWeight()

```
float problem.definition.ObjetiveFunction.getWeight () [inline]
```

Obtener el peso relativo de esta función objetivo.

**Devuelve**

peso (float)

Definición en la línea 22 del archivo [ObjetiveFunction.java](#).

```
00022             {
00023         return weight;
00024     }
```

Hace referencia a [weight](#).

### 8.70.2.4 setTypeProblem()

```
void problem.definition.ObjetiveFunction.setTypeProblem (
    ProblemType typeProblem) [inline]
```

Establecer el tipo (Maximizar/Minimizar) de la función objetivo.

**Parámetros**

<i>typeProblem</i>	tipo a asignar
--------------------	----------------

Definición en la línea 49 del archivo [ObjetiveFunction.java](#).

```
00049             {
00050         this.typeProblem = typeProblem;
00051     }
```

Hace referencia a [typeProblem](#).

### 8.70.2.5 setWeight()

```
void problem.definition.ObjetiveFunction.setWeight (
    float weight) [inline]
```

Establecer el peso relativo de la función objetivo.

**Parámetros**

<i>weight</i>	valor de peso
---------------	---------------

Definición en la línea 31 del archivo [ObjetiveFunction.java](#).

```
00031             {
00032         this.weight = weight;
00033     }
```

Hace referencia a [weight](#).

### 8.70.3 Documentación de datos miembro

#### 8.70.3.1 typeProblem

```
ProblemType problem.definition.ObjectiveFunction.typeProblem [private]
```

Definición en la línea 14 del archivo [ObjectiveFunction.java](#).

Referenciado por [getTypeProblem\(\)](#) y [setTypeProblem\(\)](#).

#### 8.70.3.2 weight

```
float problem.definition.ObjectiveFunction.weight [private]
```

Definición en la línea 15 del archivo [ObjectiveFunction.java](#).

Referenciado por [getWeight\(\)](#) y [setWeight\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [ObjectiveFunction.java](#)

## 8.71 Referencia de la clase [evolutionary\\_algorithms.complement.OnePointCrossover](#)

[OnePointCrossover](#) - applies the one-point crossover operator.

Diagrama de herencia de [evolutionary\\_algorithms.complement.OnePointCrossover](#)

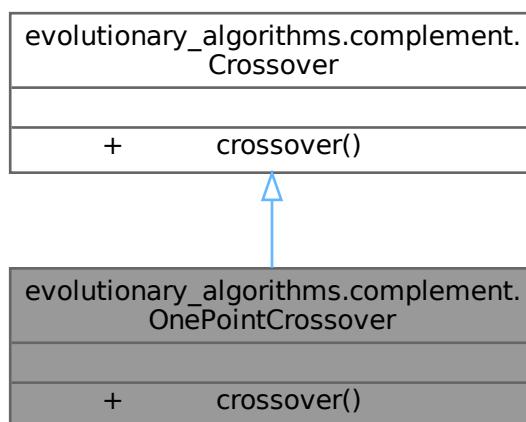
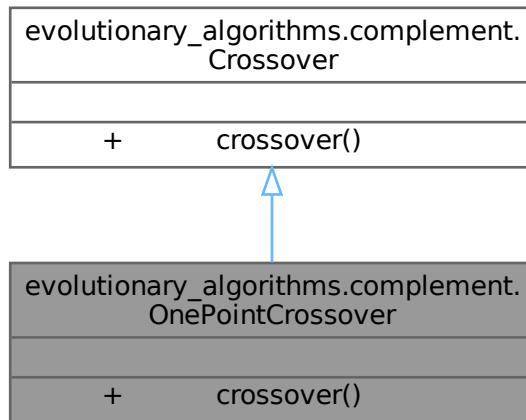


Diagrama de colaboración de evolutionary\_algorithms.complement.OnePointCrossover:



## Métodos públicos

- `State crossover (State father1, State father2, double PC)`  
*Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.*

### 8.71.1 Descripción detallada

`OnePointCrossover` - applies the one-point crossover operator.

Definición en la línea 15 del archivo [OnePointCrossover.java](#).

### 8.71.2 Documentación de funciones miembro

#### 8.71.2.1 `crossover()`

```
State evolutionary_algorithms.complement.OnePointCrossover.crossover (
    State father1,
    State father2,
    double PC) [inline]
```

Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.

This RNG is used for evolutionary algorithm operations (crossover point, selection between children, etc.) and is not used for security-sensitive purposes. Therefore we suppress the Sonar security hotspot S2245 here. `crossover` - applies the crossover operation to two parent states.

## Parámetros

<i>father1</i>	
<i>father2</i>	
<i>PC</i>	

## Devuelve

returns the offspring resulting from the crossover between father1 and father2

Reimplementado de [evolutionary\\_algorithms.complement.Crossover](#).

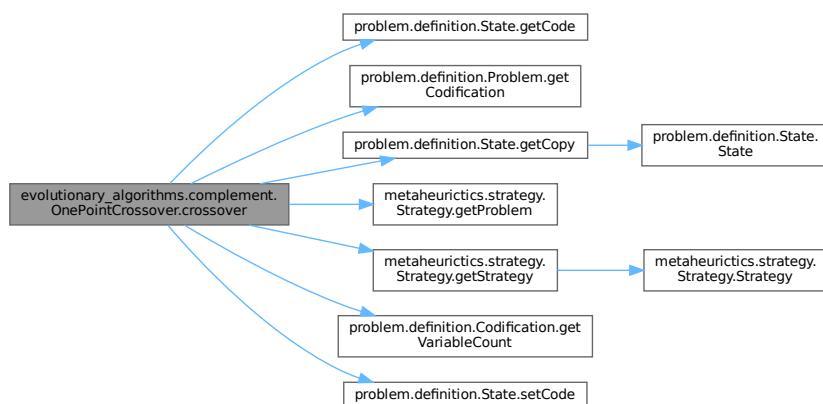
Definición en la línea 32 del archivo [OnePointCrossover.java](#).

```

00032
00033
00034     State newInd = (State) father1.getCopy();
00035
00036     List<Object> ind1 = new ArrayList<Object>();
00037     List<Object> ind2 = new ArrayList<Object>();
00038
00039     double number = ThreadLocalRandom.current().nextDouble(); // thread-safe, [0.0, 1.0)
00040     if(number <= PC){
00041         //llenar los valores de cada hijo
00042         int bound = Strategy.getStrategy().getProblem().getCodification().getVariableCount();
00043         int pos = (bound > 0) ? ThreadLocalRandom.current().nextInt(bound) : 0;
00044         for (int i = 0; i < father1.getCode().size(); i++) {
00045             if(i <= pos){
00046                 ind1.add(father1.getCode().get(i));
00047                 ind2.add(father2.getCode().get(i));
00048             }
00049             else{
00050                 ind1.add(father2.getCode().get(i));
00051                 ind2.add(father1.getCode().get(i));
00052             }
00053         }
00054
00055         //generar un numero aleatorio 0 o 1, si es 0 me quedo con ind1 si es 1 con ind2.
00056         int random = ThreadLocalRandom.current().nextInt(2);
00057         if(random == 0)
00058             newInd.setCode((ArrayList<Object>) ind1);
00059         else newInd.setCode((ArrayList<Object>) ind2);
00060     }
00061     return newInd;
00062 }
```

Hace referencia a [problem.definition.State.getCode\(\)](#), [problem.definition.Problem.getCodification\(\)](#), [problem.definition.State.getCopy\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Codification.getVariableCount\(\)](#) y [problem.definition.State.setCode\(\)](#).

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [OnePointCrossover.java](#)

## 8.72 Referencia de la clase evolutionary\_algorithms.complement.OnePointMutation

[OnePointMutation](#) - applies the one-point mutation operator.

Diagrama de herencia de `evolutionary_algorithms.complement.OnePointMutation`

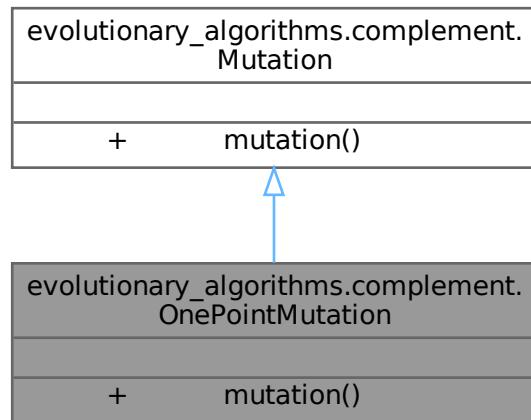
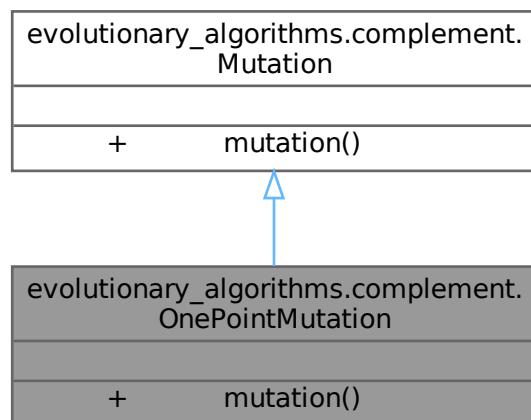


Diagrama de colaboración de `evolutionary_algorithms.complement.OnePointMutation`:



## Métodos públicos

- **State mutation (State state, double PM)**  
*Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.*

### 8.72.1 Descripción detallada

[OnePointMutation](#) - applies the one-point mutation operator.

Definición en la línea 11 del archivo [OnePointMutation.java](#).

### 8.72.2 Documentación de funciones miembro

#### 8.72.2.1 mutation()

```
State evolutionary_algorithms.complement.OnePointMutation.mutation (
    State state,
    double PM) [inline]
```

Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.

This RNG is used to decide whether a mutation occurs in the evolutionary algorithm and is not used for security-sensitive purposes. Suppress Sonar security hotspot S2245 for this usage. mutation - applies the mutation to the state.

#### Parámetros

<i>state</i>	
<i>PM</i>	

#### Devuelve

returns the mutated state

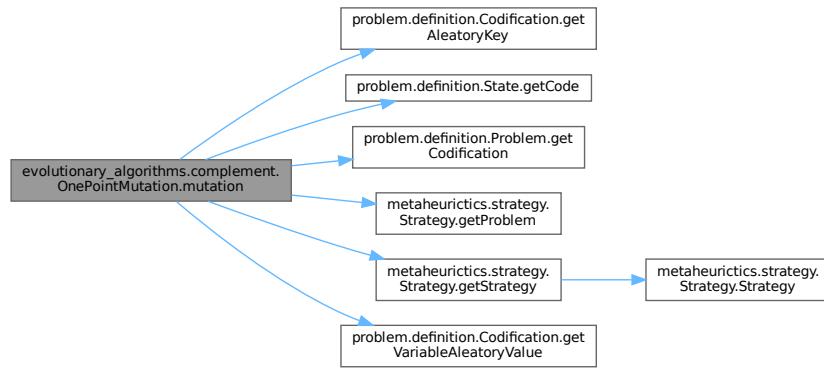
Reimplementado de [evolutionary\\_algorithms.complement.Mutation](#).

Definición en la línea 28 del archivo [OnePointMutation.java](#).

```
00028             {
00029         double probM = ThreadLocalRandom.current().nextDouble();
00030         if(PM >= probM)
00031     {
00032         Object key = Strategy.getStrategy().getProblem().getCodification().getAleatoryKey();
00033         Object value =
00034             Strategy.getStrategy().getProblem().getCodification().getVariableAleatoryValue((Integer)key);
00035             state.getCode().set((Integer) key, value);
00036     }
00037     return state;
00038 }
```

Hace referencia a [problem.definition.Codification.getAleatoryKey\(\)](#), [problem.definition.State.getCode\(\)](#), [problem.definition.Problem.getMetaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#) y [problem.definition.Codification.getVariableAleatoryValue\(\)](#).

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [OnePointMutation.java](#)

## 8.73 Referencia de la clase problem.definition.Operator

[Operator](#).

Diagrama de herencia de `problem.definition.Operator`

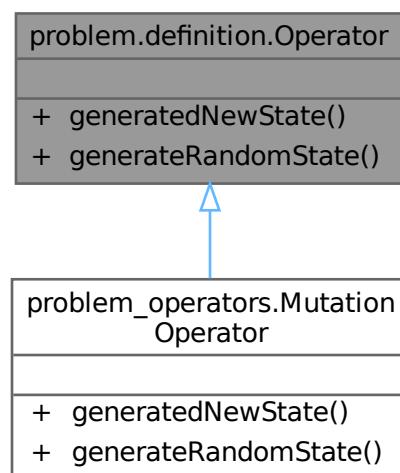
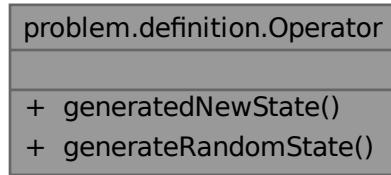


Diagrama de colaboración de problem.definition.Operator:



## Métodos públicos

- abstract List< [State](#) > [generatedNewState](#) ([State](#) stateCurrent, Integer operatornumber)  
*Generar nuevos estados a partir de un estado dado.*
- abstract List< [State](#) > [generateRandomState](#) (Integer operatornumber)  
*Generar un conjunto de estados aleatorios.*

### 8.73.1 Descripción detallada

#### [Operator](#).

Interfaz abstracta para operadores que generan nuevos estados a partir de un estado actual o que producen estados aleatorios.

Definición en la línea 11 del archivo [Operator.java](#).

### 8.73.2 Documentación de funciones miembro

#### 8.73.2.1 [generatedNewState\(\)](#)

```
abstract List< State > problem.definition.Operator.generatedNewState (
    State stateCurrent,
    Integer operatornumber) [abstract]
```

Generar nuevos estados a partir de un estado dado.

#### Parámetros

<code>stateCurrent</code>	estado base
<code>operatornumber</code>	índice de operador (uso dependiente de la implementación)

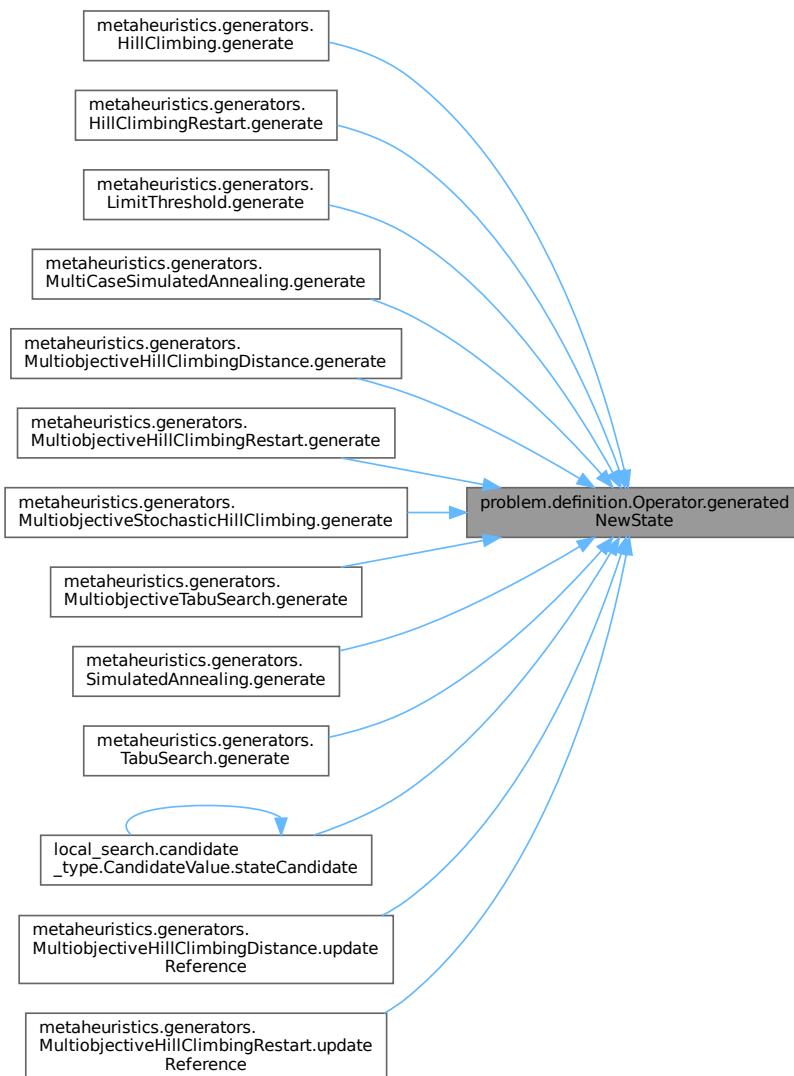
Devuelve

lista de estados generados

Reimplementado en [problem\\_operators.MutationOperator](#).

Referenciado por [metaheuristics.generators.HillClimbing.generate\(\)](#), [metaheuristics.generators.HillClimbingRestart.generate\(\)](#), [metaheuristics.generators.LimitThreshold.generate\(\)](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing.generate\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance.generate\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart\(\)](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing.generate\(\)](#), [metaheuristics.generators.MultiobjectiveTabuSearch.generate\(\)](#), [metaheuristics.generators.SimulatedAnnealing.generate\(\)](#), [metaheuristics.generators.TabuSearch.generate\(\)](#), [local\\_search.candidate\\_type.CandidateValue.stateCandidate\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance.updateReference\(\)](#) y [metaheuristics.generators.MultiobjectiveHillClimbingRestart.updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.73.2.2 generateRandomState()

```
abstract List< State > problem.definition.Operator.generateRandomState (
    Integer operatornumber) [abstract]
```

Generar un conjunto de estados aleatorios.

#### Parámetros

<i>operatornumber</i>	índice/semilla opcional para la generación
-----------------------	--

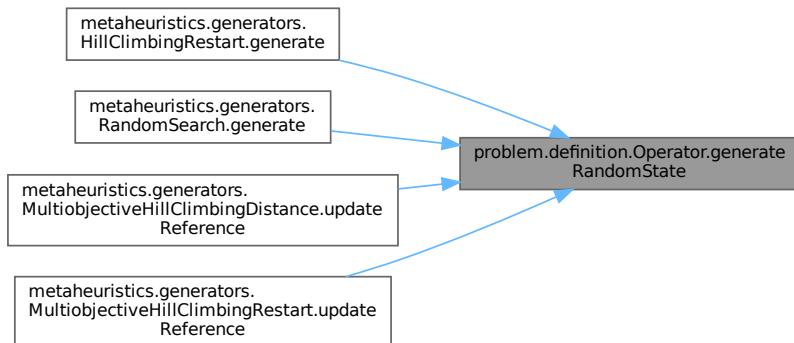
#### Devuelve

lista de estados aleatorios

Reimplementado en [problem\\_operators.MutationOperator](#).

Referenciado por [metaheuristics.generators.HillClimbingRestart.generate\(\)](#), [metaheuristics.generators.RandomSearch.generate\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance.updateReference\(\)](#) y [metaheuristics.generators.MultiobjectiveHillClimbingRestart.updateReference\(\)](#)

Gráfico de llamadas a esta función:



La documentación de esta clase está generada del siguiente archivo:

- [Operator.java](#)

## 8.74 Referencia de la clase metaheuristics.generators.Particle

[Particle](#) - class that represents a particle in the [Particle](#) Swarm Optimization algorithm.

Diagrama de herencia de metaheuristics.generators.Particle

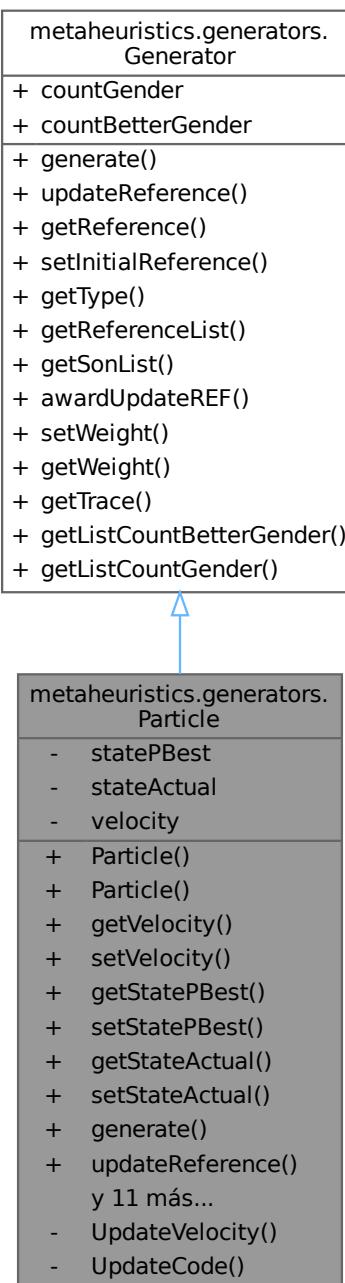
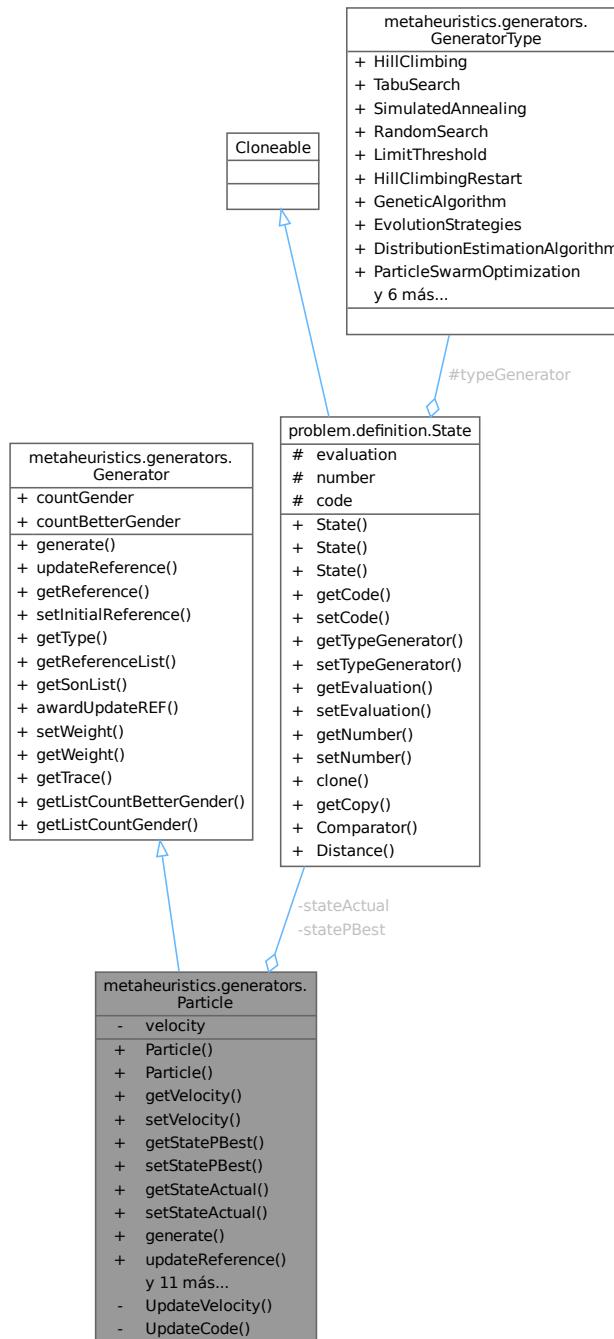


Diagrama de colaboración de metaheuristics.generators.Particle:



## Métodos públicos

- **Particle ()**  
`Particle` - method constructor.
- **Particle (State statePBest, State stateActual, ArrayList< Object > velocity)**  
`Particle` - method constructor.
- **ArrayList< Object > getVelocity ()**

- `getVelocity` - get the velocity of the particle.
- void `setVelocity` (ArrayList< Object > `velocity`)  
*setVelocity - set the velocity of the particle.*
- State `getStatePBest` ()  
*getStatePBest - get the personal best state of the particle.*
- void `setStatePBest` (State `statePBest`)  
*setStatePBest - set the personal best state of the particle.*
- State `getStateActual` ()  
*getStateActual - get the current state of the particle.*
- void `setStateActual` (State `stateActual`)  
*setStateActual - set the current state of the particle.*
- State `generate` (Integer `operatorNumber`) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException  
*generate - generate a new state for the particle.*
- void `updateReference` (State `stateCandidate`, Integer `countIterationsCurrent`) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException  
*updateReference - update the reference state and the personal best state of the particle.*
- State `getReference` ()  
*getReference - get the reference state.*
- void `setInitialReference` (State `stateInitialRef`)  
*setInitialReference - set the initial reference state for the particle.*
- GeneratorType `getType` ()  
*getType - get the type of the generator.*
- List< State > `getReferenceList` ()  
*getReferenceList - get the list of reference states.*
- List< State > `getSonList` ()  
*getSonList - get the list of son states.*
- boolean `awardUpdateREF` (State `stateCandidate`)  
*awardUpdateREF - award the update of the reference state.*
- void `setWeight` (float `weight`)  
*setWeight - set the weight of the particle.*
- float `getWeight` ()  
*getWeight - get the weight of the particle.*
- float[] `getTrace` ()  
*getTrace - get the trace of the particle.*
- int[] `getListCountBetterGender` ()  
*getListCountBetterGender - get the list of counts of better genders.*
- int[] `getListCountGender` ()  
*getListCountGender - get the list of counts of genders.*

## Métodos privados

- ArrayList< Object > `UpdateVelocity` ()  
*Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.*
- ArrayList< Object > `UpdateCode` (ArrayList< Object > `actualVelocity`)  
*Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness when computing binary code updates.*

### Atributos privados

- State statePBest
- State stateActual
- ArrayList< Object > velocity

### Otros miembros heredados

#### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- int countGender
- int countBetterGender

### 8.74.1 Descripción detallada

[Particle](#) - class that represents a particle in the [Particle](#) Swarm Optimization algorithm.

Definición en la línea 16 del archivo [Particle.java](#).

### 8.74.2 Documentación de constructores y destructores

#### 8.74.2.1 Particle() [1/2]

```
metaheuristics.generators.Particle.Particle () [inline]
```

[Particle](#) - method constructor.

Definición en la línea 26 del archivo [Particle.java](#).

```
00026             {
00027     super();
00028     this.stateActual = new State();
00029     this.statePBest = new State();
00030     this.velocity = new ArrayList<Object>();
00031 }
```

#### 8.74.2.2 Particle() [2/2]

```
metaheuristics.generators.Particle.Particle (
    State statePBest,
    State stateActual,
    ArrayList< Object > velocity) [inline]
```

[Particle](#) - method constructor.

### Parámetros

statePBest	
stateActual	
velocity	

Definición en la línea 39 del archivo [Particle.java](#).

```
00039
00040     super();
00041     this.statePBest = statePBest;
00042     this.stateActual = stateActual;
00043     this.velocity = velocity;
00044 }
```

Hace referencia a [stateActual](#), [statePBest](#) y [velocity](#).

### 8.74.3 Documentación de funciones miembro

#### 8.74.3.1 awardUpdateREF()

```
boolean metaheuristics.generators.Particle.awardUpdateREF (
    State stateCandidate) [inline]
```

awardUpdateREF - award the update of the reference state.

##### Parámetros

<i>stateCandidate</i>	
-----------------------	--

##### Devuelve

return true if the reference state was updated, false otherwise

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 280 del archivo [Particle.java](#).

```
00280
00281     // TODO Auto-generated method stub
00282     return false;
00283 }
```

#### 8.74.3.2 generate()

```
State metaheuristics.generators.Particle.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException [inline]
```

generate - generate a new state for the particle.

##### Parámetros

<i>operatornumber</i>	
-----------------------	--

##### Devuelve

##### Excepciones

<i>IllegalArgumentException</i>	
<i>SecurityException</i>	
<i>ClassNotFoundException</i>	
<i>InstantiationException</i>	

<i>IllegalAccessException</i>	
<i>InvocationTargetException</i>	
<i>NoSuchMethodException</i>	

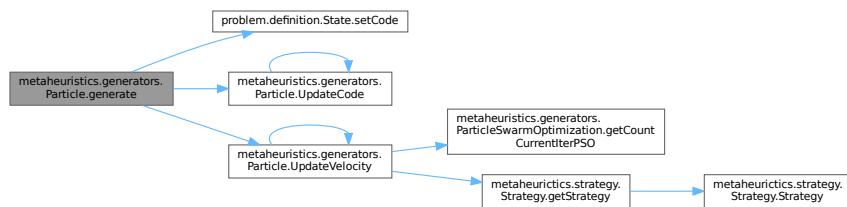
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 106 del archivo [Particle.java](#).

```
00110          {
00111      // TODO Auto-generated method stub
00112
00113      ArrayList<Object> actualVelocity = UpdateVelocity();
00114      ArrayList<Object> newCode = UpdateCode(actualVelocity);
00115      this.velocity = actualVelocity;
00116      this.stateActual.setCode(newCode);
00117      return null;
00118  }
```

Hace referencia a [problem.definition.State.setCode\(\)](#), [UpdateCode\(\)](#) y [UpdateVelocity\(\)](#).

Gráfico de llamadas de esta función:



### 8.74.3.3 getListCountBetterGender()

```
int[] metaheuristics.generators.Particle.getListCountBetterGender () [inline]
```

getListCountBetterGender - get the list of counts of better genders.

Devuelve

list of counts of better genders

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 320 del archivo [Particle.java](#).

```
00320          {
00321      // TODO Auto-generated method stub
00322      return new int[0];
00323  }
```

#### 8.74.3.4 getListCountGender()

```
int[] metaheuristics.generators.Particle.getListCountGender () [inline]
```

getListCountGender - get the list of counts of genders.

Devuelve

list of counts of genders

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 330 del archivo [Particle.java](#).

```
00330 {  
00331     // TODO Auto-generated method stub  
00332     return new int[0];  
00333 }
```

#### 8.74.3.5 getReference()

```
State metaheuristics.generators.Particle.getReference () [inline]
```

getReference - get the reference state.

Devuelve

reference state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 229 del archivo [Particle.java](#).

```
00229 {  
00230     // TODO Auto-generated method stub  
00231     return null;  
00232 }
```

#### 8.74.3.6 getReferenceList()

```
List< State > metaheuristics.generators.Particle.getReferenceList () [inline]
```

getReferenceList - get the list of reference states.

Devuelve

list of reference states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 259 del archivo [Particle.java](#).

```
00259 {  
00260     // TODO Auto-generated method stub  
00261     return null;  
00262 }
```

### 8.74.3.7 getSonList()

```
List< State > metaheuristics.generators.Particle.getSonList () [inline]
```

getSonList - get the list of son states.

Devuelve

list of son states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 269 del archivo [Particle.java](#).

```
00269
00270     // TODO Auto-generated method stub
00271     return null;
00272 }
```

### 8.74.3.8 getStateActual()

```
State metaheuristics.generators.Particle.getStateActual () [inline]
```

getStateActual - get the current state of the particle.

Devuelve

the current state of the particle

Definición en la línea 82 del archivo [Particle.java](#).

```
00082
00083     return stateActual;
00084 }
```

Hace referencia a [stateActual](#).

### 8.74.3.9 getStatePBest()

```
State metaheuristics.generators.Particle.getStatePBest () [inline]
```

getStatePBest - get the personal best state of the particle.

Devuelve

the personal best state of the particle

Definición en la línea 66 del archivo [Particle.java](#).

```
00066
00067     return statePBest;
00068 }
```

Hace referencia a [statePBest](#).

Referenciado por [metaheuristics.generators.ParticleSwarmOptimization.updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.74.3.10 getTrace()

```
float[ ] metaheuristics.generators.Particle.getTrace () [inline]
```

getTrace - get the trace of the particle.

Devuelve

```
return the trace of the particle
```

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 310 del archivo [Particle.java](#).

```
00310          {
00311      // TODO Auto-generated method stub
00312      return new float[0];
00313 }
```

### 8.74.3.11 getType()

```
GeneratorType metaheuristics.generators.Particle.getType () [inline]
```

getType - get the type of the generator.

Devuelve

```
return the type of the generator
```

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 249 del archivo [Particle.java](#).

```
00249          {
00250      // TODO Auto-generated method stub
00251      return null;
00252 }
```

### 8.74.3.12 getVelocity()

```
ArrayList< Object > metaheuristics.generators.Particle.getVelocity () [inline]
```

getVelocity - get the velocity of the particle.

Devuelve

```
the velocity of the particle
```

Definición en la línea 50 del archivo [Particle.java](#).

```
00050          {
00051      return velocity;
00052 }
```

Hace referencia a [velocity](#).

### 8.74.3.13 getWeight()

```
float metaheuristics.generators.Particle.getWeight () [inline]
```

getWeight - get the weight of the particle.

#### Devuelve

return the weight of the particle

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 300 del archivo [Particle.java](#).

```
00300          {
00301      // TODO Auto-generated method stub
00302      return 0;
00303 }
```

### 8.74.3.14 setInitialReference()

```
void metaheuristics.generators.Particle.setInitialReference (
    State stateInitialRef) [inline]
```

setInitialReference - set the initial reference state for the particle.

#### Parámetros

<code>stateInitialRef</code>	<input type="button" value=""/>
------------------------------	---------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 239 del archivo [Particle.java](#).

```
00239          {
00240      // TODO Auto-generated method stub
00241
00242 }
```

### 8.74.3.15 setStateActual()

```
void metaheuristics.generators.Particle.setStateActual (
    State stateActual) [inline]
```

setStateActual - set the current state of the particle.

#### Parámetros

<code>stateActual</code>	the current state to set
--------------------------	--------------------------

Definición en la línea 90 del archivo [Particle.java](#).

```
00090          {
00091      this.stateActual = stateActual;
00092 }
```

Hace referencia a [stateActual](#).

**Parámetros**

<code>statePBest</code>	the personal best state to set
-------------------------	--------------------------------

Definición en la línea 74 del archivo [Particle.java](#).

```
00074         {  
00075             this.statePBest = statePBest;  
00076         }
```

Hace referencia a [statePBest](#).

**8.74.3.17 setVelocity()**

```
void metaheuristics.generators.Particle.setVelocity (  
    ArrayList< Object > velocity) [inline]
```

setVelocity - set the velocity of the particle.

**Parámetros**

<code>velocity</code>	the velocity to set
-----------------------	---------------------

Definición en la línea 58 del archivo [Particle.java](#).

```
00058         {  
00059             this.velocity = velocity;  
00060         }
```

Hace referencia a [velocity](#).

**8.74.3.18 setWeight()**

```
void metaheuristics.generators.Particle.setWeight (  
    float weight) [inline]
```

setWeight - set the weight of the particle.

**Parámetros**

<code>weight</code>	
---------------------	--

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 290 del archivo [Particle.java](#).

```
00290         {  
00291             // TODO Auto-generated method stub  
00292         }  
00293     }
```

### 8.74.3.19 UpdateCode()

```
ArrayList< Object > metaheuristics.generators.Particle.UpdateCode (
    ArrayList< Object > actualVelocity) [inline], [private]
```

Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness when computing binary code updates.

Not security-sensitive; suppress S2245.

Definición en la línea 164 del archivo [Particle.java](#).

```
00164                                         { // CALCULO DE LA NUEA
00165     POSICION DE LA PARTICULA
00166     ArrayList<Object> newCode = new ArrayList<Object>();
00167     //poner la condicion de si se esta trabajando con valores continuos o binarios
00168     if(ParticleSwarmOptimization.binary == false){
00169         for (int i = 0; i < stateActual.getCode().size(); i++) {
00170             newCode.add( (Double)(stateActual.getCode().get(i)) +
00171                         (Double)(actualVelocity.get(i)) );
00172         }
00173     } else{                                         //cálculo de la posicion para
00174         codificación binaria
00175         ArrayList<Object> binaryCode = new ArrayList<Object>();
00176         for (int i = 0; i < stateActual.getCode().size(); i++){
00177             double rand = ThreadLocalRandom.current().nextDouble();
00178             double s = 1/(1 + 1.72 * (Double)(actualVelocity.get(i))); //
00179             if (rand < s){
00180                 binaryCode.add(1);
00181             } else{
00182                 binaryCode.add(0);
00183             }
00184         }
00185         return binaryCode;
00186     }
00187 }
00188 }
```

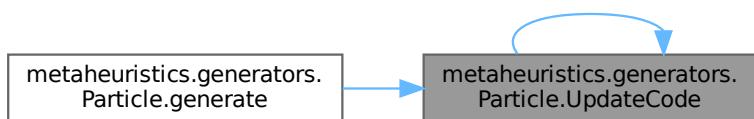
Hace referencia a [metaheuristics.generators.ParticleSwarmOptimization.binary](#), [stateActual](#) y [UpdateCode\(\)](#).

Referenciado por [generate\(\)](#) y [UpdateCode\(\)](#).

Gráfico de llamadas de esta función:



Gráfico de llamadas a esta función:



### 8.74.3.20 updateReference()

```
void metaheuristics.generators.Particle.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

updateReference - update the reference state and the personal best state of the particle.

#### Parámetros

<i>stateCandidate</i>	
<i>countIterationsCurrent</i>	

#### Excepciones

<i>IllegalArgumentException</i>	
<i>SecurityException</i>	
<i>ClassNotFoundException</i>	
<i>InstantiationException</i>	
<i>IllegalAccessException</i>	
<i>InvocationTargetException</i>	
<i>NoSuchMethodException</i>	

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 203 del archivo [Particle.java](#).

```
00207
00208     // TODO Auto-generated method stub
00209     if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00210         if(stateActual.getEvaluation().get(0) > statePBest.getEvaluation().get(0)){
00211             statePBest.setCode(new ArrayList<Object>(stateActual.getCode()));
00212             statePBest.setEvaluation(stateActual.getEvaluation());
00213         }
00214     }
00215     else{
00216         if(stateCandidate.getEvaluation().get(0) < statePBest.getEvaluation().get(0)){
00217             statePBest.setCode(new ArrayList<Object>(stateCandidate.getCode()));
00218             statePBest.setEvaluation(stateCandidate.getEvaluation());
00219         }
00220     }
00221 }
00222 }
```

Hace referencia a [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Problem.getTypeProblem\(\)](#), [problem.definition.Problem.ProblemType.Maximizar](#), [stateActual](#) y [statePBest](#).

Referenciado por [metaheuristics.generators.ParticleSwarmOptimization.updateReference\(\)](#).

Gráfico de llamadas de esta función:

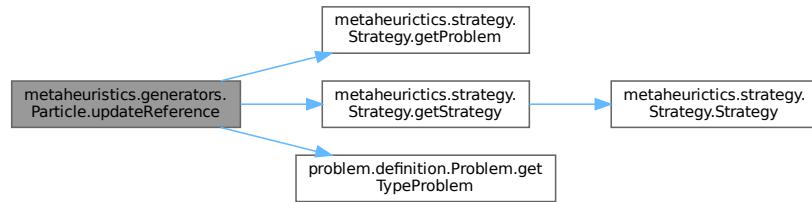


Gráfico de llamadas a esta función:



### 8.74.3.21 UpdateVelocity()

```
ArrayList< Object > metaheuristics.generators.Particle.UpdateVelocity () [inline], [private]
```

Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.

This RNG updates particle velocity and is not security-sensitive. Suppress Sonar hotspot S2245 for this usage.

Definición en la línea 127 del archivo [Particle.java](#).

```

00127             { // actualizar velocidad
00128         double w = ParticleSwarmOptimization.wmax - ((ParticleSwarmOptimization.wmax -
00129             ParticleSwarmOptimization.wmin) / Strategy.getStrategy().getCountMax()) *
00130             ParticleSwarmOptimization.getCountCurrentIterPSO(); //CALCULO DE LA INERCIA
00131         double rand1 = ThreadLocalRandom.current().nextDouble();
00132         double rand2 = ThreadLocalRandom.current().nextDouble();
00133         double inertia, cognitive, social;
00134         int learning = ParticleSwarmOptimization.learning1 + ParticleSwarmOptimization.learning2; // ratios de aprendizaje cognitivo y social
00135         ParticleSwarmOptimization.constriction = 2/(Math.abs(2 - learning-Math.sqrt((learning *
00136             learning)- 4 * learning)); // Factor de costriccion
00137         ArrayList<Object> actualVelocity = new ArrayList<Object>();
00138         if(velocity.isEmpty()){
00139             for (int i = 0; i < Strategy.getStrategy().getProblem().getState().getCode().size(); i++){
00140                 velocity.add(0.0);
00141             }
00142             // recorre el vector velocidad y lo actualiza
00143             for (int i = 0; i < Strategy.getStrategy().getProblem().getState().getCode().size(); i++) {
00144                 // cumulo donde se encuentra la particula
00145                 int swarm = ParticleSwarmOptimization.getCountParticle() /
00146                     ParticleSwarmOptimization.getCountParticleBySwarm();
00147                 inertia = w * (Double)velocity.get(i);
00148                 if(ParticleSwarmOptimization.binary == true){
00149                     cognitive = (Double)(ParticleSwarmOptimization.learning1 * rand1 *
00150                         ((Integer)(this.statePBest.getCode().get(i)) - (Integer)(stateActual.getCode().get(i))));
00151                     social = (Double)(ParticleSwarmOptimization.learning2 * rand2 * (((Integer)((State)
00152                         ParticleSwarmOptimization.lBest[swarm]).getCode().get(i)) -
00153                         (Integer)(stateActual.getCode().get(i)))));
00154                 }
00155             }
00156         }
00157     }

```

```

00149         else{
00150             cognitive = (Double)(ParticleSwarmOptimization.learning1 * rand1 *
00151 ((Double)(this.statePBest.getCode().get(i)) - (Double)(stateActual.getCode().get(i))));
00152             social = (Double)(ParticleSwarmOptimization.learning2 * rand2 * (((Double)((State)
00153 ParticleSwarmOptimization.lBest[swarm]).getCode().get(i))) -
00154 ((Double)(stateActual.getCode().get(i)))));
00155         }
00156         actualVelocity.add(ParticleSwarmOptimization.constriction*(inertia + cognitive + social));
00157     }
00158     return actualVelocity;
00159 }
```

Hace referencia a [metaheuristics.generators.ParticleSwarmOptimization.getCountCurrentIterPSO\(\)](#), [metaheuristics.strategy.Strategy](#), [metaheuristics.generators.ParticleSwarmOptimization.learning1](#), [metaheuristics.generators.ParticleSwarmOptimization.learning2](#), [UpdateVelocity\(\)](#), [velocity](#), [metaheuristics.generators.ParticleSwarmOptimization.wmax](#) y [metaheuristics.generators.ParticleSwarmOptimization.constriction](#).

Referenciado por [generate\(\)](#) y [UpdateVelocity\(\)](#).

Gráfico de llamadas de esta función:

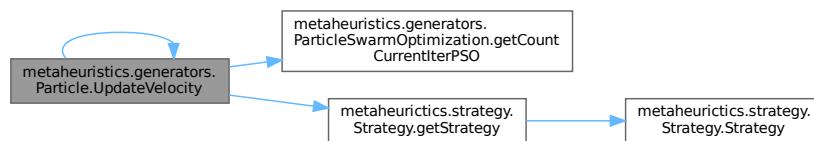


Gráfico de llamadas a esta función:



## 8.74.4 Documentación de datos miembro

### 8.74.4.1 stateActual

`State` `metaheuristics.generators.Particle.stateActual` [private]

Definición en la línea 19 del archivo [Particle.java](#).

Referenciado por [getStateActual\(\)](#), [Particle\(\)](#), [setStateActual\(\)](#), [UpdateCode\(\)](#) y [updateReference\(\)](#).

#### 8.74.4.2 statePBest

```
State metaheuristics.generators.Particle.statePBest [private]
```

Definición en la línea 18 del archivo [Particle.java](#).

Referenciado por [getStatePBest\(\)](#), [Particle\(\)](#), [setStatePBest\(\)](#) y [updateReference\(\)](#).

#### 8.74.4.3 velocity

```
ArrayList<Object> metaheuristics.generators.Particle.velocity [private]
```

Definición en la línea 20 del archivo [Particle.java](#).

Referenciado por [getVelocity\(\)](#), [Particle\(\)](#), [setVelocity\(\)](#) y [UpdateVelocity\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [Particle.java](#)

### 8.75 Referencia de la clase

## metaheuristics.generators.ParticleSwarmOptimization

[ParticleSwarmOptimization](#) - class that implements the [Particle](#) Swarm Optimization metaheuristic.

Diagrama de herencia de metaheuristics.generators.ParticleSwarmOptimization

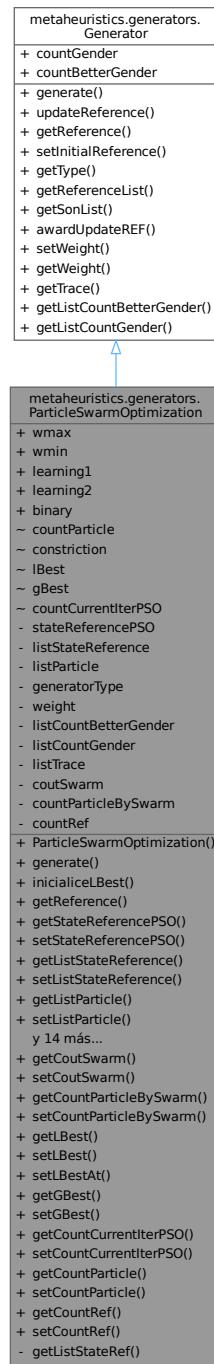
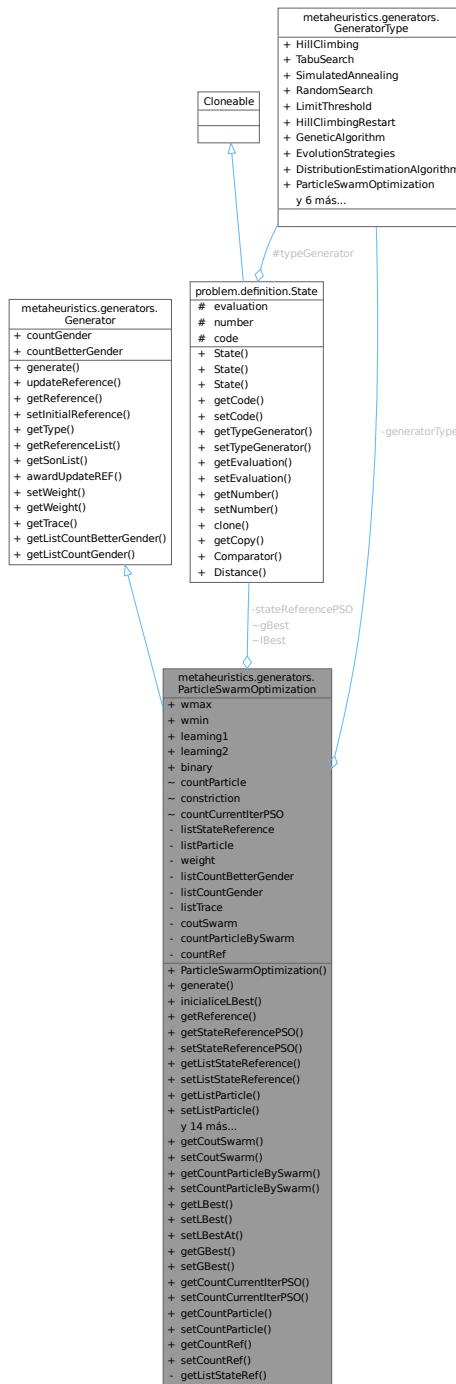


Diagrama de colaboración de `metaheuristics.generators.ParticleSwarmOptimization`:



## Métodos públicos

- `ParticleSwarmOptimization ()`

*Create a new `ParticleSwarmOptimization` instance.*

- `State generate (Integer operatornumber)` throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

- Generate a new state by advancing the next particle in the swarm.
- void **inicialiceLBest** ()  
Initialize the local best `lBest` array for each swarm.
  - State **getReference** ()  
Return the single reference state used by single-reference generators.
  - State **getStateReferencePSO** ()  
Return the PSO-specific reference state used by some helpers.
  - void **setStateReferencePSO** (State `stateReferencePSO`)  
Set the PSO-specific reference state.
  - List< Particle > **getListStateReference** ()  
Return the current list of particles used internally.
  - void **setListStateReference** (List< State > `listStateReference`)  
Replace the internal state reference list.
  - List< Particle > **getListParticle** ()  
Get the internal particle list.
  - List< Particle > **setListParticle** (List< Particle > `listParticle`)  
Replace the internal particle list and return the newly set list.
  - GeneratorType **getGeneratorType** ()  
Get the generator type associated with this instance.
  - void **setGeneratorType** (GeneratorType `generatorType`)  
Set the generator type for this instance.
  - void **updateReference** (State `stateCandidate`, Integer `countIterationsCurrent`) throws `IllegalArgumentException`,  
`SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`,  
`InvocationTargetException`, `NoMethodException`  
Update the PSO internal references based on a newly generated candidate state.
  - State **gBestIncial** ()  
Compute the initial global best (`gBest`) from the local bests contained in `lBest`.
  - void **setInitialReference** (State `stateInitialRef`)  
Set the initial reference(s) for the generator.
  - GeneratorType **getType** ()  
Return the concrete generator type of this generator.
  - List< State > **getReferenceList** ()  
Return a copy of the list used as reference solutions by PSO.
  - List< State > **getSonList** ()  
Return the currently generated offspring/state list for this generator.
  - boolean **awardUpdateREF** (State `stateCandidate`)  
Decide whether the provided candidate should update references.
  - void **setWeight** (float `weight`)  
Set the internal inertia weight used by particle velocity updates.
  - float **getWeight** ()  
Get the current inertia weight used in PSO.
  - int[] **getListCountBetterGender** ()  
Return a defensive copy of the counters that track how many times a particle improved during a period.
  - int[] **getListCountGender** ()  
Return a defensive copy of usage counters per period.
  - float[] **getTrace** ()  
Return a copy of the internal trace array used for diagnostic information (weights over time, etc.).

## Métodos públicos estáticos

- static int [getCountSwarm \(\)](#)  
*getCountSwarm - get the number of swarms.*
- static void [setCountSwarm \(int v\)](#)  
*setCountSwarm - set the number of swarms.*
- static int [getCountParticleBySwarm \(\)](#)  
*getCountParticleBySwarm - get the number of particles per swarm.*
- static void [setCountParticleBySwarm \(int v\)](#)  
*setCountParticleBySwarm - set the number of particles per swarm.*
- static [State\[\] getLBest \(\)](#)
- static void [setLBest \(State\[\] arr\)](#)  
*setLBest - set the local best array.*
- static void [setLBestAt \(int idx, State s\)](#)  
*setLBest - set the local best at index.*
- static [State getGBest \(\)](#)
- static void [setGBest \(State s\)](#)
- static int [getCountCurrentIterPSO \(\)](#)
- static void [setCountCurrentIterPSO \(int v\)](#)
- static int [getCountParticle \(\)](#)
- static void [setCountParticle \(int v\)](#)
- static int [getCountRef \(\)](#)
- static void [setCountRef \(int countRef\)](#)

## Atributos públicos estáticos

- static final double [wmax](#) = 0.9
- static final double [wmin](#) = 0.2
- static final int [learning1](#) = 2
- static final int [learning2](#) = 2
- static final boolean [binary](#) = false

## Métodos privados

- List< [Particle](#) > [getListStateRef \(\)](#)  
*Build or return the list of [Particle](#) instances used by PSO.*

## Atributos privados

- [State stateReferencePSO](#)
- List< [State](#) > [listStateReference](#) = new ArrayList<[State](#)>()
- List< [Particle](#) > [listParticle](#) = new ArrayList<[Particle](#)> ()
- [GeneratorType generatorType](#)
- float [weight](#) = 50
- int[] [listCountBetterGender](#) = new int[10]
- int[] [listCountGender](#) = new int[10]
- float[] [listTrace](#) = new float[1200000]

### Atributos estáticos privados

- static int `coutSwarm` = 0
- static int `countParticleBySwarm` = 0
- static int `countRef` = `coutSwarm` \* `countParticleBySwarm`

### Otros miembros heredados

### Atributos públicos heredados de `metaheuristics.generators.Generator`

- int `countGender`
- int `countBetterGender`

## 8.75.1 Descripción detallada

`ParticleSwarmOptimization` - class that implements the `Particle` Swarm Optimization metaheuristic.

Definición en la línea 19 del archivo `ParticleSwarmOptimization.java`.

## 8.75.2 Documentación de constructores y destructores

### 8.75.2.1 `ParticleSwarmOptimization()`

```
metaheuristics.generators.ParticleSwarmOptimization.ParticleSwarmOptimization () [inline]
```

Create a new `ParticleSwarmOptimization` instance.

The constructor initializes internal arrays and sets default values for weights, counters and generators. It intentionally does not perform expensive operations; heavy initialization is deferred to `getListStateRef()` when needed.

Definición en la línea 132 del archivo `ParticleSwarmOptimization.java`.

```
00132             super();
00133             this.setListParticle(getListStateRef());
00134             this.generatorType = GeneratorType.ParticleSwarmOptimization;
00135             this.weight = 50;
00136             setLBest(new State[coutSwarm]);
00137             if(!listParticle.isEmpty()){
00138                 setCountCurrentIterPSO(1);
00139                 inicialiceLBest();
00140                 setGBest(gBestInicial());
00141             }
00142             setCountParticle(0);
00143             listTrace[0] = this.weight;
00144             listCountBetterGender[0] = 0;
00145             listCountGender[0] = 0;
00146         }
```

Hace referencia a `coutSwarm`, `gBestInicial()`, `getListStateRef()`, `inicialiceLBest()`, `listCountBetterGender`, `listCountGender`, `listParticle`, `listTrace`, `metaheuristics.generators.GeneratorType.ParticleSwarmOptimization`, `setCountCurrentIterPSO()`, `setCountParticle()`, `setGBest()`, `setLBest()`, `setListParticle()` y `weight`.

Referenciado por `getListStateRef()`.

Gráfico de llamadas de esta función:

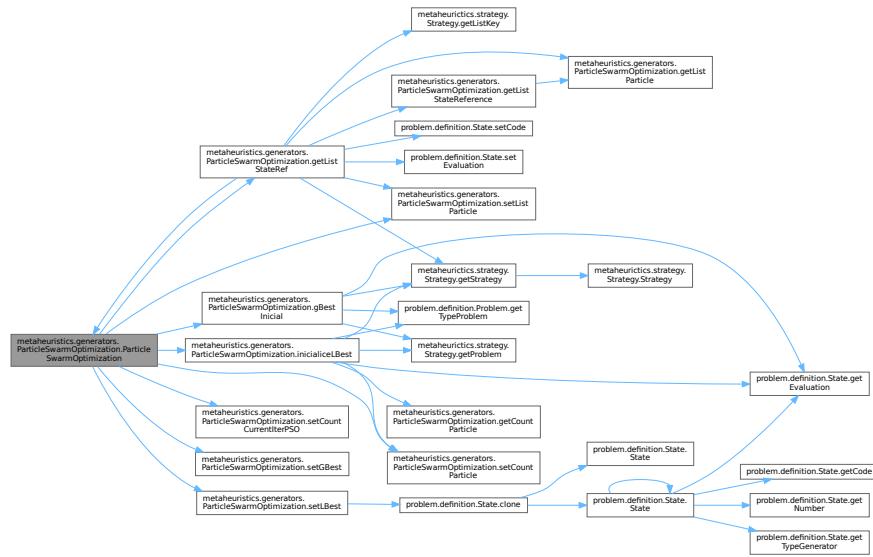
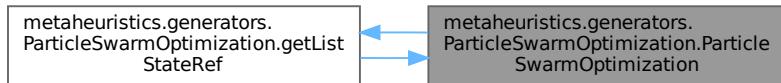


Gráfico de llamadas a esta función:



### 8.75.3 Documentación de funciones miembro

#### 8.75.3.1 awardUpdateREF()

```
boolean metaheuristics.generators.ParticleSwarmOptimization.awardUpdateREF (
```

Decide whether the provided candidate should update references.

PSO uses `updateReference` directly; this helper returns `false` by default and can be implemented if integration requires it.

## Parámetros

*stateCandidate* candidate state to evaluate

Devuelve

false by default

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 475 del archivo [ParticleSwarmOptimization.java](#).

```
00475         return false;
00476     }
00477 }
```

### 8.75.3.2 gBestInicial()

[State](#) [metaheuristics.generators.ParticleSwarmOptimization.gBestInicial \(\)](#) [inline]

Compute the initial global best (gBest) from the local bests contained in lBest.

Devuelve

the best [State](#) among lBest

Definición en la línea 403 del archivo [ParticleSwarmOptimization.java](#).

```
00403     {
00404         State stateBest = lBest[0];
00405         for (int i = 1; i < lBest.length; i++) {
00406             if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00407                 if (lBest[i].getEvaluation().get(0) > stateBest.getEvaluation().get(0)){
00408                     stateBest = lBest[i];
00409                 }
00410             }
00411             else{
00412                 if (lBest[i].getEvaluation().get(0) < stateBest.getEvaluation().get(0)){
00413                     stateBest = lBest[i];
00414                 }
00415             }
00416         }
00417         return stateBest;
00418     }
```

Hace referencia a [problem.definition.State.getEvaluation\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Problem.getTypeProblem\(\)](#) y [problem.definition.Problem.ProblemType](#).

Referenciado por [ParticleSwarmOptimization\(\)](#).

Gráfico de llamadas de esta función:

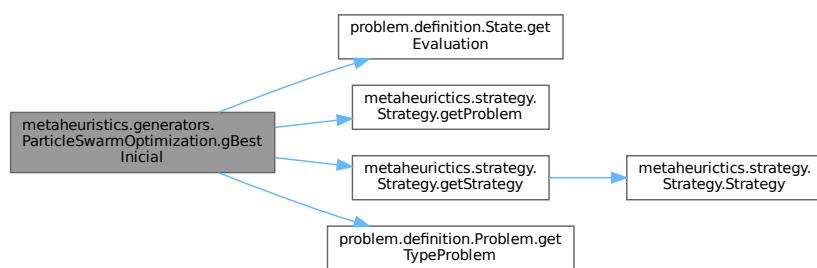


Gráfico de llamadas a esta función:



### 8.75.3.3 generate()

```
State metaheuristics.generators.ParticleSwarmOptimization.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

Generate a new state by advancing the next particle in the swarm.

#### Parámetros

<i>operatornumber</i>	unused for PSO (kept for API compatibility)
-----------------------	---

#### Devuelve

the newly generated [State](#) produced by the advanced particle

#### Excepciones

<i>ReflectiveOperationException</i>	forwarded from nested generator calls
-------------------------------------	---------------------------------------

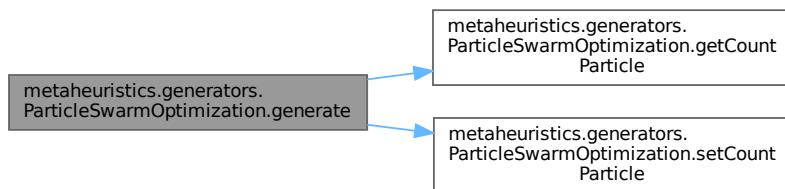
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 157 del archivo [ParticleSwarmOptimization.java](#).

```
00157     { // PSO
00158         if (getCountParticle() >= countRef) {
00159             setCountParticle(0);
00160         }
00161         // Advance the current particle and return its active state.
00162         listParticle.get(getCountParticle()).generate(1);
00163         return listParticle.get(getCountParticle()).getStateActual();
00164     }
```

Hace referencia a [countRef](#), [getCountParticle\(\)](#), [listParticle](#) y [setCountParticle\(\)](#).

Gráfico de llamadas de esta función:



### 8.75.3.4 getCountCurrentIterPSO()

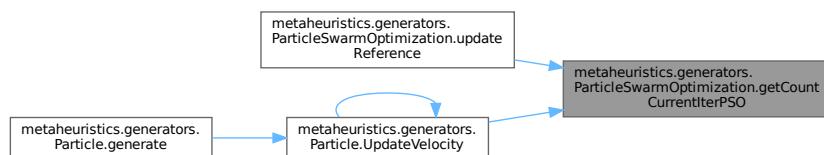
```
int metaheuristics.generators.ParticleSwarmOptimization.getCountCurrentIterPSO () [inline],  
[static]
```

Definición en la línea 114 del archivo [ParticleSwarmOptimization.java](#).

```
00114 { return countCurrentIterPSO; }
```

Referenciado por [updateReference\(\)](#) y [metaheuristics.generators.Particle.UpdateVelocity\(\)](#).

Gráfico de llamadas a esta función:



### 8.75.3.5 getCountParticle()

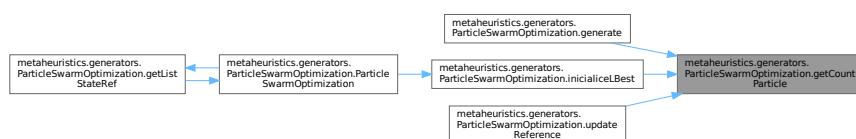
```
int metaheuristics.generators.ParticleSwarmOptimization.getCountParticle () [inline], [static]
```

Definición en la línea 116 del archivo [ParticleSwarmOptimization.java](#).

```
00116 { return countParticle; }
```

Referenciado por [generate\(\)](#), [inicialiceLBest\(\)](#) y [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.75.3.6 getCountParticleBySwarm()

```
int metaheuristics.generators.ParticleSwarmOptimization.getCountParticleBySwarm () [inline],  
[static]
```

`getCountParticleBySwarm` - get the number of particles per swarm.

Devuelve

the number of particles per swarm

Definición en la línea 54 del archivo [ParticleSwarmOptimization.java](#).

```
00054 {  
00055     return countParticleBySwarm;  
00056 }
```

Hace referencia a [countParticleBySwarm](#).

### 8.75.3.7 getCountRef()

```
int metaheuristics.generators.ParticleSwarmOptimization.getCountRef () [inline], [static]
```

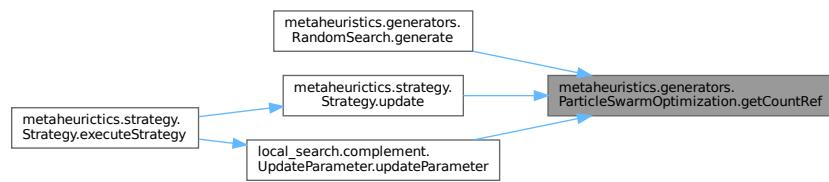
Definición en la línea 344 del archivo [ParticleSwarmOptimization.java](#).

```
00344             {
00345         return countRef;
00346     }
```

Hace referencia a [countRef](#).

Referenciado por [metaheuristics.generators.RandomSearch.generate\(\)](#), [metaheuristics.strategy.Strategy.update\(\)](#) y [local\\_search.complement.UpdateParameter.updateParameter\(\)](#).

Gráfico de llamadas a esta función:



### 8.75.3.8 getCoutSwarm()

```
int metaheuristics.generators.ParticleSwarmOptimization.getCoutSwarm () [inline], [static]
```

`getCoutSwarm` - get the number of swarms.

Devuelve

return the number of swarms

Definición en la línea 35 del archivo [ParticleSwarmOptimization.java](#).

```
00035             {
00036         return coutSwarm;
00037     }
```

Hace referencia a [coutSwarm](#).

### 8.75.3.9 getGBest()

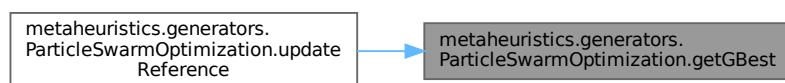
```
State metaheuristics.generators.ParticleSwarmOptimization.getGBest () [inline], [static]
```

Definición en la línea 112 del archivo [ParticleSwarmOptimization.java](#).

```
00112 { return gBest; }
```

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.75.3.10 getGeneratorType()

```
GeneratorType metaheuristics.generators.ParticleSwarmOptimization.getGeneratorType () [inline]
```

Get the generator type associated with this instance.

Devuelve

generator type ([ParticleSwarmOptimization](#))

Definición en la línea 331 del archivo [ParticleSwarmOptimization.java](#).

```
00331 {  
00332     return generatorType;  
00333 }
```

Hace referencia a [generatorType](#).

### 8.75.3.11 getLBest()

```
State[] metaheuristics.generators.ParticleSwarmOptimization.getLBest () [inline], [static]
```

Definición en la línea 80 del archivo [ParticleSwarmOptimization.java](#).

```
00080 {  
00081     if (lBest == null) return new State[0];  
00082     State[] copy = new State[lBest.length];  
00083     for (int i = 0; i < lBest.length; i++) {  
00084         copy[i] = (lBest[i] == null) ? null : lBest[i].clone();  
00085     }  
00086     return copy;  
00087 }
```

### 8.75.3.12 getListCountBetterGender()

```
int[] metaheuristics.generators.ParticleSwarmOptimization.getListCountBetterGender () [inline]
```

Return a defensive copy of the counters that track how many times a particle improved during a period.

Devuelve

copy of listCountBetterGender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 506 del archivo [ParticleSwarmOptimization.java](#).

```
00506 {  
00507     return (this.listCountBetterGender == null) ? new int[0] :  
00508         Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);  
00509 }
```

Hace referencia a [listCountBetterGender](#).

### 8.75.3.13 getListCountGender()

```
int[ ] metaheuristics.generators.ParticleSwarmOptimization.getListCountGender () [inline]
```

Return a defensive copy of usage counters per period.

Devuelve

copy of listCountGender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 516 del archivo [ParticleSwarmOptimization.java](#).

```
00516                               {
00517         return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00518             this.listCountGender.length);
00518     }
```

Hace referencia a [listCountGender](#).

### 8.75.3.14 getListParticle()

```
List< Particle > metaheuristics.generators.ParticleSwarmOptimization.getListParticle () [inline]
```

Get the internal particle list.

Devuelve

internal list of [Particle](#) instances

Definición en la línea 311 del archivo [ParticleSwarmOptimization.java](#).

```
00311                               {
00312         return listParticle;
00313     }
```

Hace referencia a [listParticle](#).

Referenciado por [getListStateRef\(\)](#) y [getListStateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.75.3.15 getListStateRef()

```
List< Particle > metaheuristics.generators.ParticleSwarmOptimization.getListStateRef () [inline],  
[private]
```

Build or return the list of `Particle` instances used by PSO.

If an existing PSO generator instance has an initialized particle list, that list is reused. Otherwise, states from `RandomSearch#listStateReference` are converted into particles.

Devuelve

list of particles representing the swarm population

Definición en la línea 220 del archivo `ParticleSwarmOptimization.java`.

```
00220           {
00221             Boolean found = false;
00222             List<String> key = Strategy.getStrategy().getListKey();
00223             int count = 0;
00224             if(RandomSearch.listStateReference.size() == 0){
00225               return this.setListParticle(new ArrayList<Particle>());
00226             }
00227             while((found.equals(false)) && (Strategy.getStrategy().mapGenerators.size() > count)){
00228               //recorrer la lista de generadores, hasta que encuentre el PSO
00229               if(key.get(count).equals(GeneratorType.ParticleSwarmOptimization.toString())){
00230                 //creo el generador PSO, y si su lista de particulas esta vacia entonces es la primera
00231                 vez que lo estoy creando, y cada estado lo convierto en particulas
00232                 GeneratorType keyGenerator = GeneratorType.valueOf(String.valueOf(key.get(count)));
00233                 ParticleSwarmOptimization generator = (ParticleSwarmOptimization)
00234                   Strategy.getStrategy().mapGenerators.get(keyGenerator);
00235                 if(generator.getListParticle().isEmpty()){
00236                   //convertir los estados en particulas
00237                   for (int j = 0; j < RandomSearch.listStateReference.size(); j++) {
00238                     //si el estado es creado con el generador RandomSearch entonces la convierto
00239                     en particula
00240                     if(getListParticle().size() != countRef){
00241                       ArrayList<Object> velocity = new ArrayList<Object>();
00242                         State stateAct = (State) RandomSearch.listStateReference.get(j).getCopy();
00243                         stateAct.setCode(new
00244                           ArrayList<Object>(RandomSearch.listStateReference.get(j).getCode()));
00245                         stateAct.setEvaluation(RandomSearch.listStateReference.get(j).getEvaluation());
00246                         State statePBest = (State)
00247                           RandomSearch.listStateReference.get(j).getCopy();
00248                         statePBest.setCode(new
00249                           ArrayList<Object>(RandomSearch.listStateReference.get(j).getCode()));
00250                         statePBest.setEvaluation(RandomSearch.listStateReference.get(j).getEvaluation());
00251                         Particle particle = new Particle(stateAct, statePBest, velocity);
00252                         getListParticle().add(particle);
00253                     }
00254                   }
00255                 }
00256               }
00257             count++;
00258           }
00259         return getListParticle();
00260     }
```

Hace referencia a `countRef`, `metaheuristics.strategy.Strategy.getListKey()`, `getListParticle()`, `getListStateReference()`, `metaheuristics.strategy.Strategy.getStrategy()`, `metaheuristics.generators.RandomSearch.listStateReference`, `metaheuristics.strategy.Strategy.mapGenerators`, `metaheuristics.generators.GeneratorType.ParticleSwarmOptimization`, `ParticleSwarmOptimization()`, `problem.definition.State.setCode()`, `problem.definition.State.setEvaluation()` y `setListParticle()`.

Referenciado por `ParticleSwarmOptimization()`.

Gráfico de llamadas de esta función:

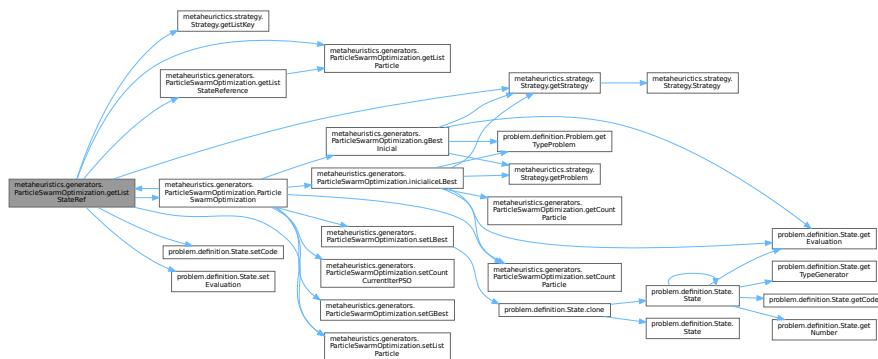
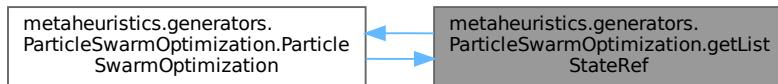


Gráfico de llamadas a esta función:



### 8.75.3.16 getListStateReference()

```
List< Particle > metaheuristics.generators.ParticleSwarmOptimization.getListStateReference ()  
[inline]
```

Return the current list of particles used internally.

The returned list is the internal storage reference; callers that need mutability isolation should copy it.

**Devuelve**

internal list of particles

Definición en la línea 292 del archivo [ParticleSwarmOptimization.java](#).

```
00292  
00293     return this.getListParticle();  
00294 }
```

Hace referencia a [getListParticle\(\)](#).

Referenciado por [getListStateRef\(\)](#).

Gráfico de llamadas de esta función:

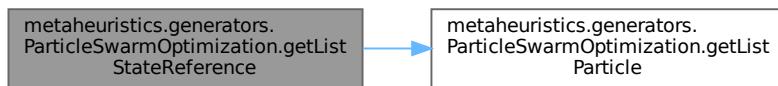


Gráfico de llamadas a esta función:



### 8.75.3.17 getReference()

`State` `metaheuristics.generators.ParticleSwarmOptimization.getReference ()` [inline]

Return the single reference state used by single-reference generators.

PSO manages multiple references (per-swarm and global), so this method returns `null` by design in the current implementation.

Devuelve

`null` in this PSO implementation

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 207 del archivo [ParticleSwarmOptimization.java](#).

```

00207           {
00208             return null;
00209         }
  
```

### 8.75.3.18 getReferenceList()

`List<State>` `metaheuristics.generators.ParticleSwarmOptimization.getReferenceList ()` [inline]

Return a copy of the list used as reference solutions by PSO.

Devuelve

copy of internal reference list

Reimplementado de [metaheuristics.generators.Generator](#).

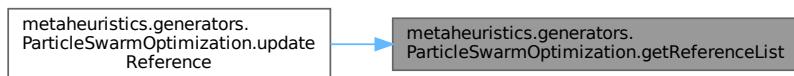
Definición en la línea 448 del archivo [ParticleSwarmOptimization.java](#).

```
00448         {
00449             return new ArrayList<State>(this.listStateReference);
00450         }
```

Hace referencia a [listStateReference](#).

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.75.3.19 getSonList()

```
List< State > metaheuristics.generators.ParticleSwarmOptimization.getSonList () [inline]
```

Return the currently generated offspring/state list for this generator.

PSO does not maintain a son list in the classical sense and thus returns null.

Devuelve

null for PSO

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 460 del archivo [ParticleSwarmOptimization.java](#).

```
00460         {
00461             return null;
00462         }
```

### 8.75.3.20 getStateReferencePSO()

```
State metaheuristics.generators.ParticleSwarmOptimization.getStateReferencePSO () [inline]
```

Return the PSO-specific reference state used by some helpers.

Devuelve

stored stateReferencePSO (may be null)

Definición en la línea 268 del archivo [ParticleSwarmOptimization.java](#).

```
00268         {
00269             return stateReferencePSO;
00270         }
```

Hace referencia a [stateReferencePSO](#).

### 8.75.3.21 getTrace()

```
float[] metaheuristics.generators.ParticleSwarmOptimization.getTrace () [inline]
```

Return a copy of the internal trace array used for diagnostic information (weights over time, etc.).

Devuelve

copy of listTrace

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 527 del archivo [ParticleSwarmOptimization.java](#).

```
00527         {
00528             return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00529                 this.listTrace.length);
00529         }
```

Hace referencia a [listTrace](#).

### 8.75.3.22 getType()

```
GeneratorType metaheuristics.generators.ParticleSwarmOptimization.getType () [inline]
```

Return the concrete generator type of this generator.

Devuelve

the [GeneratorType](#) for this generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 438 del archivo [ParticleSwarmOptimization.java](#).

```
00438         {
00439             return this.generatorType;
00440         }
```

Hace referencia a [generatorType](#).

### 8.75.3.23 getWeight()

```
float metaheuristics.generators.ParticleSwarmOptimization.getWeight () [inline]
```

Get the current inertia weight used in PSO.

Devuelve

current weight value

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 495 del archivo [ParticleSwarmOptimization.java](#).

```
00495         {
00496             return this.weight;
00497         }
```

Hace referencia a [weight](#).

### 8.75.3.24 inicialiceLBest()

```
void metaheuristics.generators.ParticleSwarmOptimization.inicialiceLBest () [inline]
```

Initialize the local best lBest array for each swarm.

The method inspects particle personal-bests within each swarm and selects the best among them according to the problem's objective type (maximize or minimize).

Definición en la línea 173 del archivo [ParticleSwarmOptimization.java](#).

```
00173      {
00174          for (int j = 0; j < coutSwarm; j++) {
00175              // pick initial reference from the particle's personal best
00176              State reference = listParticle.getCountParticle().getStatePBest();
00177              int iterator = countParticleBySwarm + getCountParticle();
00178              if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)) {
00179                  for (int i = countParticle; i < iterator; i++) {
00180                      if (listParticle.get(i).getStatePBest().getEvaluation().get(0) >
reference.getEvaluation().get(0))
00181                          reference = listParticle.get(i).getStatePBest();
00182                      setCountParticle(getCountParticle() + 1);
00183                  }
00184              } else{
00185                  for (int i = countParticle; i < iterator; i++) {
00186                      if (listParticle.get(i).getStatePBest().getEvaluation().get(0) <
reference.getEvaluation().get(0))
00187                          reference = listParticle.get(i).getStatePBest();
00188                      setCountParticle(getCountParticle() + 1);
00189                  }
00190              }
00191          }
00192          lBest[j] = reference;
00193      }
```

Hace referencia a [countParticleBySwarm](#), [coutSwarm](#), [getCountParticle\(\)](#), [problem.definition.State.getEvaluation\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Problem.getTypeProblem\(\)](#), [listParticle](#), [problem.definition.Problem.ProblemType.Maximizar](#) y [setCountParticle\(\)](#).

Referenciado por [ParticleSwarmOptimization\(\)](#).

Gráfico de llamadas de esta función:

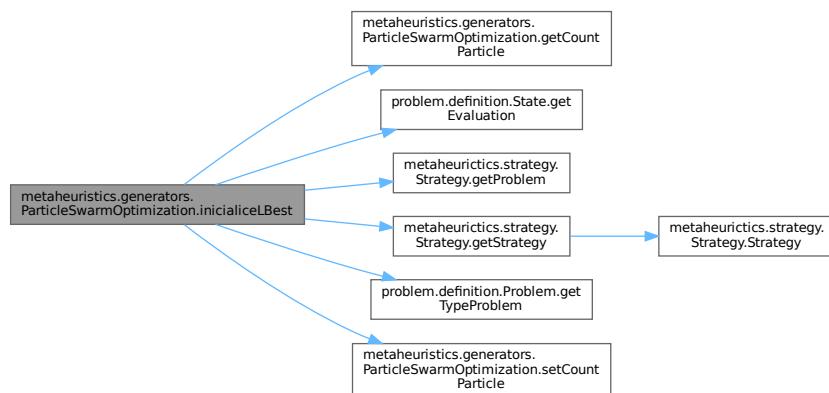
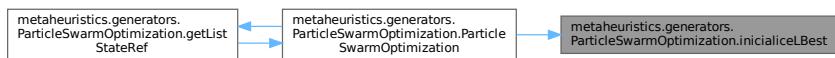


Gráfico de llamadas a esta función:



### 8.75.3.25 setCountCurrentIterPSO()

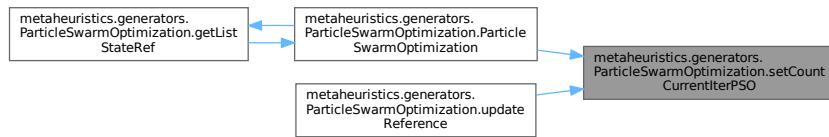
```
void metaheuristics.generators.ParticleSwarmOptimization.setCountCurrentIterPSO (
    int v) [inline], [static]
```

Definición en la línea 115 del archivo [ParticleSwarmOptimization.java](#).

```
00115 { countCurrentIterPSO = v; }
```

Referenciado por [ParticleSwarmOptimization\(\)](#) y [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.75.3.26 setCountParticle()

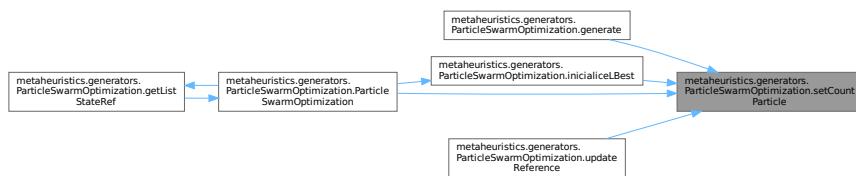
```
void metaheuristics.generators.ParticleSwarmOptimization.setCountParticle (
    int v) [inline], [static]
```

Definición en la línea 117 del archivo [ParticleSwarmOptimization.java](#).

```
00117 { countParticle = v; }
```

Referenciado por [generate\(\)](#), [inicialiceLBest\(\)](#), [ParticleSwarmOptimization\(\)](#) y [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.75.3.27 setCountParticleBySwarm()

```
void metaheuristics.generators.ParticleSwarmOptimization.setCountParticleBySwarm (
    int v) [inline], [static]
```

`setCountParticleBySwarm` - set the number of particles per swarm.

## Parámetros

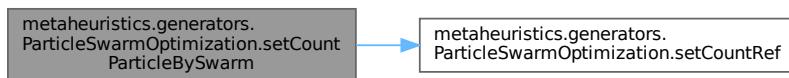
v	the number of particles per swarm to set
---	--

Definición en la línea 61 del archivo [ParticleSwarmOptimization.java](#).

```
00061         if (v < 0) throw new IllegalArgumentException("countParticleBySwarm must be >= 0");
00062         countParticleBySwarm = v;
00063         // keep dependent countRef in sync
00064         setCountRef(coutSwarm * countParticleBySwarm);
00065     }
00066 }
```

Hace referencia a [countParticleBySwarm](#), [coutSwarm](#) y [setCountRef\(\)](#).

Gráfico de llamadas de esta función:



### 8.75.3.28 setCountRef()

```
void metaheuristics.generators.ParticleSwarmOptimization.setCountRef (
    int countRef) [inline], [static]
```

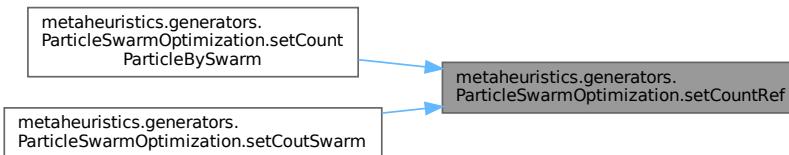
Definición en la línea 348 del archivo [ParticleSwarmOptimization.java](#).

```
00348         {
00349             ParticleSwarmOptimization.countRef = countRef;
00350         }
```

Hace referencia a [countRef](#).

Referenciado por [setCountParticleBySwarm\(\)](#) y [setCoutSwarm\(\)](#).

Gráfico de llamadas a esta función:



### 8.75.3.29 setCoutSwarm()

```
void metaheuristics.generators.ParticleSwarmOptimization.setCoutSwarm (
    int v) [inline], [static]
```

Generado por Doxygen

setCoutSwarm - set the number of swarms.

## Parámetros

<code>v</code>	the number of swarms to set
----------------	-----------------------------

Definición en la línea 42 del archivo [ParticleSwarmOptimization.java](#).

```
00042         if (v < 0) throw new IllegalArgumentException("coutSwarm must be >= 0");
00043         coutSwarm = v;
00044         // keep dependent countRef in sync
00045         setCountRef(coutSwarm * countParticleBySwarm);
00046
00047     }
```

Hace referencia a [countParticleBySwarm](#), [coutSwarm](#) y [setCountRef\(\)](#).

Gráfico de llamadas de esta función:



### 8.75.3.30 setGBest()

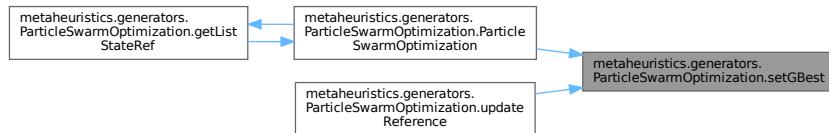
```
void metaheuristics.generators.ParticleSwarmOptimization.setGBest (
    State s) [inline], [static]
```

Definición en la línea 113 del archivo [ParticleSwarmOptimization.java](#).

```
00113 { gBest = s; }
```

Referenciado por [ParticleSwarmOptimization\(\)](#) y [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.75.3.31 setGeneratorType()

```
void metaheuristics.generators.ParticleSwarmOptimization.setGeneratorType (
    GeneratorType generatorType) [inline]
```

Set the generator type for this instance.

## Parámetros

<i>generatorType</i>	generator type to set
----------------------	-----------------------

Definición en la línea 340 del archivo [ParticleSwarmOptimization.java](#).

```
00340
00341     this.generatorType = generatorType;
00342 }
```

Hace referencia a [generatorType](#).

### 8.75.3.32 setInitialReference()

```
void metaheuristics.generators.ParticleSwarmOptimization.setInitialReference (
    State stateInitialRef) [inline]
```

Set the initial reference(s) for the generator.

PSO uses swarm references and therefore this method is intentionally left as a no-op or to be implemented by callers that bridge different data.

## Parámetros

<i>stateInitialRef</i>	initial state to use as a reference
------------------------	-------------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 428 del archivo [ParticleSwarmOptimization.java](#).

```
00428
00429     // Intentionally left blank for PSO (no single initial reference)
00430 }
```

### 8.75.3.33 setLBest()

```
void metaheuristics.generators.ParticleSwarmOptimization.setLBest (
    State[] arr) [inline], [static]
```

setLBest - set the local best array.

## Parámetros

<i>arr</i>	
------------	--

Definición en la línea 92 del archivo [ParticleSwarmOptimization.java](#).

```
00092
00093     if (arr == null) {
00094         lBest = null;
00095         return;
00096     }
00097     lBest = new State[arr.length];
00098     for (int i = 0; i < arr.length; i++) {
00099         lBest[i] = (arr[i] == null) ? null : arr[i].clone();
00100     }
00101 }
```

Hace referencia a [problem.definition.State.clone\(\)](#).

Referenciado por [ParticleSwarmOptimization\(\)](#).

Gráfico de llamadas de esta función:

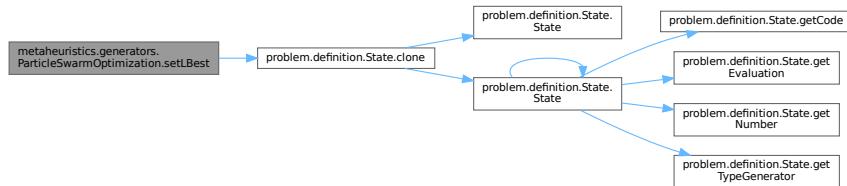
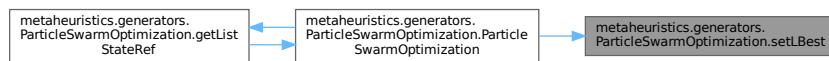


Gráfico de llamadas a esta función:



### 8.75.3.34 setLBestAt()

```

void metaheuristics.generators.ParticleSwarmOptimization.setLBestAt (
    int idx,
    State s) [inline], [static]
  
```

**setLBest** - set the local best at index.

#### Parámetros

idx	
s	

Definición en la línea 107 del archivo [ParticleSwarmOptimization.java](#).

```

00107
00108      if (lBest == null) return;
00109      if (idx < 0 || idx >= lBest.length) return;
00110      lBest[idx] = (s == null) ? null : s.clone();
00111  }
  
```

Hace referencia a [problem.definition.State.clone\(\)](#).

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas de esta función:

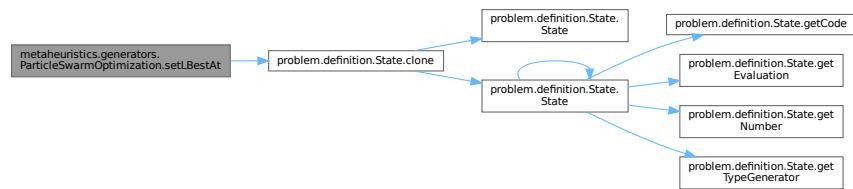
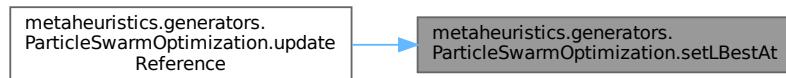


Gráfico de llamadas a esta función:



### 8.75.3.35 setListParticle()

```
List< Particle > metaheuristics.generators.ParticleSwarmOptimization.setListParticle (
    List< Particle > listParticle) [inline]
```

Replace the internal particle list and return the newly set list.

#### Parámetros

<i>listParticle</i>	list to install
---------------------	-----------------

#### Devuelve

the installed list

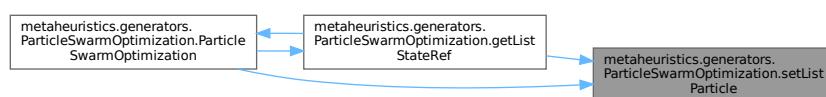
Definición en la línea 321 del archivo [ParticleSwarmOptimization.java](#).

```
00321
00322     this.listParticle = listParticle;
00323     return listParticle;
00324 }
```

Hace referencia a [listParticle](#).

Referenciado por [getListStateRef\(\)](#) y [ParticleSwarmOptimization\(\)](#).

Gráfico de llamadas a esta función:



### 8.75.3.36 setListStateReference()

```
void metaheuristics.generators.ParticleSwarmOptimization.setListStateReference (
    List< State > listStateReference) [inline]
```

Replace the internal state reference list.

Provided for API compatibility; most callers should use [setListParticle](#) instead.

#### Parámetros

<a href="#">listStateReference</a>	new list of states
------------------------------------	--------------------

Definición en la línea 302 del archivo [ParticleSwarmOptimization.java](#).

```
00302
00303     this.listStateReference = listStateReference;
00304 }
```

Hace referencia a [listStateReference](#).

### 8.75.3.37 setStateReferencePSO()

```
void metaheuristics.generators.ParticleSwarmOptimization.setStateReferencePSO (
    State stateReferencePSO) [inline]
```

Set the PSO-specific reference state.

A defensive copy is not performed because [State](#) provides its own copy/clone methods; callers should clone if they want isolation.

#### Parámetros

<a href="#">stateReferencePSO</a>	the state to store as PSO reference
-----------------------------------	-------------------------------------

Definición en la línea 280 del archivo [ParticleSwarmOptimization.java](#).

```
00280
00281     this.stateReferencePSO = stateReferencePSO;
00282 }
```

Hace referencia a [stateReferencePSO](#).

### 8.75.3.38 setWeight()

```
void metaheuristics.generators.ParticleSwarmOptimization.setWeight (
    float weight) [inline]
```

Set the internal inertia weight used by particle velocity updates.

#### Parámetros

---

<i>weight</i>	inertia weight value
---------------	----------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 485 del archivo [ParticleSwarmOptimization.java](#).

```
00485
00486     this.weight = weight;
00487 }
```

Hace referencia a [weight](#).

### 8.75.3.39 updateReference()

```
void metaheuristics.generators.ParticleSwarmOptimization.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

Update the PSO internal references based on a newly generated candidate state.

This method updates the particle personal-bests and possibly the local/global bests depending on the objective.

#### Parámetros

<i>stateCandidate</i>	newly generated candidate state
<i>countIterationsCurrent</i>	current iteration counter

#### Excepciones

<i>ReflectiveOperationException</i>	forwarded from nested calls
-------------------------------------	-----------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 364 del archivo [ParticleSwarmOptimization.java](#).

```
00364
00365     {
00366         Particle particle = listParticle.get(countParticle);
00367         int swarm = countParticle/countParticleBySwarm;
00368         if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00369             if ((lBest[swarm]).getEvaluation().get(0) <
00370                 particle.getStatePBest().getEvaluation().get(0)){
00371                 setLBestAt(swarm, particle.getStatePBest());
00372                 if(lBest[swarm].getEvaluation().get(0) >
00373                     getReferenceList().get(getReferenceList().size() - 1).getEvaluation().get(0)){
00374                     State tmp = new State();
00375                     tmp.setCode(new ArrayList<Object>(lBest[swarm].getCode()));
00376                     tmp.setEvaluation(lBest[swarm].getEvaluation());
00377                     tmp.setTypeGenerator(lBest[swarm].getTypeGenerator());
00378                     setGBest(tmp);
00379                 }
00380             }
00381             else {
00382                 particle.updateReference(stateCandidate, countIterationsCurrent);
00383                 if ((lBest[swarm]).getEvaluation().get(0) >
00384                     particle.getStatePBest().getEvaluation().get(0)){
00385                     setLBestAt(swarm, particle.getStatePBest());
00386                 }
00387             }
00388         }
00389     }
```

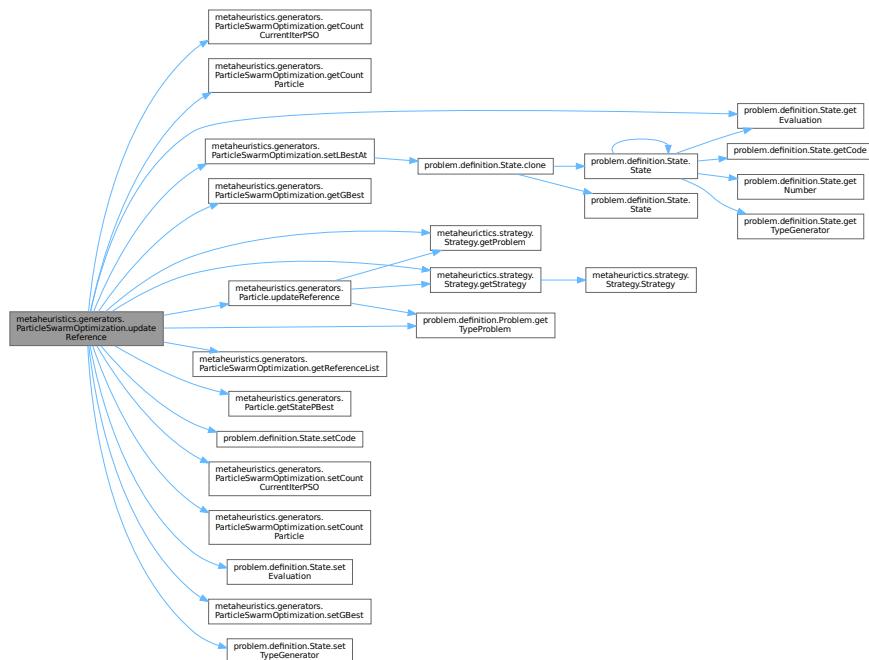
```

00383             if(lBest[swarm].getEvaluation().get(0) <
00384                 getReferenceList().get(getReferenceList().size() - 1).getEvaluation().get(0)){
00385                 State tmp = new State();
00386                 tmp.setCode(new ArrayList<Object>(lBest[swarm].getCode()));
00387                 tmp.setEvaluation(lBest[swarm].getEvaluation());
00388                 tmp.setTypeGenerator(lBest[swarm].getTypeGenerator());
00389                 setGBest(tmp);
00390             }
00391         }
00392         listStateReference.add(getGBest());
00393         setCountParticle(getCountParticle() + 1);
00394         setCountCurrentIterPSO(getCountCurrentIterPSO() + 1);
00395     }

```

Hace referencia a [countParticleBySwarm](#), [getCountCurrentIterPSO\(\)](#), [getCountParticle\(\)](#), [problem.definition.State.getEvaluation\(\)](#), [getGBest\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [getReferenceList\(\)](#), [metaheuristics.generators.Particle.getStatePBest\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Problem.getTypeProblem\(\)](#), [listParticle](#), [listStateReference](#), [problem.definition.ProblemType.Maximizar](#), [problem.definition.State.setCode\(\)](#), [setCountCurrentIterPSO\(\)](#), [setCountParticle\(\)](#), [problem.definition.State.setEvaluation\(\)](#), [setGBest\(\)](#), [setLBestAt\(\)](#), [problem.definition.State.setTypeGenerator\(\)](#) y [metaheuristics.generators.Particle.updateReference\(\)](#).

Gráfico de llamadas de esta función:



## 8.75.4 Documentación de datos miembro

### 8.75.4.1 binary

```
final boolean metaheuristics.generators.ParticleSwarmOptimization.binary = false [static]
```

Definición en la línea 73 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [metaheuristics.generators.Particle.UpdateCode\(\)](#).

#### 8.75.4.2 countParticleBySwarm

```
int metaheuristics.generators.ParticleSwarmOptimization.countParticleBySwarm = 0 [static],  
[private]
```

Definición en la línea 49 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [getCountParticleBySwarm\(\)](#), [inicialiceLBest\(\)](#), [setCountParticleBySwarm\(\)](#), [setCoutSwarm\(\)](#) y [updateReference\(\)](#).

#### 8.75.4.3 countRef

```
int metaheuristics.generators.ParticleSwarmOptimization.countRef = coutSwarm * countParticleBySwarm  
[static], [private]
```

Definición en la línea 67 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [generate\(\)](#), [getCountRef\(\)](#), [getListStateRef\(\)](#) y [setCountRef\(\)](#).

#### 8.75.4.4 coutSwarm

```
int metaheuristics.generators.ParticleSwarmOptimization.coutSwarm = 0 [static], [private]
```

Definición en la línea 30 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [getCoutSwarm\(\)](#), [inicialiceLBest\(\)](#), [ParticleSwarmOptimization\(\)](#), [setCountParticleBySwarm\(\)](#) y [setCoutSwarm\(\)](#).

#### 8.75.4.5 generatorType

```
GeneratorType metaheuristics.generators.ParticleSwarmOptimization.generatorType [private]
```

Definición en la línea 24 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [getGeneratorType\(\)](#), [getType\(\)](#) y [setGeneratorType\(\)](#).

#### 8.75.4.6 learning1

```
final int metaheuristics.generators.ParticleSwarmOptimization.learning1 = 2 [static]
```

Definición en la línea 71 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [metaheuristics.generators.Particle.UpdateVelocity\(\)](#).

#### 8.75.4.7 learning2

```
final int metaheuristics.generators.ParticleSwarmOptimization.learning2 = 2 [static]
```

Definición en la línea 71 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [metaheuristics.generators.Particle.UpdateVelocity\(\)](#).

#### 8.75.4.8 listCountBetterGender

```
int [] metaheuristics.generators.ParticleSwarmOptimization.listCountBetterGender = new int[10]  
[private]
```

Definición en la línea 120 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [getListCountBetterGender\(\)](#) y [ParticleSwarmOptimization\(\)](#).

#### 8.75.4.9 listCountGender

```
int [] metaheuristics.generators.ParticleSwarmOptimization.listCountGender = new int[10] [private]
```

Definición en la línea 121 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [getListCountGender\(\)](#) y [ParticleSwarmOptimization\(\)](#).

#### 8.75.4.10 listParticle

```
List<Particle> metaheuristics.generators.ParticleSwarmOptimization.listParticle = new Array←  
List<Particle> () [private]
```

Definición en la línea 23 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [generate\(\)](#), [getListParticle\(\)](#), [inicialiceLBest\(\)](#), [ParticleSwarmOptimization\(\)](#), [setListParticle\(\)](#) y [updateReference\(\)](#).

#### 8.75.4.11 listStateReference

```
List<State> metaheuristics.generators.ParticleSwarmOptimization.listStateReference = new  
ArrayList<State>() [private]
```

Definición en la línea 22 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [getReferenceList\(\)](#), [setListStateReference\(\)](#) y [updateReference\(\)](#).

#### 8.75.4.12 listTrace

```
float [] metaheuristics.generators.ParticleSwarmOptimization.listTrace = new float[1200000]  
[private]
```

Definición en la línea 122 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [getTrace\(\)](#) y [ParticleSwarmOptimization\(\)](#).

#### 8.75.4.13 stateReferencePSO

```
State metaheuristics.generators.ParticleSwarmOptimization.stateReferencePSO [private]
```

Definición en la línea 21 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [getStateReferencePSO\(\)](#) y [setStateReferencePSO\(\)](#).

#### 8.75.4.14 weight

```
float metaheuristics.generators.ParticleSwarmOptimization.weight = 50 [private]
```

Definición en la línea 68 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [getWeight\(\)](#), [ParticleSwarmOptimization\(\)](#) y [setWeight\(\)](#).

#### 8.75.4.15 wmax

```
final double metaheuristics.generators.ParticleSwarmOptimization.wmax = 0.9 [static]
```

Definición en la línea 69 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [metaheuristics.generators.Particle.UpdateVelocity\(\)](#).

#### 8.75.4.16 wmin

```
final double metaheuristics.generators.ParticleSwarmOptimization.wmin = 0.2 [static]
```

Definición en la línea 70 del archivo [ParticleSwarmOptimization.java](#).

Referenciado por [metaheuristics.generators.Particle.UpdateVelocity\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [ParticleSwarmOptimization.java](#)

## 8.76 Referencia de la clase

### **evolutionary\_algorithms.complement.ProbabilisticSampling**

[ProbabilisticSampling](#) - applies probabilistic sampling to select individuals.

Diagrama de herencia de `evolutionary_algorithms.complement.ProbabilisticSampling`

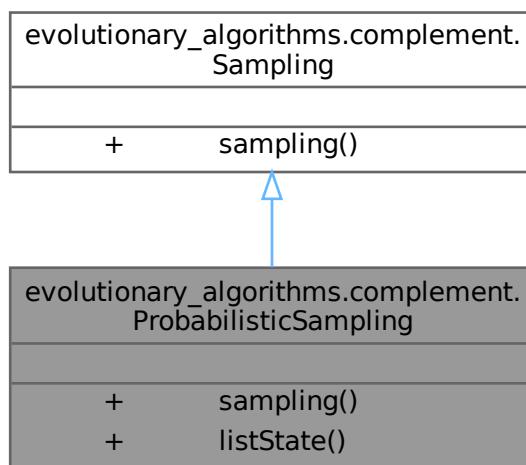
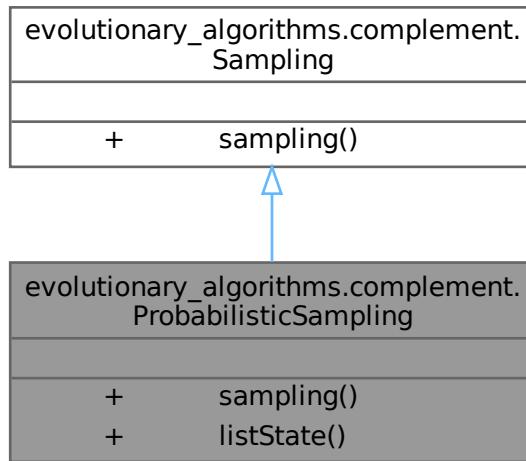


Diagrama de colaboración de evolutionary\_algorithms.complement.ProbabilisticSampling:



## Métodos públicos

- List< State > **sampling** (List< State > fathers, int countInd)  
*Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.*
- List< State > **listState** (int countInd)  
*listState - initializes the list of individuals.*

### 8.76.1 Descripción detallada

[ProbabilisticSampling](#) - applies probabilistic sampling to select individuals.

Definición en la línea 17 del archivo [ProbabilisticSampling.java](#).

### 8.76.2 Documentación de funciones miembro

#### 8.76.2.1 listState()

```
List< State > evolutionary_algorithms.complement.ProbabilisticSampling.listState (
    int countInd) [inline]
```

listState - initializes the list of individuals.

#### Parámetros

<i>countInd</i>	
-----------------	--

Devuelve

Definición en la línea 95 del archivo [ProbabilisticSampling.java](#).

```

00095         {
00096             List<State> staList = new ArrayList<State>(countInd);
00097             for (int i = 0; i < countInd; i++) {
00098                 State state = new State();
00099                 state.setCode(new ArrayList<Object>());
00100                 state.setNumber(Strategy.getStrategy().getCountCurrent());
00101                 state.setTypeGenerator(GeneratorType.DistributionEstimationAlgorithm);
00102                 staList.add(state);
00103             }
00104             return staList;
00105         }

```

Hace referencia a [metaheuristics.generators.GeneratorType.DistributionEstimationAlgorithm](#), [metaheuristics.strategy.Strategy.getCountCurrent\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.State.setCode\(\)](#), [problem.definition.State.setNumber\(\)](#) y [problem.definition.State.setTypeGenerator\(\)](#).

Referenciado por [sampling\(\)](#).

Gráfico de llamadas de esta función:

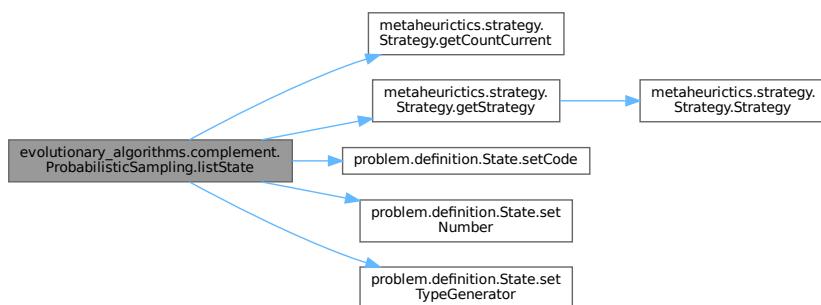


Gráfico de llamadas a esta función:



### 8.76.2.2 sampling()

```

List< State > evolutionary_algorithms.complement.ProbablisticSampling.sampling (
    List< State > fathers,
    int countInd) [inline]

```

Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.

This RNG is used for evolutionary algorithm operations (crossover point, selection between children, etc.) and is not used for security-sensitive purposes. Therefore we suppress the Sonar security hotspot S2245 here. sampling - applies probabilistic sampling to select individuals.

## Parámetros

<i>fathers</i>	
<i>countInd</i>	

## Devuelve

returns a list of sampled individuals

Reimplementado de [evolutionary\\_algorithms.complement.Sampling](#).

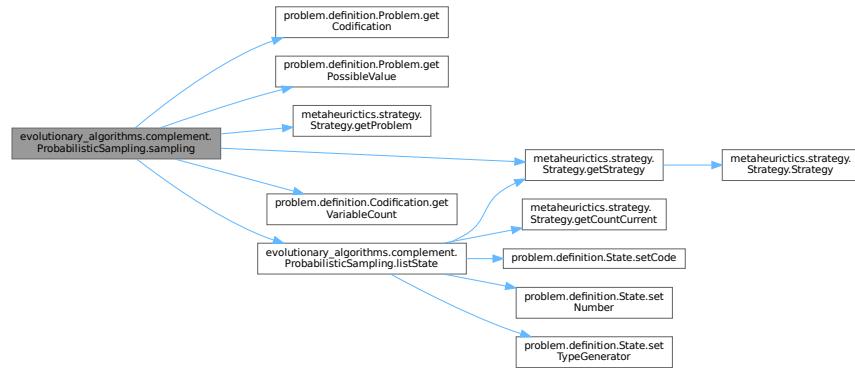
Definición en la línea 33 del archivo [ProbabilisticSampling.java](#).

```

00033
00034     // TODO Auto-generated method stub
00035     int cantV = fathers.get(0).getCode().size();
00036     List<State> staList = listState(countInd);
00037     for (int i = 0; i < cantV; i++) {
00038         Object[] values = new Object[father.size()]; // arreglo de valores de una variable
00039         Object[] arrtemp = new Object[Strategy.getStrategy().getProblem().getPossibleValue()];
00040         //llenar el arreglo con todos los valores posibles
00041         for (int j = 0; j < arrtemp.length; j++) {
00042             arrtemp[j] = j;
00043         }
00044         for (int j = 0; j < values.length; j++) {
00045             values[j] = fathers.get(j).getCode().get(i);
00046         }
00047         int k = 0;
00048         int sum = 0; // suma acumulativa por cada valor posible
00049         int arrOcc[] = new int[arrtemp.length]; // arreglo paralelo para contar la cantidad de
ocurrencias de un valor posible
00050         //llenar el arreglo con la cantidad de ocurrencias de cada valor posible, para cada
variable
00051         //recorrer el arreglo de valores de una variable
00052         while (k < arrtemp.length) {
00053             int count = 0;
00054             for (int j = 0; j < values.length; j++) {
00055                 if((Integer)values[j] != -1 && values[j] == arrtemp[k]){ ///
00056                     count++;
00057                     values[j] = -1;
00058                 }
00059             }
00060             arrOcc[k] = count;
00061             sum = sum + count;
00062             k++;
00063         }
00064         for (int l = 0; l < countInd; l++) {
00065             boolean find = false;
00066             int p = 0;
00067             int random;
00068             if (sum > 0) random = ThreadLocalRandom.current().nextInt(sum) + 1;
00069             else random = 1;
00070             while (p < arrOcc.length && find == false) {
00071                 random = random - arrOcc[p];
00072                 if(random <= 0){
00073                     staList.get(l).getCode().add(arrtemp[p]);
00074                     find = true;
00075                 }
00076                 else p++;
00077             }
00078             if(find == false){
00079                 int bound =
Strategy.getStrategy().getProblem().getCodification().getVariableCount() * 10;
00080                 int value = (bound > 0) ? ThreadLocalRandom.current().nextInt(bound) : 0;
00081                 staList.get(l).getCode().add(Integer.valueOf(value));
00082             }
00083         }
00084     }
00085 }
00086 return staList;
00087 }
```

Hace referencia a [problem.definition.Problem.getCodification\(\)](#), [problem.definition.Problem.getPossibleValue\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Codification.getVariableCount\(\)](#) y [listState\(\)](#).

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [ProbabilisticSampling.java](#)

## 8.77 Referencia de la clase **evolutionary\_algorithms.complement.Probability**

**Probability** - represents the probability of a certain event.

Diagrama de colaboración de `evolutionary_algorithms.complement.Probability`:

<code>evolutionary_algorithms.complement.Probability</code>	
-	key
-	value
-	probability
+	getProbability()
+	setProbability()
+	getKey()
+	setKey()
+	getValue()
+	setValue()

## Métodos públicos

- float [getProbability \(\)](#)  
*getProbability - getter for the probability.*
- void [setProbability \(float probability\)](#)  
*setProbability - setter for the probability.*
- Object [getKey \(\)](#)  
*getKey - getter for the key.*
- void [setKey \(Object key\)](#)  
*setKey - setter for the key.*
- Object [getValue \(\)](#)  
*getValue - getter for the value.*
- void [setValue \(Object value\)](#)  
*setValue - setter for the value.*

## Atributos privados

- Object [key](#)
- Object [value](#)
- float [probability](#)

### 8.77.1 Descripción detallada

[Probability](#) - represents the probability of a certain event.

Definición en la línea 6 del archivo [Probability.java](#).

### 8.77.2 Documentación de funciones miembro

#### 8.77.2.1 [getKey\(\)](#)

Object evolutionary\_algorithms.complement.Probability.getKey () [inline]

getKey - getter for the key.

Devuelve

returns the key

Definición en la línea 30 del archivo [Probability.java](#).

```
00030
00031      return key;
00032 }
```

Hace referencia a [key](#).

### 8.77.2.2 getProbability()

```
float evolutionary_algorithms.complement.Probability.getProbability () [inline]
getProbability - getter for the probability.
```

Devuelve

returns the probability

Definición en la línea 16 del archivo [Probability.java](#).

```
00016             {
00017         return probability;
00018     }
```

Hace referencia a [probability](#).

### 8.77.2.3 getValue()

```
Object evolutionary_algorithms.complement.Probability.getValue () [inline]
getValue - getter for the value.
```

Devuelve

returns the value

Definición en la línea 44 del archivo [Probability.java](#).

```
00044             {
00045         return value;
00046     }
```

Hace referencia a [value](#).

### 8.77.2.4 setKey()

```
void evolutionary_algorithms.complement.Probability.setKey (
    Object key) [inline]
```

setKey - setter for the key.

#### Parámetros

<i>key</i>	
------------	--

Definición en la línea 37 del archivo [Probability.java](#).

```
00037             {
00038         this.key = key;
00039     }
```

Hace referencia a [key](#).

Referenciado por [evolutionary\\_algorithms.complement.Univariate.distribution\(\)](#).

Gráfico de llamadas a esta función:



### 8.77.2.5 setProbability()

```
void evolutionary_algorithms.complement.Probability.setProbability (
    float probability) [inline]
```

setProbability - setter for the probability.

#### Parámetros

probability	
-------------	--

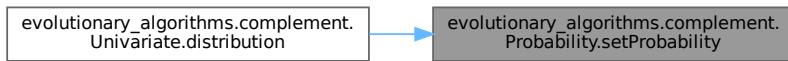
Definición en la línea 23 del archivo [Probability.java](#).

```
00023
00024     this.probability = probability;
00025 }
```

Hace referencia a [probability](#).

Referenciado por [evolutionary\\_algorithms.complement.Univariate.distribution\(\)](#).

Gráfico de llamadas a esta función:



### 8.77.2.6 setValue()

```
void evolutionary_algorithms.complement.Probability.setValue (
    Object value) [inline]
```

setValue - setter for the value.

#### Parámetros

value	
-------	--

Definición en la línea 51 del archivo [Probability.java](#).

```
00051
00052     this.value = value;
00053 }
```

Hace referencia a [value](#).

Referenciado por [evolutionary\\_algorithms.complement.Univariate.distribution\(\)](#).

Gráfico de llamadas a esta función:



### 8.77.3 Documentación de datos miembro

#### 8.77.3.1 key

Object evolutionary\_algorithms.complement.Probability.key [private]

Definición en la línea 7 del archivo [Probability.java](#).

Referenciado por [getKey\(\)](#) y [setKey\(\)](#).

#### 8.77.3.2 probability

float evolutionary\_algorithms.complement.Probability.probability [private]

Definición en la línea 9 del archivo [Probability.java](#).

Referenciado por [getProbability\(\)](#) y [setProbability\(\)](#).

#### 8.77.3.3 value

Object evolutionary\_algorithms.complement.Probability.value [private]

Definición en la línea 8 del archivo [Probability.java](#).

Referenciado por [getValue\(\)](#) y [setValue\(\)](#).

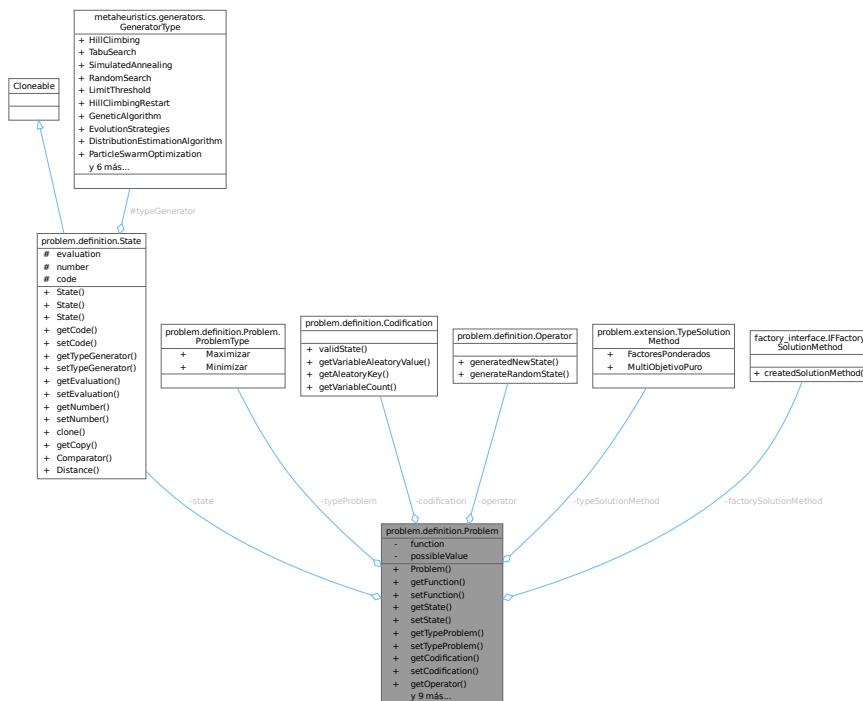
La documentación de esta clase está generada del siguiente archivo:

- [Probability.java](#)

## 8.78 Referencia de la clase problem.definition.Problem

[Problem](#).

Diagrama de colaboración de `problem.definition.Problem`:



## Clases

- enum [ProblemType](#)

*ProblemType - descripción (añade detalles).*

## Métodos públicos

- [Problem \(\)](#)

*Constructor por defecto.*

- [ArrayList< ObjetiveFunction > getFunction \(\)](#)

*Obtener la lista de funciones objetivo del problema.*

- [void setFunction \(ArrayList< ObjetiveFunction > function\)](#)

*Establecer las funciones objetivo del problema.*

- [State getState \(\)](#)

*Obtener una copia del estado actual del problema.*

- [void setState \(State state\)](#)

*Establecer el estado actual del problema (copia defensiva).*

- [ProblemType getTypeProblem \(\)](#)

*Obtener el tipo de problema (Maximizar / Minimizar).*

- [void setTypeProblem \(ProblemType typeProblem\)](#)

*Establecer el tipo de problema.*

- [Codification getCodification \(\)](#)

*Obtener la codificación usada para representar estados.*

- [void setCodification \(Codification codification\)](#)

*Establecer la codificación de variables para el problema.*

- [Operator getOperator \(\)](#)

*Obtener el operador encargado de generar nuevos estados.*

- [void setOperator \(Operator operator\)](#)

*Establecer el operador de generación de estados.*

- [int getPossibleValue \(\)](#)

*Obtener el número de valores posibles (uso dependiente de la codificación).*

- [void setPossibleValue \(int possibleValue\)](#)

*Establecer el número de valores posibles.*

- [void Evaluate \(State state\) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException](#)

*Evaluuar un estado de solución.*

- [TypeSolutionMethod getTypeSolutionMethod \(\)](#)

*Obtener el tipo de método de solución configurado.*

- [void setTypeSolutionMethod \(TypeSolutionMethod typeSolutionMethod\)](#)

*Establecer el tipo de método de solución.*

- [IFFactorySolutionMethod getFactorySolutionMethod \(\)](#)

*Obtener la fábrica de métodos de solución.*

- [void setFactorySolutionMethod \(IFFactorySolutionMethod factorySolutionMethod\)](#)

- [SolutionMethod newSolutionMethod \(TypeSolutionMethod typeSolutionMethod\) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException](#)

*Crear una instancia de [SolutionMethod](#) para el tipo solicitado.*

## Atributos privados

- ArrayList< [ObjetiveFunction](#) > function
- [State](#) state
- [ProblemType](#) typeProblem
- [Codification](#) codification
- [Operator](#) operator
- int possibleValue
- [TypeSolutionMethod](#) typeSolutionMethod
- [IFFactorySolutionMethod](#) factorySolutionMethod

## 8.78.1 Descripción detallada

[Problem](#).

Encapsula la definición del problema a resolver: funciones objetivo, codificación, operador generador de estados y método de solución.

Esta clase actúa como contenedor de las piezas necesarias para evaluar y manipular estados de solución dentro de los metaheurísticos. No modifica el comportamiento de evaluación —solo documenta y expone getters/setters usados por los generadores y estrategias.

Definición en la línea [24](#) del archivo [Problem.java](#).

## 8.78.2 Documentación de constructores y destructores

### 8.78.2.1 Problem()

```
problem.definition.Problem() [inline]
```

Constructor por defecto.

Inicializa una instancia vacía; los componentes concretos (funciones, codificación, operador) se deben injectar mediante los setters antes de usar [Evaluate\(State\)](#).

Definición en la línea [47](#) del archivo [Problem.java](#).

```
00047     super();
00048
00049 }
```

## 8.78.3 Documentación de funciones miembro

### 8.78.3.1 Evaluate()

```
void problem.definition.Problem.Evaluate(
    State state) throws IllegalArgumentException, SecurityException, ClassNotFoundException,
InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException [inline]
```

Evaluar un estado de solución.

Dependiendo del [typeSolutionMethod](#) realiza una evaluación simple usando la primera función objetivo o delega en un [SolutionMethod](#). Generado por Doxygen

## Parámetros

<code>state</code>	estado a evaluar (se modifica su lista de evaluaciones)
--------------------	---

## Excepciones

<code>ReflectiveOperationException</code>	si falla la creación reflexiva del método de solución
---	---

Definición en la línea 171 del archivo [Problem.java](#).

```

00171
00172     double eval = 0;
00173     ArrayList<Double> evaluation = new ArrayList<Double>(this.function.size());
00174     if (typeSolutionMethod == null) {
00175         eval= function.get(0).Evaluation(state);
00176         evaluation.add(evaluation.size(), eval);
00177         state.setEvaluation(evaluation);
00178     }
00179     else {
00180         SolutionMethod method = newSolutionMethod(typeSolutionMethod);
00181         method.evaluationState(state);
00182     }
00183 }
```

Hace referencia a [problem.extension.SolutionMethod.evaluationState\(\)](#), [newSolutionMethod\(\)](#), [state](#) y [typeSolutionMethod](#).

Referenciado por [metaheuristics.generators.MultiGenerator.createInstanceGeneratorsBPP\(\)](#), [metaheuristics.generators.HillClimbingRestart.generate\(\)](#), [local\\_search.candidate\\_type.NotDominatedCandidate.stateSearch\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance.updateReference\(\)](#) y [metaheuristics.generators.MultiobjectiveHillClimbingRestart.updateReference\(\)](#).

Gráfico de llamadas de esta función:

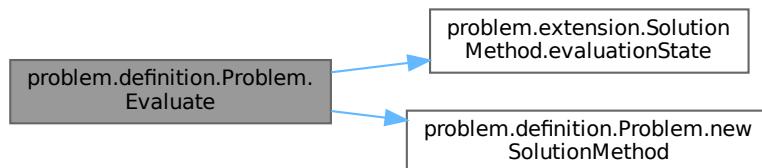
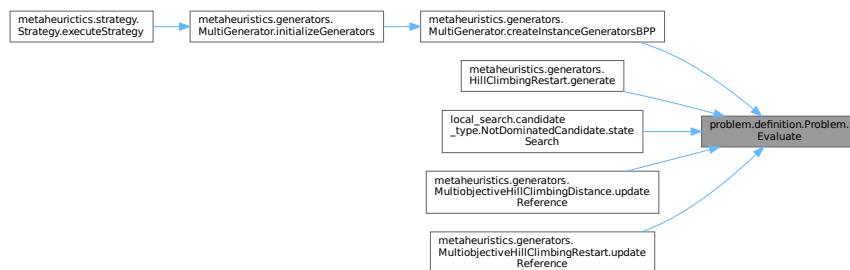


Gráfico de llamadas a esta función:



### 8.78.3.2 getCodification()

`Codification problem.definition.Problem.getCodification () [inline]`

Obtener la codificación usada para representar estados.

Devuelve

la `Codification` configurada

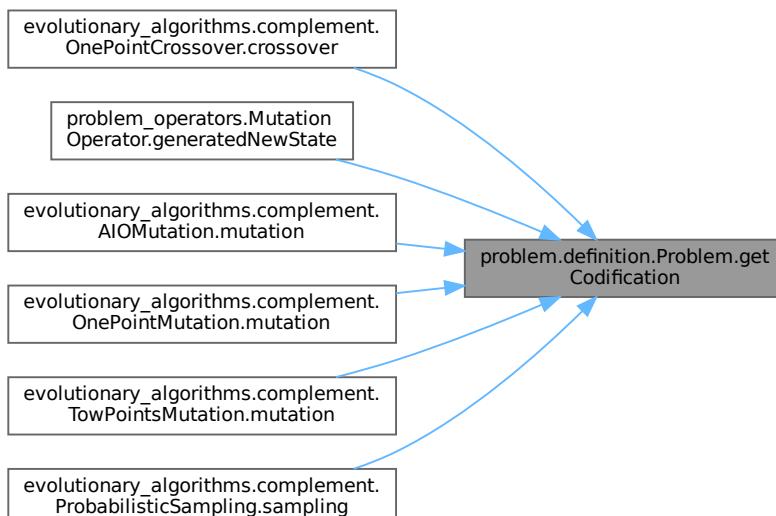
Definición en la línea 114 del archivo `Problem.java`.

```
00114 {  
00115     return codification;  
00116 }
```

Hace referencia a `codification`.

Referenciado por `evolutionary_algorithms.complement.OnePointCrossover.crossover()`, `problem_operators.MutationOperator.generatedNewState`, `evolutionary_algorithms.complement.AIOMutation.mutation()`, `evolutionary_algorithms.complement.OnePointMutation.mutation()`, `evolutionary_algorithms.complement.TowPointsMutation.mutation()` y `evolutionary_algorithms.complement.ProbabilisticSampling.sampling`

Gráfico de llamadas a esta función:



### 8.78.3.3 getFactorySolutionMethod()

`IFFactorySolutionMethod problem.definition.Problem.getFactorySolutionMethod () [inline]`

Obtener la fábrica de métodos de solución.

Devuelve

fábrica configurada (puede ser null)

Definición en la línea 206 del archivo `Problem.java`.

```
00206 {  
00207     return factorySolutionMethod;  
00208 }
```

Hace referencia a `factorySolutionMethod`.

### 8.78.3.4 getFunction()

```
ArrayList< ObjectiveFunction > problem.definition.Problem.getFunction () [inline]
```

Obtener la lista de funciones objetivo del problema.

Devuelve

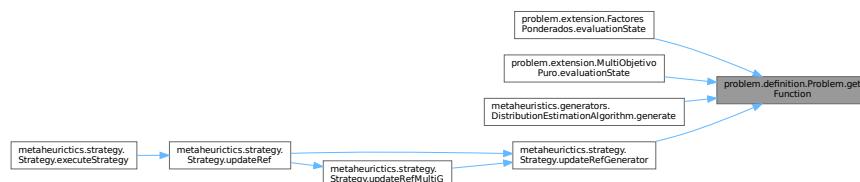
copia defensiva de la lista de `ObjectiveFunction`; nunca devuelve la referencia interna (puede ser lista vacía).

Definición en la línea 57 del archivo `Problem.java`.

```
00057
00058     return (function == null) ? new ArrayList<ObjectiveFunction>() : new
00059         ArrayList<ObjectiveFunction>(function);
00059 }
```

Referenciado por `problem.extension.FactoresPonderados.evaluationState()`, `problem.extension.MultiObjetivoPuro.evaluationState()`, `metaheuristics.generators.DistributionEstimationAlgorithm.generate()` y `metaheuristics.strategy.Strategy.updateRefGenerator()`.

Gráfico de llamadas a esta función:



### 8.78.3.5 getOperator()

```
Operator problem.definition.Problem.getOperator () [inline]
```

Obtener el operador encargado de generar nuevos estados.

Devuelve

operador configurado

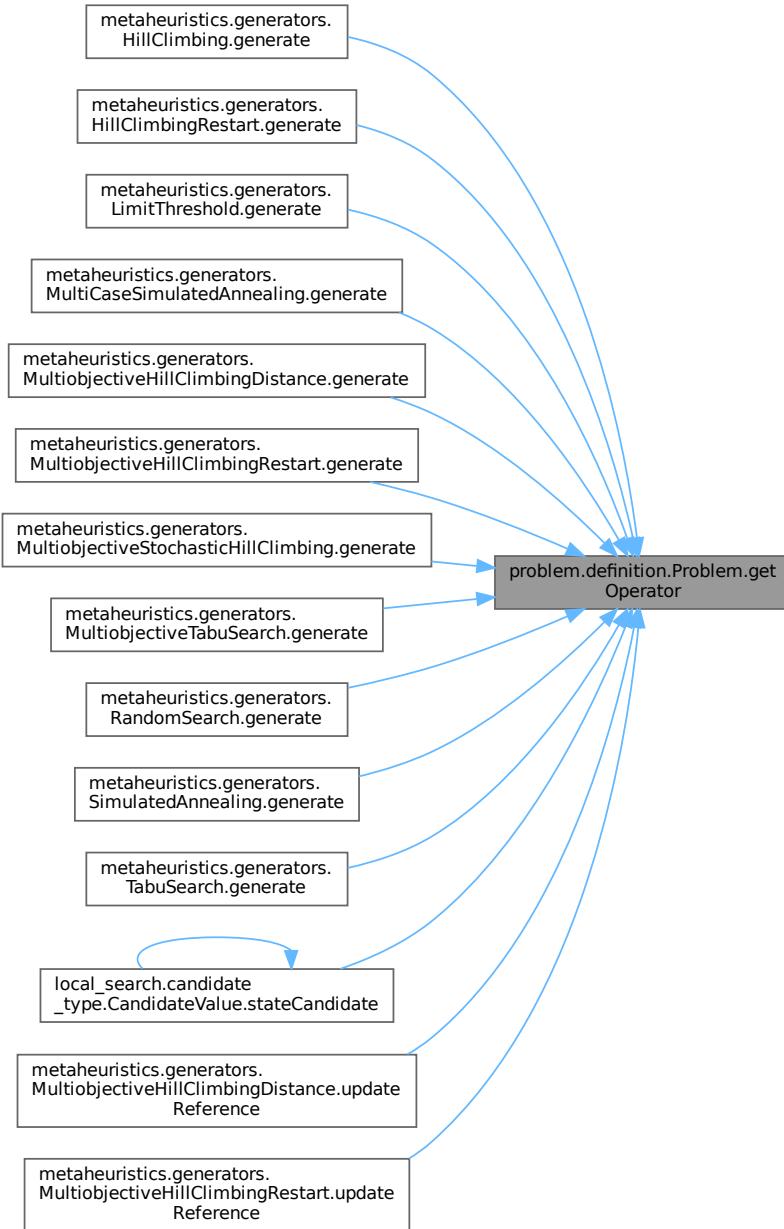
Definición en la línea 131 del archivo `Problem.java`.

```
00131
00132     return operator;
00133 }
```

Hace referencia a `operator`.

Referenciado por `metaheuristics.generators.HillClimbing.generate()`, `metaheuristics.generators.HillClimbingRestart.generate()`, `metaheuristics.generators.LimitThreshold.generate()`, `metaheuristics.generators.MultiCaseSimulatedAnnealing.generate()`, `metaheuristics.generators.MultiobjectiveHillClimbingDistance.generate()`, `metaheuristics.generators.MultiobjectiveHillClimbingRestart()`, `metaheuristics.generators.MultiobjectiveStochasticHillClimbing.generate()`, `metaheuristics.generators.MultiobjectiveTabuSearch.generate()`, `metaheuristics.generators.RandomSearch.generate()`, `metaheuristics.generators.SimulatedAnnealing.generate()`, `metaheuristics.generators.TabuSearch.generate()`, `local_search.candidate_type.CandidateValue.stateCandidate()`, `metaheuristics.generators.MultiobjectiveHillClimbingDistance.updateReference()` y `metaheuristics.generators.MultiobjectiveHillClimbi`

Gráfico de llamadas a esta función:



#### 8.78.3.6 `getPossibleValue()`

```
int problem.definition.Problem.getPossibleValue () [inline]
```

Obtener el número de valores posibles (uso dependiente de la codificación).

Devuelve

número de valores posibles

Definición en la línea 149 del archivo [Problem.java](#).

```
00149         {
00150             return possibleValue;
00151         }
```

Hace referencia a [possibleValue](#).

Referenciado por [evolutionary\\_algorithms.complement.ProbabilisticSampling.sampling\(\)](#).

Gráfico de llamadas a esta función:



### 8.78.3.7 getState()

```
State problem.definition.Problem.getState () [inline]
```

Obtener una copia del estado actual del problema.

Devuelve una copia para evitar aliasing con la representación interna.

Devuelve

copia de [State](#) o null si no hay estado inicial.

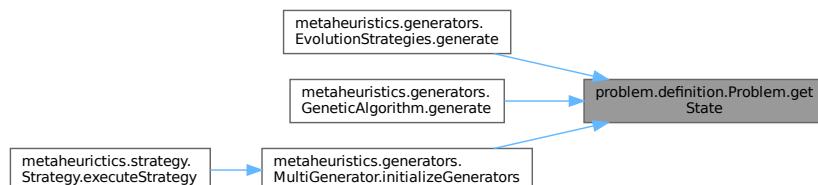
Definición en la línea 79 del archivo [Problem.java](#).

```
00079         {
00080             return (state == null) ? null : new State(state);
00081         }
```

Hace referencia a [state](#).

Referenciado por [metaheuristics.generators.EvolutionStrategies.generate\(\)](#), [metaheuristics.generators.GeneticAlgorithm.generate\(\)](#) y [metaheuristics.generators.MultiGenerator.initializeGenerators\(\)](#).

Gráfico de llamadas a esta función:



### 8.78.3.8 getTypeProblem()

```
ProblemType problem.definition.Problem.getTypeProblem () [inline]
```

Obtener el tipo de problema (Maximizar / Minimizar).

Devuelve

el [ProblemType](#) configurado para este problema

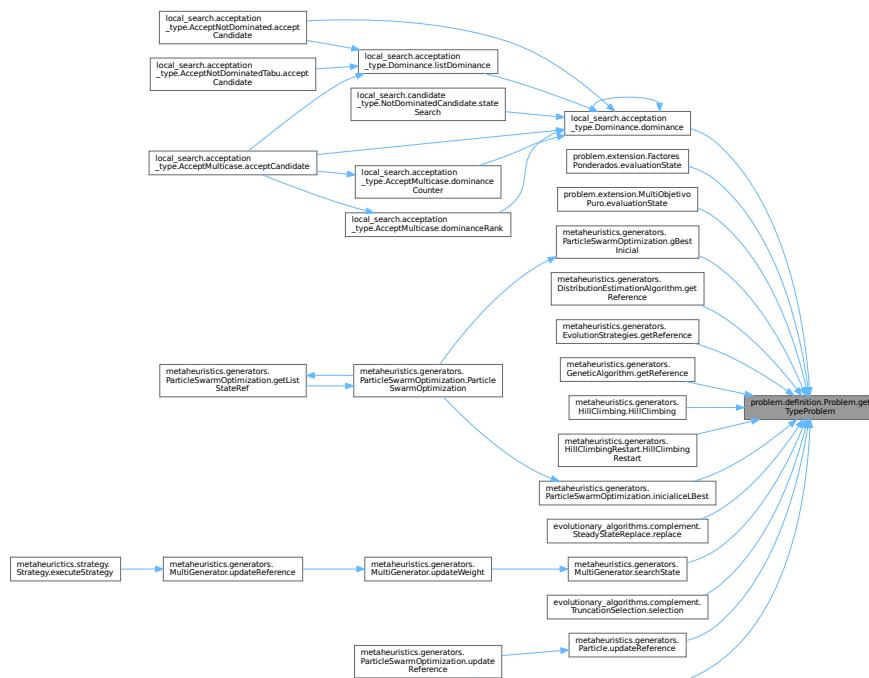
Definición en la línea 97 del archivo [Problem.java](#).

```
00097
00098     return typeProblem;
00099 }
```

Hace referencia a [typeProblem](#).

Referenciado por [local\\_search.acceptation\\_type.Dominance.dominance\(\)](#), [problem.extension.FactoresPonderados.evaluationState\(\)](#), [problem.extension.MultiObjetivoPuro.evaluationState\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.gBestIncial\(\)](#), [metaheuristics.generators.DistributionEstimationAlgorithm.getReference\(\)](#), [metaheuristics.generators.EvolutionStrategies.getReference\(\)](#), [metaheuristics.generators.GeneticAlgorithm.getReference\(\)](#), [metaheuristics.generators.HillClimbing.HillClimbing\(\)](#), [metaheuristics.generators.HillClimbingRestart.HillClimbingRestart\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.inicialiceL\(\)](#), [evolutionary\\_algorithms.complement.SteadyStateReplace.replace\(\)](#), [metaheuristics.generators.MultiGenerator.searchState\(\)](#), [evolutionary\\_algorithms.complement.TruncationSelection.selection\(\)](#), [metaheuristics.generators.Particle.updateReference\(\)](#) y [metaheuristics.generators.ParticleSwarmOptimization.updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.78.3.9 getTypeSolutionMethod()

```
TypeSolutionMethod problem.definition.Problem.getTypeSolutionMethod () [inline]
```

Obtener el tipo de método de solución configurado.

#### Devuelve

tipo de método de solución o null

Definición en la línea 190 del archivo [Problem.java](#).

```
00190
00191     return typeSolutionMethod;
00192 }
```

Hace referencia a [typeSolutionMethod](#).

### 8.78.3.10 newSolutionMethod()

```
SolutionMethod problem.definition.Problem.newSolutionMethod (
    TypeSolutionMethod typeSolutionMethod) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException [inline]
```

Crear una instancia de [SolutionMethod](#) para el tipo solicitado.

#### Parámetros

<code>typeSolutionMethod</code>	tipo de método de solución a crear
---------------------------------	------------------------------------

#### Devuelve

instancia de [SolutionMethod](#)

#### Excepciones

<code>ReflectiveOperationException</code>	si la creación vía fábrica falla
---	----------------------------------

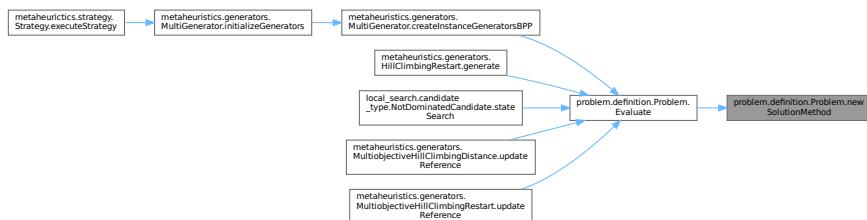
Definición en la línea 221 del archivo [Problem.java](#).

```
00221
00222 {
00223     factorySolutionMethod = new FactorySolutionMethod();
00224     SolutionMethod solutionMethod =
00225         factorySolutionMethod.createdSolutionMethod(typeSolutionMethod);
00226     return solutionMethod;
00227 }
```

Hace referencia a [factorySolutionMethod](#) y [typeSolutionMethod](#).

Referenciado por [Evaluate\(\)](#).

Gráfico de llamadas a esta función:



### 8.78.3.11 setCodification()

```
void problem.definition.Problem.setCodification (
    Codification codification) [inline]
```

Establecer la codificación de variables para el problema.

#### Parámetros

<i>codification</i>	implementación de <a href="#">Codification</a>
---------------------	--

Definición en la línea 122 del archivo [Problem.java](#).

```
00122
00123     this.codification = codification;
00124 }
```

Hace referencia a [codification](#).

### 8.78.3.12 setFactorySolutionMethod()

```
void problem.definition.Problem.setFactorySolutionMethod (
    IFFactorySolutionMethod factorySolutionMethod) [inline]
```

Definición en la línea 209 del archivo [Problem.java](#).

```
00210
00211     this.factorySolutionMethod = factorySolutionMethod;
00212 }
```

Hace referencia a [factorySolutionMethod](#).

### 8.78.3.13 setFunction()

```
void problem.definition.Problem.setFunction (
    ArrayList< ObjectiveFunction > function) [inline]
```

Establecer las funciones objetivo del problema.

Se realiza una copia defensiva de la lista proporcionada.

**Parámetros**

<i>function</i>	lista de funciones objetivo
-----------------	-----------------------------

Definición en la línea 68 del archivo [Problem.java](#).

```
00068          {
00069      this.function = (function == null) ? new ArrayList<ObjectiveFunction>() : new
00070      ArrayList<ObjectiveFunction>(function);
}
```

**8.78.3.14 setOperator()**

```
void problem.definition.Problem.setOperator (
    Operator operator) [inline]
```

Establecer el operador de generación de estados.

**Parámetros**

<i>operator</i>	operador que implementa la generación de estados
-----------------	--

Definición en la línea 140 del archivo [Problem.java](#).

```
00140          {
00141      this.operator = operator;
00142 }
```

Hace referencia a [operator](#).

**8.78.3.15 setPossibleValue()**

```
void problem.definition.Problem.setPossibleValue (
    int possibleValue) [inline]
```

Establecer el número de valores posibles.

**Parámetros**

<i>possibleValue</i>	número de valores
----------------------	-------------------

Definición en la línea 158 del archivo [Problem.java](#).

```
00158          {
00159      this.possibleValue = possibleValue;
00160 }
```

Hace referencia a [possibleValue](#).

**8.78.3.16 setState()**

```
void problem.definition.Problem.setState (
    State state) [inline]
```

## Parámetros

<code>state</code>	estado a asignar; si es null se resetea el estado interno.
--------------------	--

Definición en la línea 88 del archivo [Problem.java](#).

```
00088      {
00089          this.state = (state == null) ? null : new State(state);
00090      }
```

Hace referencia a [state](#).

Referenciado por [metaheuristics.strategy.Strategy.executeStrategy\(\)](#).

Gráfico de llamadas a esta función:



### 8.78.3.17 setTypeProblem()

```
void problem.definition.Problem.setTypeProblem (
    ProblemType typeProblem) [inline]
```

Establecer el tipo de problema.

## Parámetros

<code>typeProblem</code>	tipo de problema (Maximizar o Minimizar)
--------------------------	--

Definición en la línea 105 del archivo [Problem.java](#).

```
00105      {
00106          this.typeProblem = typeProblem;
00107      }
```

Hace referencia a [typeProblem](#).

### 8.78.3.18 setTypeSolutionMethod()

```
void problem.definition.Problem.setTypeSolutionMethod (
    TypeSolutionMethod typeSolutionMethod) [inline]
```

Establecer el tipo de método de solución.

## Parámetros

<code>typeSolutionMethod</code>	tipo a usar para evaluar estados
---------------------------------	----------------------------------

Definición en la línea 198 del archivo [Problem.java](#).

```
00198
00199     this.typeSolutionMethod = typeSolutionMethod;
00200 }
```

Hace referencia a [typeSolutionMethod](#).

### 8.78.4 Documentación de datos miembro

#### 8.78.4.1 codification

`Codification` problem.definition.Problem.codification [private]

Definición en la línea 34 del archivo [Problem.java](#).

Referenciado por [getCodification\(\)](#) y [setCodification\(\)](#).

#### 8.78.4.2 factorySolutionMethod

`IFFactorySolutionMethod` problem.definition.Problem.factorySolutionMethod [private]

Definición en la línea 38 del archivo [Problem.java](#).

Referenciado por [getFactorySolutionMethod\(\)](#), [newSolutionMethod\(\)](#) y [setFactorySolutionMethod\(\)](#).

#### 8.78.4.3 function

`ArrayList<ObjectiveFunction>` problem.definition.Problem.function [private]

Definición en la línea 31 del archivo [Problem.java](#).

#### 8.78.4.4 operator

`Operator` problem.definition.Problem.operator [private]

Definición en la línea 35 del archivo [Problem.java](#).

Referenciado por [getOperator\(\)](#) y [setOperator\(\)](#).

#### 8.78.4.5 possibleValue

`int` problem.definition.Problem.possibleValue [private]

Definición en la línea 36 del archivo [Problem.java](#).

Referenciado por [getPossibleValue\(\)](#) y [setPossibleValue\(\)](#).

#### 8.78.4.6 state

```
State problem.definition.Problem.state [private]
```

Definición en la línea 32 del archivo [Problem.java](#).

Referenciado por [Evaluate\(\)](#), [getState\(\)](#) y [setState\(\)](#).

#### 8.78.4.7 typeProblem

```
ProblemType problem.definition.Problem.typeProblem [private]
```

Definición en la línea 33 del archivo [Problem.java](#).

Referenciado por [getTypeProblem\(\)](#) y [setTypeProblem\(\)](#).

#### 8.78.4.8 typeSolutionMethod

```
TypeSolutionMethod problem.definition.Problem.typeSolutionMethod [private]
```

Definición en la línea 37 del archivo [Problem.java](#).

Referenciado por [Evaluate\(\)](#), [getTypeSolutionMethod\(\)](#), [newSolutionMethod\(\)](#) y [setTypeSolutionMethod\(\)](#).

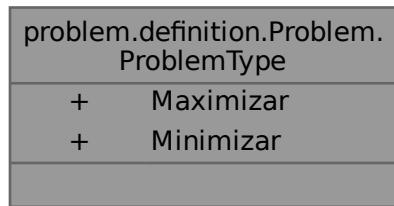
La documentación de esta clase está generada del siguiente archivo:

- [Problem.java](#)

### 8.79 Referencia de la enumeración **problem.definition.Problem.ProblemType**

[ProblemType](#) - descripción (añade detalles).

Diagrama de colaboración de `problem.definition.Problem.ProblemType`:



## Atributos públicos

- Maximizar
- Minimizar

### 8.79.1 Descripción detallada

ProblemType - descripción (añade detalles).

Definición en la línea 29 del archivo [Problem.java](#).

### 8.79.2 Documentación de datos miembro

#### 8.79.2.1 Maximizar

`problem.definition.Problem.ProblemType.Maximizar`

Definición en la línea 29 del archivo [Problem.java](#).

Referenciado por [local\\_search.acceptation\\_type.AcceptBest.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotBad.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotBadT.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotBadU.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.Dominance.dominance\(\)](#), [problem.extension.FactoresPonderados.evaluationState\(\)](#), [problem.extension.MultiObjetivoPuro.evaluationState\(\)](#), [metaheuristics.strategy.Strategy.executeStrategy\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.gBestInicial\(\)](#), [metaheuristics.generators.DistributionEstimationAlgorithm.getReference\(\)](#), [metaheuristics.generators.EvolutionStrategies.getReference\(\)](#), [metaheuristics.generators.GeneticAlgorithm.getReference\(\)](#), [metaheuristics.generators.HillClimbing.HillClimbing\(\)](#), [metaheuristics.generators.HillClimbingRestart.HillClimbingRestart\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.inicialiceLBest\(\)](#), [metaheuristics.generators.LimitThreshold.LimitThreshold\(\)](#), [evolutionary\\_algorithms.complement.SteadyStateReplace.replace\(\)](#), [metaheuristics.generators.MultiGenerator.searchState\(\)](#), [evolutionary\\_algorithms.complement.TruncationSelection.selection\(\)](#), [metaheuristics.generators.TabuSearch.TabuSearch\(\)](#), [metaheuristics.generators.Particle.updateReference\(\)](#) y [metaheuristics.generators.ParticleSwarmOptimization.updateReference\(\)](#).

#### 8.79.2.2 Minimizar

`problem.definition.Problem.ProblemType.Minimizar`

Definición en la línea 29 del archivo [Problem.java](#).

Referenciado por [metaheuristics.strategy.Strategy.executeStrategy\(\)](#), [evolutionary\\_algorithms.complement.SteadyStateReplace.replace\(\)](#) y [evolutionary\\_algorithms.complement.TruncationSelection.selection\(\)](#).

La documentación de esta enumeración está generada del siguiente archivo:

- [Problem.java](#)

## 8.80 Referencia de la clase

### **local\_search.candidate\_type.RandomCandidate**

[RandomCandidate](#) - select a neighbor at random.

Diagrama de herencia de local\_search.candidate\_type.RandomCandidate

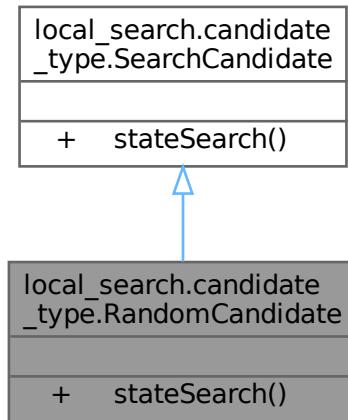
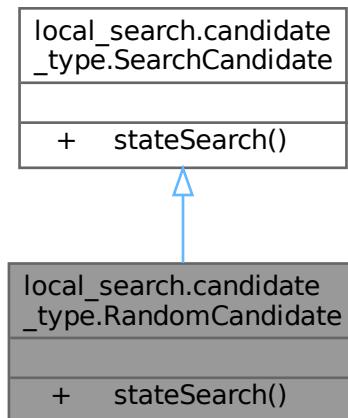


Diagrama de colaboración de local\_search.candidate\_type.RandomCandidate:



#### Métodos públicos

- [State stateSearch \(List< State > listNeighborhood\)](#)

*Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.*

### 8.80.1 Descripción detallada

[RandomCandidate](#) - select a neighbor at random.

Returns a uniformly random neighbor from the provided list. Uses ThreadLocalRandom for thread-safe, efficient random number generation.

Definición en la línea 18 del archivo [RandomCandidate.java](#).

### 8.80.2 Documentación de funciones miembro

#### 8.80.2.1 stateSearch()

```
State local_search.candidate_type.RandomCandidate.stateSearch (
    List< State > listNeighborhood) [inline]
```

Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.

This RNG is used to decide whether a mutation occurs in the evolutionary algorithm and is not used for security-sensitive purposes. Suppress Sonar security hotspot S2245 for this usage. Choose a random neighbor from the list.

#### Parámetros

<i>listNeighborhood</i>	list of neighbor states
-------------------------	-------------------------

#### Devuelve

a randomly selected [State](#) from the list

Reimplementado de [local\\_search.candidate\\_type.SearchCandidate](#).

Definición en la línea 33 del archivo [RandomCandidate.java](#).

```
00033                               {
00034         int size = listNeighborhood.size();
00035         int pos = (size > 0) ? ThreadLocalRandom.current().nextInt(size) : 0;
00036         State stateAleatory = listNeighborhood.get(pos);
00037         return stateAleatory;
00038     }
```

La documentación de esta clase está generada del siguiente archivo:

- [RandomCandidate.java](#)

## 8.81 Referencia de la clase metaheuristics.generators.RandomSearch

[RandomSearch](#) - class that implements the Random Search metaheuristic.

Diagrama de herencia de `metaheuristics.generators.RandomSearch`

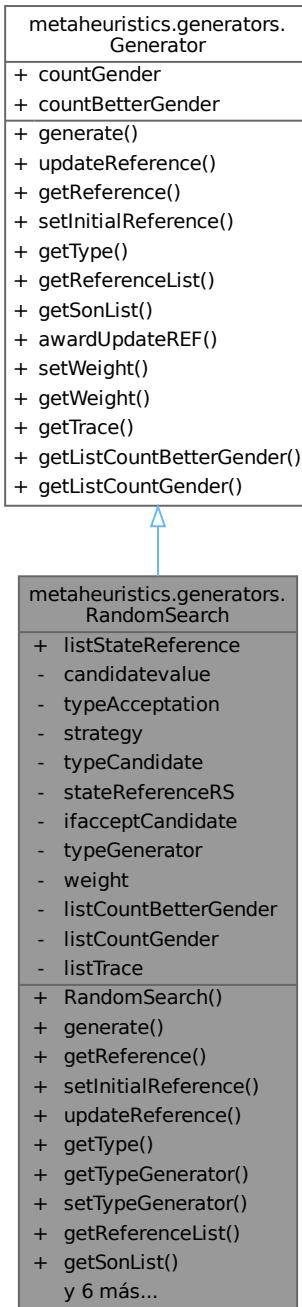
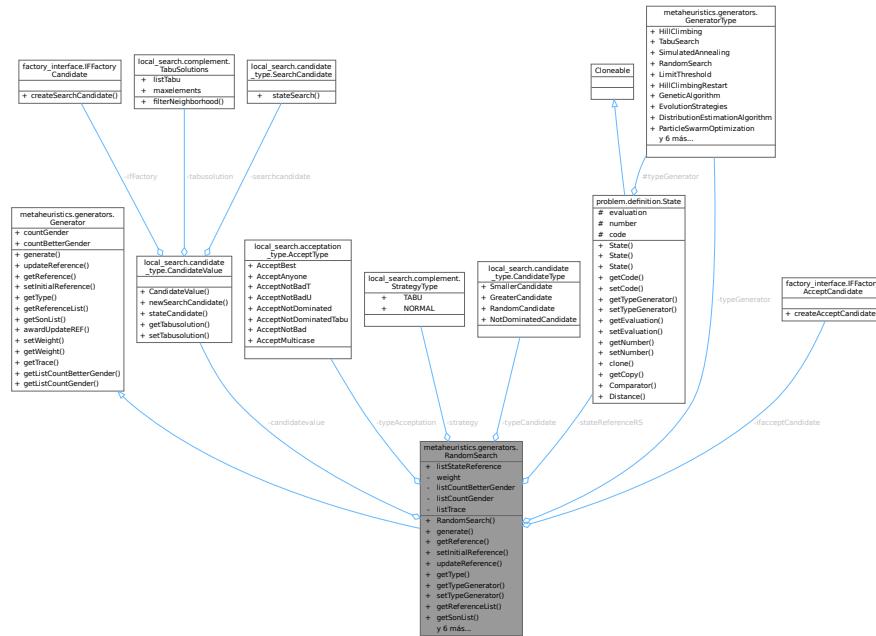


Diagrama de colaboración de metaheuristics.generators.RandomSearch:



## Métodos públicos

- **RandomSearch ()**

*RandomSearch - class that implements the Random Search metaheuristic.*

- **State generate (Integer operatornumber)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*generate - generate a new state for the random search.*

- **State getReference ()**

*getReference - get the reference state for the random search.*

- **void setInitialReference (State statelInitialRef)**

*setInitialReference - set the initial reference state for the random search.*

- **void updateReference (State stateCandidate, Integer countIterationsCurrent)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*updateReference - update the reference state and the personal best state of the particle.*

- **GeneratorType getType ()**

*getType - get the type of the generator.*

- **GeneratorType getTypeGenerator ()**

*getTypeGenerator - get the type of the generator.*

- **void setTypeGenerator (GeneratorType typeGenerator)**

*setTypeGenerator - set the type of the generator.*

- **List< State > getReferenceList ()**

*getReferenceList - get the list of reference states for the random search.*

- **List< State > getSonList ()**

*getSonList - get the son list.*

- **boolean awardUpdateREF (State stateCandidate)**

*awardUpdateREF - award the update of the reference state.*

- float [getWeight \(\)](#)  
*getWeight - get the weight of the generator.*
- void [setWeight \(float weight\)](#)  
*setWeight - set the weight of the generator.*
- int[] [getListCountBetterGender \(\)](#)  
*getListCountBetterGender - get the list of count better gender.*
- int[] [getListCountGender \(\)](#)  
*getListCountGender - get the list of count gender.*
- float[] [getTrace \(\)](#)  
*getTrace - get the trace of the generator.*

### Atributos públicos estáticos

- static final List< [State](#) > [listStateReference](#) = Collections.synchronizedList(new ArrayList< [State](#) >())

### Atributos privados

- [CandidateValue candidatevalue](#)
- [AcceptType typeAcceptation](#)
- [StrategyType strategy](#)
- [CandidateType typeCandidate](#)
- [State stateReferenceRS](#)
- [IFFactoryAcceptCandidate ifacceptCandidate](#)
- [GeneratorType typeGenerator](#)
- float [weight](#)
- int[] [listCountBetterGender](#) = new int[10]
- int[] [listCountGender](#) = new int[10]
- float[] [listTrace](#) = new float[1200000]

### Otros miembros heredados

#### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- int [countGender](#)
- int [countBetterGender](#)

### 8.81.1 Descripción detallada

[RandomSearch](#) - class that implements the Random Search metaheuristic.

Definición en la línea 24 del archivo [RandomSearch.java](#).

## 8.81.2 Documentación de constructores y destructores

### 8.81.2.1 RandomSearch()

```
metaheuristics.generators.RandomSearch RandomSearch () [inline]
```

[RandomSearch](#) - class that implements the Random Search metaheuristic.

Definición en la línea 46 del archivo [RandomSearch.java](#).

```
00046          {
00047      super();
00048      this.typeAcceptation = AcceptType.AcceptBest;
00049      this.strategy = StrategyType.NORMAL;
00050      this.typeCandidate = CandidateType.RandomCandidate;
00051      this.candidatevalue = new CandidateValue();
00052      this.typeGenerator = GeneratorType.RandomSearch;
00053      this.weight = 50;
00054      listTrace[0] = this.weight;
00055      listCountBetterGender[0] = 0;
00056      listCountGender[0] = 0;
00057      listStateReference.clear();
00058  }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptType.AcceptBest](#), [listCountBetterGender](#), [listCountGender](#), [listStateReference](#), [listTrace](#), [local\\_search.complement.StrategyType.NORMAL](#), [local\\_search.candidate\\_type.CandidateType.Random](#) y [metaheuristics.generators.GeneratorType.RandomSearch](#).

## 8.81.3 Documentación de funciones miembro

### 8.81.3.1 awardUpdateREF()

```
boolean metaheuristics.generators.RandomSearch.awardUpdateREF (
    State stateCandidate) [inline]
```

[awardUpdateREF](#) - award the update of the reference state.

#### Parámetros

<i>stateCandidate</i>	
-----------------------	--

#### Devuelve

return true if the update is awarded, false otherwise

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 158 del archivo [RandomSearch.java](#).

```
00158          // TODO Auto-generated method stub
00159      return false;
00160  }
```

### 8.81.3.2 generate()

```
State metaheuristics.generators.RandomSearch.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException [inline]
```

[generate](#) - generate a new state for the random search

## Parámetros

*operatornumber*

Devuelve

the generated state

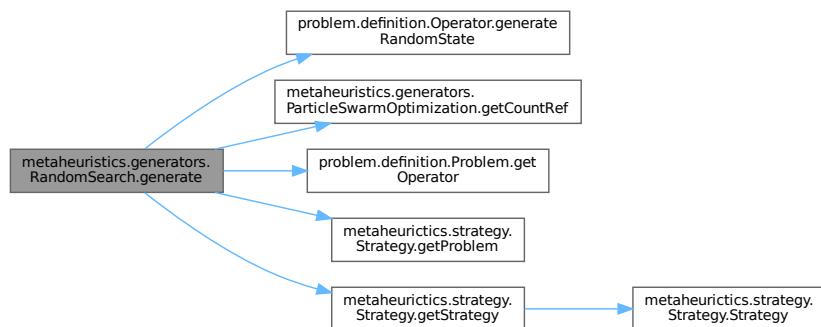
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 66 del archivo [RandomSearch.java](#).

```
00066
00067     {
00068         List<State> neighborhood =
00069             Strategy.getStrategy().getProblem().getOperator().generateRandomState(operatornumber);
00070         State statecandidate = candidatevalue.stateCandidate(stateReferenceRS, typeCandidate,
00071             strategy, operatornumber, neighborhood);
00072         if(GeneticAlgorithm.countRef != 0 || EvolutionStrategies.countRef != 0 ||
00073             DistributionEstimationAlgorithm.countRef != 0 || ParticleSwarmOptimization.getCountRef() != 0)
00074             listStateReference.add(statecandidate);
00075         return statecandidate;
00076     }
```

Hace referencia a [candidatevalue](#), [metaheuristics.generators.DistributionEstimationAlgorithm.countRef](#), [metaheuristics.generators.EvolutionStrategies.countRef](#), [metaheuristics.generators.GeneticAlgorithm.countRef](#), [problem.definition.Operator.generateRandomState\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.getCountRef\(\)](#), [problem.definition.Problem.getOperator\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [listStateReference](#), [stateReferenceRS](#), [strategy](#) y [typeCandidate](#).

Gráfico de llamadas de esta función:



### 8.81.3.3 getListCountBetterGender()

```
int[] metaheuristics.generators.RandomSearch.getListCountBetterGender () [inline]
getListCountBetterGender - get the list of count better gender.
```

Devuelve

the list of count better gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 189 del archivo [RandomSearch.java](#).

```
00189
00190     // TODO Auto-generated method stub
00191     return (this.listCountBetterGender == null) ? new int[0] :
00192         Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
```

#### 8.81.3.4 getListCountGender()

```
int[] metaheuristics.generators.RandomSearch.getListCountGender () [inline]
```

getListCountGender - get the list of count gender.

Devuelve

the list of count gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 199 del archivo [RandomSearch.java](#).

```
00199         {
00200             // TODO Auto-generated method stub
00201             return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00202                 this.listCountGender.length);
00203         }
```

#### 8.81.3.5 getReference()

```
State metaheuristics.generators.RandomSearch.getReference () [inline]
```

getReference - get the reference state for the random search.

Devuelve

the reference state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 79 del archivo [RandomSearch.java](#).

```
00079         {
00080             return stateReferenceRS;
00081         }
```

Hace referencia a [stateReferenceRS](#).

#### 8.81.3.6 getReferenceList()

```
List< State > metaheuristics.generators.RandomSearch.getReferenceList () [inline]
```

getReferenceList - get the list of reference states for the random search.

Devuelve

the list of reference states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 137 del archivo [RandomSearch.java](#).

```
00137         {
00138             listStateReference.add(stateReferenceRS);
00139             return listStateReference;
00140         }
```

Hace referencia a [listStateReference](#) y [stateReferenceRS](#).

### 8.81.3.7 getSonList()

```
List< State > metaheuristics.generators.RandomSearch.getSonList () [inline]
```

getSonList - get the son list.

Devuelve

the son list

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 147 del archivo [RandomSearch.java](#).

```
00147                               {  
00148         // TODO Auto-generated method stub  
00149         return null;  
00150     }
```

### 8.81.3.8 getTrace()

```
float[ ] metaheuristics.generators.RandomSearch.getTrace () [inline]
```

getTrace - get the trace of the generator.

Devuelve

the trace of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 209 del archivo [RandomSearch.java](#).

```
00209                               {  
00210         // TODO Auto-generated method stub  
00211         return \(this.listTrace == null\) ? new float\[0\] : Arrays.copyOf\(this.listTrace,  
00212             this.listTrace.length\);  
00212     }
```

### 8.81.3.9 getType()

```
GeneratorType metaheuristics.generators.RandomSearch.getType () [inline]
```

getType - get the type of the generator.

Devuelve

the type of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 112 del archivo [RandomSearch.java](#).

```
00112                               {  
00113         return this.typeGenerator;  
00114     }
```

### 8.81.3.10 getTypeGenerator()

```
GeneratorType metaheuristics.generators.RandomSearch.getTypeGenerator () [inline]
```

getTypeGenerator - get the type of the generator.

Devuelve

the type of the generator

Definición en la línea 120 del archivo [RandomSearch.java](#).

```
00120 {  
00121     return typeGenerator;  
00122 }
```

Hace referencia a [typeGenerator](#).

### 8.81.3.11 getWeight()

```
float metaheuristics.generators.RandomSearch.getWeight () [inline]
```

getWeight - get the weight of the generator.

Devuelve

the weight of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 169 del archivo [RandomSearch.java](#).

```
00169 {  
00170     // TODO Auto-generated method stub  
00171     return this.weight;  
00172 }
```

### 8.81.3.12 setInitialReference()

```
void metaheuristics.generators.RandomSearch.setInitialReference (  
    State stateInitialRef) [inline]
```

setInitialReference - set the initial reference state for the random search.

#### Parámetros

stateInitialRef	<input type="text"/>
-----------------	----------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 88 del archivo [RandomSearch.java](#).

```
00088 {  
00089     this.stateReferenceRS = stateInitialRef;  
00090 }
```

### 8.81.3.13 setTypeGenerator()

```
void metaheuristics.generators.RandomSearch.setTypeGenerator (
```

**Parámetros**

<i>typeGenerator</i>	
----------------------	--

Definición en la línea 128 del archivo [RandomSearch.java](#).

```
00128
00129     this.typeGenerator = typeGenerator;
00130 }
```

Hace referencia a [typeGenerator](#).

**8.81.3.14 setWeight()**

```
void metaheuristics.generators.RandomSearch.setWeight (
    float weight) [inline]
```

`setWeight` - set the weight of the generator.

**Parámetros**

<i>weight</i>	
---------------	--

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 179 del archivo [RandomSearch.java](#).

```
00179
00180     // TODO Auto-generated method stub
00181     this.weight = weight;
00182 }
```

Hace referencia a [weight](#).

**8.81.3.15 updateReference()**

```
void metaheuristics.generators.RandomSearch.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

`updateReference` - update the reference state and the personal best state of the particle.

**Parámetros**

<i>stateCandidate</i>	
-----------------------	--

<i>countIterationsCurrent</i>	
-------------------------------	--

Reimplementado de [metaheuristics.generators.Generator](#).

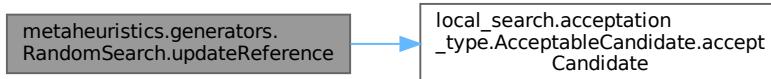
Definición en la línea 98 del archivo [RandomSearch.java](#).

```
00098
{
```

```
00099     ifacceptCandidate = new FactoryAcceptCandidate();
00100     AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00101     Boolean accept = candidate.acceptCandidate(stateReferenceRS, stateCandidate);
00102     if(accept.equals(true))
00103         stateReferenceRS = stateCandidate;
00104 }
```

Hace referencia a `local_search.acceptation_type.AcceptableCandidate.acceptCandidate()`, `ifacceptCandidate`, `stateReferenceRS` y `typeAcceptation`.

Gráfico de llamadas de esta función:



## 8.81.4 Documentación de datos miembro

### 8.81.4.1 candidatevalue

`CandidateValue` `metaheuristics.generators.RandomSearch.candidatevalue` [private]

Definición en la línea 26 del archivo `RandomSearch.java`.

Referenciado por [generate\(\)](#).

### 8.81.4.2 ifacceptCandidate

`IFFactoryAcceptCandidate` `metaheuristics.generators.RandomSearch.ifacceptCandidate` [private]

Definición en la línea 31 del archivo `RandomSearch.java`.

Referenciado por [updateReference\(\)](#).

### 8.81.4.3 listCountBetterGender

`int []` `metaheuristics.generators.RandomSearch.listCountBetterGender` = new int[10] [private]

Definición en la línea 39 del archivo `RandomSearch.java`.

Referenciado por [RandomSearch\(\)](#).

### 8.81.4.4 listCountGender

`int []` `metaheuristics.generators.RandomSearch.listCountGender` = new int[10] [private]

Definición en la línea 40 del archivo `RandomSearch.java`.

Referenciado por [RandomSearch\(\)](#).

#### 8.81.4.5 listStateReference

```
final List<State> metaheuristics.generators.RandomSearch.listStateReference = Collections.←  
synchronizedList(new ArrayList<State>()) [static]
```

Definición en la línea 36 del archivo [RandomSearch.java](#).

Referenciado por [metaheuristics.strategy.Strategy.destroyExecute\(\)](#), [generate\(\)](#), [metaheuristics.generators.DistributionEstimationAlg](#)  
[metaheuristics.generators.EvolutionStrategies.getListStateRef\(\)](#), [metaheuristics.generators.GeneticAlgorithm.getListStateRef\(\)](#),  
[metaheuristics.generators.ParticleSwarmOptimization.getListStateRef\(\)](#), [getReferenceList\(\)](#) y [RandomSearch\(\)](#).

#### 8.81.4.6 listTrace

```
float [] metaheuristics.generators.RandomSearch.listTrace = new float[1200000] [private]
```

Definición en la línea 41 del archivo [RandomSearch.java](#).

Referenciado por [RandomSearch\(\)](#).

#### 8.81.4.7 stateReferenceRS

```
State metaheuristics.generators.RandomSearch.stateReferenceRS [private]
```

Definición en la línea 30 del archivo [RandomSearch.java](#).

Referenciado por [generate\(\)](#), [getReference\(\)](#), [getReferenceList\(\)](#) y [updateReference\(\)](#).

#### 8.81.4.8 strategy

```
StrategyType metaheuristics.generators.RandomSearch.strategy [private]
```

Definición en la línea 28 del archivo [RandomSearch.java](#).

Referenciado por [generate\(\)](#).

#### 8.81.4.9 typeAcceptation

```
AcceptType metaheuristics.generators.RandomSearch.typeAcceptation [private]
```

Definición en la línea 27 del archivo [RandomSearch.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.81.4.10 typeCandidate

```
CandidateType metaheuristics.generators.RandomSearch.typeCandidate [private]
```

Definición en la línea 29 del archivo [RandomSearch.java](#).

Referenciado por [generate\(\)](#).

#### 8.81.4.11 typeGenerator

```
GeneratorType metaheuristics.generators.RandomSearch.typeGenerator [private]
```

Definición en la línea 32 del archivo [RandomSearch.java](#).

Referenciado por [getTypeGenerator\(\)](#) y [setTypeGenerator\(\)](#).

#### 8.81.4.12 weight

```
float metaheuristics.generators.RandomSearch.weight [private]
```

Definición en la línea 34 del archivo [RandomSearch.java](#).

Referenciado por [setWeight\(\)](#).

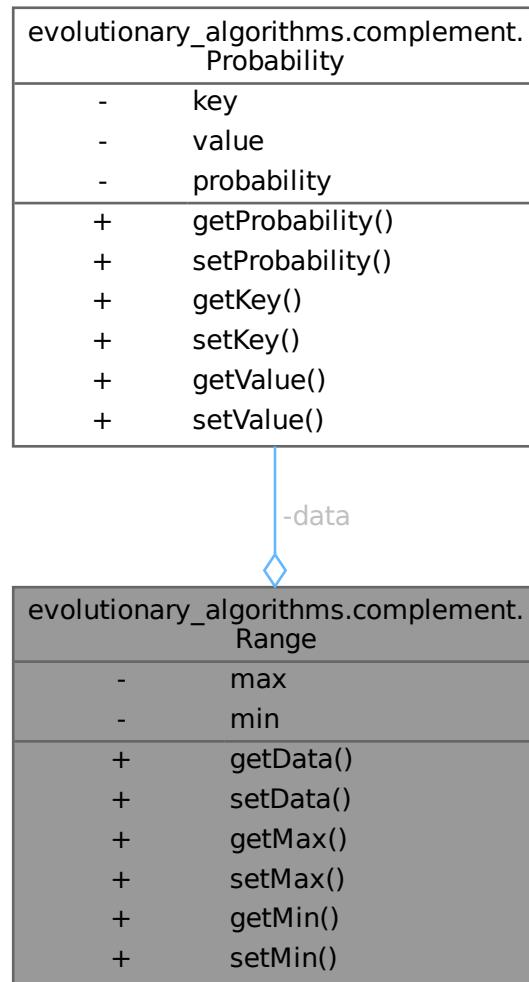
La documentación de esta clase está generada del siguiente archivo:

- [RandomSearch.java](#)

## 8.82 Referencia de la clase evolutionary\_algorithms.complement.Range

[Range](#) - represents a range of values with associated probabilities.

Diagrama de colaboración de `evolutionary_algorithms.complement.Range`:



## Métodos públicos

- `Probability getData ()`  
`getData` - getter for the data.
- `void setData (Probability data)`  
`setData` - setter for the data.
- `float getMax ()`  
`getMax` - getter for the max.
- `void setMax (float max)`  
`setMax` - setter for the max.
- `float getMin ()`  
`getMin` - getter for the min.
- `void setMin (float min)`  
`setMin` - setter for the min.

### Atributos privados

- [Probability data](#)
- float [max](#)
- float [min](#)

## 8.82.1 Descripción detallada

[Range](#) - represents a range of values with associated probabilities.

Definición en la línea 6 del archivo [Range.java](#).

## 8.82.2 Documentación de funciones miembro

### 8.82.2.1 getData()

[Probability](#) `evolutionary_algorithms.complement.Range.getData () [inline]`

getData - getter for the data.

Devuelve

returns the data

Definición en la línea 15 del archivo [Range.java](#).

```
00015          {  
00016      return data;  
00017  }
```

Hace referencia a [data](#).

### 8.82.2.2 getMax()

`float evolutionary_algorithms.complement.Range.getMax () [inline]`

getMax - getter for the max.

Devuelve

returns the max

Definición en la línea 29 del archivo [Range.java](#).

```
00029          {  
00030      return max;  
00031  }
```

Hace referencia a [max](#).

### 8.82.2.3 getMin()

```
float evolutionary_algorithms.complement.Range.getMin () [inline]
```

getMin - getter for the min.

#### Devuelve

returns the min

Definición en la línea 43 del archivo [Range.java](#).

```
00043             {
00044     return min;
00045 }
```

Hace referencia a [min](#).

### 8.82.2.4 setData()

```
void evolutionary_algorithms.complement.Range.setData (
    Probability data) [inline]
```

setData - setter for the data.

#### Parámetros

<i>data</i>	<input type="text"/>
-------------	----------------------

Definición en la línea 22 del archivo [Range.java](#).

```
00022             {
00023     this.data = data;
00024 }
```

Hace referencia a [data](#).

### 8.82.2.5 setMax()

```
void evolutionary_algorithms.complement.Range.setMax (
    float max) [inline]
```

setMax - setter for the max.

#### Parámetros

<i>max</i>	<input type="text"/>
------------	----------------------

Definición en la línea 36 del archivo [Range.java](#).

```
00036             {
00037     this.max = max;
00038 }
```

Hace referencia a [max](#).

### 8.82.2.6 setMin()

## Parámetros

<i>min</i>	<input type="text"/>
------------	----------------------

Definición en la línea 50 del archivo [Range.java](#).

```
00050          {  
00051      this.min = min;  
00052 }
```

Hace referencia a [min](#).

### 8.82.3 Documentación de datos miembro

#### 8.82.3.1 data

[Probability](#) evolutionary\_algorithms.complement.Range.data [private]

Definición en la línea 7 del archivo [Range.java](#).

Referenciado por [getData\(\)](#) y [setData\(\)](#).

#### 8.82.3.2 max

float evolutionary\_algorithms.complement.Range.max [private]

Definición en la línea 8 del archivo [Range.java](#).

Referenciado por [getMax\(\)](#) y [setMax\(\)](#).

#### 8.82.3.3 min

float evolutionary\_algorithms.complement.Range.min [private]

Definición en la línea 9 del archivo [Range.java](#).

Referenciado por [getMin\(\)](#) y [setMin\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [Range.java](#)

## 8.83 Referencia de la clase evolutionary\_algorithms.complement.Replace

[Replace](#) - applies replacement strategies for individuals in the population.

Diagrama de herencia de `evolutionary_algorithms.complement.Replace`

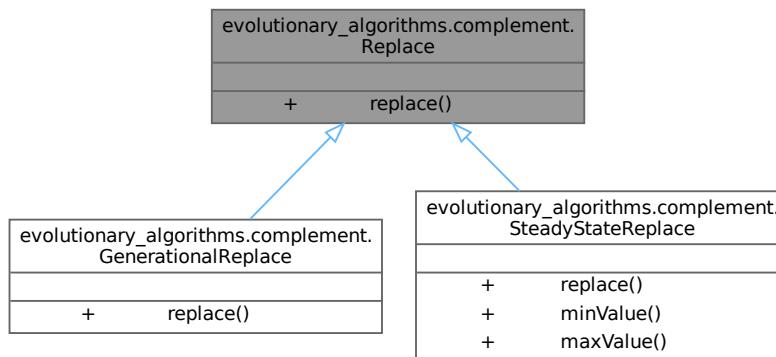
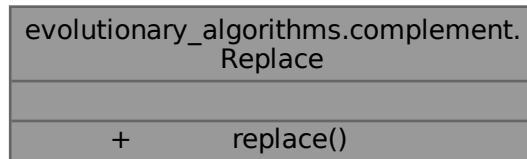


Diagrama de colaboración de `evolutionary_algorithms.complement.Replace`:



### Métodos públicos

- abstract List< `State` > `replace` (`State` stateCandidate, List< `State` >listState) throws `IllegalArgumentException`, `Exception`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*replace - replace the worst individual in the population.*

### 8.83.1 Descripción detallada

[Replace](#) - applies replacement strategies for individuals in the population.

Definición en la línea 12 del archivo [Replace.java](#).

## 8.83.2 Documentación de funciones miembro

### 8.83.2.1 replace()

```
abstract List< State > evolutionary_algorithms.complement.Replace.replace (
    State stateCandidate,
    List< State > listState) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [abstract]
```

replace - replace the worst individual in the population.

#### Parámetros

<i>stateCandidate</i>	
<i>listState</i>	

Devuelve

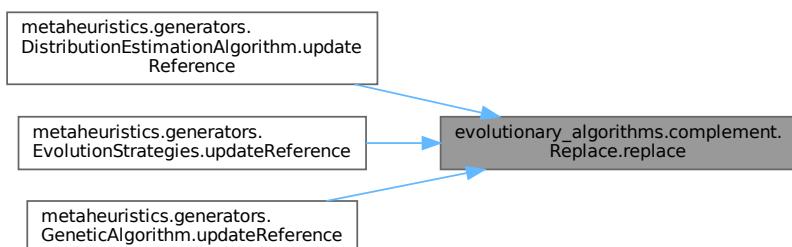
#### Excepciones

<i>IllegalArgumentException</i>	
<i>SecurityException</i>	
<i>ClassNotFoundException</i>	
<i>InstantiationException</i>	
<i>IllegalAccessException</i>	
<i>InvocationTargetException</i>	
<i>NoSuchMethodException</i>	

Reimplementado en [evolutionary\\_algorithms.complement.GenerationalReplace](#) y [evolutionary\\_algorithms.complement.SteadyStateReplace](#).

Referenciado por [metaheuristics.generators.DistributionEstimationAlgorithm.updateReference\(\)](#), [metaheuristics.generators.EvolutionStrategies.updateReference\(\)](#) y [metaheuristics.generators.GeneticAlgorithm.updateReference\(\)](#).

Gráfico de llamadas a esta función:



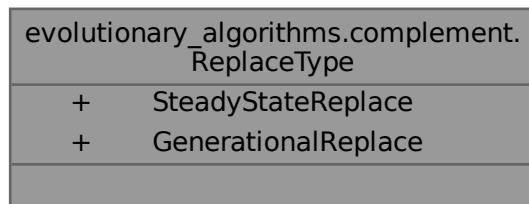
La documentación de esta clase está generada del siguiente archivo:

- [Replace.java](#)

## 8.84 Referencia de la enumeración `evolutionary_algorithms.complement.ReplaceType`

`ReplaceType` - represents the type of replacement strategy.

Diagrama de colaboración de `evolutionary_algorithms.complement.ReplaceType`:



### Atributos públicos

- `SteadyStateReplace`
- `GenerationalReplace`

### 8.84.1 Descripción detallada

`ReplaceType` - represents the type of replacement strategy.

Definición en la línea 6 del archivo [ReplaceType.java](#).

### 8.84.2 Documentación de datos miembro

#### 8.84.2.1 GenerationalReplace

`evolutionary_algorithms.complement.ReplaceType.GenerationalReplace`

Definición en la línea 7 del archivo [ReplaceType.java](#).

#### 8.84.2.2 SteadyStateReplace

`evolutionary_algorithms.complement.ReplaceType.SteadyStateReplace`

Definición en la línea 7 del archivo [ReplaceType.java](#).

La documentación de esta enumeración está generada del siguiente archivo:

- [ReplaceType.java](#)

## 8.85 Referencia de la clase evolutionary\_algorithms.complement.RouletteSelection

[RouletteSelection](#) - applies roulette selection to choose parents.

Diagrama de herencia de evolutionary\_algorithms.complement.RouletteSelection

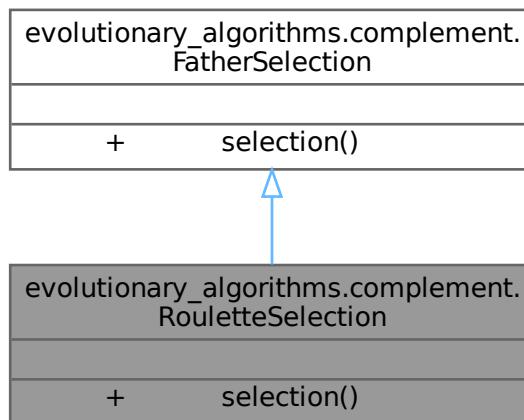
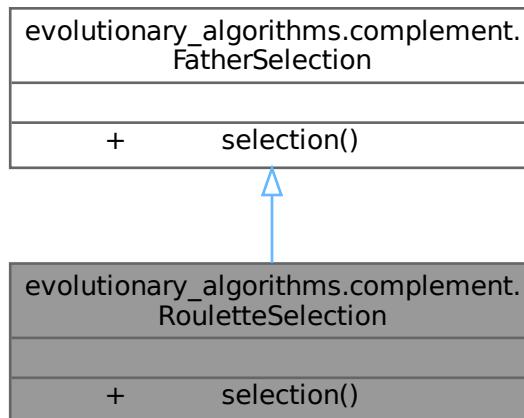


Diagrama de colaboración de `evolutionary_algorithms.complement.RouletteSelection`:



### Métodos públicos

- `List< State > selection (List< State > listState, int truncation)`  
*selection - applies roulette selection to choose parents.*

## 8.85.1 Descripción detallada

[RouletteSelection](#) - applies roulette selection to choose parents.

Definición en la línea 14 del archivo [RouletteSelection.java](#).

## 8.85.2 Documentación de funciones miembro

### 8.85.2.1 selection()

```
List< State > evolutionary_algorithms.complement.RouletteSelection.selection (
    List< State > listState,
    int truncation) [inline]
```

selection - applies roulette selection to choose parents.

#### Parámetros

<i>listState</i>	
<i>truncation</i>	

#### Devuelve

Reimplementado de [evolutionary\\_algorithms.complement.FatherSelection](#).

Definición en la línea 23 del archivo [RouletteSelection.java](#).

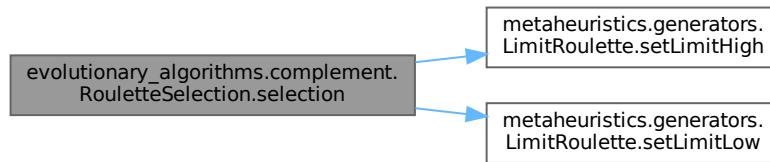
```
00023         float totalWeight = 0;
00024         for (int i = 0; i < listState.size(); i++) {
00025             totalWeight = (float) (listState.get(i).getEvaluation().get(0) + totalWeight);
00026         }
00027         List<Float> listProb = new ArrayList<Float>();
00028         for (int i = 0; i < listState.size(); i++) {
00029             float probF = (float) (listState.get(i).getEvaluation().get(0) / totalWeight);
00030             listProb.add(probF);
00031         }
00032         List<LimitRoulette> listLimit = new ArrayList<LimitRoulette>();
00033         float limitHigh = 0;
00034         float limitLow = 0;
00035         for (int i = 0; i < listProb.size(); i++) {
00036             LimitRoulette limitRoulette = new LimitRoulette();
00037             limitHigh = listProb.get(i) + limitHigh;
00038             limitRoulette.setLimitHigh(limitHigh);
00039             limitRoulette.setLimitLow(limitLow);
00040             limitLow = limitHigh;
00041             // limitRoulette.setGenerator(listGenerators.get(i));
00042             listLimit.add(limitRoulette);
00043         }
00044         List<State> fatherList = new ArrayList<State>();
00045         for (int j = 0; j < listState.size(); j++) {
00046             // Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.
00047             // This RNG decides selection in the evolutionary algorithm and is not
00048             // used for security-sensitive purposes. Suppress Sonar hotspot S2245.
00049             @SuppressWarnings("squid:S2245")
00050             float numbAleatory = (float) ThreadLocalRandom.current().nextDouble();
00051             boolean find = false;
00052             int i = 0;
00053             while ((find == false) && (i < listLimit.size())){
00054                 if((listLimit.get(i).getLimitLow() <= numbAleatory) && (numbAleatory <=
00055                     listLimit.get(i).getLimitHigh())){
00056                     find = true;
00057                     fatherList.add(listState.get(i));
```

```

00058         }
00059         else i++;
00060     }
00061 }
00062 return fatherList;
00063 }
```

Hace referencia a [metaheuristics.generators.LimitRoulette.setLimitHigh\(\)](#) y [metaheuristics.generators.LimitRoulette.setLimitLow\(\)](#).

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [RouletteSelection.java](#)

## 8.86 Referencia de la clase evolutionary\_algorithms.complement.Sampling

[Sampling](#) - represents a sampling strategy for selecting individuals.

Diagrama de herencia de `evolutionary_algorithms.complement.Sampling`

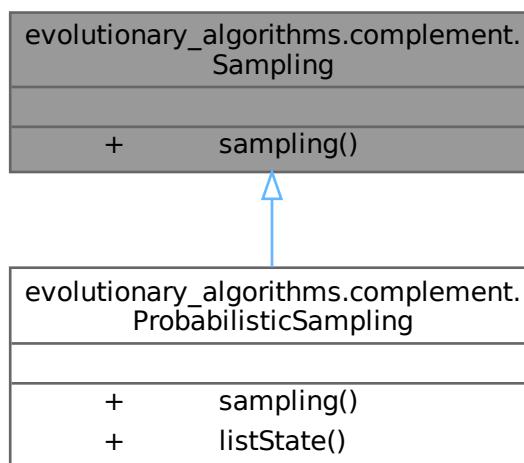
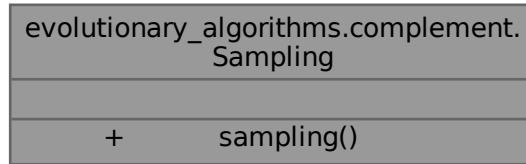


Diagrama de colaboración de `evolutionary_algorithms.complement.Sampling`:



## Métodos públicos

- abstract List< [State](#) > [sampling](#) (List< [State](#) > fathers, int countInd)  
*sampling - samples a given number of individuals from the list of fathers.*

### 8.86.1 Descripción detallada

[Sampling](#) - represents a sampling strategy for selecting individuals.

Definición en la línea 11 del archivo [Sampling.java](#).

### 8.86.2 Documentación de funciones miembro

#### 8.86.2.1 [sampling\(\)](#)

```
abstract List< State > evolutionary_algorithms.complement.Sampling.sampling (
    List< State > fathers,
    int countInd) [abstract]
```

[sampling](#) - samples a given number of individuals from the list of fathers.

#### Parámetros

<i>fathers</i>	
<i>countInd</i>	

Devuelve

Reimplementado en [evolutionary\\_algorithms.complement.ProbabilisticSampling](#).

Referenciado por [metaheuristics.generators.DistributionEstimationAlgorithm.generate\(\)](#).

Gráfico de llamadas a esta función:



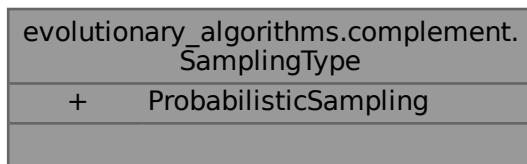
La documentación de esta clase está generada del siguiente archivo:

- [Sampling.java](#)

## 8.87 Referencia de la enumeración evolutionary\_algorithms.complement.SamplingType

[SamplingType](#) - represents the type of sampling strategy.

Diagrama de colaboración de [evolutionary\\_algorithms.complement.SamplingType](#):



### Atributos públicos

- [ProbabilisticSampling](#)

#### 8.87.1 Descripción detallada

[SamplingType](#) - represents the type of sampling strategy.

Definición en la línea 6 del archivo [SamplingType.java](#).

## 8.87.2 Documentación de datos miembro

### 8.87.2.1 ProbabilisticSampling

`evolutionary_algorithms.complement.SamplingType.ProbabilisticSampling`

Definición en la línea 7 del archivo `SamplingType.java`.

Referenciado por [metaheuristics.generators.DistributionEstimationAlgorithm.DistributionEstimationAlgorithm\(\)](#).

La documentación de esta enumeración está generada del siguiente archivo:

- [SamplingType.java](#)

## 8.88 Referencia de la clase

### `local_search.candidate_type.SearchCandidate`

[SearchCandidate](#) - abstract base class for candidate selection strategies.

Diagrama de herencia de `local_search.candidate_type.SearchCandidate`

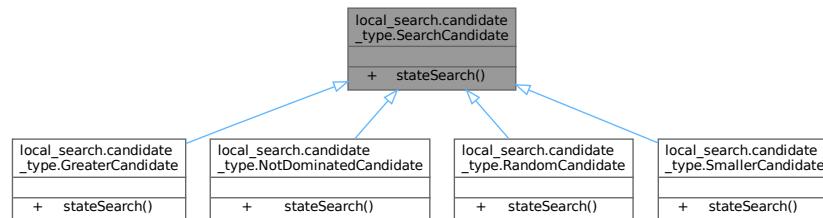
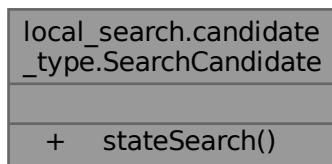


Diagrama de colaboración de `local_search.candidate_type.SearchCandidate`:



## Métodos públicos

- abstract `State stateSearch (List< State > listNeighborhood) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException`

*Choose a candidate state from the provided neighborhood.*

### 8.88.1 Descripción detallada

`SearchCandidate` - abstract base class for candidate selection strategies.

Concrete subclasses implement `stateSearch` to select a candidate `State` from a provided neighborhood.

Definición en la línea 19 del archivo `SearchCandidate.java`.

### 8.88.2 Documentación de funciones miembro

#### 8.88.2.1 stateSearch()

```
abstract State local_search.candidate_type.SearchCandidate.stateSearch (
    List< State > listNeighborhood) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [abstract]
```

*Choose a candidate state from the provided neighborhood.*

#### Parámetros

<code>listNeighborhood</code>	the list of neighboring States to choose from
-------------------------------	---

#### Devuelve

`chosen State`

#### Excepciones

<code>ReflectiveOperationException</code>	when reflective operations fail
---	---------------------------------

Reimplementado en `local_search.candidate_type.GreaterCandidate`, `local_search.candidate_type.NotDominatedCandidate`, `local_search.candidate_type.RandomCandidate` y `local_search.candidate_type.SmallerCandidate`.

Referenciado por `local_search.candidate_type.CandidateValue.stateCandidate()`.

Gráfico de llamadas a esta función:



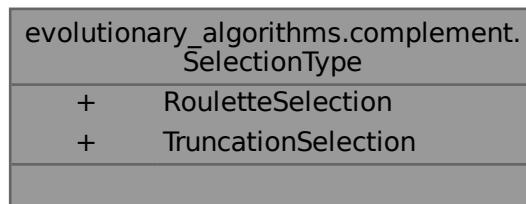
La documentación de esta clase está generada del siguiente archivo:

- `SearchCandidate.java`

## 8.89 Referencia de la enumeración evolutionary\_algorithms.complement.SelectionType

[SelectionType](#) - represents the type of selection strategy.

Diagrama de colaboración de `evolutionary_algorithms.complement.SelectionType`:



### Atributos públicos

- [RouletteSelection](#)
- [TruncationSelection](#)

### 8.89.1 Descripción detallada

[SelectionType](#) - represents the type of selection strategy.

Definición en la línea 6 del archivo [SelectionType.java](#).

### 8.89.2 Documentación de datos miembro

#### 8.89.2.1 RouletteSelection

`evolutionary_algorithms.complement.SelectionType.RouletteSelection`

Definición en la línea 7 del archivo [SelectionType.java](#).

#### 8.89.2.2 TruncationSelection

`evolutionary_algorithms.complement.SelectionType.TruncationSelection`

Definición en la línea 7 del archivo [SelectionType.java](#).

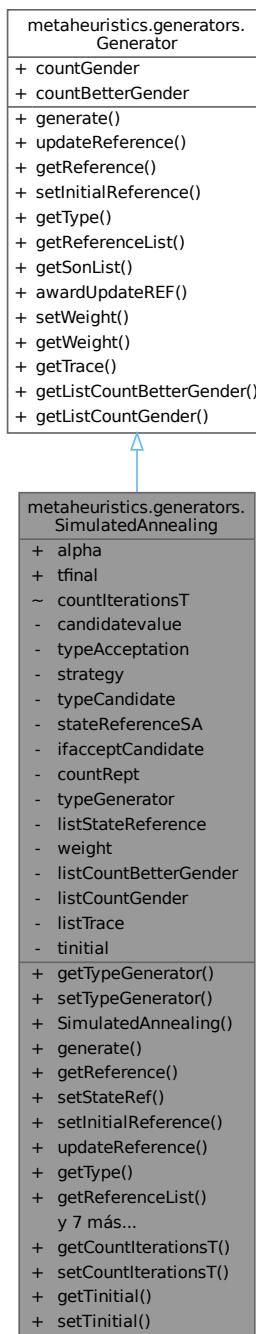
La documentación de esta enumeración está generada del siguiente archivo:

- [SelectionType.java](#)

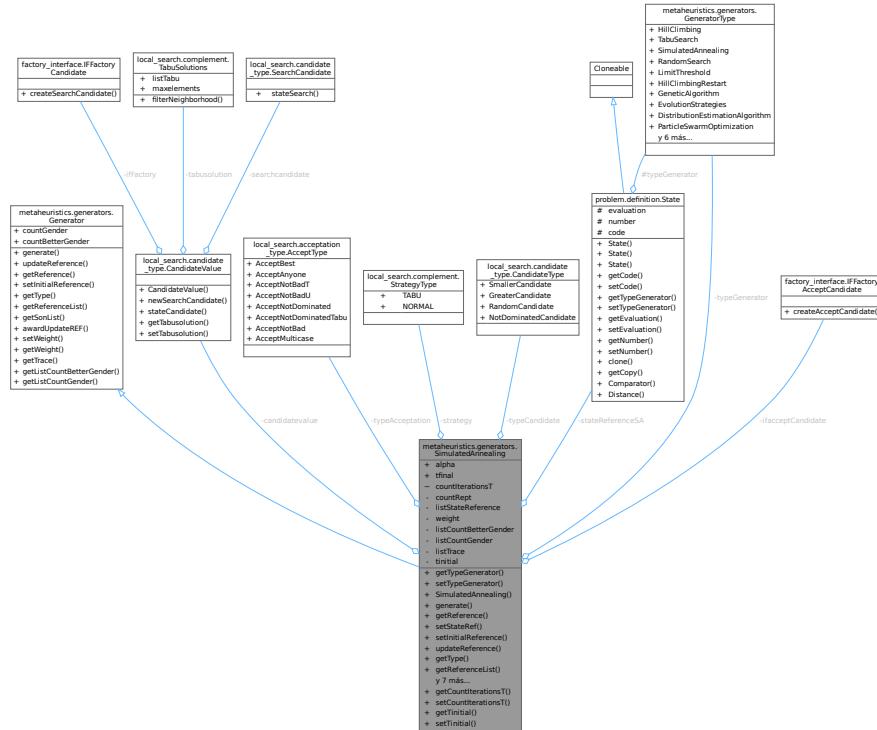
## 8.90 Referencia de la clase metaheuristics.generators.SimulatedAnnealing

[SimulatedAnnealing](#) - class that implements the Simulated Annealing metaheuristic.

Diagrama de herencia de metaheuristics.generators.SimulatedAnnealing



## Diagrama de colaboración de metaheuristics.generators.SimulatedAnnealing:



## Métodos públicos

- **GeneratorType getTypeGenerator ()**  
*getTypeGenerator - get the type of the generator.*
  - void **setTypeGenerator (GeneratorType typeGenerator)**  
*setTypeGenerator - set the type of the generator.*
  - **SimulatedAnnealing ()**  
*SimulatedAnnealing - descripción (añade detalles).*
  - **State generate (Integer operatornumber)** throws IllegalArgumentException, SecurityException, ClassNotFoundException, NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException  
*generate - method to generate a new state.*
  - **State getReference ()**  
*getReference - get the reference state.*
  - void **setStateRef (State stateRef)**  
*setStateRef - set the reference state.*
  - void **setInitialReference (State stateInitialRef)**  
*setInitialReference - set the initial reference state.*
  - void **updateReference (State stateCandidate, Integer countIterationsCurrent)** throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException  
*updateReference - update the reference state.*
  - **GeneratorType getType ()**  
*getType - get the type of the generator.*
  - List< State > **getReferenceList ()**

- `getReferenceList` - get the list of reference states.
- List< `State` > `getSonList` ()
  - getSonList* - get the list of child states.
- boolean `awardUpdateREF` (`State` stateCandidate)
  - awardUpdateREF* - award the update of the reference state.
- float `getWeight` ()
  - getWeight* - get the weight of the generator.
- void `setWeight` (float `weight`)
  - setWeight* - set the weight of the generator.
- int[] `getListCountBetterGender` ()
  - getListCountBetterGender* - get the list of better gender counts.
- int[] `getListCountGender` ()
  - getListCountGender* - get the list of gender counts.
- float[] `getTrace` ()
  - getTrace* - get the trace of the generator.

## Métodos públicos estáticos

- static int `getCountIterationsT` ()
  - getCountIterationsT* - get the count of iterations for temperature update.
- static void `setCountIterationsT` (int c)
  - setCountIterationsT* - set the count of iterations for temperature update.
- static Double `getTinitial` ()
  - Getter for initial temperature.*
- static void `setTinitial` (Double t)
  - Setter for initial temperature.*

## Atributos públicos estáticos

- static final double `alpha` = 0.93
- static final Double `tfinal` = 41.66

## Atributos privados

- `CandidateValue` candidatevalue
- `AcceptType` typeAcceptation
- `StrategyType` strategy
- `CandidateType` typeCandidate
- `State` stateReferenceSA
- `IFFactoryAcceptCandidate` ifacceptCandidate
- int `countRept`
- `GeneratorType` typeGenerator
- List< `State` > `listStateReference` = new ArrayList<`State`>()
- float `weight`
- int[] `listCountBetterGender` = new int[10]
- int[] `listCountGender` = new int[10]
- float[] `listTrace` = new float[1200000]

### Atributos estáticos privados

- static Double `tinitial` = 250.0

### Otros miembros heredados

### Atributos públicos heredados de `metaheuristics.generators.Generator`

- int `countGender`
- int `countBetterGender`

## 8.90.1 Descripción detallada

`SimulatedAnnealing` - class that implements the Simulated Annealing metaheuristic.

Definición en la línea 23 del archivo `SimulatedAnnealing.java`.

## 8.90.2 Documentación de constructores y destructores

### 8.90.2.1 `SimulatedAnnealing()`

```
metaheuristics.generators.SimulatedAnnealing.SimulatedAnnealing () [inline]
```

`SimulatedAnnealing` - descripción (añade detalles).

Definición en la línea 98 del archivo `SimulatedAnnealing.java`.

```
00098         {
00099
00100     super();
00101     this.typeAcceptation = AcceptType.AcceptNotBadT;
00102     this.strategy = StrategyType.NORMAL;
00103     this.typeCandidate = CandidateType.RandomCandidate;
00104     this.candidatevalue = new CandidateValue();
00105     this.typeGenerator = GeneratorType.SimulatedAnnealing;
00106     this.weight = 50;
00107     listTrace[0] = this.weight;
00108     listCountBetterGender[0] = 0;
00109     listCountGender[0] = 0;
00110 }
```

Hace referencia a `local_search.acceptation_type.AcceptType.AcceptNotBadT`, `listCountBetterGender`, `listCountGender`, `listTrace`, `local_search.complement.StrategyType.NORMAL`, `local_search.candidate_type.CandidateType.RandomCandidate`, `metaheuristics.generators.GeneratorType.SimulatedAnnealing` y `weight`.

Referenciado por `updateReference()`.

Gráfico de llamadas a esta función:



## 8.90.3 Documentación de funciones miembro

**Parámetros**

<i>stateCandidate</i>	<input type="button" value=""/>
-----------------------	---------------------------------

**Devuelve**

return true if the update is awarded, false otherwise

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 205 del archivo [SimulatedAnnealing.java](#).

```
00205
00206     // TODO Auto-generated method stub
00207     return false;
00208 }
```

**8.90.3.2 generate()**

```
State metaheuristics.generators.SimulatedAnnealing.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

generate - method to generate a new state.

**Parámetros**

<i>operatornumber</i>	<input type="button" value=""/>
-----------------------	---------------------------------

**Devuelve**

return the generated state

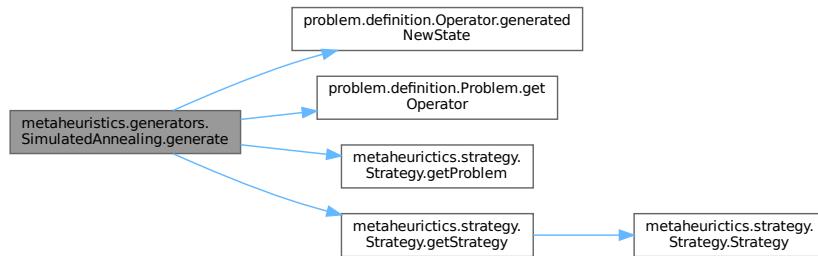
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 118 del archivo [SimulatedAnnealing.java](#).

```
00118
00119     {
00120         List<State> neighborhood =
00121             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceSA, operatornumber);
00122         State statecandidate = candidatevalue.stateCandidate(stateReferenceSA, typeCandidate,
00123             strategy, operatornumber, neighborhood);
00124         return statecandidate;
00125     }
```

Hace referencia a [candidatevalue](#), [problem.definition.Operator.generatedNewState\(\)](#), [problem.definition.Problem.getOperator\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [stateReferenceSA](#), [strategy](#) y [typeCandidate](#).

Gráfico de llamadas de esta función:



### 8.90.3.3 getCountIterationsT()

```
int metaheuristics.generators.SimulatedAnnealing.getCountIterationsT () [inline], [static]
getCountIterationsT - get the count of iterations for temperature update.
```

Devuelve

the count of iterations for temperature update

Definición en la línea 42 del archivo [SimulatedAnnealing.java](#).

```
00042
00043     return countIterationsT;
00044 }
```

Referenciado por [updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.90.3.4 getListCountBetterGender()

```
int[] metaheuristics.generators.SimulatedAnnealing.getListCountBetterGender () [inline]
getListCountBetterGender - get the list of better gender counts.
```

Devuelve

the list of better gender counts

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 236 del archivo [SimulatedAnnealing.java](#).

```
00236
00237     // TODO Auto-generated method stub
00238     return (this.listCountBetterGender == null) ? new int[0] :
00239         Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
```

Hace referencia a [listCountBetterGender](#).

### 8.90.3.5 getListCountGender()

```
int[] metaheuristics.generators.SimulatedAnnealing.getListCountGender () [inline]
```

getListCountGender - get the list of gender counts.

Devuelve

the list of gender counts

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 246 del archivo [SimulatedAnnealing.java](#).

```
00246      {
00247          // TODO Auto-generated method stub
00248          return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00249              this.listCountGender.length);
}
```

Hace referencia a [listCountGender](#).

### 8.90.3.6 getReference()

```
State metaheuristics.generators.SimulatedAnnealing.getReference () [inline]
```

getReference - get the reference state.

Devuelve

the reference state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 129 del archivo [SimulatedAnnealing.java](#).

```
00129      {
00130          return stateReferenceSA;
00131      }
```

Hace referencia a [stateReferenceSA](#).

### 8.90.3.7 getReferenceList()

```
List< State > metaheuristics.generators.SimulatedAnnealing.getReferenceList () [inline]
```

getReferenceList - get the list of reference states.

Devuelve

the list of reference states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 184 del archivo [SimulatedAnnealing.java](#).

```
00184      {
00185          listStateReference.add(stateReferenceSA);
00186          return new ArrayList<State>(listStateReference);
00187      }
```

Hace referencia a [listStateReference](#) y [stateReferenceSA](#).

### 8.90.3.8 getSonList()

```
List< State > metaheuristics.generators.SimulatedAnnealing.getSonList () [inline]
```

getSonList - get the list of child states.

Devuelve

the list of child states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 194 del archivo [SimulatedAnnealing.java](#).

```
00194 {
00195     // TODO Auto-generated method stub
00196     return null;
00197 }
```

### 8.90.3.9 getTinitial()

```
Double metaheuristics.generators.SimulatedAnnealing.getTinitial () [inline], [static]
```

Getter for initial temperature.

Definición en la línea 84 del archivo [SimulatedAnnealing.java](#).

```
00084 {
00085     return tinitial;
00086 }
```

Hace referencia a [tinitial](#).

Referenciado por [local\\_search.acceptation\\_type.AcceptNotBadT.acceptCandidate\(\)](#).

Gráfico de llamadas a esta función:



### 8.90.3.10 getTrace()

```
float[] metaheuristics.generators.SimulatedAnnealing.getTrace () [inline]
```

getTrace -get the trace of the generator.

Devuelve

the trace of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 256 del archivo [SimulatedAnnealing.java](#).

```
00256 {
00257     // TODO Auto-generated method stub
00258     return \(this.listTrace == null\) ? new float\[0\] : Arrays.copyOf\(this.listTrace,
00259         this.listTrace.length\);
```

Hace referencia a [listTrace](#).

### 8.90.3.11 getType()

```
GeneratorType metaheuristics.generators.SimulatedAnnealing.getType () [inline]
```

getType - get the type of the generator.

Devuelve

the type of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 175 del archivo [SimulatedAnnealing.java](#).

```
00175             {
00176         return this.typeGenerator;
00177     }
```

### 8.90.3.12 getTypeGenerator()

```
GeneratorType metaheuristics.generators.SimulatedAnnealing.getTypeGenerator () [inline]
```

getTypeGenerator - get the type of the generator.

Devuelve

the type of the generator

Definición en la línea 69 del archivo [SimulatedAnnealing.java](#).

```
00069             {
00070         return typeGenerator;
00071     }
```

Hace referencia a [typeGenerator](#).

### 8.90.3.13 getWeight()

```
float metaheuristics.generators.SimulatedAnnealing.getWeight () [inline]
```

getWeight - get the weight of the generator.

Devuelve

the weight of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 216 del archivo [SimulatedAnnealing.java](#).

```
00216             {
00217         // TODO Auto-generated method stub
00218         return this.weight;
00219     }
```

Hace referencia a [weight](#).

## Parámetros

c	the count of iterations to set
---	--------------------------------

Definición en la línea 50 del archivo [SimulatedAnnealing.java](#).

```
00050
00051     countIterationsT = c;
00052 }
```

### 8.90.3.15 setInitialReference()

```
void metaheuristics.generators.SimulatedAnnealing.setInitialReference (
    State stateInitialRef) [inline]
```

setInitialReference - set the initial reference state.

## Parámetros

stateInitialRef	the initial reference state to set
-----------------	------------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 146 del archivo [SimulatedAnnealing.java](#).

```
00146
00147     this.stateReferenceSA = stateInitialRef;
00148 }
```

### 8.90.3.16 setStateRef()

```
void metaheuristics.generators.SimulatedAnnealing.setStateRef (
    State stateRef) [inline]
```

setStateRef - set the reference state.

## Parámetros

stateRef	the reference state to set
----------	----------------------------

Definición en la línea 137 del archivo [SimulatedAnnealing.java](#).

```
00137
00138     this.stateReferenceSA = stateRef;
00139 }
```

### 8.90.3.17 setTinitial()

```
void metaheuristics.generators.SimulatedAnnealing.setTinitial (
    Double t) [inline], [static]
```

Setter for initial temperature.

Definición en la línea 91 del archivo [SimulatedAnnealing.java](#).

```
00091
00092     tinitial = t;
00093 }
```

Hace referencia a [tinitial](#).

### 8.90.3.18 setTypeGenerator()

```
void metaheuristics.generators.SimulatedAnnealing.setTypeGenerator (
    GeneratorType typeGenerator) [inline]
```

setTypeGenerator - set the type of the generator.

#### Parámetros

<i>typeGenerator</i>	
----------------------	--

Definición en la línea 77 del archivo [SimulatedAnnealing.java](#).

```
00077
00078      this.typeGenerator = typeGenerator;
00079 }
```

Hace referencia a [typeGenerator](#).

### 8.90.3.19 setWeight()

```
void metaheuristics.generators.SimulatedAnnealing.setWeight (
    float weight) [inline]
```

setWeight - set the weight of the generator.

#### Parámetros

<i>weight</i>	the weight to set
---------------	-------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 226 del archivo [SimulatedAnnealing.java](#).

```
00226
00227      // TODO Auto-generated method stub
00228      this.weight = weight;
00229 }
```

Hace referencia a [weight](#).

### 8.90.3.20 updateReference()

```
void metaheuristics.generators.SimulatedAnnealing.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

updateReference - update the reference state.

## Parámetros

<i>stateCandidate</i>	
<i>countIterationsCurrent</i>	

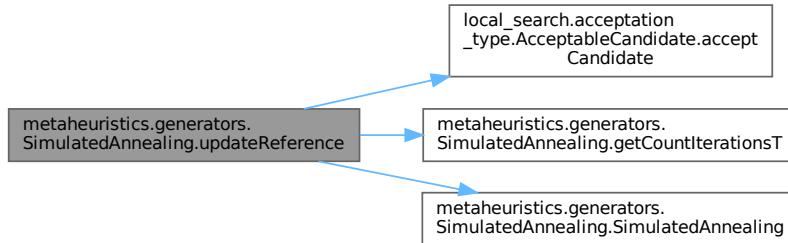
Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 156 del archivo [SimulatedAnnealing.java](#).

```
00156
00157     {
00158         countRept = countIterationsT;
00159         ifacceptCandidate = new FactoryAcceptCandidate();
00160         AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00161         Boolean accept = candidate.acceptCandidate(stateReferenceSA, stateCandidate);
00162         if(accept.equals(true))
00163             stateReferenceSA = stateCandidate;
00164         if(countIterationsCurrent.equals(getCountIterationsT())){
00165             // use static setters to avoid writing the static field from an instance method
00166             SimulatedAnnealing.setTinitial(SimulatedAnnealing.getTinitial() * alpha);
00167             SimulatedAnnealing.setCountIterationsT(SimulatedAnnealing.getCountIterationsT() +
00168             countRept);
00169         }
00170     }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptableCandidate.acceptCandidate\(\)](#), [alpha](#), [countRept](#), [getCountIterationsT\(\)](#), [ifacceptCandidate](#), [SimulatedAnnealing\(\)](#), [stateReferenceSA](#) y [typeAcceptation](#).

Gráfico de llamadas de esta función:



## 8.90.4 Documentación de datos miembro

### 8.90.4.1 alpha

```
final double metaheuristics.generators.SimulatedAnnealing.alpha = 0.93 [static]
```

Definición en la línea 32 del archivo [SimulatedAnnealing.java](#).

Referenciado por [updateReference\(\)](#).

### 8.90.4.2 candidatevalue

```
CandidateValue metaheuristics.generators.SimulatedAnnealing.candidatevalue [private]
```

Definición en la línea 25 del archivo [SimulatedAnnealing.java](#).

Referenciado por [generate\(\)](#).

#### 8.90.4.3 countRept

```
int metaheuristics.generators.SimulatedAnnealing.countRept [private]
```

Definición en la línea 53 del archivo [SimulatedAnnealing.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.90.4.4 ifacceptCandidate

```
IFFactoryAcceptCandidate metaheuristics.generators.SimulatedAnnealing.ifacceptCandidate [private]
```

Definición en la línea 30 del archivo [SimulatedAnnealing.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.90.4.5 listCountBetterGender

```
int [] metaheuristics.generators.SimulatedAnnealing.listCountBetterGender = new int[10] [private]
```

Definición en la línea 60 del archivo [SimulatedAnnealing.java](#).

Referenciado por [getListCountBetterGender\(\)](#) y [SimulatedAnnealing\(\)](#).

#### 8.90.4.6 listCountGender

```
int [] metaheuristics.generators.SimulatedAnnealing.listCountGender = new int[10] [private]
```

Definición en la línea 61 del archivo [SimulatedAnnealing.java](#).

Referenciado por [getListCountGender\(\)](#) y [SimulatedAnnealing\(\)](#).

#### 8.90.4.7 listStateReference

```
List<State> metaheuristics.generators.SimulatedAnnealing.listStateReference = new Array←  
List<State>() [private]
```

Definición en la línea 55 del archivo [SimulatedAnnealing.java](#).

Referenciado por [getReferenceList\(\)](#).

#### 8.90.4.8 listTrace

```
float [] metaheuristics.generators.SimulatedAnnealing.listTrace = new float[1200000] [private]
```

Definición en la línea 62 del archivo [SimulatedAnnealing.java](#).

Referenciado por [getTrace\(\)](#) y [SimulatedAnnealing\(\)](#).

#### 8.90.4.9 stateReferenceSA

```
State metaheuristics.generators.SimulatedAnnealing.stateReferenceSA [private]
```

Definición en la línea 29 del archivo [SimulatedAnnealing.java](#).

Referenciado por [generate\(\)](#), [getReference\(\)](#), [getReferenceList\(\)](#) y [updateReference\(\)](#).

#### 8.90.4.10 strategy

```
StrategyType metaheuristics.generators.SimulatedAnnealing.strategy [private]
```

Definición en la línea 27 del archivo [SimulatedAnnealing.java](#).

Referenciado por [generate\(\)](#).

#### 8.90.4.11 tfinal

```
final Double metaheuristics.generators.SimulatedAnnealing.tfinal = 41.66 [static]
```

Definición en la línea 35 del archivo [SimulatedAnnealing.java](#).

#### 8.90.4.12 tinitial

```
Double metaheuristics.generators.SimulatedAnnealing.tinitial = 250.0 [static], [private]
```

Definición en la línea 34 del archivo [SimulatedAnnealing.java](#).

Referenciado por [getTinitial\(\)](#) y [setTinitial\(\)](#).

#### 8.90.4.13 typeAcceptation

```
AcceptType metaheuristics.generators.SimulatedAnnealing.typeAcceptation [private]
```

Definición en la línea 26 del archivo [SimulatedAnnealing.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.90.4.14 typeCandidate

```
CandidateType metaheuristics.generators.SimulatedAnnealing.typeCandidate [private]
```

Definición en la línea 28 del archivo [SimulatedAnnealing.java](#).

Referenciado por [generate\(\)](#).

#### 8.90.4.15 typeGenerator

```
GeneratorType metaheuristics.generators.SimulatedAnnealing.typeGenerator [private]
```

Definición en la línea 54 del archivo [SimulatedAnnealing.java](#).

Referenciado por [getTypeGenerator\(\)](#) y [setTypeGenerator\(\)](#).

#### 8.90.4.16 weight

```
float metaheuristics.generators.SimulatedAnnealing.weight [private]
```

Definición en la línea 56 del archivo [SimulatedAnnealing.java](#).

Referenciado por [getWeight\(\)](#), [setWeight\(\)](#) y [SimulatedAnnealing\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [SimulatedAnnealing.java](#)

## 8.91 Referencia de la clase local\_search.candidate\_type.SmallerCandidate

[SmallerCandidate](#) - choose the neighbor with the smallest evaluation.

Diagrama de herencia de local\_search.candidate\_type.SmallerCandidate

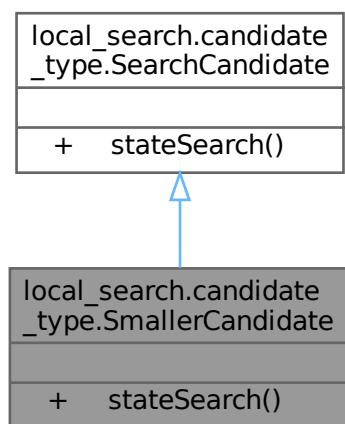
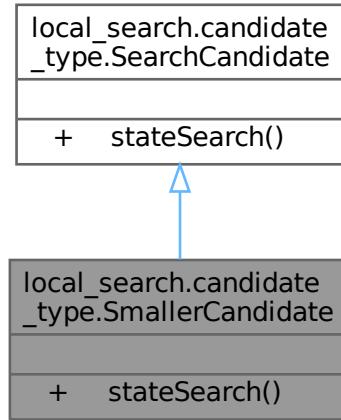


Diagrama de colaboración de local\_search.candidate\_type.SmallerCandidate:



## Métodos públicos

- **State stateSearch (List< State > listNeighborhood)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`

*Select the smallest neighbor from the list.*

### 8.91.1 Descripción detallada

[SmallerCandidate](#) - choose the neighbor with the smallest evaluation.

Scans the provided neighborhood and returns the state with the smallest primary objective value.

Definición en la línea 19 del archivo [SmallerCandidate.java](#).

### 8.91.2 Documentación de funciones miembro

#### 8.91.2.1 stateSearch()

```

State local_search.candidate_type.SmallerCandidate.stateSearch (
    List< State > listNeighborhood) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
  
```

Select the smallest neighbor from the list.

## Parámetros

<i>listNeighborhood</i>	list of neighbor states
-------------------------	-------------------------

## Devuelve

the neighbor with the smallest evaluation (primary objective)

## Excepciones

<i>ReflectiveOperationException</i>	propagated from potential evaluations
-------------------------------------	---------------------------------------

Reimplementado de [local\\_search.candidate\\_type.SearchCandidate](#).

Definición en la línea 29 del archivo [SmallerCandidate.java](#).

```

00029
00030     {
00031         State stateSmaller = null;
00032         if(listNeighborhood.size() > 1){
00033             double counter = 0;
00034             double currentCount = listNeighborhood.get(0).getEvaluation().get(0);
00035             for (int i = 1; i < listNeighborhood.size(); i++) {
00036                 counter = listNeighborhood.get(i).getEvaluation().get(0);
00037                 if (counter < currentCount) {
00038                     currentCount = counter;
00039                     stateSmaller = listNeighborhood.get(i);
00040                 }
00041             }
00042         }
00043         else stateSmaller = listNeighborhood.get(0);
00044         return stateSmaller;
00045     }

```

La documentación de esta clase está generada del siguiente archivo:

- [SmallerCandidate.java](#)

## 8.92 Referencia de la clase problem.extension.SolutionMethod

[SolutionMethod](#).

Diagrama de herencia de `problem.extension.SolutionMethod`

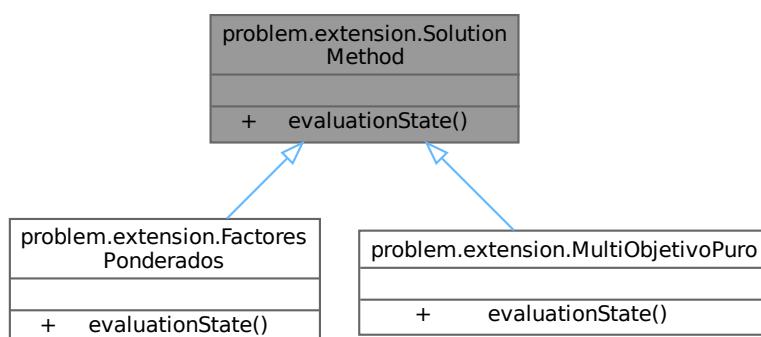
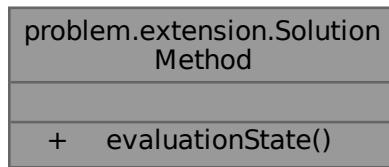


Diagrama de colaboración de problem.extension.SolutionMethod:



## Métodos públicos

- abstract void [evaluationState \(State state\)](#)  
*Evaluar un estado según el método de solución concreto.*

### 8.92.1 Descripción detallada

[SolutionMethod](#).

Abstracción para métodos concretos que evalúan un [problem.definition.State](#).

Implementaciones específicas calculan y asignan la lista de evaluaciones del estado.

Definición en la línea 12 del archivo [SolutionMethod.java](#).

### 8.92.2 Documentación de funciones miembro

#### 8.92.2.1 evaluationState()

```
abstract void problem.extension.SolutionMethod.evaluationState (
    State state) [abstract]
```

Evaluar un estado según el método de solución concreto.

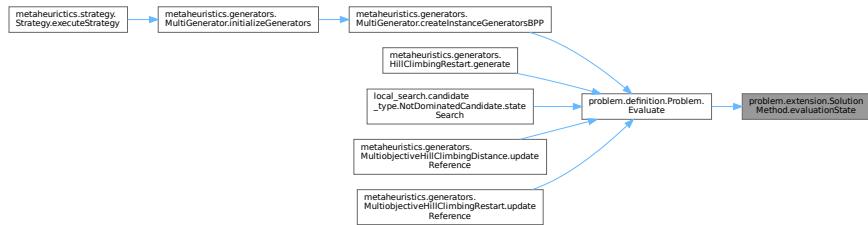
#### Parámetros

<code>state</code>	estado a evaluar; se espera que la implementación modifique la lista de evaluaciones del estado.
--------------------	--

Reimplementado en [problem.extension.FactoresPonderados](#) y [problem.extension.MultiObjetivoPuro](#).

Referenciado por [problem.definition.Problem.Evaluate\(\)](#).

Gráfico de llamadas a esta función:



La documentación de esta clase está generada del siguiente archivo:

- [SolutionMethod.java](#)

## 8.93 Referencia de la clase problem.definition.State

[State.](#)

Diagrama de herencia de problem.definition.State

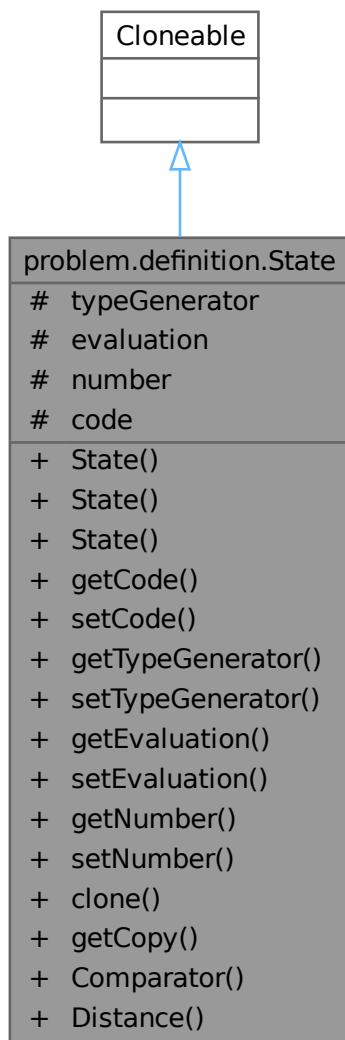
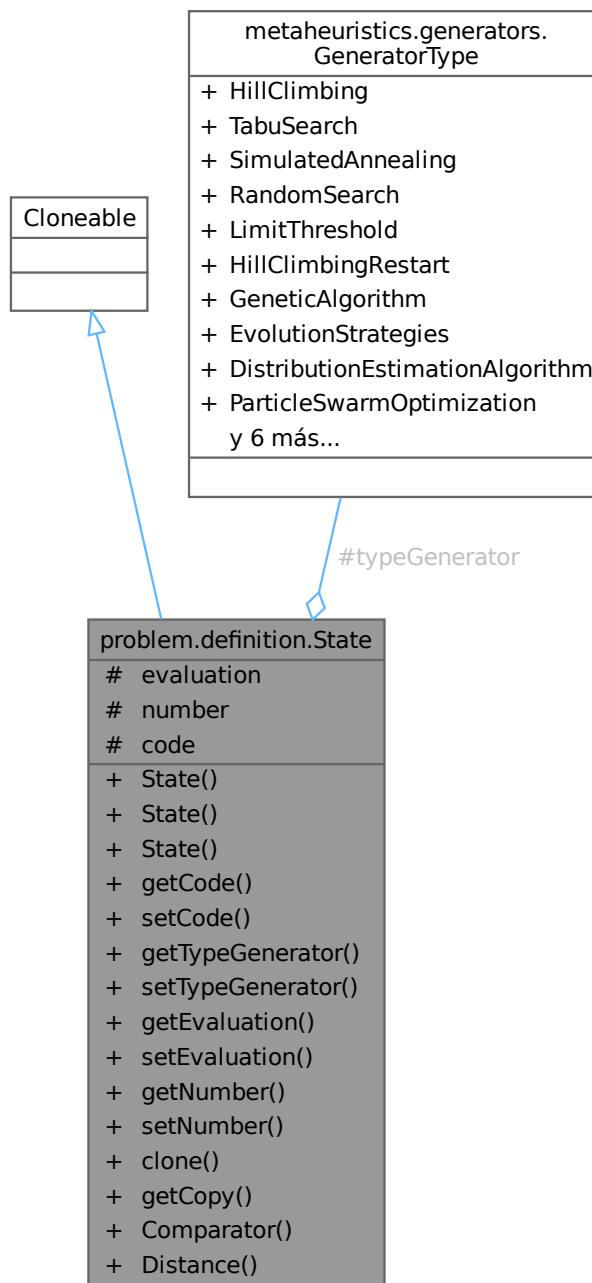


Diagrama de colaboración de problem.definition.State:



## Métodos públicos

- `State (State ps)`  
*Constructor copia.*
- `State (ArrayList< Object > code)`  
*Construir un estado a partir de la representación de código.*
- `State ()`

- Constructor por defecto: crea un estado con código vacío.*
- `ArrayList< Object > getCode ()`

*Obtener la codificación del estado (copia defensiva).*
  - `void setCode (ArrayList< Object > listCode)`

*Establecer la codificación del estado (se copia la lista entrante).*
  - `GeneratorType getTypeGenerator ()`

*Obtener el tipo de generador asociado a este estado.*
  - `void setTypeGenerator (GeneratorType typeGenerator)`

*Establecer el tipo de generador responsable de crear este estado.*
  - `ArrayList< Double > getEvaluation ()`

*Obtener la lista de evaluaciones del estado.*
  - `void setEvaluation (ArrayList< Double > evaluation)`

*Establecer la lista de evaluaciones del estado (copia defensiva).*
  - `int getNumber ()`

*Obtener el identificador numérico del estado.*
  - `void setNumber (int number)`

*Asignar el identificador numérico al estado.*
  - `State clone ()`

*clone*
  - `Object getCopy ()`

*Obtener una copia por valor del estado (implementado con copy-constructor).*
  - `boolean Comparator (State state)`

*Comparar la codificación de dos estados.*
  - `double Distance (State state)`

*Calcular la distancia (cantidad de posiciones diferentes) entre dos códigos.*

## Atributos protegidos

- `GeneratorType typeGenerator`
- `ArrayList< Double > evaluation`
- `int number`
- `ArrayList< Object > code`

### 8.93.1 Descripción detallada

`State`.

Representa una solución (estado) dentro del espacio de búsqueda.

Contiene la codificación de la solución, su evaluación (lista de valores para problemas multiobjetivo), un identificador numérico y el tipo de generador que produjo este estado.

Definición en la línea 16 del archivo `State.java`.

### 8.93.2 Documentación de constructores y destructores

#### 8.93.2.1 `State()` [1/3]

---

```
problem.definition.State.State (
    State ps)  [inline]
```

Generado por Doxygen

Constructor copia.

## Parámetros

<code>ps</code>	estado a copiar (puede ser null)
-----------------	----------------------------------

Definición en la línea 30 del archivo [State.java](#).

```

00030         {
00031     if (ps == null) {
00032         this.code = new ArrayList<Object>();
00033         this.evaluation = null;
00034         this.number = 0;
00035         this.typeGenerator = null;
00036         return;
00037     }
00038     typeGenerator = ps.getTypeGenerator();
00039     ArrayList<Double> eval = ps.getEvaluation();
00040     this.evaluation = (eval == null) ? null : new ArrayList<Double>(eval);
00041     number = ps.getNumber();
00042     ArrayList<Object> c = ps.getCode();
00043     this.code = (c == null) ? new ArrayList<Object>() : new ArrayList<Object>(c);
00044 }
```

Hace referencia a [getCode\(\)](#), [getEvaluation\(\)](#), [getNumber\(\)](#), [getTypeGenerator\(\)](#), [number](#), [State\(\)](#) y [typeGenerator](#).

Referenciado por [clone\(\)](#), [Comparator\(\)](#), [Distance\(\)](#) y [State\(\)](#).

Gráfico de llamadas de esta función:

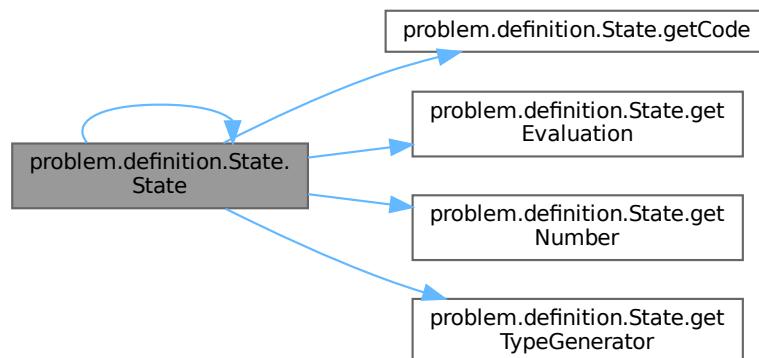
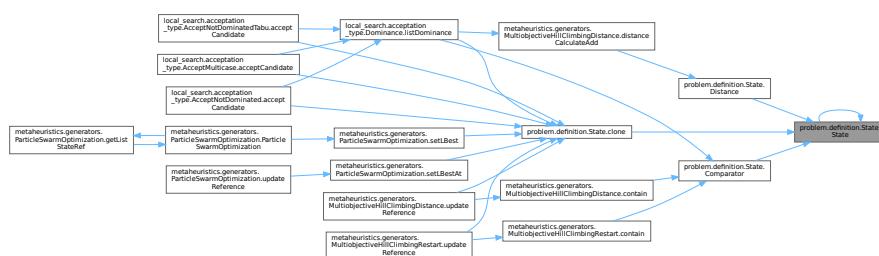


Gráfico de llamadas a esta función:



### 8.93.2.2 State() [2/3]

```
problem.definition.State.State (
    ArrayList< Object > code) [inline]
```

Construir un estado a partir de la representación de código.

#### Parámetros

<code>code</code>	lista de objetos que representan la codificación interna
-------------------	--

Definición en la línea 51 del archivo [State.java](#).

```
00051             {
00052     super();
00053     this.code = (code == null) ? new ArrayList<Object>() : new ArrayList<Object>(code);
00054 }
```

Hace referencia a `code`.

### 8.93.2.3 State() [3/3]

```
problem.definition.State.State () [inline]
```

Constructor por defecto: crea un estado con código vacío.

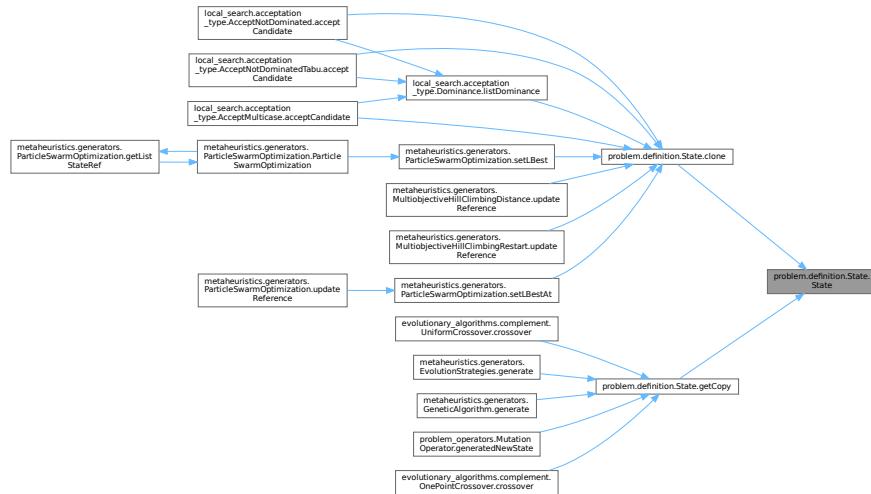
Definición en la línea 59 del archivo [State.java](#).

```
00059         {
00060     code=new ArrayList<Object>();
00061 }
```

Hace referencia a `code`.

Referenciado por `clone()` y `getCopy()`.

Gráfico de llamadas a esta función:



### 8.93.3 Documentación de funciones miembro

#### 8.93.3.1 clone()

`State` `problem.definition.State.clone ()` [inline]

`clone`

Devuelve una copia superficial segura (deep-copy de colecciones mutables) del estado.

Devuelve

nueva instancia `State` con copias de las colecciones

Definición en la línea 142 del archivo `State.java`.

```
00142             {
00143         try {
00144             State s = (State) super.clone();
00145             // deep-copy mutable collections
00146             s.code = (this.code == null) ? new ArrayList<Object>() : new ArrayList<Object>(this.code);
00147             s.evaluation = (this.evaluation == null) ? null : new ArrayList<Double>(this.evaluation);
00148             // primitive and enum-like fields are safe to copy by assignment
00149             return s;
00150         } catch (CloneNotSupportedException e) {
00151             // fallback to copy constructor
00152             return new State(this);
00153         }
00154     }
```

Hace referencia a `State()` y `State()`.

Referenciado por `local_search.acceptation_type.AcceptMulticase.acceptCandidate()`, `local_search.acceptation_type.AcceptNotDominant()`, `local_search.acceptation_type.AcceptNotDominatedTabu.acceptCandidate()`, `local_search.acceptation_type.Dominance.listDominant()`, `metaheuristics.generators.ParticleSwarmOptimization.setLBest()`, `metaheuristics.generators.ParticleSwarmOptimization.setLBestAt()`, `metaheuristics.generators.MultiobjectiveHillClimbingDistance.updateReference()` y `metaheuristics.generators.MultiobjectiveHillClimbingDistance.setLBestAt()`.

Gráfico de llamadas de esta función:

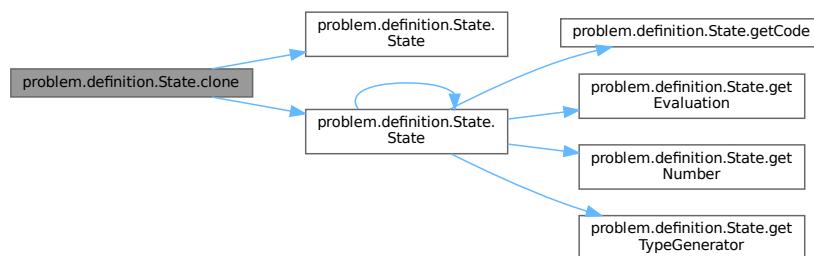
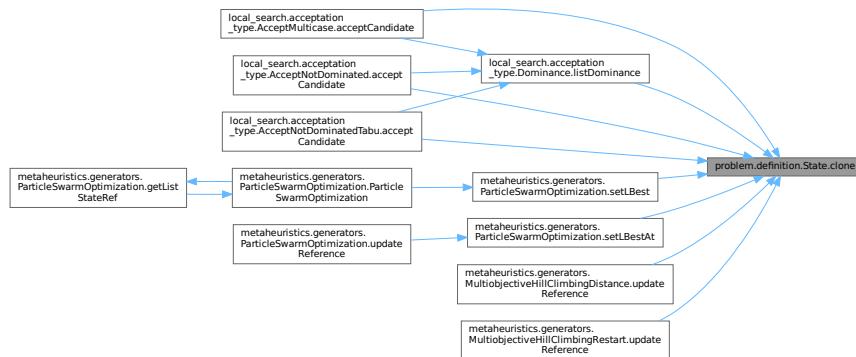


Gráfico de llamadas a esta función:



### 8.93.3.2 Comparator()

```
boolean problem.definition.State.Comparator (
    State state) [inline]
```

Comparar la codificación de dos estados.

#### Parámetros

<code>state</code>	estado con el que comparar
--------------------	----------------------------

#### Devuelve

true si las listas de código son iguales

Definición en la línea 171 del archivo [State.java](#).

```
00171
00172
00173     boolean result=false;
00174     if(state.getCode().equals(getCode())) {
00175         result=true;
00176     }
00177     return result;
00178 }
```

Hace referencia a [getCode\(\)](#) y [State\(\)](#).

Referenciado por [metaheuristics.generators.MultiobjectiveHillClimbingDistance.contains\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance.contains\(\)](#) y [local\\_search.acceptation\\_type.Dominance.listDominance\(\)](#).

Gráfico de llamadas de esta función:

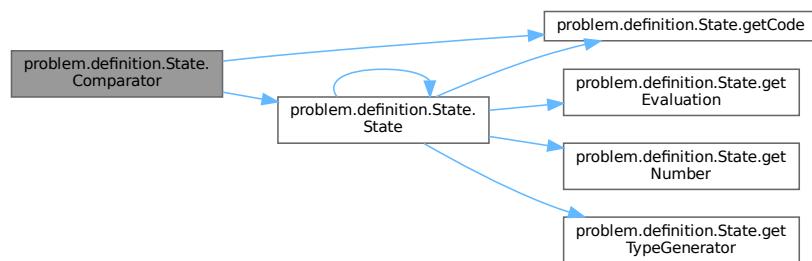
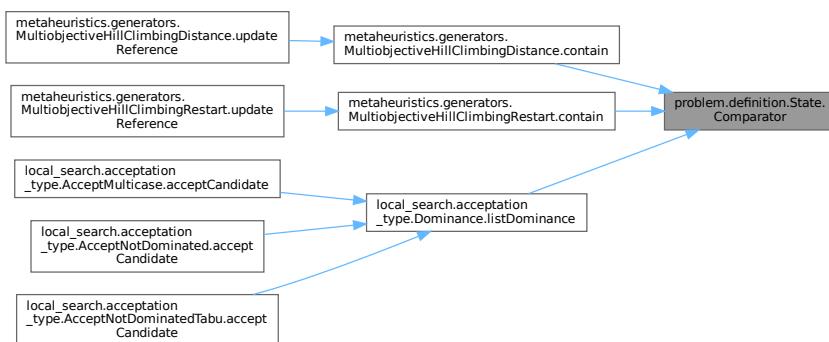


Gráfico de llamadas a esta función:



### 8.93.3.3 Distance()

```
double problem.definition.State.Distance (
    State state) [inline]
```

Calcular la distancia (cantidad de posiciones diferentes) entre dos códigos.

#### Parámetros

<code>state</code>	otro estado a comparar
--------------------	------------------------

#### Devuelve

número de posiciones distintas

Definición en la línea 185 del archivo [State.java](#).

```
00185         double distancia = 0;
00186         for (int i = 0; i < state.getCode().size(); i++) {
00187             if (!state.getCode().get(i).equals(this.getCode().get(i))) {
```

```

00189             distancia++;
00190         }
00191     }
00192     return distancia;
00193 }
```

Hace referencia a [getCode\(\)](#) y [State\(\)](#).

Referenciado por [metaheuristics.generators.MultiobjectiveHillClimbingDistance.distanceCalculateAdd\(\)](#).

Gráfico de llamadas de esta función:

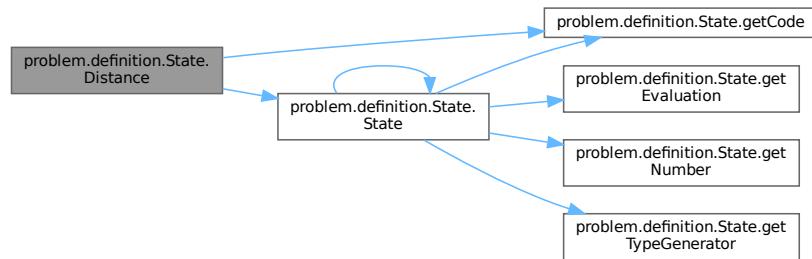
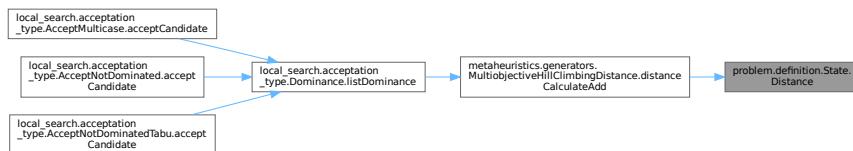


Gráfico de llamadas a esta función:



#### 8.93.3.4 getCode()

```
ArrayList< Object > problem.definition.State.getCode () [inline]
```

Obtener la codificación del estado (copia defensiva).

Devuelve

lista con los elementos de código

Definición en la línea 68 del archivo [State.java](#).

```

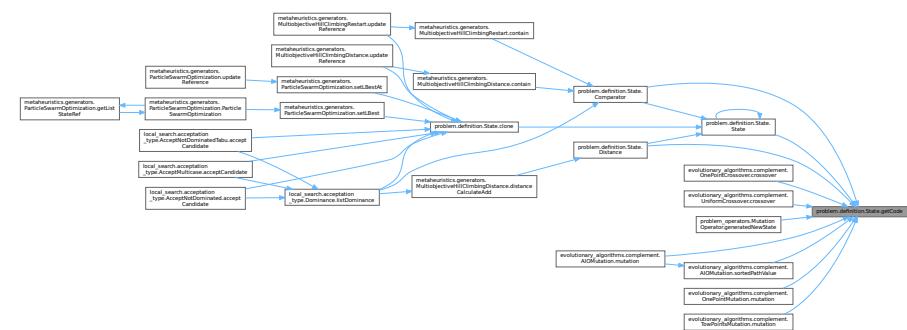
00068             {
00069         return (code == null) ? new ArrayList<Object>() : new ArrayList<Object>(code);
00070     }
```

Hace referencia a [code](#).

Referenciado por [Comparator\(\)](#), [evolutionary\\_algorithms.complement.OnePointCrossover.crossover\(\)](#), [evolutionary\\_algorithms.complement.Distance\(\)](#), [problem\\_operators.MutationOperator.generatedNewState\(\)](#), [evolutionary\\_algorithms.complement.AIOMutation.mutation\(\)](#)

`evolutionary_algorithms.complement.OnePointMutation.mutation()`, `evolutionary_algorithms.complement.TowPointsMutation.mutation()`,  
`evolutionary_algorithms.complement.AIMutation.sortedPathValue()` y `State()`.

Gráfico de llamadas a esta función:



### 8.93.3.5 getCopy()

```
Object problem.definition.State.getCopy () [inline]
```

Obtener una copia por valor del estado (implementado con copy-constructor).

## Devuelve

objeto State nuevo

Definición en la línea 161 del archivo State.java.

```
00161             {
00162         return new State(this);
00163     }
```

Hace referencia a [State\(\)](#).

Referenciado por `evolutionary_algorithms.complement.OnePointCrossover.crossover()`, `evolutionary_algorithms.complement.UniformMetaheuristics.generators.EvolutionStrategies.generate()`, `metaheuristics.generators.GeneticAlgorithm.generate()` y `problem_operators.MutationOperator.generatedNewState()`.

Gráfico de llamadas de esta función:

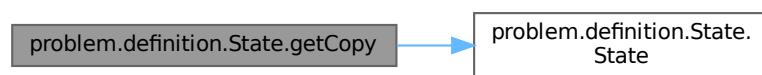
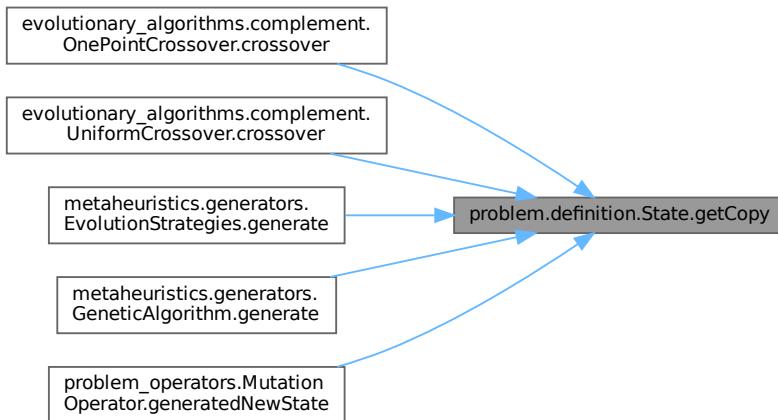


Gráfico de llamadas a esta función:



#### 8.93.3.6 getEvaluation()

```
ArrayList< Double > problem.definition.State.getEvaluation () [inline]
```

Obtener la lista de evaluaciones del estado.

**Devuelve**

copia de la lista de evaluaciones (puede ser null)

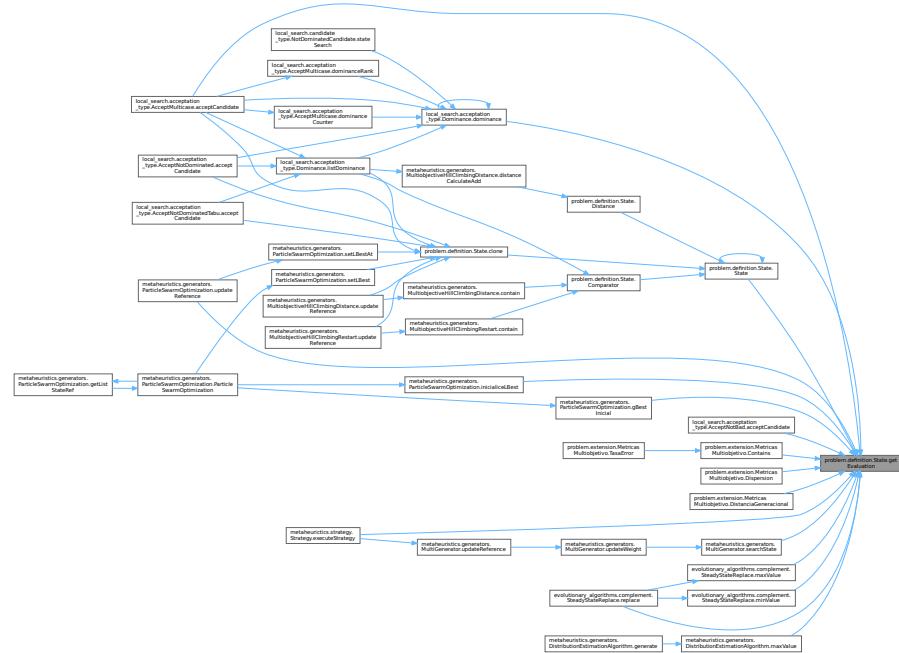
Definición en la línea 104 del archivo [State.java](#).

```
00104
00105     return (evaluation == null) ? null : new ArrayList<Double>(evaluation);
00106 }
```

Hace referencia a [evaluation](#).

Referenciado por [local\\_search.acceptation\\_type.AcceptMulticase.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotBad.accept\(\)](#), [problem.extension.MetricasMultiobjetivo.Contains\(\)](#), [problem.extension.MetricasMultiobjetivo.Dispersion\(\)](#), [problem.extension.MetricasMultiobjetivo.Dominance\(\)](#), [metaheuristics.strategy.Strategy.executeStrategy\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.gBestInitial\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.inicialice\(\)](#), [evolutionary\\_algorithms.complement.SteadyStateReplace maxValue\(\)](#), [metaheuristics.generators.DistributionEstimationAlgorithm.maxValue\(\)](#), [evolutionary\\_algorithms.complement.SteadyStateReplace minValue\(\)](#), [evolutionary\\_algorithms.complement.SteadyStateReplace.replace\(\)](#), [metaheuristics.generators.MultiGenerator.searchState\(\)](#), [State\(\)](#) y [metaheuristics.generators.ParticleSwarmOptimization.updateReferencia\(\)](#).

Gráfico de llamadas a esta función:



### 8.93.3.7 getNumber()

```
int problem.definition.State.getNumber () [inline]
```

Obtener el identificador numérico del estado.

## Devuelve

número asignado al estado

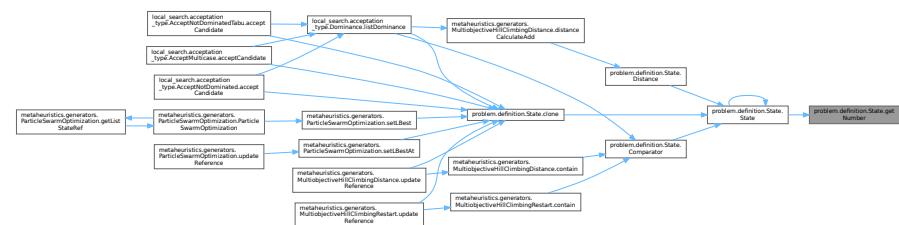
Definición en la línea 122 del archivo State.java.

```
00122  
00123      return number;  
00124  }
```

Hace referencia a [number](#).

Referenciado por [State\(\)](#).

Gráfico de llamadas a esta función:



### 8.93.3.8 getTypeGenerator()

```
GeneratorType problem.definition.State.getTypeGenerator () [inline]
```

Obtener el tipo de generador asociado a este estado.

Devuelve

tipo de generador

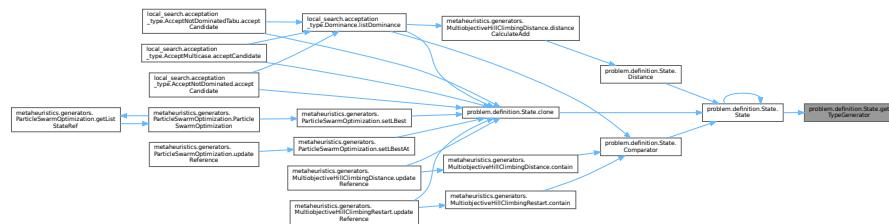
Definición en la línea 86 del archivo [State.java](#).

```
00086
00087     return typeGenerator;
00088 }
```

Hace referencia a [typeGenerator](#).

Referenciado por [State\(\)](#).

Gráfico de llamadas a esta función:



### 8.93.3.9 setCode()

```
void problem.definition.State.setCode (
    ArrayList< Object > listCode) [inline]
```

Establecer la codificación del estado (se copia la lista entrante).

#### Parámetros

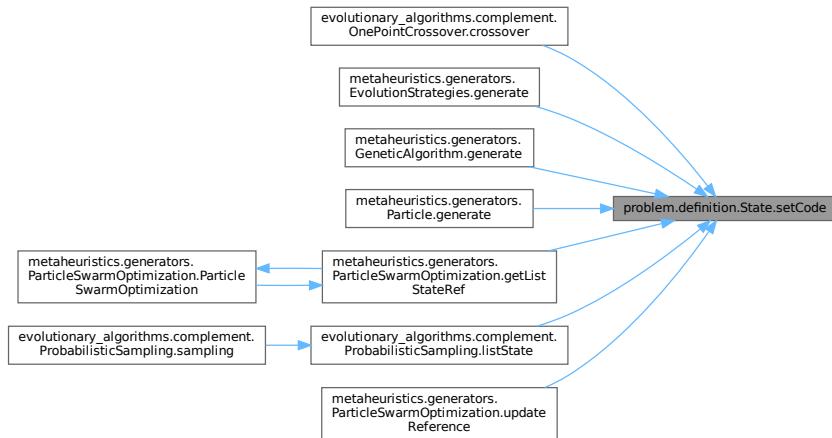
<code>listCode</code>	nueva codificación
-----------------------	--------------------

Definición en la línea 77 del archivo [State.java](#).

```
00077
00078     this.code = (listCode == null) ? new ArrayList<Object>() : new ArrayList<Object>(listCode);
00079 }
```

Referenciado por [evolutionary\\_algorithms.complement.OnePointCrossover.crossover\(\)](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm.generate\(\)](#), [metaheuristics.generators.Particle.generate\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization](#), [evolutionary\\_algorithms.complement.ProbabilisticSampling.listState\(\)](#) y [metaheuristics.generators.ParticleSwarmOptimization.updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.93.3.10 setEvaluation()

```
void problem.definition.State.setEvaluation (
    ArrayList< Double > evaluation) [inline]
```

Establecer la lista de evaluaciones del estado (copia defensiva).

#### Parámetros

<code>evaluation</code>	lista de doubles que representan la evaluación
-------------------------	--

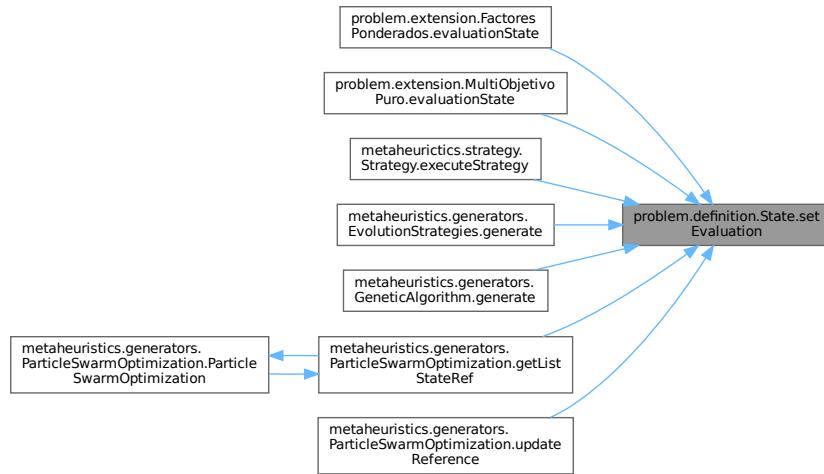
Definición en la línea 113 del archivo [State.java](#).

```
00113
00114     this.evaluation = (evaluation == null) ? null : new ArrayList<Double>(evaluation);
00115 }
```

Hace referencia a [evaluation](#).

Referenciado por [problem.extension.FactoresPonderados.evaluationState\(\)](#), [problem.extension.MultiObjetivoPuro.evaluationState\(\)](#), [metaheuristics.strategy.Strategy.executeStrategy\(\)](#), [metaheuristics.generators.EvolutionStrategies.generate\(\)](#), [metaheuristics.generators.GeneticAlgorithm.generate\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.getListStateRef\(\)](#) y [metaheuristics.generators.ParticleSwarmOptimization.updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.93.3.11 setNumber()

```
void problem.definition.State.setNumber (
    int number) [inline]
```

Asignar el identificador numérico al estado.

#### Parámetros

<code>number</code>	nuevo identificador
---------------------	---------------------

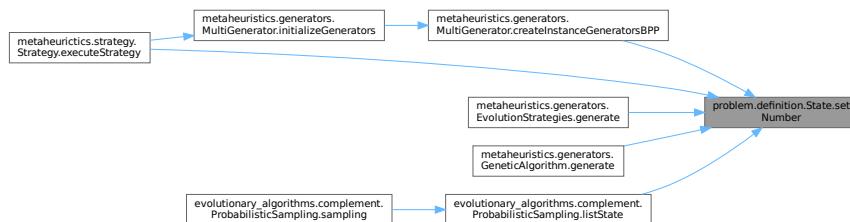
Definición en la línea 130 del archivo [State.java](#).

```
00130
00131     this.number = number;
00132 }
```

Hace referencia a [number](#).

Referenciado por [metaheuristics.generators.MultiGenerator.createInstanceGeneratorsBPP\(\)](#), [metaheuristics.strategy.Strategy.execute\(\)](#), [metaheuristics.generators.EvolutionStrategies.generate\(\)](#), [metaheuristics.generators.GeneticAlgorithm.generate\(\)](#) y [evolutionary\\_algorithms.complement.ProbabilisticSampling.listState\(\)](#).

Gráfico de llamadas a esta función:



### 8.93.3.12 setTypeGenerator()

```
void problem.definition.State.setTypeGenerator (
    GeneratorType typeGenerator) [inline]
```

Establecer el tipo de generador responsable de crear este estado.

#### Parámetros

<code>typeGenerator</code>	tipo de generador
----------------------------	-------------------

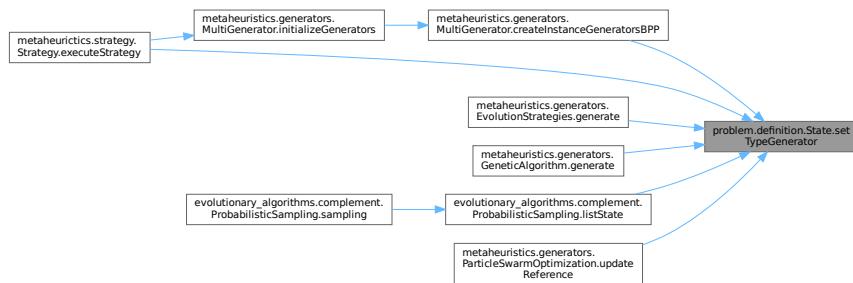
Definición en la línea 94 del archivo [State.java](#).

```
00094
00095     this.typeGenerator = typeGenerator;
00096 }
```

Hace referencia a [typeGenerator](#).

Referenciado por [metaheuristics.generators.MultiGenerator.createInstanceGeneratorsBPP\(\)](#), [metaheuristics.strategy.Strategy.execute\(\)](#), [metaheuristics.generators.EvolutionStrategies.generate\(\)](#), [metaheuristics.generators.GeneticAlgorithm.generate\(\)](#), [evolutionary\\_algorithms.complement.ProbabilisticSampling.listState\(\)](#) y [metaheuristics.generators.ParticleSwarmOptimization.updateReference\(\)](#).

Gráfico de llamadas a esta función:



### 8.93.4 Documentación de datos miembro

#### 8.93.4.1 code

```
ArrayList<Object> problem.definition.State.code [protected]
```

Definición en la línea 21 del archivo [State.java](#).

Referenciado por [getCode\(\)](#), [State\(\)](#) y [State\(\)](#).

#### 8.93.4.2 evaluation

```
ArrayList<Double> problem.definition.State.evaluation [protected]
```

Definición en la línea 19 del archivo [State.java](#).

Referenciado por [getEvaluation\(\)](#) y [setEvaluation\(\)](#).

#### 8.93.4.3 number

```
int problem.definition.State.number [protected]
```

Definición en la línea 20 del archivo [State.java](#).

Referenciado por [getNumber\(\)](#), [setNumber\(\)](#) y [State\(\)](#).

#### 8.93.4.4 typeGenerator

```
GeneratorType problem.definition.State.typeGenerator [protected]
```

Definición en la línea 18 del archivo [State.java](#).

Referenciado por [getTypeGenerator\(\)](#), [setTypeGenerator\(\)](#) y [State\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [State.java](#)

### 8.94 Referencia de la clase

#### **[evolutionary\\_algorithms.complement.SteadyStateReplace](#)**

[SteadyStateReplace](#) - applies steady-state replacement strategy.

Diagrama de herencia de [evolutionary\\_algorithms.complement.SteadyStateReplace](#)

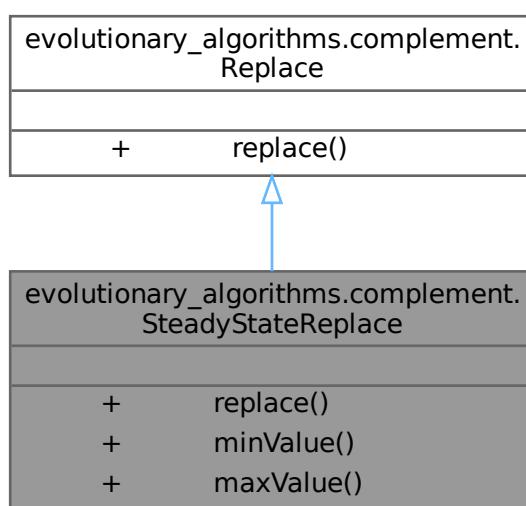
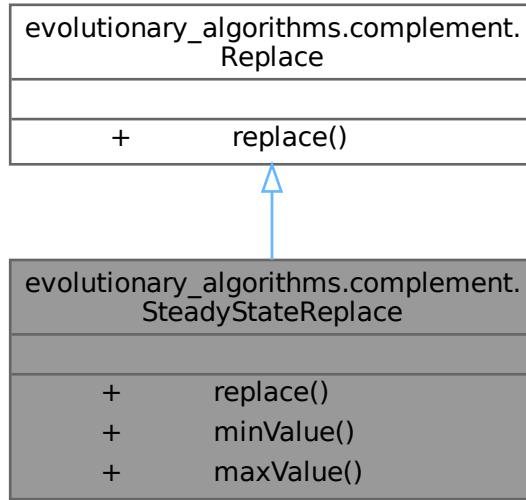


Diagrama de colaboración de evolutionary\_algorithms.complement.SteadyStateReplace:



## Métodos públicos

- List< State > **replace** (State stateCandidate, List< State > listState)  
*replace - applies steady-state replacement strategy.*
- State **minValue** (List< State > listState)  
*minValue - applies steady-state replacement strategy.*
- State **maxValue** (List< State > listState)  
*maxValue - applies steady-state replacement strategy.*

### 8.94.1 Descripción detallada

**SteadyStateReplace** - applies steady-state replacement strategy.

Definición en la línea 13 del archivo [SteadyStateReplace.java](#).

### 8.94.2 Documentación de funciones miembro

#### 8.94.2.1 maxValue()

```
State evolutionary_algorithms.complement.SteadyStateReplace.maxValue (
    List< State > listState) [inline]
```

**maxValue** - applies steady-state replacement strategy.

## Parámetros

*listState*

## Devuelve

returns the state with the maximum evaluation value.

Definición en la línea 81 del archivo [SteadyStateReplace.java](#).

```

00081      State value = listState.get(0);
00082      double max = listState.get(0).getEvaluation().get(0);
00083      for (int i = 1; i < listState.size(); i++) {
00084          if (listState.get(i).getEvaluation().get(0) > max) {
00085              max = listState.get(i).getEvaluation().get(0);
00086              value = listState.get(i);
00087          }
00088      }
00089  }
00090  return value;
00091 }
```

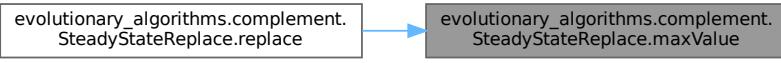
Hace referencia a [problem.definition.State.getEvaluation\(\)](#).

Referenciado por [replace\(\)](#).

Gráfico de llamadas de esta función:



Gráfico de llamadas a esta función:



### 8.94.2.2 minValue()

```
State evolutionary_algorithms.complement.SteadyStateReplace.minValue (
    List< State > listState) [inline]
```

minValue - applies steady-state replacement strategy.

## Parámetros

*listState*

## Devuelve

returns the state with the minimum evaluation value.

Definición en la línea 64 del archivo [SteadyStateReplace.java](#).

```
00064      State value = listState.get(0);
00065      double min = listState.get(0).getEvaluation().get(0);
00066      for (int i = 1; i < listState.size(); i++) {
00067          if (listState.get(i).getEvaluation().get(0) < min) {
00068              min = listState.get(i).getEvaluation().get(0);
00069              value = listState.get(i);
00070          }
00071      }
00072  }
00073 return value;
00074 }
```

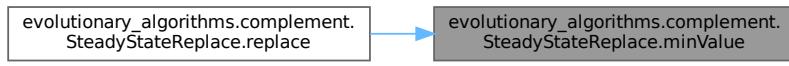
Hace referencia a [problem.definition.State.getEvaluation\(\)](#).

Referenciado por [replace\(\)](#).

Gráfico de llamadas de esta función:



Gráfico de llamadas a esta función:



### 8.94.2.3 replace()

```
List< State > evolutionary_algorithms.complement.SteadyStateReplace.replace (
    State stateCandidate,
    List< State > listState) [inline]
```

replace - applies steady-state replacement strategy.

## Parámetros

<i>stateCandidate</i>	
<i>listState</i>	

## Devuelve

returns the updated list of states after replacement.

Reimplementado de [evolutionary\\_algorithms.complement.Replace](#).

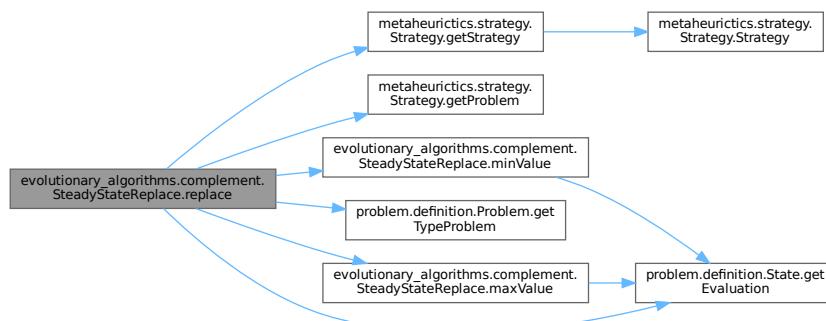
Definición en la línea 22 del archivo [SteadyStateReplace.java](#).

```

00022
00023     State stateREP = null;
00024     if (Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)) {
00025         stateREP = minValue(listState);
00026         if(stateCandidate.getEvaluation().get(0) >= stateREP.getEvaluation().get(0)){
00027             Boolean find = false;
00028             int count = 0;
00029             while ((find.equals(false)) && (listState.size() > count)){
00030                 if(listState.get(count).equals(stateREP)){
00031                     listState.remove(count);
00032                     listState.add(count, stateCandidate);
00033                     find = true;
00034                 }
00035                 else count++;
00036             }
00037         }
00038     } else {
00039
00040         if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Minimizar)){
00041             stateREP = maxValue(listState);
00042             if(stateCandidate.getEvaluation().get(0) <= stateREP.getEvaluation().get(0)){
00043                 Boolean find = false;
00044                 int count = 0;
00045                 while ((find.equals(false)) && (listState.size() > count)){
00046                     if(listState.get(count).equals(stateREP)){
00047                         listState.remove(count);
00048                         listState.add(count, stateCandidate);
00049                         find = true;
00050                     }
00051                     else count++;
00052                 }
00053             }
00054         }
00055     }
00056     return listState;
00057 }
```

Hace referencia a [problem.definition.State.getEvaluation\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Problem.getTypeProblem\(\)](#), [problem.definition.Problem.ProblemType.maxValue\(\)](#), [problem.definition.Problem.ProblemType.Minimizar](#) y [minValue\(\)](#).

Gráfico de llamadas de esta función:



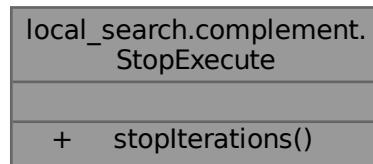
La documentación de esta clase está generada del siguiente archivo:

- [SteadyStateReplace.java](#)

## 8.95 Referencia de la clase local\_search.complement.StopExecute

[StopExecute](#) - termination predicate for local search loops.

Diagrama de colaboración de local\_search.complement.StopExecute:



### Métodos públicos

- Boolean [stopIterations](#) (int countIterationsCurrent, int countmaxIterations)  
*Check whether the maximum number of iterations has been reached.*

### 8.95.1 Descripción detallada

[StopExecute](#) - termination predicate for local search loops.

Utility that provides a simple iterations-based stopping condition for local search algorithms.

Definición en la línea 10 del archivo [StopExecute.java](#).

### 8.95.2 Documentación de funciones miembro

#### 8.95.2.1 stopIterations()

```
Boolean local_search.complement.StopExecute.stopIterations (
    int countIterationsCurrent,
    int countmaxIterations) [inline]
```

Check whether the maximum number of iterations has been reached.

## Parámetros

<code>countIterationsCurrent</code>	current iteration count (0-based or 1-based depending on caller convention)
<code>countmaxIterations</code>	maximum allowed iterations

## Devuelve

true if execution should stop (current  $\geq$  max), false otherwise

Definición en la línea 20 del archivo StopExecute.java.

```
00020
00021      if (countIterationsCurrent < countmaxIterations) {
00022          return false;
00023      } else {
00024          return true;
00025      }
00026  }
```

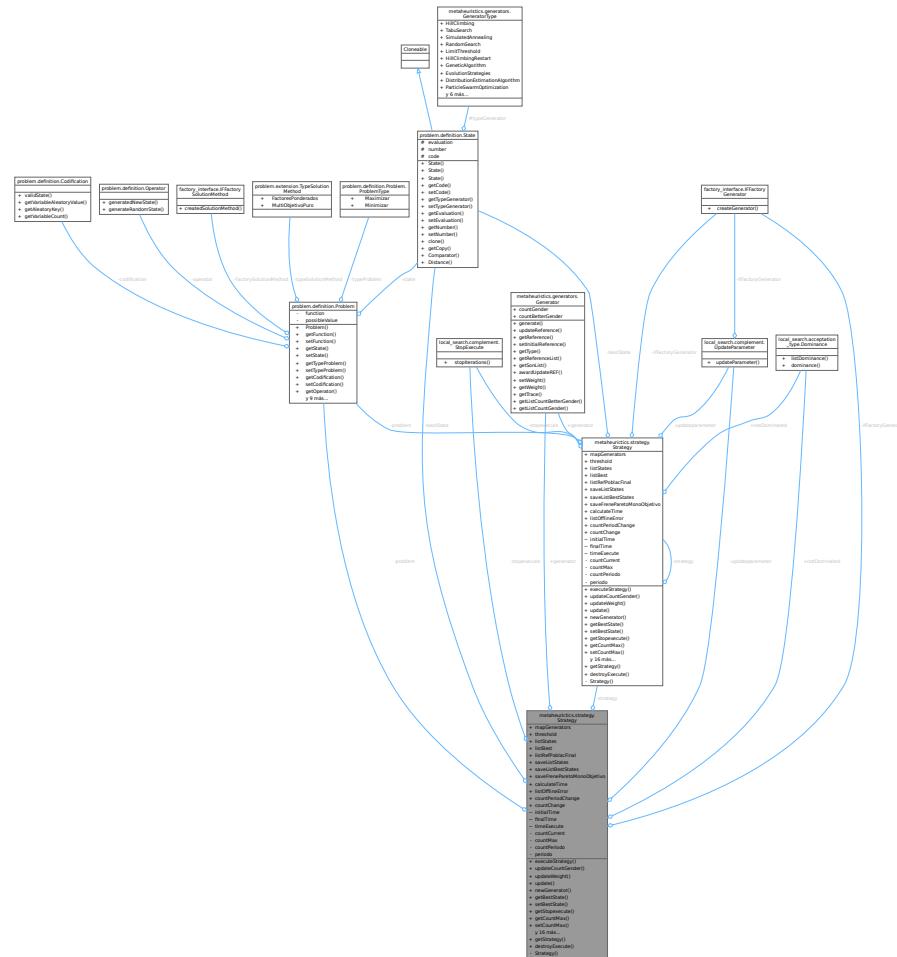
La documentación de esta clase está generada del siguiente archivo:

- StopExecute.java

## 8.96 Referencia de la clase metaheuristics.strategy.Strategy

## Strategy -

Diagrama de colaboración de metaheuristics.strategy.Strategy:



## Métodos públicos

- void **executeStrategy** (int countmaxIterations, int countIterationsChange, int operatornumber, **GeneratorType** generatorType) throws **IllegalArgumentException**, **SecurityException**, **ClassNotFoundException**, **InstantiationException**, **IllegalAccessException**, **InvocationTargetException**, **NoSuchMethodException**  
*executeStrategy - method to execute the strategy.*
- void **updateCountGender** ()
- void **updateWeight** ()  
*updateWeight - method to update the weight of the generators.*
- void **update** (Integer countIterationsCurrent) throws **IllegalArgumentException**, **SecurityException**, **ClassNotFoundException**, **InstantiationException**, **IllegalAccessException**, **InvocationTargetException**, **NoSuchMethodException**
- **Generator newGenerator** (**GeneratorType** Generatortype) throws **IllegalArgumentException**, **SecurityException**, **ClassNotFoundException**, **InstantiationException**, **IllegalAccessException**, **InvocationTargetException**, **NoSuchMethodException**  
*newGenerator - method to create a new generator.*
- **State getBestState** ()  
*getBestState - method to get the best state.*
- void **setBestState** (**State** besState)  
*setBestState - method to set the best state.*
- **StopExecute getStopexecute** ()  
*getStopexecute - method to get the stop execute.*
- int **getCountMax** ()  
*getCountMax - method to get the maximum count.*
- void **setCountMax** (int **countMax**)  
*setCountMax - method to set the maximum count.*
- void **setStopexecute** (**StopExecute** stopexecute)  
*setStopexecute - method to set the stop execute.*
- **UpdateParameter getUpdateparameter** ()  
*getUpdateparameter - method to get the update parameter.*
- void **setUpdateparameter** (**UpdateParameter** updateparameter)  
*setUpdateparameter - method to set the update parameter.*
- **Problem getProblem** ()  
*getProblem - method to get the problem.*
- void **setProblem** (**Problem** problem)  
*setProblem - method to set the problem.*
- **ArrayList< String > getListKey** ()  
*getListKey - method to get the list of keys.*
- void **initializeGenerators** () throws **IllegalArgumentException**, **SecurityException**, **ClassNotFoundException**, **InstantiationException**, **IllegalAccessException**, **InvocationTargetException**, **NoSuchMethodException**  
*initializeGenerators - method to initialize the generators.*
- void **initialize** () throws **IllegalArgumentException**, **SecurityException**, **ClassNotFoundException**, **InstantiationException**, **IllegalAccessException**, **InvocationTargetException**, **NoSuchMethodException**  
*initialize - method to initialize the generators.*
- int **getCountCurrent** ()  
*getCountCurrent - method to get the current count.*
- void **setCountCurrent** (int **countCurrent**)  
*setCountCurrent - method to set the current count.*
- double **getThreshold** ()  
*getThreshold - method to get the threshold.*
- void **setThreshold** (double **threshold**)  
*setThreshold - method to set the threshold.*

- void `calculateOffLinePerformance` (float sumMax, int countOff)  
`calculateOffLinePerformance` - method to calculate the offline performance.
- void `updateRef` (GeneratorType generatorType)  
`updateRef` - method to update the reference.
- void `updateRefMultiG` ()  
`updateRefMultiG` - method to update the reference for multi-generators.
- void `updateRefGenerator` (Generator generator)  
`updateRefGenerator` - method to update the reference for generators.

### Métodos públicos estáticos

- static Strategy `getStrategy` ()  
`getStrategy` - getter of `Strategy` singleton instance
- static void `destroyExecute` ()

### Atributos públicos

- SortedMap< GeneratorType, Generator > `mapGenerators`
- Generator `generator`
- double `threshold`
- List< State > `listStates`
- List< State > `listBest`
- List< State > `listRefPoblacFinal` = new ArrayList<State> ()
- Dominance `notDominated`
- boolean `saveListStates`
- boolean `saveListBestStates`
- boolean `saveFreneParetoMonoObjetivo`
- boolean `calculateTime`
- float[] `listOfflineError` = new float[100]
- int `countPeriodChange` = 0
- int `countChange` = 0

### Métodos privados

- Strategy ()  
`Strategy` - method constructor of `Strategy` singleton.

### Atributos privados

- State `bestState`
- Problem `problem`
- StopExecute `stopexecute`
- UpdateParameter `updateparameter`
- IFFactoryGenerator `ifFactoryGenerator`
- int `countCurrent`
- int `countMax`
- int `countPeriodo`
- int `periodo`

### Atributos estáticos privados

- static [Strategy strategy](#) = null

## 8.96.1 Descripción detallada

[Strategy](#) -.

Definición en la línea [33](#) del archivo [Strategy.java](#).

## 8.96.2 Documentación de constructores y destructores

### 8.96.2.1 Strategy()

```
metaheuristics.strategy.Strategy.Strategy () [inline], [private]
```

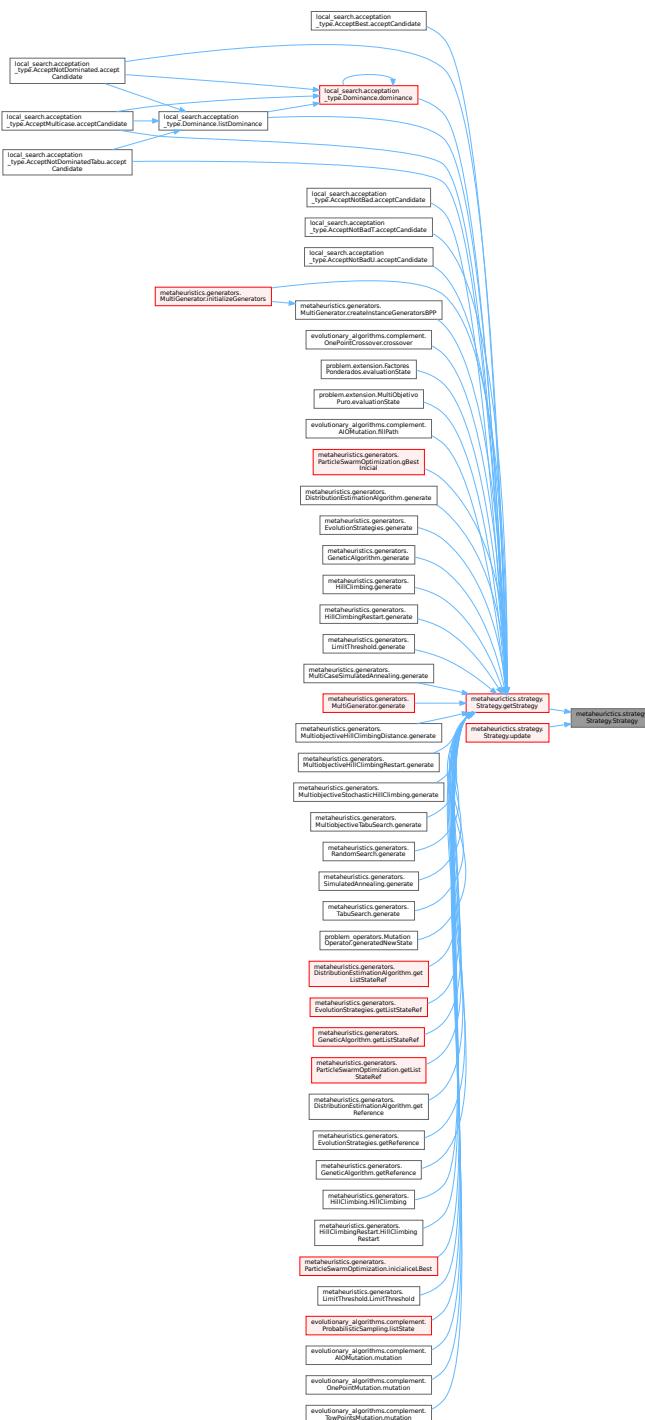
[Strategy](#) - method constructor of [Strategy](#) singleton.

Definición en la línea [73](#) del archivo [Strategy.java](#).

```
00073     {
00074         super ();
00075     }
```

Referenciado por [getStrategy\(\)](#) y [update\(\)](#).

Gráfico de llamadas a esta función:



### 8.96.3 Documentación de funciones miembro

### 8.96.3.1 calculateOffLinePerformance()

```
void metaheuristics.strategy.Strategy.calculateOffLinePerformance (
```

float sumMax,	int countOff)	[inline]
---------------	---------------	----------

calculateOffLinePerformance - method to calculate the offline performance.

#### Parámetros

<i>sumMax</i>	
<i>countOff</i>	

Definición en la línea 509 del archivo [Strategy.java](#).

```
00509
00510     float off = sumMax / countPeriodChange;
00511     listOfflineError[countOff] = off;
00512 }
```

Hace referencia a [countPeriodChange](#) y [listOfflineError](#).

Referenciado por [executeStrategy\(\)](#).

Gráfico de llamadas a esta función:



#### 8.96.3.2 destroyExecute()

```
void metaheuristics.strategy.Strategy.destroyExecute () [inline], [static]
```

Definición en la línea 483 del archivo [Strategy.java](#).

```
00483
00484     strategy = null;
00485     RandomSearch.listStateReference.clear();
00486 }
```

Hace referencia a [metaheuristics.generators.RandomSearch.listStateReference](#) y [strategy](#).

#### 8.96.3.3 executeStrategy()

```
void metaheuristics.strategy.Strategy.executeStrategy (
    int countmaxIterations,
    int countIterationsChange,
    int operatornumber,
    GeneratorType generatorType) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

executeStrategy - method to execute the strategy.

## Parámetros

<i>countmaxIterations</i>	
<i>countIterationsChange</i>	
<i>operatornumber</i>	
<i>generatorType</i>	

run - method to grant privileged permission for reflective access to non-public clone()

Devuelve

Definición en la línea 94 del archivo [Strategy.java](#).

```

00094 {
00095     // si se quiere calcular el tiempo de ejecucion del un algoritmo
00096     if(calculateTime == true){
00097         initialTime = System.currentTimeMillis();
00098     }
00099     this.countMax = countmaxIterations; // max cantidad de iteraciones
00100     //generar estado inicial de la estrategia
00101     Generator randomInitial = new RandomSearch();
00102     State initialState = randomInitial.generate(operatornumber);
00103     problem.Evaluate(initialState); //evaluar ese estado
00104     initialState.setTypeGenerator(generatorType);
00105     getProblem().setState(initialState);
00106     //si se va a salvar la lista de estados generados, adicionar el estado
00107     if(saveListStates == true){
00108         listStates = new ArrayList<State>(); //list de estados generados
00109         listStates.add(initialState);
00110     }
00111     //si se va a salvar la lista de mejores soluciones encontradas en cada iteracion
00112     if(saveListBestStates == true){
00113         listBest = new ArrayList<State>(); // list de mejores estados encontrados
00114         listBest.add(initialState);
00115     }
00116     if(saveFrenteParetoMonoObjetivo == true){
00117         notDominated = new Dominance();
00118     }
00119     // crear el generador a ejecutar
00120     generator = newGenerator(generatorType);
00121     generator.setInitialReference(initialState);
00122     bestState = initialState;
00123     countCurrent = 0;
00124     listRefPoblacFinal = new ArrayList<State>();
00125     MultiGenerator multiGenerator = null;
00126     countPeriodChange = countIterationsChange;
00127     countChange = countIterationsChange;
00128     countPeriodo = countIterationsChange / 10; //cantidad de iteraciones de un periodo
00129     //verificar que es portafolio e inicializar los generadores del portafolio
00130     if(generatorType.equals(GeneratorType.MultiGenerator)){
00131         initializeGenerators();
00132         MultiGenerator.initializeGenerators();
00133         MultiGenerator.listGeneratedPP.clear();
00134         // create a clone of the MultiGenerator using reflection to access protected clone()
00135         try {
00136             final java.lang.reflect.Method cloneMethod =
00137                 generator.getClass().getDeclaredMethod("clone");
00138             // Grant privileged permission for reflective access to non-public clone()
00139             AccessController.doPrivileged(new PrivilegedAction<Void>() {
00140                 public Void run() {
00141                     cloneMethod.setAccessible(true);
00142                     return null;
00143                 }
00144             });
00145             multiGenerator = (MultiGenerator) cloneMethod.invoke(generator);
00146         } catch (Exception e) {
00147             throw new RuntimeException("Failed to clone MultiGenerator", e);
00148         }
00149     }
00150     else initialize(); //crea el mapa de generadores
00151     update(countCurrent);
00152 }
00153 float sumMax = 0; // suma acumulativa para almacenar la evaluacion de la mejor solucion
00154     encontrada y calcular el OfflinePerformance

```

```

00157     int countOff = 0; // variable para contar los OfflinePerformance que se van salvando en el
00158     arreglo
00159     //ciclo de ejecución del algoritmo
00160     while (!stopexecute.stopIterations(countCurrent, countmaxIterations)){
00161         //si se detecta un cambio
00162         if(countCurrent == countChange){
00163             //calcular offlinePerformance
00164             calculateOfflinePerformance(sumMax, countOff);
00165             countOff++;
00166             sumMax = 0;
00167             countIterationsChange = countIterationsChange + countPeriodChange; // actualizar la
00168             cantidad de iteraciones
00169             //actualizar la referencia luego de un cambio
00170             updateRef(generatorType);
00171             countChange = countChange + countPeriodChange;
00172             //generar un nuevo candidato en la iteración, dependiendo del generador
00173             State stateCandidate = null;
00174             if(generatorType.equals(GeneratorType.MultiGenerator)){
00175                 if(countPeriodo == countCurrent){
00176                     updateCountGender();
00177                     countPeriodo = countPeriodo + countPeriodChange / 10;
00178                     periodo = 0;
00179                     MultiGenerator.activeGenerator.countBetterGender = 0;
00180                 }
00181                 updateWeight(); //actualizar el peso de los generadores si se reinician cuando
00182                 ocurre un cambio
00183                 //generar el estado candidato de la iteración
00184                 stateCandidate = multiGenerator.generate(operatornumber);
00185                 problem.Evaluate(stateCandidate);
00186                 stateCandidate.setEvaluation(stateCandidate.getEvaluation());
00187                 stateCandidate.setNumber(countCurrent);
00188                 stateCandidate.setTypeGenerator(generatorType);
00189                 multiGenerator.updateReference(stateCandidate, countCurrent);
00190             }
00191             else {
00192                 stateCandidate = generator.generate(operatornumber);
00193                 problem.Evaluate(stateCandidate);
00194                 stateCandidate.setEvaluation(stateCandidate.getEvaluation());
00195                 stateCandidate.setNumber(countCurrent);
00196                 stateCandidate.setTypeGenerator(generatorType);
00197                 generator.updateReference(stateCandidate, countCurrent);
00198                 if(saveListStates == true){
00199                     listStates.add(stateCandidate);
00200                 }
00201             }
00202             //actualizar el mejor estado encontrado solo tiene sentido para algoritmos
00203             monoobjetivos
00204             //actualizar el mejor estado encontrado solo tiene sentido para algoritmos
00205             monoobjetivos
00206             if ((getProblem().getTypeProblem().equals(ProblemType.Maximizar)) &&
00207                 bestState.getEvaluation().get(bestState.getEvaluation().size() - 1) <
00208                 stateCandidate.getEvaluation().get(bestState.getEvaluation().size() - 1)) {
00209                 bestState = stateCandidate;
00210             }
00211             if ((problem.getTypeProblem().equals(ProblemType.Minimizar)) &&
00212                 bestState.getEvaluation().get(bestState.getEvaluation().size() - 1) >
00213                 stateCandidate.getEvaluation().get(bestState.getEvaluation().size() - 1)) {
00214                 bestState = stateCandidate;
00215             }
00216             // no ha ocurrido un cambio
00217             else {
00218                 State stateCandidate = null;
00219                 if(generatorType.equals(GeneratorType.MultiGenerator)){
00220                     if(countPeriodo == countCurrent){
00221                         updateCountGender();
00222                         countPeriodo = countPeriodo + countPeriodChange / 10;
00223                         periodo++;
00224                         MultiGenerator.activeGenerator.countBetterGender = 0;
00225                     }
00226                     stateCandidate = multiGenerator.generate(operatornumber);
00227                     problem.Evaluate(stateCandidate);
00228                     stateCandidate.setEvaluation(stateCandidate.getEvaluation());
00229                     stateCandidate.setNumber(countCurrent);
00230                     stateCandidate.setTypeGenerator(generatorType);
00231                     multiGenerator.updateReference(stateCandidate, countCurrent);
00232                 }
00233             }
00234             //generar estado candidato y evaluar si es aceptado o no

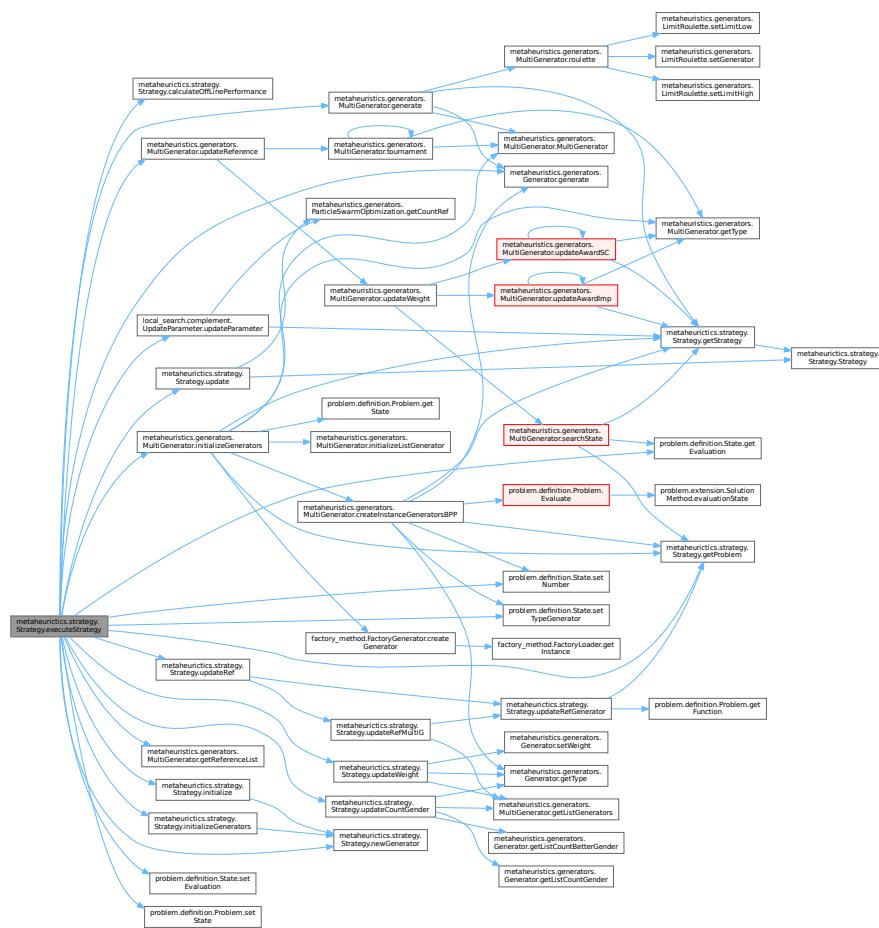
```

```

00235         stateCandidate = generator.generate(operatorNumber);
00236         problem.evaluate(stateCandidate);
00237         stateCandidate.setEvaluation(stateCandidate.getEvaluation());
00238         stateCandidate.setNumber(countCurrent);
00239         stateCandidate.setTypeGenerator(generatorType);
00240         generator.updateReference(stateCandidate, countCurrent); // actualizar la
referencia del estado
00241             if(saveListStates == true){
00242                 listStates.add(stateCandidate);
00243             }
00244             if(saveFreneParetoMonoObjetivo == true){
00245                 notDominated.listDominance(stateCandidate, listRefPoblacFinal);
00246             }
00247         }
00248         countCurrent = UpdateParameter.updateParameter(countCurrent);
00249         //actualizar el mejor estado encontrado solo tiene sentido para algoritmos
monoobjetivos
00250             if ((getProblem().getTypeProblem().equals(ProblemType.Maximizar)) &&
bestState.getEvaluation().get(bestState.getEvaluation().size() - 1) <
stateCandidate.getEvaluation().get(bestState.getEvaluation().size() - 1)) {
00251                 bestState = stateCandidate;
00252             }
00253             if ((problem.getTypeProblem().equals(ProblemType.Minimizar)) &&
bestState.getEvaluation().get(bestState.getEvaluation().size() - 1) >
stateCandidate.getEvaluation().get(bestState.getEvaluation().size() - 1)) {
00254                 bestState = stateCandidate;
00255             }
00256             System.out.println("Evaluacion: " + bestState.getEvaluation());
00257             if(saveListBestStates == true){
00258                 listBest.add(bestState);
00259             }
00260             sumMax = (float) (sumMax + bestState.getEvaluation().get(0));
00261         }
00262     //     System.out.println("Iteracion: " + countCurrent);
00263     }
00264     //calcular tiempo final
00265     if(calculateTime == true){
00266         finalTime = System.currentTimeMillis();
00267         timeExecute = finalTime - initialTime;
00268         System.out.println("El tiempo de ejecucion: " + timeExecute);
00269     }
00270     if(generatorType.equals(GeneratorType.MultiGenerator)){
00271         listBest = new ArrayList<State>(multiGenerator.getReferenceList());
00272         //calcular offlinePerformance
00273         calculateOfflinePerformance(sumMax, countOff);
00274         if(countPeriodo == countCurrent){
00275             updateCountGender();
00276         }
00277     }
00278     else{
00279         listBest = new ArrayList<State>(generator.getReferenceList());
00280         calculateOffLinePerformance(sumMax, countOff);
00281     }
00282 }
```

Hace referencia a `bestState`, `calculateOffLinePerformance()`, `calculateTime`, `countChange`, `countCurrent`, `countPeriodChange`, `countPeriodo`, `metaheuristics.generators.Generator.generate()`, `metaheuristics.generators.MultiGenerator.generator`, `problem.definition.State.getEvaluation()`, `getProblem()`, `metaheuristics.generators.MultiGenerator.getReferenceList()`, `initialize()`, `initializeGenerators()`, `metaheuristics.generators.MultiGenerator.initializeGenerators()`, `listBest`, `metaheuristics.generators.MultiGenerator.listGeneratedPP`, `listRefPoblacFinal`, `listStates`, `problem.definition.Problem.ProblemType.Maximizar`, `problem.definition.Problem.ProblemType.Minimizar`, `metaheuristics.generators.GeneratorType.MultiGenerator`, `newGenerator()`, `notDominated`, `periodo`, `problem`, `saveFreneParetoMonoObjetivo`, `saveListBestStates`, `saveListStates`, `problem.definition.State.setEvaluation()`, `problem.definition.State.setNumber()`, `problem.definition.Problem.setState()`, `problem.definition.State.setTypeGenerator()`, `stopexecute`, `update()`, `updateCountGender()`, `local_search.complement.UpdateParamete`, `updateRef()`, `metaheuristics.generators.MultiGenerator.updateReference()` y `updateWeight()`.

Gráfico de llamadas de esta función:



#### 8.96.3.4 getBestState()

`State metaheuristics.strategy.Strategy.getBestState () [inline]`

`getBestState` - method to get the best state.

## Devuelve

return the best state

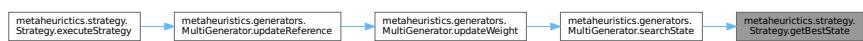
Definición en la línea 342 del archivo `Strategy.java`.

```
00342
00343      return bestState;
00344  }
```

Hace referencia a [bestState](#).

Referenciado por [metaheuristics.generators.MultiGenerator.searchState\(\)](#).

Gráfico de llamadas a esta función:



### 8.96.3.5 getCountCurrent()

```
int metaheuristics.strategy.Strategy.getCountCurrent () [inline]
```

getCountCurrent - method to get the current count.

Devuelve

return the current count

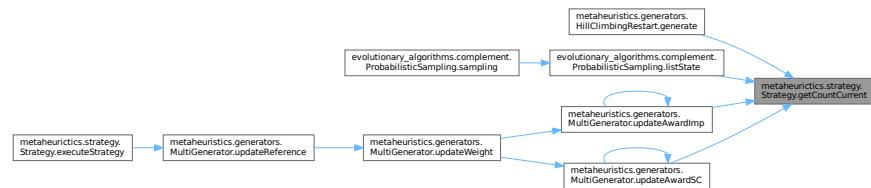
Definición en la línea 471 del archivo [Strategy.java](#).

```
00471 {  
00472     return countCurrent;  
00473 }
```

Hace referencia a [countCurrent](#).

Referenciado por [metaheuristics.generators.HillClimbingRestart.generate\(\)](#), [evolutionary\\_algorithms.complement.ProbabilisticSampling.listState\(\)](#), [metaheuristics.generators.MultiGenerator.updateAwardImp\(\)](#) y [metaheuristics.generators.MultiGenerator.updateAwardSC\(\)](#).

Gráfico de llamadas a esta función:



### 8.96.3.6 getCountMax()

```
int metaheuristics.strategy.Strategy.getCountMax () [inline]
```

getCountMax - method to get the maximum count.

Devuelve

return the maximum count

Definición en la línea 365 del archivo [Strategy.java](#).

```
00365 {  
00366     return countMax;  
00367 }
```

Hace referencia a [countMax](#).

### 8.96.3.7 getListKey()

```
ArrayList< String > metaheuristics.strategy.Strategy.getListKey () [inline]
```

getListKey - method to get the list of keys.

**Devuelve**

return the list of keys

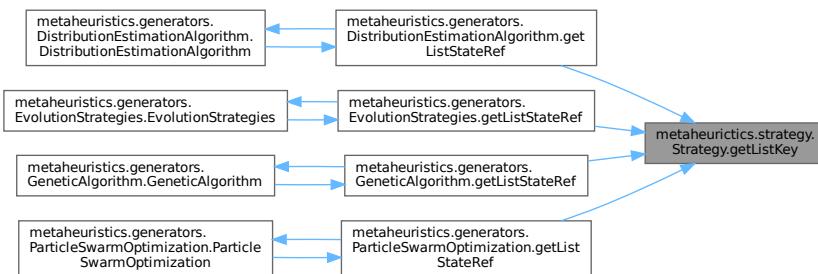
Definición en la línea 421 del archivo [Strategy.java](#).

```
00421      {
00422          ArrayList<String> listKeys = new ArrayList<String>();
00423          String key = mapGenerators.keySet().toString();
00424          String returnString = key.substring(1, key.length() - 1);
00425          returnString += ", ";
00426          int countKey = mapGenerators.size();
00427          for (int i = 0; i < countKey; i++) {
00428              String r = returnString.substring(0, returnString.indexOf(',')); 
00429              returnString = returnString.substring(returnString.indexOf(',') + 2);
00430              listKeys.add(r);
00431          }
00432      }
00433 }
```

Hace referencia a [mapGenerators](#).

Referenciado por [metaheuristics.generators.DistributionEstimationAlgorithm.getListStateRef\(\)](#), [metaheuristics.generators.EvolutionStrategies.getListStateRef\(\)](#), [metaheuristics.generators.GeneticAlgorithm.getListStateRef\(\)](#) y [metaheuristics.generators.ParticleSwarmOptimization.getListStateRef\(\)](#)

Gráfico de llamadas a esta función:



### 8.96.3.8 getProblem()

```
Problem metaheuristics.strategy.Strategy.getProblem () [inline]
```

getProblem - method to get the problem.

Devuelve

return the problem

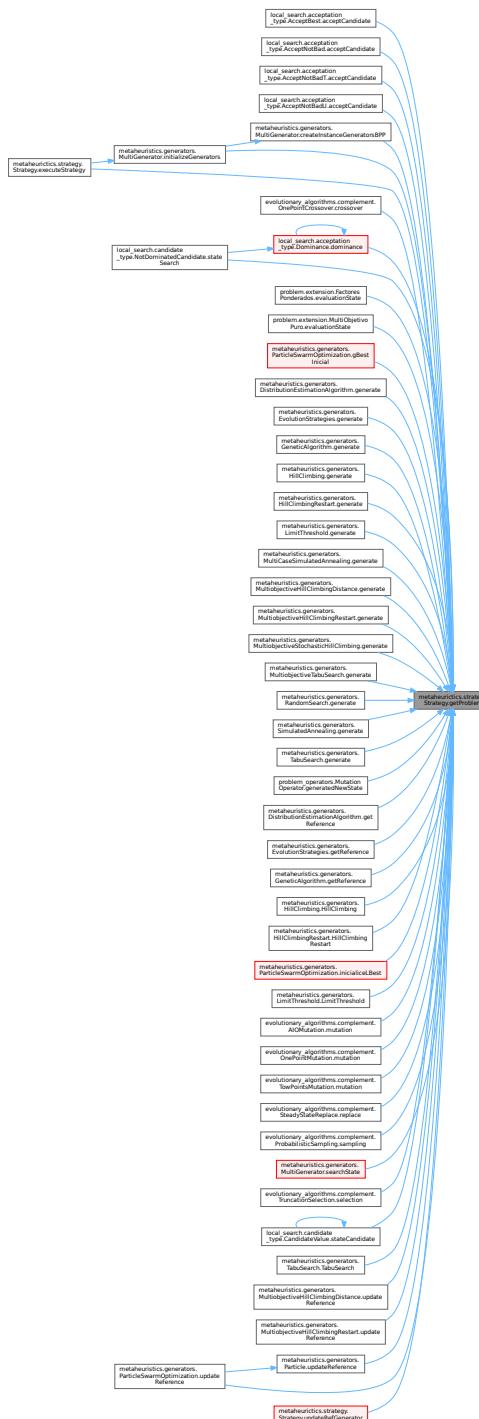
Definición en la línea 405 del archivo [Strategy.java](#).

```
00405          {
00406      return problem;
00407 }
```

Hace referencia a [problem](#).

Referenciado por [local\\_search.acceptation\\_type.AcceptBest.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotBad.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotBadT.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotBadU.acceptCandidate\(\)](#), [metaheuristics.generators.MultiGenerator.createInstanceGeneratorsBPP\(\)](#), [evolutionary\\_algorithms.complement.OnePointCrossover](#), [local\\_search.acceptation\\_type.Dominance.dominance\(\)](#), [problem.extension.FactoresPonderados.evaluationState\(\)](#), [problem.extension.MultiObjetivoPuro.evaluationState\(\)](#), [executeStrategy\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.gBestUpdate\(\)](#), [metaheuristics.generators.DistributionEstimationAlgorithm.generate\(\)](#), [metaheuristics.generators.EvolutionStrategies.generate\(\)](#), [metaheuristics.generators.GeneticAlgorithm.generate\(\)](#), [metaheuristics.generators.HillClimbing.generate\(\)](#), [metaheuristics.generators.HillClimbingRestart.generate\(\)](#), [metaheuristics.generators.LimitThreshold.generate\(\)](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing.generate\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance.generate\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart.generate\(\)](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#), [metaheuristics.generators.MultiobjectiveTabuSearch.generate\(\)](#), [metaheuristics.generators.RandomSearch.generate\(\)](#), [metaheuristics.generators.SimulatedAnnealing.generate\(\)](#), [metaheuristics.generators.TabuSearch.generate\(\)](#), [problem\\_operators.MutationOperator.generatedNewState\(\)](#), [metaheuristics.generators.DistributionEstimationAlgorithm.getReference\(\)](#), [metaheuristics.generators.EvolutionStrategies.getReference\(\)](#), [metaheuristics.generators.GeneticAlgorithm.getReference\(\)](#), [metaheuristics.generators.HillClimbing.HillClimbing\(\)](#), [metaheuristics.generators.HillClimbingRestart.HillClimbingRestart\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.inicialiceLBest\(\)](#), [metaheuristics.generators.MultiGenerator.initializeGenerators\(\)](#), [metaheuristics.generators.LimitThreshold.LimitThreshold\(\)](#), [evolutionary\\_algorithms.complement.AIOMutation.mutation\(\)](#), [evolutionary\\_algorithms.complement.OnePointMutation.mutation\(\)](#), [evolutionary\\_algorithms.complement.TowPointsMutation.mutation\(\)](#), [evolutionary\\_algorithms.complement.SteadyStateReplace.replace\(\)](#), [evolutionary\\_algorithms.complement.ProbabilisticSampling.sample\(\)](#), [metaheuristics.generators.MultiGenerator.searchState\(\)](#), [evolutionary\\_algorithms.complement.TruncationSelection.selection\(\)](#), [local\\_search.candidate\\_type.CandidateValue.stateCandidate\(\)](#), [local\\_search.candidate\\_type.NotDominatedCandidate.stateSearch\(\)](#), [metaheuristics.generators.TabuSearch.TabuSearch\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance.updateReference\(\)](#), [metaheuristics.generators.Particle.updateReference\(\)](#) y [updateRefGenerator\(\)](#).

Gráfico de llamadas a esta función:



### 8.96.3.9 getStopexecute()

`StopExecute` `metaheuristics.strategy.Strategy.getStopexecute () [inline]`

`getStopexecute` - method to get the stop execute.

Devuelve

```
return the stop execute
```

Definición en la línea 357 del archivo [Strategy.java](#).

```
00357
00358     return stopexecute;
00359 }
```

Hace referencia a [stopexecute](#).

### 8.96.3.10 getStrategy()

[Strategy](#) metaheuristics.strategy.Strategy.getStrategy () [inline], [static]

getStrategy - getter of [Strategy](#) singleton instance

Devuelve

```
Strategy instance
```

Definición en la línea 80 del archivo [Strategy.java](#).

```
00080
00081     if (strategy == null) {
00082         strategy = new Strategy();
00083     }
00084     return strategy;
00085 }
```

Hace referencia a [Strategy\(\)](#) y [strategy](#).

Referenciado por [local\\_search.acceptation\\_type.AcceptBest.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptMulticase.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotBad.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotBadT.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotBadU.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotDominated.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotDominatedTabu.acceptCandidate\(\)](#), [metaheuristics.generators.MultiGenerator.createInstance\(\)](#), [evolutionary\\_algorithms.complement.OnePointCrossover.crossover\(\)](#), [local\\_search.acceptation\\_type.Dominance.dominance\(\)](#), [problem.extension.FactoresPonderados.evaluationState\(\)](#), [problem.extension.MultiObjetivoPuro.evaluationState\(\)](#), [evolutionary\\_algorithms.complement.AIMutation.fillPath\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.gBestInicial\(\)](#), [metaheuristics.generators.DistributionEstimationAlgorithm.generate\(\)](#), [metaheuristics.generators.EvolutionStrategies.generate\(\)](#), [metaheuristics.generators.GeneticAlgorithm.generate\(\)](#), [metaheuristics.generators.HillClimbing.generate\(\)](#), [metaheuristics.generators.HillClimbingRestart.generate\(\)](#), [metaheuristics.generators.LimitThreshold.generate\(\)](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing.generate\(\)](#), [metaheuristics.generators.MultiGenerator.generate\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance.generate\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart\(\)](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing.generate\(\)](#), [metaheuristics.generators.MultiobjectiveTabuSearch.generate\(\)](#), [metaheuristics.generators.RandomSearch.generate\(\)](#), [metaheuristics.generators.SimulatedAnnealing.generate\(\)](#), [metaheuristics.generators.TabuSearch.generate\(\)](#), [problem\\_operators.MutationOperator.generatedNewState\(\)](#), [metaheuristics.generators.DistributionEstimationAlgorithm.getListStateRef\(\)](#), [metaheuristics.generators.EvolutionStrategies.getListStateRef\(\)](#), [metaheuristics.generators.GeneticAlgorithm.getListStateRef\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.getListStateRef\(\)](#), [metaheuristics.generators.DistributionEstimationAlgorithm.getReference\(\)](#), [metaheuristics.generators.EvolutionStrategies.getReference\(\)](#), [metaheuristics.generators.GeneticAlgorithm.getReference\(\)](#), [metaheuristics.generators.HillClimbing.HillClimbing\(\)](#), [metaheuristics.generators.HillClimbingRestart.HillClimbingRestart\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.inicialice\(\)](#), [metaheuristics.generators.MultiGenerator.initializeGenerators\(\)](#), [metaheuristics.generators.LimitThreshold.LimitThreshold\(\)](#), [local\\_search.acceptation\\_type.Dominance.listDominance\(\)](#), [evolutionary\\_algorithms.complement.ProbabilisticSampling.listState\(\)](#), [evolutionary\\_algorithms.complement.AIMutation.mutation\(\)](#), [evolutionary\\_algorithms.complement.OnePointMutation.mutation\(\)](#), [evolutionary\\_algorithms.complement.TowPointsMutation.mutation\(\)](#), [evolutionary\\_algorithms.complement.SteadyStateReplace.replace\(\)](#), [evolutionary\\_algorithms.complement.ProbabilisticSampling.sampling\(\)](#), [metaheuristics.generators.MultiGenerator.searchState\(\)](#), [evolutionary\\_algorithms.complement.TruncationSelection.selection\(\)](#), [local\\_search.candidate\\_type.CandidateValue.stateCandidate\(\)](#), [local\\_search.candidate\\_type.NotDominatedCandidate.stateSearch\(\)](#), [metaheuristics.generators.TabuSearch.TabuSearch\(\)](#),

metaheuristics.generators.MultiGenerator.updateAwardImp(), metaheuristics.generators.MultiGenerator.updateAwardSC(), local\_search.complement.UpdateParameter.updateParameter(), metaheuristics.generators.MultiobjectiveHillClimbingDistance.update(), metaheuristics.generators.MultiobjectiveHillClimbingRestart.updateReference(), metaheuristics.generators.Particle.updateReference(), metaheuristics.generators.ParticleSwarmOptimization.updateReference() y metaheuristics.generators.Particle.UpdateVelocity().

Gráfico de llamadas de esta función:



#### 8.96.3.11 getThreshold()

double metaheuristics.strategy.Strategy.getThreshold () [inline]

getThreshold - method to get the threshold.

Devuelve

return the threshold

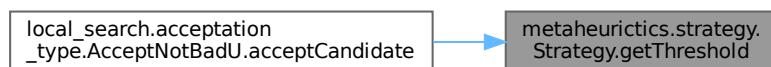
Definición en la línea 492 del archivo [Strategy.java](#).

```
00492           {  
00493         return threshold;  
00494     }
```

Hace referencia a [threshold](#).

Referenciado por [local\\_search.acceptation\\_type.AcceptNotBadU.acceptCandidate\(\)](#).

Gráfico de llamadas a esta función:



#### 8.96.3.12 getUpdateparameter()

UpdateParameter metaheuristics.strategy.Strategy.getUpdateparameter () [inline]

getUpdateparameter - method to get the update parameter.

Devuelve

return the update parameter

Definición en la línea 389 del archivo [Strategy.java](#).

```
00389           {  
00390         return updateparameter;  
00391     }
```

Hace referencia a [updateparameter](#).

### 8.96.3.13 initialize()

```
void metaheuristics.strategy.Strategy.initialize () throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException [inline]
```

initialize - method to initialize the generators.

Definición en la línea 454 del archivo [Strategy.java](#).

```
00454
{
00455     List<GeneratorType> listType = new ArrayList<GeneratorType>();
00456     this.mapGenerators = new TreeMap<GeneratorType, Generator>();
00457     GeneratorType type[] = GeneratorType.values();
00458     for (GeneratorType generator : type) {
00459         listType.add(generator);
00460     }
00461     for (int i = 0; i < listType.size(); i++) {
00462         Generator generator = newGenerator(listType.get(i));
00463         mapGenerators.put(listType.get(i), generator);
00464     }
00465 }
```

Hace referencia a [generator](#), [mapGenerators](#) y [newGenerator\(\)](#).

Referenciado por [executeStrategy\(\)](#).

Gráfico de llamadas de esta función:

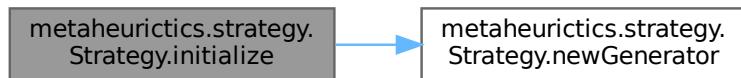


Gráfico de llamadas a esta función:



### 8.96.3.14 initializeGenerators()

```
void metaheuristics.strategy.Strategy.initializeGenerators () throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException [inline]
```

initializeGenerators - method to initialize the generators.

Definición en la línea 438 del archivo [Strategy.java](#).

```
00438
00439     {
00440         List<GeneratorType> listType = new ArrayList<GeneratorType>();
00441         this.mapGenerators = new TreeMap<GeneratorType, Generator>();
00442         GeneratorType type[] = GeneratorType.values();
00443         for (GeneratorType generator : type) {
00444             listType.add(generator);
00445         }
00446         for (int i = 0; i < listType.size(); i++) {
00447             Generator generator = newGenerator(listType.get(i));
00448             mapGenerators.put(listType.get(i), generator);
00449         }
00450     }
```

Hace referencia a [generator](#), [mapGenerators](#) y [newGenerator\(\)](#).

Referenciado por [executeStrategy\(\)](#).

Gráfico de llamadas de esta función:



Gráfico de llamadas a esta función:



### 8.96.3.15 newGenerator()

```
Generator metaheuristics.strategy.Strategy.newGenerator (
    GeneratorType Generatortype) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

`newGenerator` - method to create a new generator.

#### Parámetros

<code>Generatortype</code>	
----------------------------	--

Devuelve

`return a Generator`

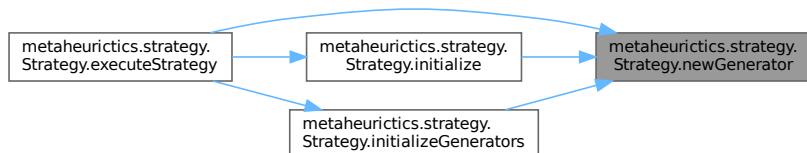
Definición en la línea 332 del archivo [Strategy.java](#).

```
00332
{
00333     ifFactoryGenerator = new FactoryGenerator();
00334     Generator generator = ifFactoryGenerator.createGenerator(Generatortype);
00335     return generator;
00336 }
```

Hace referencia a `generator` y `ifFactoryGenerator`.

Referenciado por [executeStrategy\(\)](#), [initialize\(\)](#) y [initializeGenerators\(\)](#).

Gráfico de llamadas a esta función:



### 8.96.3.16 setBestState()

```
void metaheuristics.strategy.Strategy.setBestState (
    State besState) [inline]
```

`setBestState` - method to set the best state.

#### Parámetros

<code>besState</code>	
-----------------------	--

Definición en la línea 350 del archivo [Strategy.java](#).

```
00350
00351     this.bestState = besState;
00352 }
```

### 8.96.3.17 setCountCurrent()

```
void metaheuristics.strategy.Strategy.setCountCurrent (
    int countCurrent) [inline]
```

`setCountCurrent` - method to set the current count.

**Parámetros**

<i>countCurrent</i>	<input type="button" value=""/>
---------------------	---------------------------------

Definición en la línea 479 del archivo [Strategy.java](#).

```
00479         this.countCurrent = countCurrent;
00480
00481 }
```

Hace referencia a [countCurrent](#).

**8.96.3.18 setCountMax()**

```
void metaheuristics.strategy.Strategy.setCountMax (
    int countMax) [inline]
```

setCountMax - method to set the maximum count.

**Parámetros**

<i>countMax</i>	<input type="button" value=""/>
-----------------	---------------------------------

Definición en la línea 373 del archivo [Strategy.java](#).

```
00373
00374     this.countMax = countMax;
00375 }
```

Hace referencia a [countMax](#).

**8.96.3.19 setProblem()**

```
void metaheuristics.strategy.Strategy.setProblem (
    Problem problem) [inline]
```

setProblem - method to set the problem.

**Parámetros**

<i>problem</i>	<input type="button" value=""/>
----------------	---------------------------------

Definición en la línea 413 del archivo [Strategy.java](#).

```
00413
00414     this.problem = problem;
00415 }
```

Hace referencia a [problem](#).

**8.96.3.20 setStopexecute()**

```
void metaheuristics.strategy.Strategy.setStopexecute (
    StopExecute stopexecute) [inline]
```

setStopexecute - method to set the stop execute.

**Parámetros**

<i>stopexecute</i>	<input type="button" value=""/>
--------------------	---------------------------------

Definición en la línea 381 del archivo [Strategy.java](#).

```
00381
00382     this.stopexecute = stopexecute;
00383 }
```

Hace referencia a [stopexecute](#).

**8.96.3.21 setThreshold()**

```
void metaheuristics.strategy.Strategy.setThreshold (
    double threshold) [inline]
```

setThreshold - method to set the threshold.

**Parámetros**

<i>threshold</i>	<input type="button" value=""/>
------------------	---------------------------------

Definición en la línea 500 del archivo [Strategy.java](#).

```
00500
00501     this.threshold = threshold;
00502 }
```

Hace referencia a [threshold](#).

**8.96.3.22 setUpdateparameter()**

```
void metaheuristics.strategy.Strategy.setUpdateparameter (
    UpdateParameter updateparameter) [inline]
```

setUpdateparameter - method to set the update parameter.

**Parámetros**

<i>updateparameter</i>	<input type="button" value=""/>
------------------------	---------------------------------

Definición en la línea 397 del archivo [Strategy.java](#).

```
00397
00398     this.updateparameter = updateparameter;
00399 }
```

Hace referencia a [updateparameter](#).

### 8.96.3.23 update()

```
void metaheuristics.strategy.Strategy.update (
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

Definición en la línea 307 del archivo [Strategy.java](#).

```
00307     { //HashMap<String, Object> map,
00308         //      Here update parameter for update and change generator.
00309         if(countIterationsCurrent.equals(GeneticAlgorithm.countRef - 1)){
00310             ifFactoryGenerator = new FactoryGenerator();
00311             Strategy.getStrategy().generator =
00312                 ifFactoryGenerator.createGenerator(GeneratorType.GeneticAlgorithm);
00313         }
00314         if(countIterationsCurrent.equals(EvolutionStrategies.countRef - 1)){
00315             ifFactoryGenerator = new FactoryGenerator();
00316             Strategy.getStrategy().generator =
00317                 ifFactoryGenerator.createGenerator(GeneratorType.EvolutionStrategies);
00318         }
00319         if(countIterationsCurrent.equals(DistributionEstimationAlgorithm.countRef - 1)){
00320             ifFactoryGenerator = new FactoryGenerator();
00321             Strategy.getStrategy().generator =
00322                 ifFactoryGenerator.createGenerator(GeneratorType.DistributionEstimationAlgorithm);
00323         }
00324     }
```

Hace referencia a [metaheuristics.generators.DistributionEstimationAlgorithm.countRef](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm.countRef](#), [metaheuristics.generators.GeneratorType.DistributionEstimationAlgorithm](#), [metaheuristics.generators.GeneratorType.EvolutionStrategies](#), [metaheuristics.generators.GeneratorType.GeneticAlgorithm](#), [metaheuristics.generators.ParticleSwarmOptimization.getCountRef\(\)](#), [ifFactoryGenerator](#), [metaheuristics.generators.GeneratorType](#) y [Strategy\(\)](#).

Referenciado por [executeStrategy\(\)](#).

Gráfico de llamadas de esta función:

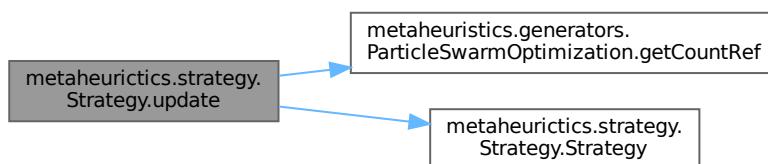
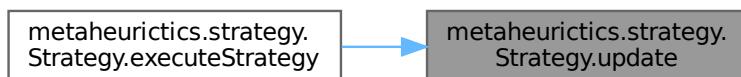


Gráfico de llamadas a esta función:



### 8.96.3.24 updateCountGender()

```
void metaheuristics.strategy.Strategy.updateCountGender () [inline]
```

Definición en la línea 284 del archivo [Strategy.java](#).

```
00284     { // actualizar la cantidad de mejoras y cantidad de veces que se
00285         uso un generador en un periodo dado
00286         for (int i = 0; i < MultiGenerator.getListGenerators().length; i++) {
00287             if (!MultiGenerator.getListGenerators()[i].getType().equals(GeneratorType.MultiGenerator))
00288             {
00289                 MultiGenerator.getListGenerators()[i].getListCountGender() [periodo] =
00290                     MultiGenerator.getListGenerators() [i].countGender +
00291                     MultiGenerator.getListGenerators() [i].getListCountGender() [periodo];
00292                 MultiGenerator.getListGenerators() [i].getListCountBetterGender() [periodo] =
00293                     MultiGenerator.getListGenerators() [i].countBetterGender +
00294                     MultiGenerator.getListGenerators() [i].getListCountBetterGender() [periodo];
00295                 MultiGenerator.getListGenerators() [i].countGender = 0;
00296                 MultiGenerator.getListGenerators() [i].countBetterGender = 0;
00297             }
00298         }
00299     }
```

Hace referencia a [metaheuristics.generators.Generator.countBetterGender](#), [metaheuristics.generators.Generator.countGender](#), [metaheuristics.generators.Generator.getListCountBetterGender\(\)](#), [metaheuristics.generators.Generator.getListCountGender\(\)](#), [metaheuristics.generators.MultiGenerator.getListGenerators\(\)](#), [metaheuristics.generators.Generator.getType\(\)](#), [metaheuristics.generators.GeneratorType.MultiGenerator](#) y [periodo](#).

Referenciado por [executeStrategy\(\)](#).

Gráfico de llamadas de esta función:

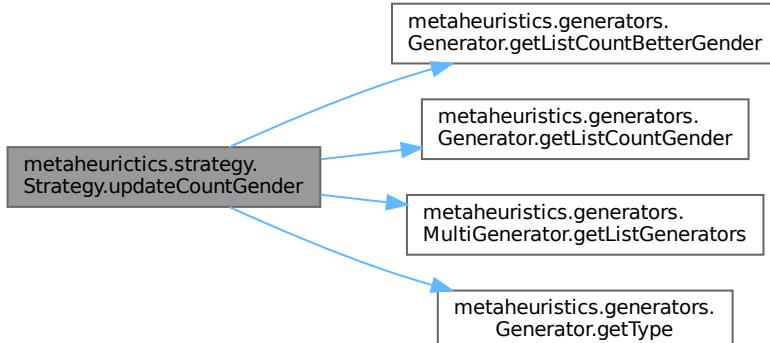


Gráfico de llamadas a esta función:



### 8.96.3.25 updateRef()

```
void metaheuristics.strategy.Strategy.updateRef (
    GeneratorType generatorType) [inline]
```

updateRef - method to update the reference.

#### Parámetros

<i>generatorType</i>	
----------------------	--

Definición en la línea 518 del archivo [Strategy.java](#).

```
00518     {
00519         if(generatorType.equals(GeneratorType.MultiGenerator)){
00520             updateRefMultiG();
00521             bestState = MultiGenerator.listStateReference.get(
00522                 MultiGenerator.listStateReference.size() - 1);
00523         }  
00524         else{
00525             updateRefGenerator(generator);
00526             bestState = generator.getReference();
00527         }
00528     }
```

Hace referencia a **bestState**, **generator**, **metaheuristics.generators.MultiGenerator.listStateReference**, **metaheuristics.generators.GeneratorType**, **updateRefGenerator()** y **updateRefMultiG()**.

Referenciado por [executeStrategy\(\)](#).

Gráfico de llamadas de esta función:

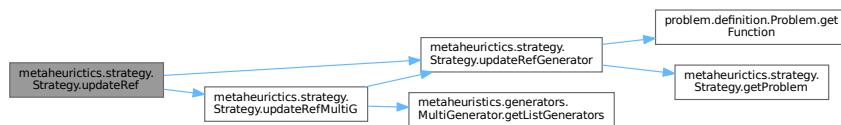


Gráfico de llamadas a esta función:



### 8.96.3.26 updateRefGenerator()

```
void metaheuristics.strategy.Strategy.updateRefGenerator (
    Generator generator) [inline]
```

**updateRefGenerator** - method to update the reference for generators.

## Parámetros

<i>generator</i>	
------------------	--

Definición en la línea 540 del archivo [Strategy.java](#).

```

00540
00541      if(generator.getType().equals(GeneratorType.HillClimbing) ||
00542          generator.getType().equals(GeneratorType.TabuSearch) ||
00543          generator.getType().equals(GeneratorType.RandomSearch) ||
00544          generator.getType().equals(GeneratorType.SimulatedAnnealing)){
00545          double evaluation =
00546          getProblem().getFunction().get(0).Evaluation(generator.getReference());
00547          generator.getReference().getEvaluation().set(0, evaluation);
00548      }
00549
00550  }
00551 }
```

Hace referencia a [metaheuristics.generators.GeneratorType.DistributionEstimationAlgorithm](#), [metaheuristics.generators.GeneratorType](#), [generator](#), [metaheuristics.generators.GeneratorType.GeneticAlgorithm](#), [problem.definition.Problem.getFunction\(\)](#), [getProblem\(\)](#), [metaheuristics.generators.GeneratorType.HillClimbing](#), [metaheuristics.generators.GeneratorType.RandomSearch](#), [metaheuristics.generators.GeneratorType.SimulatedAnnealing](#) y [metaheuristics.generators.GeneratorType.TabuSearch](#).

Referenciado por [updateRef\(\)](#) y [updateRefMultiG\(\)](#).

Gráfico de llamadas de esta función:

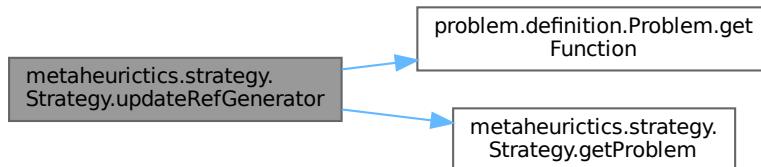
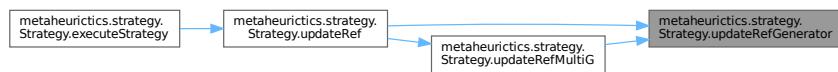


Gráfico de llamadas a esta función:



### 8.96.3.27 updateRefMultiG()

```
void metaheuristics.strategy.Strategy.updateRefMultiG () [inline]
```

updateRefMultiG - method to update the reference for multi-generators.

Definición en la línea 531 del archivo [Strategy.java](#).

```
00531         {
00532             for (int i = 0; i < MultiGenerator.getListGenerators().length; i++) {
00533                 updateRefGenerator(MultiGenerator.getListGenerators()[i]);
00534             }
00535 }
```

Hace referencia a [metaheuristics.generators.MultiGenerator.getListGenerators\(\)](#) y [updateRefGenerator\(\)](#).

Referenciado por [updateRef\(\)](#).

Gráfico de llamadas de esta función:

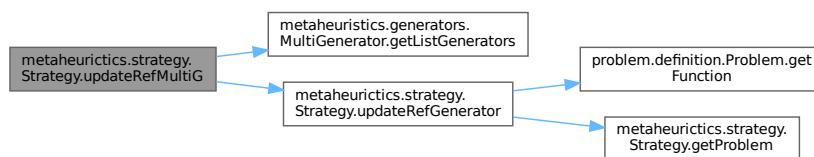
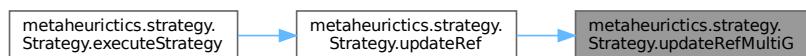


Gráfico de llamadas a esta función:



### 8.96.3.28 updateWeight()

```
void metaheuristics.strategy.Strategy.updateWeight () [inline]
```

updateWeight - method to update the weight of the generators.

Definición en la línea 298 del archivo [Strategy.java](#).

```
00298         {
00299             for (int i = 0; i < MultiGenerator.getListGenerators().length; i++) {
00300                 if (!MultiGenerator.getListGenerators()[i].getType().equals(GeneratorType.MultiGenerator)) {
00301                     MultiGenerator.getListGenerators()[i].setWeight((float) 50.0);
00302                 }
00303             }
00304 }
```

Hace referencia a [metaheuristics.generators.MultiGenerator.getListGenerators\(\)](#), [metaheuristics.generators.Generator.getType\(\)](#), [metaheuristics.generators.GeneratorType.MultiGenerator](#) y [metaheuristics.generators.Generator.setWeight\(\)](#).

Referenciado por [executeStrategy\(\)](#).

Gráfico de llamadas de esta función:

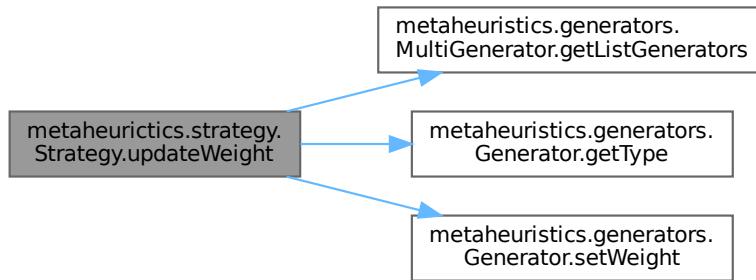
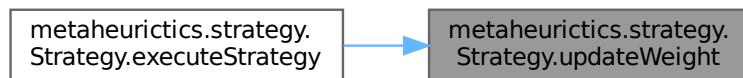


Gráfico de llamadas a esta función:



## 8.96.4 Documentación de datos miembro

### 8.96.4.1 bestState

```
State metaheuristics.strategy.Strategy.bestState [private]
```

Definición en la línea 36 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#), [getBestState\(\)](#) y [updateRef\(\)](#).

### 8.96.4.2 calculateTime

```
boolean metaheuristics.strategy.Strategy.calculateTime
```

Definición en la línea 56 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#).

#### 8.96.4.3 countChange

```
int metaheuristics.strategy.Strategy.countChange = 0
```

Definición en la línea 65 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#).

#### 8.96.4.4 countCurrent

```
int metaheuristics.strategy.Strategy.countCurrent [private]
```

Definición en la línea 42 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#), [getCountCurrent\(\)](#) y [setCountCurrent\(\)](#).

#### 8.96.4.5 countMax

```
int metaheuristics.strategy.Strategy.countMax [private]
```

Definición en la línea 43 del archivo [Strategy.java](#).

Referenciado por [getCountMax\(\)](#) y [setCountMax\(\)](#).

#### 8.96.4.6 countPeriodChange

```
int metaheuristics.strategy.Strategy.countPeriodChange = 0
```

Definición en la línea 64 del archivo [Strategy.java](#).

Referenciado por [calculateOffLinePerformance\(\)](#) y [executeStrategy\(\)](#).

#### 8.96.4.7 countPeriodo

```
int metaheuristics.strategy.Strategy.countPeriodo [private]
```

Definición en la línea 66 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#).

#### 8.96.4.8 generator

```
Generator metaheuristics.strategy.Strategy.generator
```

Definición en la línea 44 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#), [metaheuristics.generators.MultiGenerator.generate\(\)](#), [initialize\(\)](#), [initializeGenerators\(\)](#), [local\\_search.acceptation\\_type.Dominance.listDominance\(\)](#), [newGenerator\(\)](#), [local\\_search.complement.UpdateParameter.updateParaupdateRef\(\)](#) y [updateRefGenerator\(\)](#).

#### 8.96.4.9 ifFactoryGenerator

```
IFFactoryGenerator metaheuristics.strategy.Strategy.ifFactoryGenerator [private]
```

Definición en la línea 41 del archivo [Strategy.java](#).

Referenciado por [newGenerator\(\)](#) y [update\(\)](#).

#### 8.96.4.10 listBest

```
List<State> metaheuristics.strategy.Strategy.listBest
```

Definición en la línea 48 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#).

#### 8.96.4.11 listOfflineError

```
float [ ] metaheuristics.strategy.Strategy.listOfflineError = new float[100]
```

Definición en la línea 63 del archivo [Strategy.java](#).

Referenciado por [calculateOffLinePerformance\(\)](#).

#### 8.96.4.12 listRefPoblacFinal

```
List<State> metaheuristics.strategy.Strategy.listRefPoblacFinal = new ArrayList<State> ()
```

Definición en la línea 49 del archivo [Strategy.java](#).

Referenciado por [local\\_search.acceptation\\_type.AcceptMulticase.acceptCandidate\(\)](#), [local\\_search.acceptation\\_type.AcceptNotDominatedTabu.acceptCandidate\(\)](#), [executeStrategy\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingRestart.updateReference\(\)](#) y [metaheuristics.generators.MultiobjectiveHillClimbingRestart.updateReference\(\)](#).

#### 8.96.4.13 listStates

```
List<State> metaheuristics.strategy.Strategy.listStates
```

Definición en la línea 47 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#) y [metaheuristics.generators.MultiGenerator.initializeGenerators\(\)](#).

#### 8.96.4.14 mapGenerators

```
SortedMap<GeneratorType, Generator> metaheuristics.strategy.Strategy.mapGenerators
```

Definición en la línea 38 del archivo [Strategy.java](#).

Referenciado por [getListKey\(\)](#), [metaheuristics.generators.DistributionEstimationAlgorithm.getListStateRef\(\)](#), [metaheuristics.generators.EvolutionStrategies.getListStateRef\(\)](#), [metaheuristics.generators.GeneticAlgorithm.getListStateRef\(\)](#), [metaheuristics.generators.ParticleSwarmOptimization.getListStateRef\(\)](#), [initialize\(\)](#) y [initializeGenerators\(\)](#).

#### 8.96.4.15 notDominated

`Dominance` `metaheuristics.strategy.Strategy.notDominated`

Definición en la línea 50 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#).

#### 8.96.4.16 periodo

`int` `metaheuristics.strategy.Strategy.periodo` [private]

Definición en la línea 67 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#) y [updateCountGender\(\)](#).

#### 8.96.4.17 problem

`Problem` `metaheuristics.strategy.Strategy.problem` [private]

Definición en la línea 37 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#), [getProblem\(\)](#) y [setProblem\(\)](#).

#### 8.96.4.18 saveFreneParetoMonoObjetivo

`boolean` `metaheuristics.strategy.Strategy.saveFreneParetoMonoObjetivo`

Definición en la línea 55 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#).

#### 8.96.4.19 saveListBestStates

`boolean` `metaheuristics.strategy.Strategy.saveListBestStates`

Definición en la línea 54 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#).

#### 8.96.4.20 saveListStates

`boolean` `metaheuristics.strategy.Strategy.saveListStates`

Definición en la línea 53 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#).

#### 8.96.4.21 stopexecute

`StopExecute` metaheuristics.strategy.Strategy.stopexecute [private]

Definición en la línea 39 del archivo [Strategy.java](#).

Referenciado por [executeStrategy\(\)](#), [getStopexecute\(\)](#) y [setStopexecute\(\)](#).

#### 8.96.4.22 strategy

`Strategy` metaheuristics.strategy.Strategy.strategy = null [static], [private]

Definición en la línea 35 del archivo [Strategy.java](#).

Referenciado por [destroyExecute\(\)](#) y [getStrategy\(\)](#).

#### 8.96.4.23 threshold

`double` metaheuristics.strategy.Strategy.threshold

Definición en la línea 45 del archivo [Strategy.java](#).

Referenciado por [getThreshold\(\)](#) y [setThreshold\(\)](#).

#### 8.96.4.24 updateparameter

`UpdateParameter` metaheuristics.strategy.Strategy.updateparameter [private]

Definición en la línea 40 del archivo [Strategy.java](#).

Referenciado por [getUpdateparameter\(\)](#) y [setUpdateparameter\(\)](#).

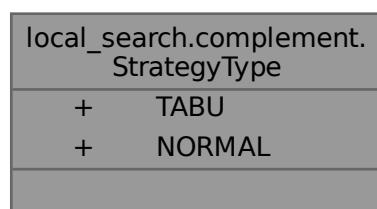
La documentación de esta clase está generada del siguiente archivo:

- [Strategy.java](#)

### 8.97 Referencia de la enumeración `local_search.complement.StrategyType`

`StrategyType` - enumeration of local search strategy modes.

Diagrama de colaboración de `local_search.complement.StrategyType`:



## Atributos públicos

- [TABU](#)  
*Use tabu-list based neighborhood filtering and behavior.*
- [NORMAL](#)  
*Use normal (non-tabu) neighborhood processing.*

### 8.97.1 Descripción detallada

[StrategyType](#) - enumeration of local search strategy modes.

Represents the two supported local search strategy types: a TABU variant which uses tabu lists and a NORMAL (non-tabu) variant.

Definición en la línea 13 del archivo [StrategyType.java](#).

### 8.97.2 Documentación de datos miembro

#### 8.97.2.1 NORMAL

`local_search.complement.StrATEGYType.NORMAL`

Use normal (non-tabu) neighborhood processing.

Definición en la línea 18 del archivo [StrategyType.java](#).

Referenciado por [metaheuristics.generators.HillClimbing.HillClimbing\(\)](#), [metaheuristics.generators.HillClimbingRestart.HillClimbingRestart\(\)](#), [metaheuristics.generators.LimitThreshold.LimitThreshold\(\)](#), [metaheuristics.generators.MultiCaseSimulatedAnnealing.MultiCaseSimulatedAnnealing\(\)](#), [metaheuristics.generators.MultiobjectiveHillClimbingDistance.MultiobjectiveHillClimbingDistance\(\)](#), [metaheuristics.generators.MultiobjectiveStochasticHillClimbing.MultiobjectiveStochasticHillClimbing\(\)](#), [metaheuristics.generators.Ran](#) y [metaheuristics.generators.SimulatedAnnealing.SimulatedAnnealing\(\)](#).

#### 8.97.2.2 TABU

`local_search.complement.StrATEGYType.TABU`

Use tabu-list based neighborhood filtering and behavior.

Definición en la línea 16 del archivo [StrategyType.java](#).

Referenciado por [metaheuristics.generators.MultiobjectiveTabuSearch.MultiobjectiveTabuSearch\(\)](#), [local\\_search.candidate\\_type.CandidateType](#), [metaheuristics.generators.TabuSearch.TabuSearch\(\)](#), [metaheuristics.generators.MultiobjectiveTabuSearch.updateReference\(\)](#) y [metaheuristics.generators.TabuSearch.updateReference\(\)](#).

La documentación de esta enumeración está generada del siguiente archivo:

- [StrategyType.java](#)

## 8.98 Referencia de la clase metaheuristics.generators.TabuSearch

[TabuSearch](#) - class that implements the Tabu Search metaheuristic.

Diagrama de herencia de metaheuristics.generators.TabuSearch

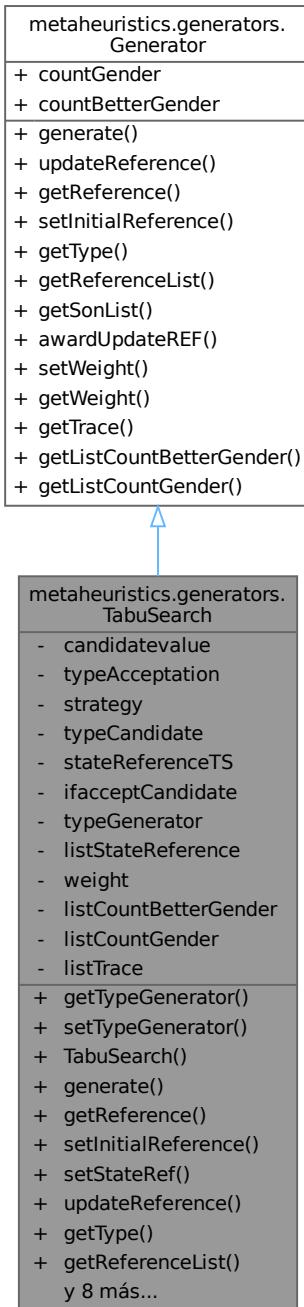
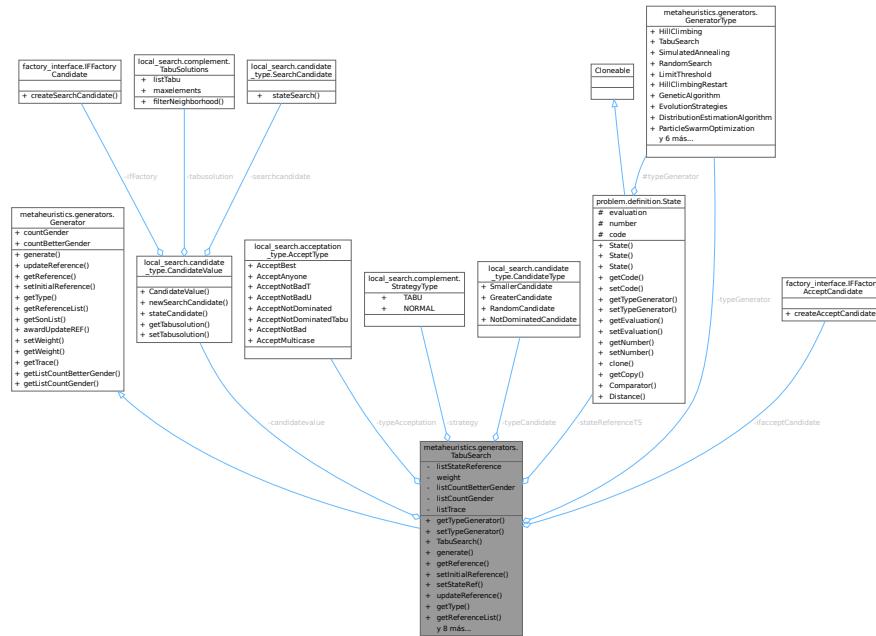


Diagrama de colaboración de metaheuristics.generators.TabuSearch:



## Métodos públicos

- **GeneratorType getTypeGenerator ()**  
*getTypeGenerator - get the type of the generator.*
- **void setTypeGenerator (GeneratorType typeGenerator)**  
*setTypeGenerator - set the type of the generator.*
- **TabuSearch ()**  
*TabuSearch - constructor for the Tabu Search metaheuristic.*
- **State generate (Integer operatornumber)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`  
*generate - method to generate a new state.*
- **State getReference ()**  
*getReference - get the reference state.*
- **void setInitialReference (State statelinitialRef)**  
*setInitialReference - set the initial reference state.*
- **void setStateRef (State stateRef)**  
*setStateRef - set the reference state.*
- **void updateReference (State stateCandidate, Integer countIterationsCurrent)** throws `IllegalArgumentException`, `SecurityException`, `ClassNotFoundException`, `InstantiationException`, `IllegalAccessException`, `InvocationTargetException`, `NoSuchMethodException`  
*updateReference - update the reference state.*
- **GeneratorType getType ()**  
*getType - get the type of the generator.*
- **List< State > getReferenceList ()**  
*getReferenceList - get the list of reference states.*
- **List< State > getSonList ()**  
*getSonList - get the list of child states.*

- void [setTypeCandidate \(CandidateType typeCandidate\)](#)  
*setTypeCandidate - set the type of the candidate.*
- boolean [awardUpdateREF \(State stateCandidate\)](#)  
*awardUpdateREF - award the update of the reference state.*
- float [getWeight \(\)](#)  
*getWeight - get the weight of the generator.*
- void [setWeight \(float weight\)](#)  
*setWeight - set the weight of the generator.*
- int[] [getListCountBetterGender \(\)](#)  
*getListCountBetterGender - get the list of counts for better gender.*
- int[] [getListCountGender \(\)](#)  
*getListCountGender - get the list of counts for gender.*
- float[] [getTrace \(\)](#)  
*getTrace - get the trace of the generator.*

### Atributos privados

- CandidateValue candidatevalue
- AcceptType typeAcceptation
- StrategyType strategy
- CandidateType typeCandidate
- State stateReferenceTS
- IFFactoryAcceptCandidate ifacceptCandidate
- GeneratorType typeGenerator
- List< State > listStateReference = new ArrayList<State>()
- float weight
- int[] listCountBetterGender = new int[10]
- int[] listCountGender = new int[10]
- float[] listTrace = new float[1200000]

### Otros miembros heredados

#### Atributos públicos heredados de [metaheuristics.generators.Generator](#)

- int countGender
- int countBetterGender

### 8.98.1 Descripción detallada

[TabuSearch](#) - class that implements the Tabu Search metaheuristic.

Definición en la línea 28 del archivo [TabuSearch.java](#).

## 8.98.2 Documentación de constructores y destructores

### 8.98.2.1 TabuSearch()

```
metaheuristics.generators.TabuSearch.TabuSearch () [inline]
```

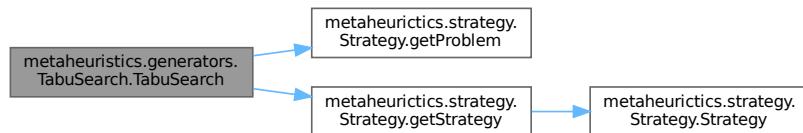
**TabuSearch** - constructor for the Tabu Search metaheuristic.

Definición en la línea 65 del archivo [TabuSearch.java](#).

```
00065      {
00066          super();
00067          this.typeAcceptation = AcceptType.AcceptAnyone;
00068          this.strategy = StrategyType.TABU;
00069
00070
00071          Problem problem = Strategy.getStrategy().getProblem();
00072
00073          if(problem.getTypeProblem().equals(ProblemType.Maximizar)) {
00074              this.typeCandidate = CandidateType.GreaterCandidate;
00075          }
00076          else{
00077              this.typeCandidate = CandidateType.SmallerCandidate;
00078          }
00079
00080          this.candidatevalue = new CandidateValue();
00081          this.typeGenerator = GeneratorType.TabuSearch;
00082          this.weight = 50;
00083          listTrace[0] = this.weight;
00084          listCountBetterGender[0] = 0;
00085          listCountGender[0] = 0;
00086
00087      }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptType.AcceptAnyone](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [local\\_search.candidate\\_type.CandidateType.GreaterCandidate](#), [listCountBetterGender](#), [listCountGender](#), [listTrace](#), [problem.definition.Problem.ProblemType.Maximizar](#), [local\\_search.candidate\\_type.local\\_search.complement.StrategyType.TABU](#) y [metaheuristics.generators.GeneratorType.TabuSearch](#).

Gráfico de llamadas de esta función:



## 8.98.3 Documentación de funciones miembro

### 8.98.3.1 awardUpdateREF()

```
boolean metaheuristics.generators.TabuSearch.awardUpdateREF (
    State stateCandidate) [inline]
```

**awardUpdateREF** - award the update of the reference state.

**Parámetros**

<i>stateCandidate</i>	<input type="button" value=""/>
-----------------------	---------------------------------

**Devuelve**Reimplementado de [metaheuristics.generators.Generator](#).Definición en la línea 213 del archivo [TabuSearch.java](#).

```
00213
00214     // TODO Auto-generated method stub
00215     return false;
00216 }
```

**8.98.3.2 generate()**

```
State metaheuristics.generators.TabuSearch.generate (
    Integer operatornumber) throws IllegalArgumentException, SecurityException, Class←
NotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline]
```

generate - method to generate a new state.

**Parámetros**

<i>operatornumber</i>	<input type="button" value=""/>
-----------------------	---------------------------------

**Devuelve**

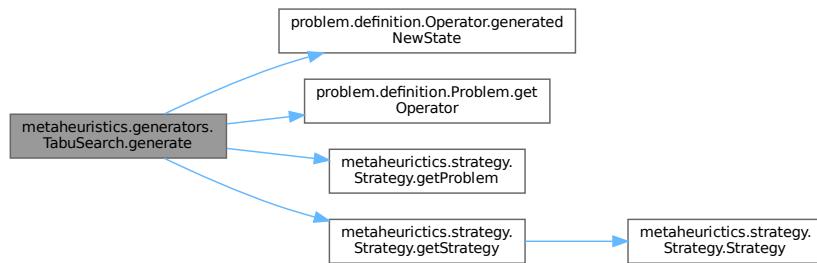
the generated state

Reimplementado de [metaheuristics.generators.Generator](#).Definición en la línea 95 del archivo [TabuSearch.java](#).

```
00095
00096     {
00097         List<State> neighborhood =
00098             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceTS, operatornumber);
00099         State statecandidate = candidatevalue.stateCandidate(stateReferenceTS, typeCandidate,
00100             strategy, operatornumber, neighborhood);
00101         return statecandidate;
00102     }
```

Hace referencia a [candidatevalue](#), [problem.definition.Operator.generatedNewState\(\)](#), [problem.definition.Problem.getOperator\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [stateReferenceTS](#), [strategy](#) y [typeCandidate](#).

Gráfico de llamadas de esta función:



### 8.98.3.3 getListCountBetterGender()

```
int[] metaheuristics.generators.TabuSearch.getListCountBetterGender () [inline]
```

getListCountBetterGender - get the list of counts for better gender.

**Devuelve**

the list of counts for better gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 242 del archivo [TabuSearch.java](#).

```
00242      {
00243          // TODO Auto-generated method stub
00244          return (this.listCountBetterGender == null) ? new int[0] :
00245              Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
```

### 8.98.3.4 getListCountGender()

```
int[] metaheuristics.generators.TabuSearch.getListCountGender () [inline]
```

getListCountGender - get the list of counts for gender.

**Devuelve**

the list of counts for gender

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 252 del archivo [TabuSearch.java](#).

```
00252      {
00253          // TODO Auto-generated method stub
00254          return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00255              this.listCountGender.length);
```

### 8.98.3.5 **getReference()**

```
State metaheuristics.generators.TabuSearch.getReference () [inline]
```

getReference - get the reference state.

Devuelve

the reference state

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 106 del archivo [TabuSearch.java](#).

```
00106         {
00107             return stateReferenceTS;
00108         }
```

Hace referencia a [stateReferenceTS](#).

### 8.98.3.6 **getReferenceList()**

```
List< State > metaheuristics.generators.TabuSearch.getReferenceList () [inline]
```

getReferenceList - get the list of reference states.

Devuelve

the list of reference states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 184 del archivo [TabuSearch.java](#).

```
00184         {
00185             listStateReference.add(stateReferenceTS);
00186             return new ArrayList<State>(listStateReference);
00187         }
```

Hace referencia a [listStateReference](#) y [stateReferenceTS](#).

### 8.98.3.7 **getSonList()**

```
List< State > metaheuristics.generators.TabuSearch.getSonList () [inline]
```

getSonList - get the list of child states.

Devuelve

the list of child states

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 194 del archivo [TabuSearch.java](#).

```
00194         {
00195             // TODO Auto-generated method stub
00196             return null;
00197         }
```

### 8.98.3.8 getTrace()

```
float[ ] metaheuristics.generators.TabuSearch.getTrace () [inline]
```

getTrace - get the trace of the generator.

Devuelve

the trace of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 262 del archivo [TabuSearch.java](#).

```
00262      {
00263          // TODO Auto-generated method stub
00264          return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00265              this.listTrace.length);
}
```

### 8.98.3.9 getType()

```
GeneratorType metaheuristics.generators.TabuSearch.getType () [inline]
```

getType - get the type of the generator.

Devuelve

the type of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 175 del archivo [TabuSearch.java](#).

```
00175      {
00176          return this.typeGenerator;
00177      }
```

### 8.98.3.10 getTypeGenerator()

```
GeneratorType metaheuristics.generators.TabuSearch.getTypeGenerator () [inline]
```

getTypeGenerator - get the type of the generator.

Devuelve

the type of the generator

Definición en la línea 50 del archivo [TabuSearch.java](#).

```
00050      {
00051          return typeGenerator;
00052      }
```

Hace referencia a [typeGenerator](#).

### 8.98.3.11 getWeight()

```
float metaheuristics.generators.TabuSearch.getWeight () [inline]
```

getWeight - get the weight of the generator.

#### Devuelve

the weight of the generator

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 222 del archivo [TabuSearch.java](#).

```
00222                     {
00223             // TODO Auto-generated method stub
00224             return this.weight;
00225         }
```

### 8.98.3.12 setInitialReference()

```
void metaheuristics.generators.TabuSearch.setInitialReference (
    State stateInitialRef) [inline]
```

setInitialReference - set the initial reference state.

#### Parámetros

<code>stateInitialRef</code>	the initial reference state to set
------------------------------	------------------------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 115 del archivo [TabuSearch.java](#).

```
00115                     {
00116             this.stateReferenceTS = stateInitialRef;
00117         }
```

### 8.98.3.13 setStateRef()

```
void metaheuristics.generators.TabuSearch.setStateRef (
    State stateRef) [inline]
```

setStateRef - set the reference state.

#### Parámetros

<code>stateRef</code>	the reference state to set
-----------------------	----------------------------

Definición en la línea 123 del archivo [TabuSearch.java](#).

```
00123                     {
00124             this.stateReferenceTS = stateRef;
00125         }
```

### 8.98.3.14 setTypeCandidate()

```
void metaheuristics.generators.TabuSearch.setTypeCandidate (
```

**Parámetros**

<i>typeCandidate</i>	the type of the candidate to set
----------------------	----------------------------------

Definición en la línea 203 del archivo [TabuSearch.java](#).

```
00203
00204     this.typeCandidate = typeCandidate;
00205 }
```

Hace referencia a [typeCandidate](#).

**8.98.3.15 setTypeGenerator()**

```
void metaheuristics.generators.TabuSearch.setTypeGenerator (
    GeneratorType typeGenerator) [inline]
```

[setTypeGenerator](#) - set the type of the generator.

**Parámetros**

<i>typeGenerator</i>	the type of the generator to set
----------------------	----------------------------------

Definición en la línea 58 del archivo [TabuSearch.java](#).

```
00058
00059     this.typeGenerator = typeGenerator;
00060 }
```

Hace referencia a [typeGenerator](#).

**8.98.3.16 setWeight()**

```
void metaheuristics.generators.TabuSearch.setWeight (
    float weight) [inline]
```

[setWeight](#) - set the weight of the generator.

**Parámetros**

<i>weight</i>	the weight to set
---------------	-------------------

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 232 del archivo [TabuSearch.java](#).

```
00232
00233     // TODO Auto-generated method stub
00234     this.weight = weight;
00235 }
```

Hace referencia a [weight](#).

**8.98.3.17 updateReference()**

---

Generado por Doxygen

```
void metaheuristics.generators.TabuSearch.updateReference (
    State stateCandidate,
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
```

## Parámetros

<i>stateCandidate</i>	
<i>countIterationsCurrent</i>	

Reimplementado de [metaheuristics.generators.Generator](#).

Definición en la línea 133 del archivo [TabuSearch.java](#).

```

00133
00134     ifacceptCandidate = new FactoryAcceptCandidate();
00135     AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00136     Boolean accept = candidate.acceptCandidate(stateReferenceTS, stateCandidate);
00137     if(accept.equals(true))
00138         stateReferenceTS = stateCandidate;
00139
00140     if (strategy.equals(StrategyType.TABU) && accept.equals(true)) {
00141         if (TabuSolutions.listTabu.size() < TabuSolutions.maxelements) {
00142             Boolean find = false;
00143             int count = 0;
00144             while ((TabuSolutions.listTabu.size() > count) && (find.equals(false))) {
00145                 if ((TabuSolutions.listTabu.get(count).Comparator(stateCandidate))==true) {
00146                     find = true;
00147                 }
00148                 count++;
00149             }
00150             if (find.equals(false)) {
00151                 TabuSolutions.listTabu.add(stateCandidate);
00152             }
00153         } else {
00154             TabuSolutions.listTabu.remove(0);
00155             Boolean find = false;
00156             int count = 0;
00157             while (TabuSolutions.listTabu.size() > count && find.equals(false)) {
00158                 if ((TabuSolutions.listTabu.get(count).Comparator(stateCandidate))==true) {
00159                     find = true;
00160                 }
00161                 count++;
00162             }
00163             if (find.equals(false)) {
00164                 TabuSolutions.listTabu.add(stateCandidate);
00165             }
00166         }
00167     }
00168 }
```

Hace referencia a [local\\_search.acceptation\\_type.AcceptableCandidate.acceptCandidate\(\)](#), [ifacceptCandidate](#), [local\\_search.complement.TabuSolutions.listTabu](#), [local\\_search.complement.TabuSolutions.maxelements](#), [stateReferenceTS](#), [strategy](#), [local\\_search.complement.StrategyType.TABU](#) y [typeAcceptation](#).

Gráfico de llamadas de esta función:



## 8.98.4 Documentación de datos miembro

### 8.98.4.1 candidatevalue

`CandidateValue` [metaheuristics.generators.TabuSearch.candidatevalue](#) [private]

Definición en la línea 30 del archivo [TabuSearch.java](#).

Referenciado por [generate\(\)](#).

#### 8.98.4.2 ifacceptCandidate

```
IFFactoryAcceptCandidate metaheuristics.generators.TabuSearch.ifacceptCandidate [private]
```

Definición en la línea 35 del archivo [TabuSearch.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.98.4.3 listCountBetterGender

```
int [] metaheuristics.generators.TabuSearch.listCountBetterGender = new int[10] [private]
```

Definición en la línea 41 del archivo [TabuSearch.java](#).

Referenciado por [TabuSearch\(\)](#).

#### 8.98.4.4 listCountGender

```
int [] metaheuristics.generators.TabuSearch.listCountGender = new int[10] [private]
```

Definición en la línea 42 del archivo [TabuSearch.java](#).

Referenciado por [TabuSearch\(\)](#).

#### 8.98.4.5 listStateReference

```
List<State> metaheuristics.generators.TabuSearch.listStateReference = new ArrayList<State>()  
[private]
```

Definición en la línea 37 del archivo [TabuSearch.java](#).

Referenciado por [getReferenceList\(\)](#).

#### 8.98.4.6 listTrace

```
float [] metaheuristics.generators.TabuSearch.listTrace = new float[1200000] [private]
```

Definición en la línea 43 del archivo [TabuSearch.java](#).

Referenciado por [TabuSearch\(\)](#).

#### 8.98.4.7 stateReferenceTS

```
State metaheuristics.generators.TabuSearch.stateReferenceTS [private]
```

Definición en la línea 34 del archivo [TabuSearch.java](#).

Referenciado por [generate\(\)](#), [getReference\(\)](#), [getReferenceList\(\)](#) y [updateReference\(\)](#).

#### 8.98.4.8 strategy

```
StrategyType metaheuristics.generators.TabuSearch.strategy [private]
```

Definición en la línea 32 del archivo [TabuSearch.java](#).

Referenciado por [generate\(\)](#) y [updateReference\(\)](#).

#### 8.98.4.9 typeAcceptation

```
AcceptType metaheuristics.generators.TabuSearch.typeAcceptation [private]
```

Definición en la línea 31 del archivo [TabuSearch.java](#).

Referenciado por [updateReference\(\)](#).

#### 8.98.4.10 typeCandidate

```
CandidateType metaheuristics.generators.TabuSearch.typeCandidate [private]
```

Definición en la línea 33 del archivo [TabuSearch.java](#).

Referenciado por [generate\(\)](#) y [setTypeCandidate\(\)](#).

#### 8.98.4.11 typeGenerator

```
GeneratorType metaheuristics.generators.TabuSearch.typeGenerator [private]
```

Definición en la línea 36 del archivo [TabuSearch.java](#).

Referenciado por [getTypeGenerator\(\)](#) y [setTypeGenerator\(\)](#).

#### 8.98.4.12 weight

```
float metaheuristics.generators.TabuSearch.weight [private]
```

Definición en la línea 38 del archivo [TabuSearch.java](#).

Referenciado por [setWeight\(\)](#).

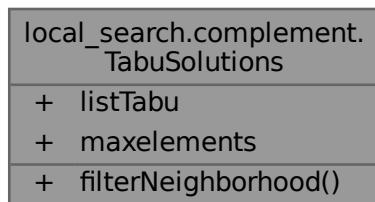
La documentación de esta clase está generada del siguiente archivo:

- [TabuSearch.java](#)

## 8.99 Referencia de la clase local\_search.complement.TabuSolutions

[TabuSolutions](#) - helper that manages a tabu list and filters neighborhoods.

Diagrama de colaboración de local\_search.complement.TabuSolutions:



### Métodos públicos

- List< [State](#) > [filterNeighborhood](#) (List< [State](#) > [listNeighborhood](#)) throws Exception  
*Remove tabu solutions from the supplied neighborhood list.*

### Atributos públicos estáticos

- static final List< [State](#) > [listTabu](#) = Collections.synchronizedList(new ArrayList< [State](#) >())  
*Global synchronized tabu list shared by the local search components.*
- static final int [maxelements](#) = 0  
*Maximum number of elements allowed in the tabu list (unused placeholder).*

### 8.99.1 Descripción detallada

[TabuSolutions](#) - helper that manages a tabu list and filters neighborhoods.

Provides a synchronized global tabu list and a utility to remove tabu entries from candidate neighborhoods. The filtering method will throw an Exception if all neighbors are tabu.

Definición en la línea 16 del archivo [TabuSolutions.java](#).

### 8.99.2 Documentación de funciones miembro

#### 8.99.2.1 [filterNeighborhood\(\)](#)

```
List< State > local_search.complement.TabuSolutions.filterNeighborhood (
    List< State > listNeighborhood) throws Exception [inline]
```

[Remove tabu solutions from the supplied neighborhood list.](#)

Generado por Doxygen

This method iterates the provided neighborhood and removes any element that equals an element in the global [listTabu](#). If after filtering the neighborhood becomes empty the method throws an Exception to signal the caller that no valid neighbors remain.

## Parámetros

<i>listNeighborhood</i>	list of neighbor States to filter
-------------------------	-----------------------------------

Devuelve

a filtered List<State> with tabu entries removed

## Excepciones

<i>Exception</i>	when no neighbors remain after filtering
------------------	--

Definición en la línea 36 del archivo [TabuSolutions.java](#).

```

00036
00037     List<State> listFiltrate;
00038     // If there are entries in the tabu list, remove matching neighbors
00039     if (!listTabu.isEmpty()) {
00040         for (int i = listNeighborhood.size() - 1; i >= 0 ; i--) {
00041             int count_tabu = 0;
00042             while (listTabu.size() > count_tabu) {
00043                 if (listNeighborhood.get(i).equals(listTabu.get(count_tabu))) {
00044                     listNeighborhood.remove(i);
00045                 }
00046                 count_tabu++;
00047             }
00048         }
00049         listFiltrate = listNeighborhood;
00050         if (listFiltrate.isEmpty()) {
00051             throw new Exception();
00052         }
00053     } else {
00054         listFiltrate = listNeighborhood;
00055     }
00056     return listFiltrate;
00057 }
```

Hace referencia a [listTabu](#).

Referenciado por [local\\_search.candidate\\_type.CandidateValue.stateCandidate\(\)](#).

Gráfico de llamadas a esta función:



## 8.99.3 Documentación de datos miembro

### 8.99.3.1 listTabu

```
final List<State> local_search.complement.TabuSolutions.listTabu = Collections.synchronizedList(new ArrayList<State>()) [static]
```

Global synchronized tabu list shared by the local search components.

Definición en la línea 19 del archivo [TabuSolutions.java](#).

Referenciado por [filterNeighborhood\(\)](#), [metaheuristics.generators.MultiobjectiveTabuSearch.updateReference\(\)](#) y [metaheuristics.generators.TabuSearch.updateReference\(\)](#).

### 8.99.3.2 maxelements

```
final int local_search.complement.TabuSolutions.maxelements = 0 [static]
```

Maximum number of elements allowed in the tabu list (unused placeholder).

Definición en la línea 22 del archivo [TabuSolutions.java](#).

Referenciado por [metaheuristics.generators.MultiobjectiveTabuSearch.updateReference\(\)](#) y [metaheuristics.generators.TabuSearch.up](#)

La documentación de esta clase está generada del siguiente archivo:

- [TabuSolutions.java](#)

## 8.100 Referencia de la clase evolutionary\_algorithms.complement.TowPointsMutation

[TowPointsMutation](#) - applies two-points mutation.

Diagrama de herencia de `evolutionary_algorithms.complement.TowPointsMutation`

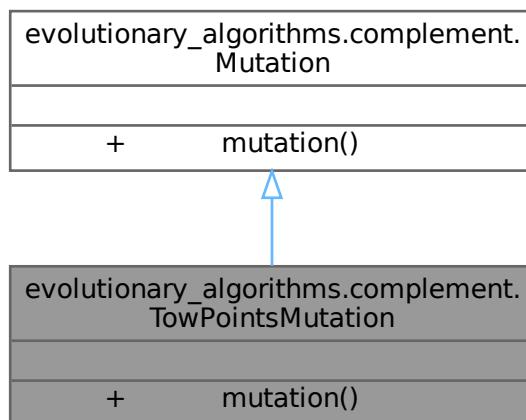
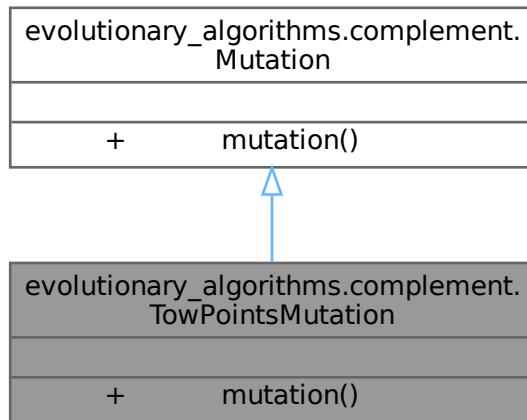


Diagrama de colaboración de `evolutionary_algorithms.complement.TowPointsMutation`:



## Métodos públicos

- `State mutation (State newind, double PM)`

*mutation - applies two-points mutation.*

### 8.100.1 Descripción detallada

[TowPointsMutation](#) - applies two-points mutation.

Definición en la línea 9 del archivo [TowPointsMutation.java](#).

### 8.100.2 Documentación de funciones miembro

#### 8.100.2.1 mutation()

```

State evolutionary_algorithms.complement.TowPointsMutation.mutation (
    State newind,
    double PM)  [inline]
  
```

*mutation - applies two-points mutation.*

#### Parámetros

<i>newind</i>	
<i>PM</i>	

Devuelve

returns the mutated state

Reimplementado de [evolutionary\\_algorithms.complement.Mutation](#).

Definición en la línea 20 del archivo [TowPointsMutation.java](#).

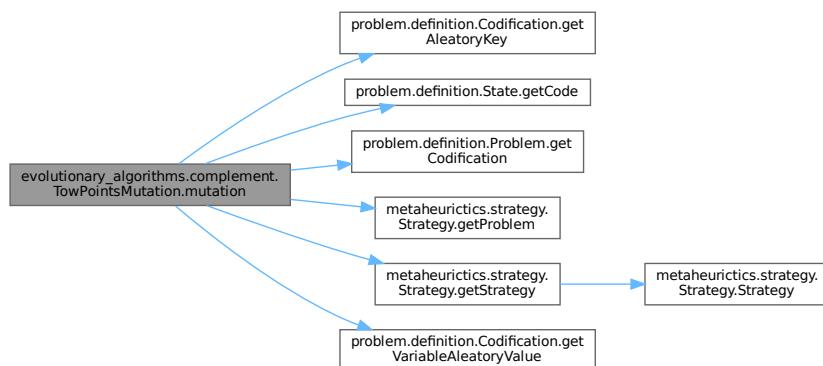
```

00020           {
00021     Object key1 = Strategy.getStrategy().getProblem().getCodification().getAleatoryKey();
00022     Object key2 = Strategy.getStrategy().getProblem().getCodification().getAleatoryKey();
00023     Object value1 =
00024       Strategy.getStrategy().getProblem().getCodification().getVariableAleatoryValue((Integer) key1);
00025     Object value2 =
00026       Strategy.getStrategy().getProblem().getCodification().getVariableAleatoryValue((Integer) key2);
00027     newind.getCode().set((Integer) key1, (Integer)value2);
00028     newind.getCode().set((Integer) key2, (Integer)value1);
00029     return newind;
00030   }

```

Hace referencia a [problem.definition.Codification.getAleatoryKey\(\)](#), [problem.definition.State.getCode\(\)](#), [problem.definition.Problem.get](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#) y [problem.definition.Codification.getVariableAleatoryValue\(\)](#)

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [TowPointsMutation.java](#)

## 8.101 Referencia de la clase

### [evolutionary\\_algorithms.complement.TruncationSelection](#)

[TruncationSelection](#) - applies truncation selection strategy.

Diagrama de herencia de evolutionary\_algorithms.complement.TruncationSelection

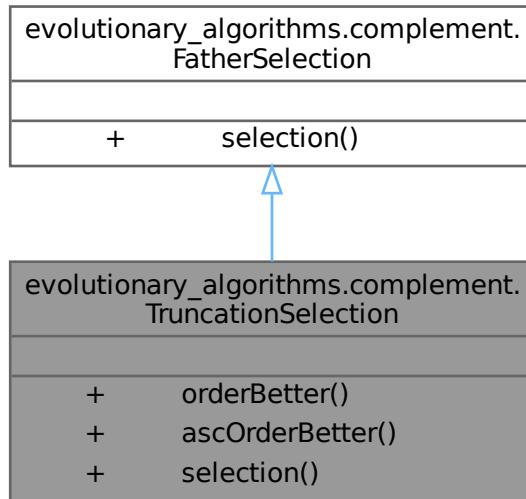
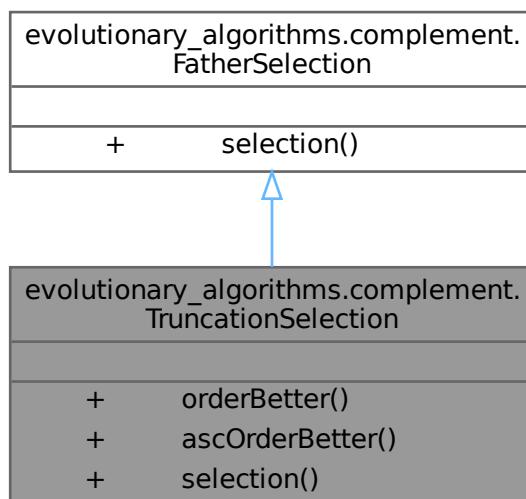


Diagrama de colaboración de `evolutionary_algorithms.complement.TruncationSelection`:



## Métodos públicos

- `List< State > orderBetter (List< State > listState)`  
*orderBetter - applies truncation selection strategy.*

- List< State > **ascOrderBetter** (List< State > listState)
   
ascOrderBetter - applies truncation selection strategy.
- List< State > **selection** (List< State > listState, int truncation)
   
selection - applies truncation selection strategy.

### 8.101.1 Descripción detallada

[TruncationSelection](#) - applies truncation selection strategy.

Definición en la línea 15 del archivo [TruncationSelection.java](#).

### 8.101.2 Documentación de funciones miembro

#### 8.101.2.1 ascOrderBetter()

```
List< State > evolutionary_algorithms.complement.TruncationSelection.ascOrderBetter (
    List< State > listState) [inline]
```

ascOrderBetter - applies truncation selection strategy.

##### Parámetros

<i>listState</i>	
------------------	--

##### Devuelve

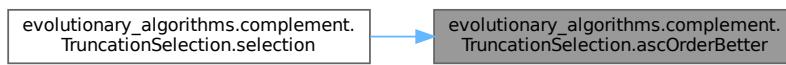
returns the list of states ordered from worst to best.

Definición en la línea 41 del archivo [TruncationSelection.java](#).

```
00041             State var = null;
00042             for (int i = 0; i < listState.size() - 1; i++) {
00043                 for (int j = i + 1; j < listState.size(); j++) {
00044                     if (listState.get(i).getEvaluation().get(0) > listState.get(j).getEvaluation().get(0)) {
00045                         var = listState.get(i);
00046                         listState.set(i, listState.get(j));
00047                         listState.set(j, var);
00048                     }
00049                 }
00050             }
00051         }
00052     return listState;
00053 }
```

Referenciado por [selection\(\)](#).

Gráfico de llamadas a esta función:



## Parámetros

<i>listState</i>	<input type="button" value=""/>
------------------	---------------------------------

## Devuelve

returns the list of states ordered from best to worst.

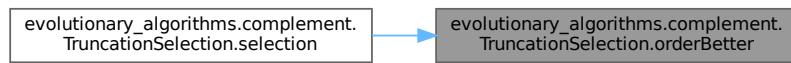
Definición en la línea 22 del archivo [TruncationSelection.java](#).

```

00022
00023     State var = null;
00024     for (int i = 0; i < listState.size()- 1; i++) {
00025         for (int j = i+1; j < listState.size(); j++) {
00026             if(listState.get(i).getEvaluation().get(0) < listState.get(j).getEvaluation().get(0)) {
00027                 var = listState.get(i);
00028                 listState.set(i, listState.get(j));
00029                 listState.set(j,var);
00030             }
00031         }
00032     }
00033     return listState;
00034 }
```

Referenciado por [selection\(\)](#).

Gráfico de llamadas a esta función:



### 8.101.2.3 selection()

```
List< State > evolutionary_algorithms.complement.TruncationSelection.selection (
    List< State > listState,
    int truncation) [inline]
```

selection - applies truncation selection strategy.

## Parámetros

<i>listState</i>	<input type="button" value=""/>
<i>truncation</i>	<input type="button" value=""/>

Devuelve

returns the selected list of states based on truncation.

Reimplementado de [evolutionary\\_algorithms.complement.FatherSelection](#).

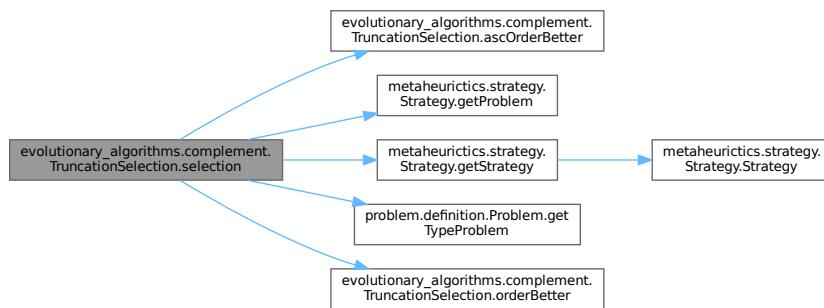
Definición en la línea 62 del archivo [TruncationSelection.java](#).

```

00062      List<State> AuxList = new ArrayList<State>();
00063      if (Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)) {
00064          listState = orderBetter(listState);
00065      } else {
00066          if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Minimizar))
00067              listState = ascOrderBetter(listState);
00068      }
00069      int i = 0;
00070      while(AuxList.size()< truncation){
00071          AuxList.add(listState.get(i));
00072          i++;
00073      }
00074  }
00075  return AuxList;
00076 }
```

Hace referencia a [ascOrderBetter\(\)](#), [metaheuristics.strategy.Strategy.getProblem\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [problem.definition.Problem.getTypeProblem\(\)](#), [problem.definition.Problem.ProblemType.Maximizar](#), [problem.definition.Problem.ProblemType.Minimizar](#) y [orderBetter\(\)](#).

Gráfico de llamadas de esta función:



La documentación de esta clase está generada del siguiente archivo:

- [TruncationSelection.java](#)

## 8.102 Referencia de la clase config.tspDynamic.TSPState

[TSPState](#) - descripción (añade detalles).

Diagrama de colaboración de config.tspDynamic.TSPState:

config.tspDynamic.TSPState	
-	value
-	idCity
+	getValue()
+	setValue()
+	getIdCity()
+	setIdCity()

### Métodos públicos

- int [getValue \(\)](#)  
*getValue - descripción (añade detalles).*
- void [setValue \(int value\)](#)  
*setValue - descripción (añade detalles).*
- int [getIdCity \(\)](#)  
*getIdCity - descripción (añade detalles).*
- void [setIdCity \(int idCity\)](#)  
*setIdCity - descripción (añade detalles).*

### Atributos privados

- int [value](#)
- int [idCity](#)

## 8.102.1 Descripción detallada

[TSPState](#) - descripción (añade detalles).

Definición en la línea 6 del archivo [TSPState.java](#).

## 8.102.2 Documentación de funciones miembro

### 8.102.2.1 [getIdCity\(\)](#)

```
int config.tspDynamic.TSPState.getIdCity () [inline]
```

[getIdCity](#) - descripción (añade detalles).

Devuelve

Definición en la línea 39 del archivo [TSPState.java](#).

```
00039
00040     return idCity;
00041 }
```

Hace referencia a [idCity](#).

### 8.102.2.2 getValue()

```
int config.tspDynamic.TSPState.getValue () [inline]
```

getValue - descripción (añade detalles).

Devuelve

Definición en la línea 16 del archivo [TSPState.java](#).

```
00016           {
00017       return value;
00018 }
```

Hace referencia a [value](#).

### 8.102.2.3 setIdCity()

```
void config.tspDynamic.TSPState.setIdCity (
    int idCity) [inline]
```

setIdCity - descripción (añade detalles).

Parámetros

<i>idCity</i>	
---------------	--

Definición en la línea 46 del archivo [TSPState.java](#).

```
00046           {
00047       this.idCity = idCity;
00048 }
```

Hace referencia a [idCity](#).

### 8.102.2.4 setValue()

```
void config.tspDynamic.TSPState.setValue (
    int value) [inline]
```

setValue - descripción (añade detalles).

Parámetros

<i>value</i>	
--------------	--

Definición en la línea 23 del archivo [TSPState.java](#).

```
00023           {
00024       this.value = value;
00025 }
```

Hace referencia a [value](#).

### 8.102.3 Documentación de datos miembro

#### 8.102.3.1 idCity

```
int config.tspDynamic.TSPState.idCity [private]
```

Definición en la línea 9 del archivo [TSPState.java](#).

Referenciado por [getIdCity\(\)](#) y [setIdCity\(\)](#).

#### 8.102.3.2 value

```
int config.tspDynamic.TSPState.value [private]
```

Definición en la línea 8 del archivo [TSPState.java](#).

Referenciado por [getValue\(\)](#) y [setValue\(\)](#).

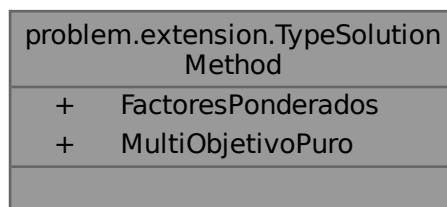
La documentación de esta clase está generada del siguiente archivo:

- [TSPState.java](#)

## 8.103 Referencia de la enumeración `problem.extension.TypeSolutionMethod`

[TypeSolutionMethod](#).

Diagrama de colaboración de `problem.extension.TypeSolutionMethod`:



#### Atributos públicos

- [FactoresPonderados](#)
- [MultiObjetivoPuro](#)

### 8.103.1 Descripción detallada

[TypeSolutionMethod](#).

Enum que lista los métodos de solución soportados por la fábrica (p. ej. factores ponderados o multi-objetivo puro).

Definición en la línea 9 del archivo [TypeSolutionMethod.java](#).

### 8.103.2 Documentación de datos miembro

#### 8.103.2.1 FactoresPonderados

`problem.extension.TypeSolutionMethod.FactoresPonderados`

Definición en la línea 11 del archivo [TypeSolutionMethod.java](#).

#### 8.103.2.2 MultiObjetivoPuro

`problem.extension.TypeSolutionMethod.MultiObjetivoPuro`

Definición en la línea 11 del archivo [TypeSolutionMethod.java](#).

La documentación de esta enumeración está generada del siguiente archivo:

- [TypeSolutionMethod.java](#)

## 8.104 Referencia de la clase evolutionary\_algorithms.complement.UniformCrossover

[UniformCrossover](#) - complements uniform crossover strategy.

Diagrama de herencia de `evolutionary_algorithms.complement.UniformCrossover`

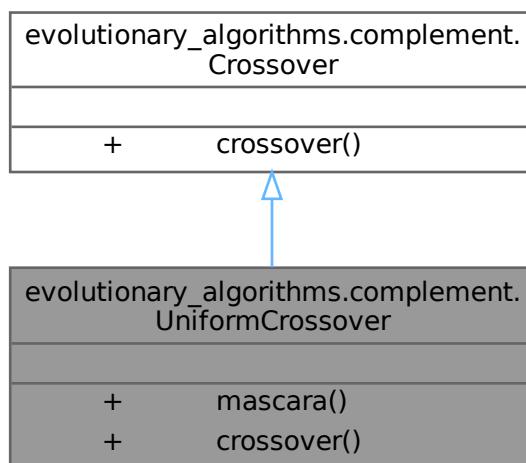
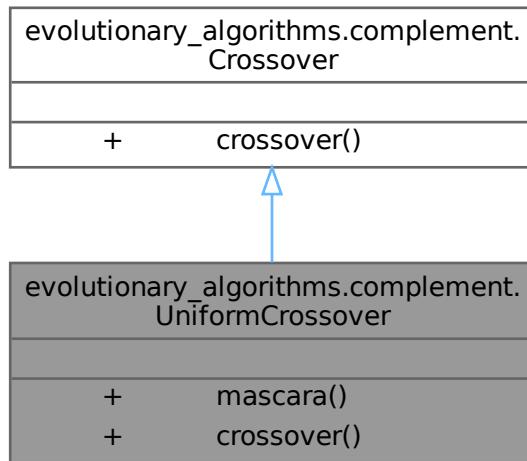


Diagrama de colaboración de `evolutionary_algorithms.complement.UniformCrossover`:



## Métodos públicos

- `int[] mascara (int length)`  
*Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.*
- `State crossover (State father1, State father2, double PC)`  
*crossover - applies uniform crossover strategy.*

### 8.104.1 Descripción detallada

[UniformCrossover](#) - complements uniform crossover strategy.

Definición en la línea 11 del archivo [UniformCrossover.java](#).

### 8.104.2 Documentación de funciones miembro

#### 8.104.2.1 `crossover()`

```
State evolutionary_algorithms.complement.UniformCrossover.crossover (
    State father1,
    State father2,
    double PC) [inline]
```

`crossover` - applies uniform crossover strategy.

#### Parámetros

<i>father1</i>	
<i>father2</i>	
<i>PC</i>	

**Devuelve**

returns the offspring state after crossover.

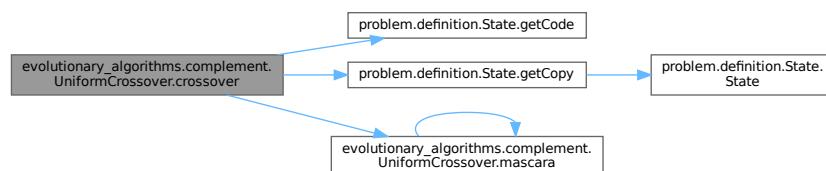
Reimplementado de [evolutionary\\_algorithms.complement.Crossover](#).

Definición en la línea 42 del archivo [UniformCrossover.java](#).

```
00042
00043     Object value;
00044     State state = (State) father1.getCopy();
00045     int[] mascara = mascara(father1.getCode().size());
00046     for (int k = 0; k < mascara.length; k++) {
00047         if(mascara[k] == 1){
00048             value = father1.getCode().get(k);
00049             state.getCode().set(k, value);
00050         }
00051         else{
00052             if(mascara[k] == 0){
00053                 value = father2.getCode().get(k);
00054                 state.getCode().set(k, value);
00055             }
00056         }
00057     }
00058     return state;
00059 }
```

Hace referencia a [problem.definition.State.getCode\(\)](#), [problem.definition.State.getCopy\(\)](#) y [mascara\(\)](#).

Gráfico de llamadas de esta función:

**8.104.2.2 mascara()**

```
int[] evolutionary_algorithms.complement.UniformCrossover.mascara (
    int length) [inline]
```

Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.

This RNG is used for evolutionary algorithm operations (crossover point, selection between children, etc.) and is not used for security-sensitive purposes. Therefore we suppress the Sonar security hotspot S2245 here. `mascara` - applies uniform crossover strategy.

**Parámetros**

<i>length</i>	
---------------	--

**Devuelve**

returns the generated mask for crossover.

Definición en la línea 25 del archivo [UniformCrossover.java](#).

```
00025          {
00026      int[] mascara = new int[length];
00027      for (int i = 0; i < mascara.length; i++) {
00028          int value = ThreadLocalRandom.current().nextInt(2);
00029          mascara[i] = value;
00030      }
00031      return mascara;
00032  }
```

Hace referencia a [mascara\(\)](#).

Referenciado por [crossover\(\)](#) y [mascara\(\)](#).

Gráfico de llamadas de esta función:

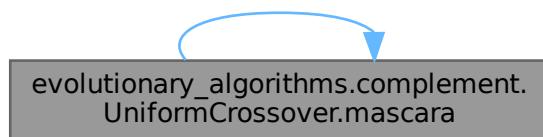


Gráfico de llamadas a esta función:



La documentación de esta clase está generada del siguiente archivo:

- [UniformCrossover.java](#)

## 8.105 Referencia de la clase evolutionary\_algorithms.complement.Univariate

[Univariate](#) - applies univariate distribution strategy.

Diagrama de herencia de evolutionary\_algorithms.complement.Univariate

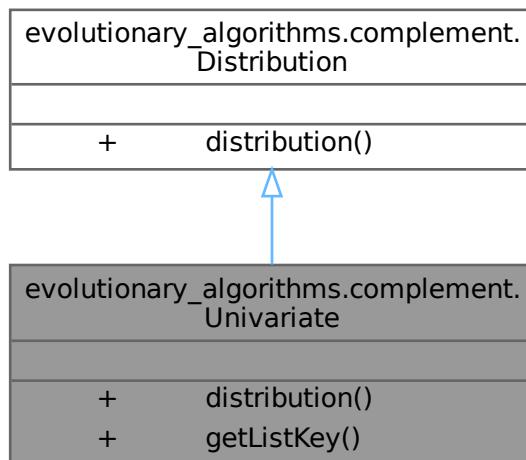
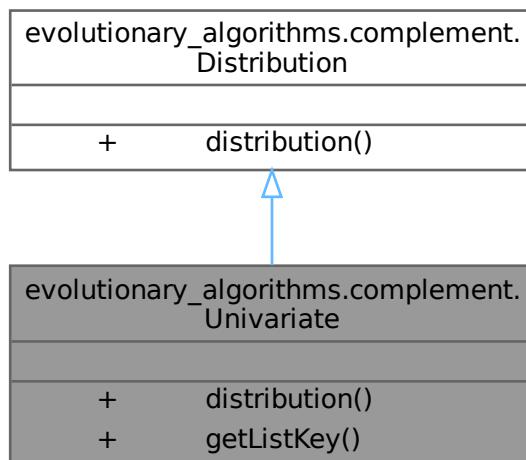


Diagrama de colaboración de evolutionary\_algorithms.complement.Univariate:



## Métodos públicos

- List< Probability > **distribution** (List< State > fathers)
   
*distribution - applies univariate distribution strategy.*
- List< String > **getListKey** (SortedMap< String, Object > map)
   
*getListKey - applies univariate distribution strategy.*

### 8.105.1 Descripción detallada

[Univariate](#) - applies univariate distribution strategy.

Definición en la línea 14 del archivo [Univariate.java](#).

### 8.105.2 Documentación de funciones miembro

#### 8.105.2.1 distribution()

```
List< Probability > evolutionary_algorithms.complement.Univariate.distribution (
    List< State > fathers) [inline]
```

distribution - applies univariate distribution strategy.

#### Parámetros

<i>fathers</i>	
----------------	--

#### Devuelve

returns the list of probabilities for each variable value.

Reimplementado de [evolutionary\\_algorithms.complement.Distribution](#).

Definición en la línea 22 del archivo [Univariate.java](#).

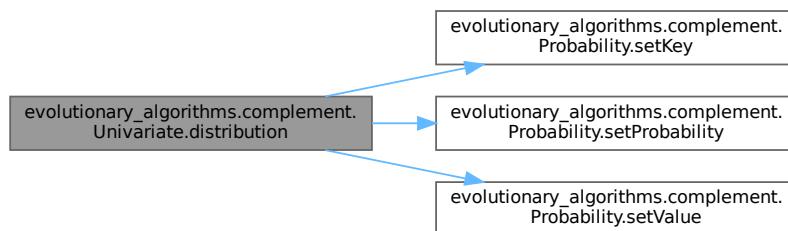
```
00022
00023
00024     List<Probability> ListProbability = new ArrayList<Probability>();
00025     int cantV = fathers.get(0).getCode().size();
00026     for (int i = 0; i < cantV; i++) {
00027         int[] values = new int[father.size()];
00028         //llenar los valores de cada variable
00029         for (int j = 0; j < fathers.size(); j++) {
00030             values[j] = (Integer) fathers.get(j).getCode().get(i);
00031         }
00032         //para cada valor contar la cantidad de veces que se repite
00033         int k = 0;
00034         while (k < values.length) {
00035             float count = 0;
00036             if(values[k] != -1) {
00037                 int l = k;
00038                 int temp = values[k];
00039                 while (l < values.length) {
00040                     if(temp == values[l]) {
00041                         count++;
00042                         values[l] = -1;
00043                     }
00044                     l++;
00045                 }
00046                 Probability probability = new Probability();
00047                 float prob = count / values.length;
```

```

00048         probability.setKey(i);
00049         probability.setValue(temp);
00050         probability.setProbability(prob);
00051         ListProbability.add(probability);
00052     }
00053     k++;
00054 }
00055 }
00056 return ListProbability;
00057 }
```

Hace referencia a [evolutionary\\_algorithms.complement.Probability.setKey\(\)](#), [evolutionary\\_algorithms.complement.Probability.setProbability\(\)](#) y [evolutionary\\_algorithms.complement.Probability.setValue\(\)](#).

Gráfico de llamadas de esta función:



### 8.105.2.2 getListKey()

```
List< String > evolutionary_algorithms.complement.Univariate.getListKey (
    SortedMap< String, Object > map) [inline]
```

getListKey - applies univariate distribution strategy.

#### Parámetros

<i>SortedMap&lt;String</i>	
<i>map</i>	

#### Devuelve

returns the list of keys from the given map.

Definición en la línea 65 del archivo [Univariate.java](#).

```

00065
00066     List<String> listKey = new ArrayList<String>();
00067     String key = map.keySet().toString();
00068     String returnString = key.substring(1, key.length() - 1);
00069     returnString = returnString + ", ";
00070     int countKey = map.size();
00071     for (int j = 0; j < countKey; j++) {
00072         String r = returnString.substring(0, returnString.indexOf(',') );
00073         returnString = returnString.substring(returnString.indexOf(',') + 2);
00074         listKey.add(r);
00075     }
00076     return listKey;
00077 }
```

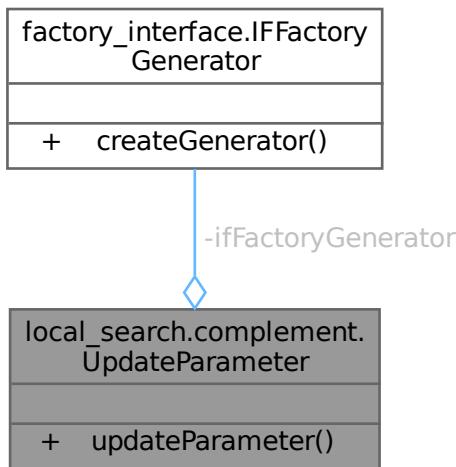
La documentación de esta clase está generada del siguiente archivo:

- [Univariate.java](#)

## 8.106 Referencia de la clase local\_search.complement.UpdateParameter

[UpdateParameter](#) - utility that updates generator parameters during search.

Diagrama de colaboración de local\_search.complement.UpdateParameter:



### Métodos públicos estáticos

- static Integer [updateParameter](#) (Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException
 

*Increment the provided iteration counter and, when thresholds for different generators are reached, replace the Strategy's generator with the corresponding implementation.*

### Atributos estáticos privados

- static IFFactoryGenerator [ifFactoryGenerator](#)

### 8.106.1 Descripción detallada

[UpdateParameter](#) - utility that updates generator parameters during search.

Responsible for incrementing an iteration counter and switching the active generator implementation in the global [Strategy](#) when certain iteration thresholds are reached.

Definición en la línea 23 del archivo [UpdateParameter.java](#).

## 8.106.2 Documentación de funciones miembro

### 8.106.2.1 updateParameter()

```
Integer local_search.complement.UpdateParameter.updateParameter (
    Integer countIterationsCurrent) throws IllegalArgumentException, SecurityException,
ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
NoSuchMethodException [inline], [static]
```

Increment the provided iteration counter and, when thresholds for different generators are reached, replace the [Strategy](#)'s generator with the corresponding implementation.

Common use: called at each iteration to advance the count and rotate the generator when configured counts are reached.

#### Parámetros

<i>countIterationsCurrent</i>	current iteration counter
-------------------------------	---------------------------

#### Devuelve

incremented iteration counter

#### Excepciones

<i>ReflectiveOperationException</i>	when reflective generator creation fails
-------------------------------------	--

Definición en la línea 39 del archivo [UpdateParameter.java](#).

```
00039     { //HashMap<String, Object> map,
00040         countIterationsCurrent = countIterationsCurrent + 1;
00041         //      Here update parameter for update and change generator.
00042         if(countIterationsCurrent.equals(GeneticAlgorithm.countRef - 1)){
00043             ifFactoryGenerator = new FactoryGenerator();
00044             Strategy.getStrategy().generator =
00045                 iffFactoryGenerator.createGenerator(GeneratorType.GeneticAlgorithm);
00046         } else{
00047             if(countIterationsCurrent.equals(EvolutionStrategies.countRef - 1)){
00048                 ifFactoryGenerator = new FactoryGenerator();
00049                 Strategy.getStrategy().generator =
00050                     iffFactoryGenerator.createGenerator(GeneratorType.EvolutionStrategies);
00051             } if(countIterationsCurrent.equals(DistributionEstimationAlgorithm.countRef - 1)){
00052                 ifFactoryGenerator = new FactoryGenerator();
00053                 Strategy.getStrategy().generator =
00054                     iffFactoryGenerator.createGenerator(GeneratorType.DistributionEstimationAlgorithm);
00055             } if(countIterationsCurrent.equals(ParticleSwarmOptimization.getCountRef() - 1)){
00056                 ifFactoryGenerator = new FactoryGenerator();
00057                 Strategy.getStrategy().generator =
00058                     iffFactoryGenerator.createGenerator(GeneratorType.ParticleSwarmOptimization);
00059             }
00060         }
00061     } return countIterationsCurrent;
```

Hace referencia a [metaheuristics.generators.DistributionEstimationAlgorithm.countRef](#), [metaheuristics.generators.EvolutionStrategies](#), [metaheuristics.generators.GeneticAlgorithm.countRef](#), [metaheuristics.generators.GeneratorType.DistributionEstimationAlgorithm](#), [metaheuristics.generators.GeneratorType.EvolutionStrategies](#), [metaheuristics.strategy.Strategy.generator](#), [metaheuristics.generators.metaheuristics.generators.ParticleSwarmOptimization.getCountRef\(\)](#), [metaheuristics.strategy.Strategy.getStrategy\(\)](#), [iffFactoryGenerator](#) y [metaheuristics.generators.GeneratorType.ParticleSwarmOptimization](#).

Referenciado por [metaheuristics.strategy.Strategy.executeStrategy\(\)](#).

Gráfico de llamadas de esta función:

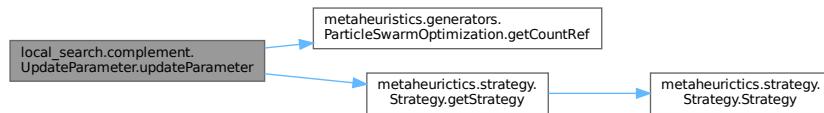


Gráfico de llamadas a esta función:



### 8.106.3 Documentación de datos miembro

#### 8.106.3.1 ifFactoryGenerator

`IFFactoryGenerator local_search.complement.UpdateParameter.ifFactoryGenerator [static], [private]`

Definición en la línea 25 del archivo [UpdateParameter.java](#).

Referenciado por [updateParameter\(\)](#).

La documentación de esta clase está generada del siguiente archivo:

- [UpdateParameter.java](#)

# Chapter 9

## Documentación de archivos

### 9.1 Referencia del archivo TSPState.java

#### Clases

- class config.tspDynamic.TSPState  
*TSPState* - descripción (añade detalles).

#### Paquetes

- package config.tspDynamic

### 9.2 TSPState.java

[Ir a la documentación de este archivo.](#)

```
00001 package config.tspDynamic;
00002
00006 public class TSPState {
00007
00008     private int value;
00009     private int idCity;
00010     //private boolean pool;
00011
00016     public int getValue() {
00017         return value;
00018     }
00023     public void setValue(int value) {
00024         this.value = value;
00025     }
00026     /*
00027     public boolean isPool() {
00028         return pool;
00029     }
00030
00031     public void setPool(boolean pool) {
00032         this.pool = pool;
00033     }
00034     */
00039     public int getIdCity() {
00040         return idCity;
00041     }
00046     public void setIdCity(int idCity) {
00047         this.idCity = idCity;
00048     }
00049 }
00050 }
```

## 9.3 Referencia del archivo App.java

### Clases

- class [es.ull.App](#)

*Hello world!*

### Paquetes

- package [es.ull](#)

## 9.4 App.java

[Ir a la documentación de este archivo.](#)

```
00001 package es.ull;
00002
00010 public class App {
00015     public static void main( String[] args )
00016     {
00017         System.out.println( "Hello World!" );
00018     }
00019 }
```

## 9.5 Referencia del archivo AIOMutation.java

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import config.tspDynamic.TSPState;
import metaheuristics.strategy.Strategy;
import problem.definition.State;
Gráfico de dependencias incluidas en AIOMutation.java:
```



### Clases

- class [evolutionary\\_algorithms.complement.AIOMutation](#)
- AIOMutation - applies the [AIOMutation](#) to the state.*

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.6 AIOMutation.java

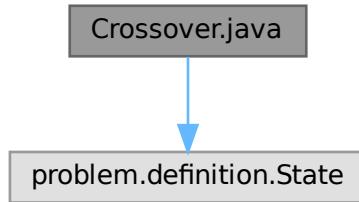
[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00003 import java.util.ArrayList;
00004 import java.util.Collections;
00005 import java.util.List;
00006
00007 import config.tspDynamic.TSPState;
00008
00009
00010 import metaheuristics.strategy.Strategy;
00011
00012 import problem.definition.State;
00013
00017 public class AIOMutation extends Mutation {
00018
00022     public static final List<Object> path = Collections.synchronizedList(new ArrayList<Object>());
00023
00024     @Override
00031     public State mutation(State state, double PM) {
00032         // TODO Auto-generated method stub
00033         int key = Strategy.getStrategy().getProblem().getCodification().getAleatoryKey();
00034         //selecciónar aleatoriamente una ciudad
00035         int key1 = 0;
00036         boolean found = false;
00037         while (found == false){
00038             key1 = Strategy.getStrategy().getProblem().getCodification().getAleatoryKey();
00039             if(key1 != key)
00040                 found = true;
00041         }
00042         sortedPathValue(state);
00043         int p1 = 0;
00044         int p2 = 0;
00045         if(key > key1){
00046             p2 = key;
00047             p1 = key1;
00048         }
00049         else{
00050             p1 = key;
00051             p2 = key1;
00052         }
00053         int length = (p2 - p1) / 2;
00054         for (int i = 1; i <= length + 1; i++) {
00055             int tempC = ((TSPState) state.getCode().get(p1 + i - 1)).getIdCity();
00056             ((TSPState)state.getCode().get(p1 + i - 1)).setIdCity(((TSPState)state.getCode().get(p2 - i + 1)).getIdCity());
00057             ((TSPState)state.getCode().get(p2 - i + 1)).setIdCity(tempC);
00058         }
00059         path.clear();
00060         return state;
00061     }
00066     public void sortedPathValue(State state) {
00067         for(int k = 0; k < state.getCode().size(); k++){
00068             path.add( state.getCode().get(k));
00069         }
00070         for(int i = 1; i < path.size(); i++){
00071             for(int j = 0; j < i; j++){
00072                 Integer data1 = ((TSPState)state.getCode().get(i)).getValue();
00073                 Integer data2 = ((TSPState)state.getCode().get(j)).getValue();
00074                 if(data1 < data2){
00075                     state.getCode().add(j, state.getCode().remove(i));
00076                     path.add(j, path.remove(i));
00077                 }
00078             }
00079         }
00080     }
00081     public static void fillPath() {
00086         for(int k = 0; k < Strategy.getStrategy().getProblem().getCodification().getVariableCount();
00087             k++){
00088             path.add(k);
00089         }
00090     }
00091 }
```

## 9.7 Referencia del archivo Crossover.java

```
import problem.definition.State;
```

Gráfico de dependencias incluidas en Crossover.java:



### Clases

- class [evolutionary\\_algorithms.complement.Crossover](#)  
*Crossover - applies the crossover operation to two parent states.*

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.8 Crossover.java

[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00003
00004 import problem.definition.State;
00005
00009 public abstract class Crossover {
00017     public abstract State crossover(State father1, State father2, double PC);
00018 }
```

## 9.9 Referencia del archivo CrossoverType.java

### Clases

- enum [evolutionary\\_algorithms.complement.CrossoverType](#)  
*CrossoverType - descripcion (añade detalles).*

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.10 CrossoverType.java

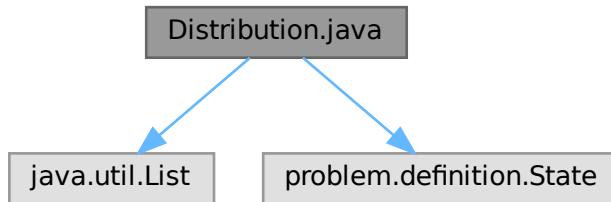
[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00006 public enum CrossoverType {
00007     OnePointCrossover, UniformCrossover;
00008 }
```

## 9.11 Referencia del archivo Distribution.java

```
import java.util.List;
import problem.definition.State;
```

Gráfico de dependencias incluidas en Distribution.java:



### Clases

- class `evolutionary_algorithms.complement.Distribution`  
*Distribution* - applies a probability distribution to a set of fathers.

### Paquetes

- package `evolutionary_algorithms.complement`

## 9.12 Distribution.java

[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00003 import java.util.List;
00004
00005 import problem.definition.State;
00006
00007
00011 public abstract class Distribution {
00017     public abstract List<Probability> distribution(List<State> fathers);
00018
00019 }
```

## 9.13 Referencia del archivo DistributionType.java

### Clases

- enum [evolutionary\\_algorithms.complement.DistributionType](#)  
*DistributionType - descripción (añade detalles).*

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.14 DistributionType.java

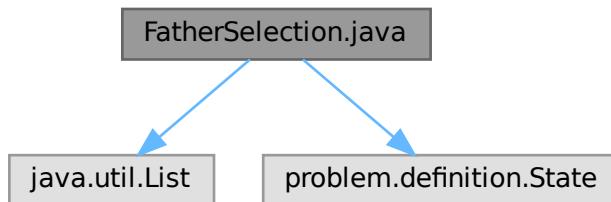
[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00006 public enum DistributionType {
00007     Univariate;
00008 }
```

## 9.15 Referencia del archivo FatherSelection.java

```
import java.util.List;
import problem.definition.State;
```

Gráfico de dependencias incluidas en FatherSelection.java:



### Clases

- class [evolutionary\\_algorithms.complement.FatherSelection](#)  
*FatherSelection - selects the best fathers from the population.*

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.16 FatherSelection.java

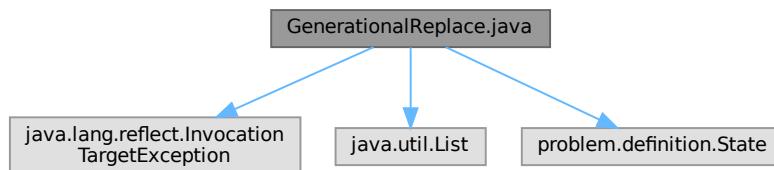
[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00003 import java.util.List;
00004
00005 import problem.definition.State;
00006
00007
00011 public abstract class FatherSelection {
00018     public abstract List<State> selection(List<State> listState, int truncation);
00019
00020 }
```

## 9.17 Referencia del archivo GenerationalReplace.java

```
import java.lang.reflect.InvocationTargetException;
import java.util.List;
import problem.definition.State;
```

Gráfico de dependencias incluidas en GenerationalReplace.java:



### Clases

- class [evolutionary\\_algorithms.complement.GenerationalReplace](#)  
*GenerationalReplace - replaces the worst individual in the population.*

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.18 GenerationalReplace.java

[Ir a la documentación de este archivo.](#)

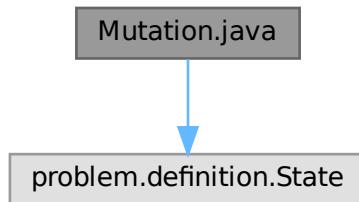
```
00001 package evolutionary_algorithms.complement;
00002
00003
00004 import java.lang.reflect.InvocationTargetException;
00005 import java.util.List;
00006
00007 import problem.definition.State;
00008
00012 public class GenerationalReplace extends Replace {
00013 }
```

```

00014     @Override
00021     public List<State> replace(State stateCandidate, List<State> listState) throws
00022         IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00023         IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00022     listState.remove(0);
00023     listState.add(stateCandidate);
00024     return listState;
00025 }
00026 }
```

## 9.19 Referencia del archivo Mutation.java

import problem.definition.State;  
 Gráfico de dependencias incluidas en Mutation.java:



### Clases

- class [evolutionary\\_algorithms.complement.Mutation](#)  
*Mutation* - applies a mutation to a given state.

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.20 Mutation.java

[Ir a la documentación de este archivo.](#)

```

00001 package evolutionary_algorithms.complement;
00002
00003 import problem.definition.State;
00004
00008 public abstract class Mutation {
00015     public abstract State mutation (State state, double PM);
00016
00017 }
```

## 9.21 Referencia del archivo MutationType.java

### Clases

- enum [evolutionary\\_algorithms.complement.MutationType](#)  
*MutationType* - descripción (añade detalles).

## Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.22 MutationType.java

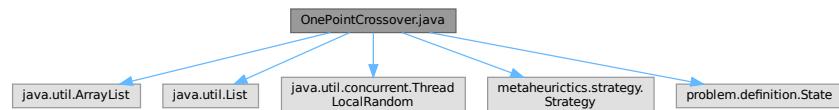
[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00006 public enum MutationType {
00007     TwoPointsMutation, OnePointMutation, AIOMutation;
00008 }
```

## 9.23 Referencia del archivo OnePointCrossover.java

```
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.ThreadLocalRandom;
import metaheuristics.strategy.Strategy;
import problem.definition.State;
```

Gráfico de dependencias incluidas en OnePointCrossover.java:



## Clases

- class [evolutionary\\_algorithms.complement.OnePointCrossover](#)  
*OnePointCrossover - applies the one-point crossover operator.*

## Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.24 OnePointCrossover.java

[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00003 import java.util.ArrayList;
00004 import java.util.List;
00005 import java.util.concurrent.ThreadLocalRandom;
00006
00007 import metaheuristics.strategy.Strategy;
00008
00009 import problem.definition.State;
00010
```

```

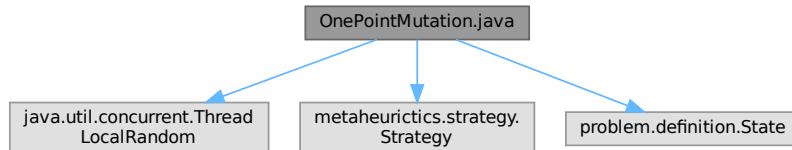
00011
00015 public class OnePointCrossover extends Crossover {
00016
00023     @SuppressWarnings("squid:S2245")
00024     @Override
00032     public State crossover(State father1, State father2, double PC) {
00033
00034         State newInd = (State) father1.getCopy();
00035
00036         List<Object> ind1 = new ArrayList<Object>();
00037         List<Object> ind2 = new ArrayList<Object>();
00038
00039         double number = ThreadLocalRandom.current().nextDouble(); // thread-safe, [0.0, 1.0)
00040         if(number <= PC){
00041             //llenar los valores de cada hijo
00042             int bound = Strategy.getStrategy().getProblem().getCodification().getVariableCount();
00043             int pos = (bound > 0) ? ThreadLocalRandom.current().nextInt(bound) : 0;
00044             for (int i = 0; i < father1.getCode().size(); i++) {
00045                 if(i <= pos){
00046                     ind1.add(father1.getCode().get(i));
00047                     ind2.add(father2.getCode().get(i));
00048                 }
00049                 else{
00050                     ind1.add(father2.getCode().get(i));
00051                     ind2.add(father1.getCode().get(i));
00052                 }
00053             }
00054
00055             //generar un numero aleatorio 0 o 1, si es 0 me quedo con ind1 si es 1 con ind2.
00056             int random = ThreadLocalRandom.current().nextInt(2);
00057             if(random == 0)
00058                 newInd.setCode((ArrayList<Object>) ind1);
00059             else newInd.setCode((ArrayList<Object>) ind2);
00060         }
00061         return newInd;
00062     }
00063
00064
00065 }
```

## 9.25 Referencia del archivo OnePointMutation.java

```

import java.util.concurrent.ThreadLocalRandom;
import metaheuristics.strategy.Strategy;
import problem.definition.State;
```

Gráfico de dependencias incluidas en OnePointMutation.java:



### Clases

- class [evolutionary\\_algorithms.complement.OnePointMutation](#)  
*OnePointMutation - applies the one-point mutation operator.*

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.26 OnePointMutation.java

[Ir a la documentación de este archivo.](#)

```

00001 package evolutionary_algorithms.complement;
00002
00003 import java.util.concurrent.ThreadLocalRandom;
00004
00005 import metaheuristics.strategy.Strategy;
00006 import problem.definition.State;
00007
00011 public class OnePointMutation extends Mutation {
00012
00013
00020     @SuppressWarnings("squid:S2245")
00021     @Override
00028     public State mutation(State state, double PM) {
00029         double probM = ThreadLocalRandom.current().nextDouble();
00030         if(PM >= probM)
00031         {
00032             Object key = Strategy.getStrategy().getProblem().getCodification().getAleatoryKey();
00033             Object value =
00034                 Strategy.getStrategy().getProblem().getCodification().getVariableAleatoryValue((Integer)key);
00035             state.getCode().set((Integer) key, value);
00036         }
00036         return state;
00037     }
00038 }
```

## 9.27 Referencia del archivo ProbabilisticSampling.java

```

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.ThreadLocalRandom;
import metaheuristics.strategy.Strategy;
import metaheuristics.generators.GeneratorType;
import problem.definition.State;
```

Gráfico de dependencias incluidas en ProbabilisticSampling.java:



### Clases

- class [evolutionary\\_algorithms.complement.ProbabilisticSampling](#)  
*ProbabilisticSampling* - applies probabilistic sampling to select individuals.

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.28 ProbabilisticSampling.java

[Ir a la documentación de este archivo.](#)

```

00001 package evolutionary_algorithms.complement;
00002
00003
00004 import java.util.ArrayList;
00005 import java.util.List;
00006 import java.util.concurrent.ThreadLocalRandom;
00007
00008 import metaheuristics.strategy.Strategy;
00009 import metaheuristics.generators.GeneratorType;
00010
00011 import problem.definition.State;
00012
00013
00017 public class ProbabilisticSampling extends Sampling {
00018
00025     @SuppressWarnings("squid:S2245")
00026     @Override
00033     public List<State> sampling(List<State> fathers, int countInd) {
00034         // TODO Auto-generated method stub
00035         int cantV = fathers.get(0).getCode().size();
00036         List<State> staList = listState(countInd);
00037         for (int i = 0; i < cantV; i++) {
00038             Object[] values = new Object[fathers.size()]; // arreglo de valores de una variable
00039             Object[] arrtemp = new Object[Strategy.getStrategy().getProblem().getPossibleValue()];
00040             //llenar el arreglo con todos los valores posibles
00041             for (int j = 0; j < arrtemp.length; j++) {
00042                 arrtemp[j] = j;
00043             }
00044             for (int j = 0; j < values.length; j++) {
00045                 values[j] = fathers.get(j).getCode().get(i);
00046             }
00047             int k = 0;
00048             int sum = 0; // suma acumulativa por cada valor posible
00049             int arrOcc[] = new int[arrtemp.length]; // arreglo paralelo para contar la cantidad de
00050             //ocurrencias de un valor posible
00051             //llenar el arreglo con la cantidad de ocurrencias de cada valor posible, para cada
00052             //variable
00053             //recorrer el arreglo de valores de una variable
00054             while (k < arrtemp.length) {
00055                 int count = 0;
00056                 for (int j = 0; j < values.length; j++) {
00057                     if((Integer)values[j] != -1 && values[j] == arrtemp[k]){ //
00058                         count++;
00059                         values[j] = -1;
00060                     }
00061                     arrOcc[k] = count;
00062                     sum = sum + count;
00063                     k++;
00064                 }
00065                 for (int l = 0; l < countInd; l++) {
00066                     boolean find = false;
00067                     int p = 0;
00068                     int random;
00069                     if (sum > 0) random = ThreadLocalRandom.current().nextInt(sum) + 1;
00070                     else random = 1;
00071                     while (p < arrOcc.length && find == false) {
00072                         random = random - arrOcc[p];
00073                         if(random <= 0){
00074                             staList.get(l).getCode().add(arrtemp[p]);
00075                             find = true;
00076                         }
00077                         else p++;
00078                     }
00079                     if(find == false){
00080                         int bound =
00081                         Strategy.getStrategy().getProblem().getCodification().getVariableCount() * 10;
00082                         int value = (bound > 0) ? ThreadLocalRandom.current().nextInt(bound) : 0;
00083                         staList.get(l).getCode().add(Integer.valueOf(value));
00084                     }
00085                 }
00086             return staList;
00087         }
00088
00089         // inicializa la lista de individuos
00095         public List<State> listState(int countInd) {
00096             List<State> staList = new ArrayList<State>(countInd);
00097             for (int i = 0; i < countInd; i++) {
00098                 State state = new State();
00099                 state.setCode(new ArrayList<Object>());

```

```

00100         state.setNumber(Strategy.getStrategy().getCountCurrent());
00101         state.setTypeGenerator(GeneratorType.DistributionEstimationAlgorithm);
00102         staList.add(state);
00103     }
00104     return staList;
00105 }
00106 }
```

## 9.29 Referencia del archivo Probability.java

### Clases

- class [evolutionary\\_algorithms.complement.Probability](#)

*Probability* - represents the probability of a certain event.

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.30 Probability.java

[Ir a la documentación de este archivo.](#)

```

00001 package evolutionary_algorithms.complement;
00002
00006 public class Probability {
00007     private Object key;
00008     private Object value;
00009     private float probability;
00010
00011
00016     public float getProbability() {
00017         return probability;
00018     }
00023     public void setProbability(float probability) {
00024         this.probability = probability;
00025     }
00030     public Object getKey() {
00031         return key;
00032     }
00037     public void setKey(Object key) {
00038         this.key = key;
00039     }
00044     public Object getValue() {
00045         return value;
00046     }
00051     public void setValue(Object value) {
00052         this.value = value;
00053     }
00054 }
```

## 9.31 Referencia del archivo Range.java

### Clases

- class [evolutionary\\_algorithms.complement.Range](#)

*Range* - represents a range of values with associated probabilities.

## Paquetes

- package [evolutionary\\_algorithms.complement](#)

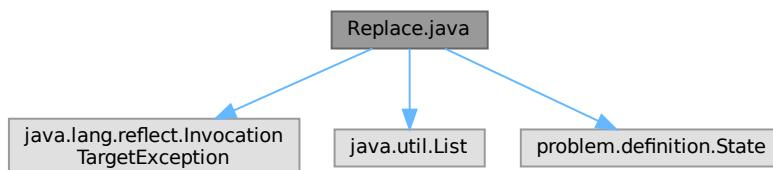
## 9.32 Range.java

[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00006 public class Range {
00007     private Probability data;
00008     private float max;
00009     private float min;
00010
00015     public Probability getData() {
00016         return data;
00017     }
00022     public void setData(Probability data) {
00023         this.data = data;
00024     }
00029     public float getMax() {
00030         return max;
00031     }
00036     public void setMax(float max) {
00037         this.max = max;
00038     }
00043     public float getMin() {
00044         return min;
00045     }
00050     public void setMin(float min) {
00051         this.min = min;
00052     }
00053 }
```

## 9.33 Referencia del archivo Replace.java

```
import java.lang.reflect.InvocationTargetException;
import java.util.List;
import problem.definition.State;
Gráfico de dependencias incluidas en Replace.java:
```



## Clases

- class [evolutionary\\_algorithms.complement.Replace](#)

*Replace - applies replacement strategies for individuals in the population.*

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.34 Replace.java

[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.List;
00005
00006 import problem.definition.State;
00007
00008
00012 public abstract class Replace {
00026     public abstract List<State> replace(State stateCandidate, List<State>listState) throws
        IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
        IllegalAccessException, InvocationTargetException, NoSuchMethodException;
00027 }
```

## 9.35 Referencia del archivo ReplaceType.java

### Clases

- enum [evolutionary\\_algorithms.complement.ReplaceType](#)
- ReplaceType - represents the type of replacement strategy.*

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

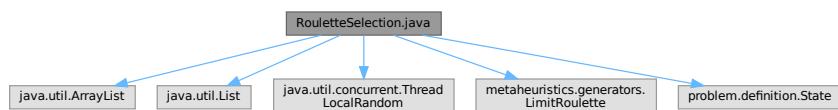
## 9.36 ReplaceType.java

[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00006 public enum ReplaceType {
00007     SteadyStateReplace, GenerationalReplace;
00008 }
```

## 9.37 Referencia del archivo RouletteSelection.java

```
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.ThreadLocalRandom;
import metaheuristics.generators.LimitRoulette;
import problem.definition.State;
Gráfico de dependencias incluidas en RouletteSelection.java:
```



## Clases

- class [evolutionary\\_algorithms.complement.RouletteSelection](#)  
*RouletteSelection - applies roulette selection to choose parents.*

## Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.38 RouletteSelection.java

[Ir a la documentación de este archivo.](#)

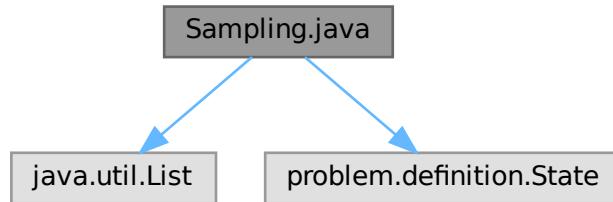
```

00001 package evolutionary_algorithms.complement;
00002
00003 import java.util.ArrayList;
00004 import java.util.List;
00005 import java.util.concurrent.ThreadLocalRandom;
00006
00007 import metaheuristics.generators.LimitRoulette;
00008
00009 import problem.definition.State;
00010
00014 public class RouletteSelection extends FatherSelection {
00015
00016     @Override
00023     public List<State> selection(List<State> listState, int truncation) {
00024         float totalWeight = 0;
00025         for (int i = 0; i < listState.size(); i++) {
00026             totalWeight = (float) (listState.get(i).getEvaluation().get(0) + totalWeight);
00027         }
00028         List<Float> listProb = new ArrayList<Float>();
00029         for (int i = 0; i < listState.size(); i++) {
00030             float probF = (float) (listState.get(i).getEvaluation().get(0) / totalWeight);
00031             listProb.add(probF);
00032         }
00033         List<LimitRoulette> listLimit = new ArrayList<LimitRoulette>();
00034         float limitHigh = 0;
00035         float limitLow = 0;
00036         for (int i = 0; i < listProb.size(); i++) {
00037             LimitRoulette limitRoulette = new LimitRoulette();
00038             limitHigh = listProb.get(i) + limitHigh;
00039             limitRoulette.setLimitHigh(limitHigh);
00040             limitRoulette.setLimitLow(limitLow);
00041             limitLow = limitHigh;
00042             // limitRoulette.setGenerator(listGenerators.get(i));
00043             listLimit.add(limitRoulette);
00044         }
00045         List<State> fatherList = new ArrayList<State>();
00046         for (int j = 0; j < listState.size(); j++) {
00047             // Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.
00048             // This RNG decides selection in the evolutionary algorithm and is not
00049             // used for security-sensitive purposes. Suppress Sonar hotspot S2245.
00050             @SuppressWarnings("squid:S2245")
00051             float numbAleatory = (float) ThreadLocalRandom.current().nextDouble();
00052             boolean find = false;
00053             int i = 0;
00054             while ((find == false) && (i < listLimit.size())){
00055                 if((listLimit.get(i).getLimitLow() <= numbAleatory) && (numbAleatory <=
listLimit.get(i).getLimitHigh())){
00056                     find = true;
00057                     fatherList.add(listState.get(i));
00058                 }
00059                 else i++;
00060             }
00061         }
00062         return fatherList;
00063     }
00064 }
```

## 9.39 Referencia del archivo Sampling.java

```
import java.util.List;
import problem.definition.State;
```

Gráfico de dependencias incluidas en Sampling.java:



### Clases

- class [evolutionary\\_algorithms.complement.Sampling](#)  
*Sampling - represents a sampling strategy for selecting individuals.*

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.40 Sampling.java

[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00003 import java.util.List;
00004
00005 import problem.definition.State;
00006
00007
00011 public abstract class Sampling {
00018     public abstract List<State> sampling (List<State> fathers, int countInd);
00019 }
```

## 9.41 Referencia del archivo SamplingType.java

### Clases

- enum [evolutionary\\_algorithms.complement.SamplingType](#)  
*SamplingType - represents the type of sampling strategy.*

## Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.42 SamplingType.java

[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00006 public enum SamplingType {
00007     ProbabilisticSampling;
00008 }
```

## 9.43 Referencia del archivo SelectionType.java

### Clases

- enum [evolutionary\\_algorithms.complement.SelectionType](#)  
*SelectionType - represents the type of selection strategy.*

## Paquetes

- package [evolutionary\\_algorithms.complement](#)

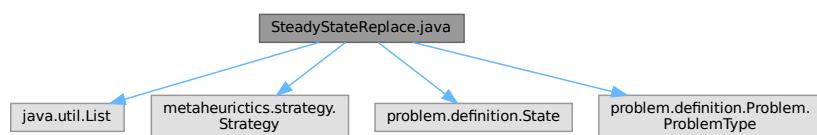
## 9.44 SelectionType.java

[Ir a la documentación de este archivo.](#)

```
00001 package evolutionary_algorithms.complement;
00002
00006 public enum SelectionType {
00007     RouletteSelection, TruncationSelection;
00008 }
```

## 9.45 Referencia del archivo SteadyStateReplace.java

```
import java.util.List;
import metaheuristics.strategy.Strategy;
import problem.definition.State;
import problem.definition.Problem.ProblemType;
Gráfico de dependencias incluidas en SteadyStateReplace.java:
```



## Clases

- class [evolutionary\\_algorithms.complement.SteadyStateReplace](#)  
*SteadyStateReplace - applies steady-state replacement strategy.*

## Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.46 SteadyStateReplace.java

[Ir a la documentación de este archivo.](#)

```

00001 package evolutionary_algorithms.complement;
00002
00003
00004 import java.util.List;
00005 import metaheuristics.strategy.Strategy;
00006 import problem.definition.State;
00007 import problem.definition.Problem.ProblemType;
00008
00009
00013 public class SteadyStateReplace extends Replace {
00014
00015     @Override
00022     public List<State> replace(State stateCandidate, List<State> listState) {
00023         State stateREP = null;
00024         if (Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)) {
00025             stateREP = minValue(listState);
00026             if(stateCandidate.getEvaluation().get(0) >= stateREP.getEvaluation().get(0)){
00027                 Boolean find = false;
00028                 int count = 0;
00029                 while ((find.equals(false)) && (listState.size() > count)){
00030                     if(listState.get(count).equals(stateREP)){
00031                         listState.remove(count);
00032                         listState.add(count, stateCandidate);
00033                         find = true;
00034                     }
00035                     else count++;
00036                 }
00037             }
00038         }
00039         else {
00040
00041             if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Minimizar)){
00042                 stateREP = maxValue(listState);
00043                 if(stateCandidate.getEvaluation().get(0) <= stateREP.getEvaluation().get(0)){
00044                     Boolean find = false;
00045                     int count = 0;
00046                     while ((find.equals(false)) && (listState.size() > count)){
00047                         if(listState.get(count).equals(stateREP)){
00048                             listState.remove(count);
00049                             listState.add(count, stateCandidate);
00050                             find = true;
00051                         }
00052                     }
00053                 }
00054             }
00055         }
00056         return listState;
00057     }
00058
00064     public State minValue(List<State> listState) {
00065         State value = listState.get(0);
00066         double min = listState.get(0).getEvaluation().get(0);
00067         for (int i = 1; i < listState.size(); i++) {
00068             if (listState.get(i).getEvaluation().get(0) < min) {
00069                 min = listState.get(i).getEvaluation().get(0);
00070                 value = listState.get(i);
00071             }
00072         }
00073         return value;
00074     }
00075
00081     public State maxValue(List<State> listState) {

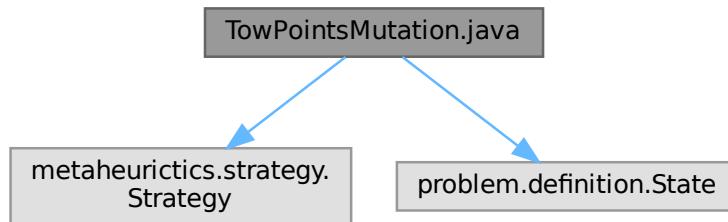
```

```

00082     State value = listState.get(0);
00083     double max = listState.get(0).getEvaluation().get(0);
00084     for (int i = 1; i < listState.size(); i++) {
00085         if (listState.get(i).getEvaluation().get(0) > max) {
00086             max = listState.get(i).getEvaluation().get(0);
00087             value = listState.get(i);
00088         }
00089     }
00090     return value;
00091 }
00092 }
```

## 9.47 Referencia del archivo TowPointsMutation.java

import metaheuristics.strategy.Strategy;  
 import problem.definition.State;  
 Gráfico de dependencias incluidas en TowPointsMutation.java:



### Clases

- class [evolutionary\\_algorithms.complement.TowPointsMutation](#)  
*TowPointsMutation* - applies two-points mutation.

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.48 TowPointsMutation.java

[Ir a la documentación de este archivo.](#)

```

00001 package evolutionary_algorithms.complement;
00002
00003
00004 import metaheuristics.strategy.Strategy;
00005 import problem.definition.State;
00009 public class TowPointsMutation extends Mutation {
00010
00011     // legacy ProblemState-based mutation removed: use State-based mutation below
00012
00013     @Override
00020     public State mutation(State newind, double PM) {
00021         Object key1 = Strategy.getStrategy().getProblem().getCodification().getAleatoryKey();
```

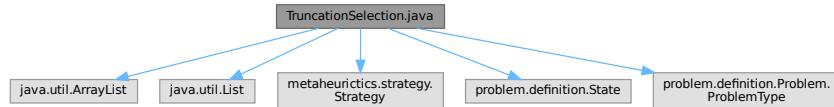
```

00022     Object key2 = Strategy.getStrategy().getProblem().getCodification().getAleatoryKey();
00023     Object value1 =
00024         Strategy.getStrategy().getProblem().getCodification().getVariableAleatoryValue((Integer) key1);
00025     Object value2 =
00026         Strategy.getStrategy().getProblem().getCodification().getVariableAleatoryValue((Integer) key2);
00027     newind.getCode().set((Integer) key1, (Integer) value2);
00028     newind.getCode().set((Integer) key2, (Integer) value1);
00029     return newind;
00030 }
```

## 9.49 Referencia del archivo TruncationSelection.java

```

import java.util.ArrayList;
import java.util.List;
import metaheuristics.strategy.Strategy;
import problem.definition.State;
import problem.definition.Problem.ProblemType;
Gráfico de dependencias incluidas en TruncationSelection.java:
```



### Clases

- class [evolutionary\\_algorithms.complement.TruncationSelection](#)  
*TruncationSelection - applies truncation selection strategy.*

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.50 TruncationSelection.java

Ir a la documentación de este archivo.

```

00001 package evolutionary_algorithms.complement;
00002
00003
00004 import java.util.ArrayList;
00005 import java.util.List;
00006
00007 import metaheuristics.strategy.Strategy;
00008
00009 import problem.definition.State;
00010 import problem.definition.Problem.ProblemType;
00011
00015 public class TruncationSelection extends FatherSelection {
00016
00022     public List<State> orderBetter (List<State> listState) {
00023         State var = null;
00024         for (int i = 0; i < listState.size()- 1; i++) {
00025             for (int j = i+1; j < listState.size(); j++) {
00026                 if(listState.get(i).getEvaluation().get(0) < listState.get(j).getEvaluation().get(0)) {
```

```

00027             var = listState.get(i);
00028             listState.set(i, listState.get(j));
00029             listState.set(j,var);
00030         }
00031     }
00032 }
00033 return listState;
00034 }
00035
00041 public List<State> ascOrderBetter (List<State> listState){
00042     State var = null;
00043     for (int i = 0; i < listState.size()- 1; i++) {
00044         for (int j = i+1; j < listState.size(); j++) {
00045             if(listState.get(i).getEvaluation().get(0) > listState.get(j).getEvaluation().get(0)) {
00046                 var = listState.get(i);
00047                 listState.set(i, listState.get(j));
00048                 listState.set(j,var);
00049             }
00050         }
00051     }
00052     return listState;
00053 }
00054
00055 @Override
00062 public List<State> selection(List<State> listState, int truncation) {
00063     List<State> AuxList = new ArrayList<State>();
00064     if (Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)) {
00065         listState = orderBetter(listState);
00066     } else {
00067         if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Minimizar))
00068             listState = ascOrderBetter(listState);
00069     }
00070     int i = 0;
00071     while(AuxList.size()< truncation){
00072         AuxList.add(listState.get(i));
00073         i++;
00074     }
00075     return AuxList;
00076 }
00077 }

```

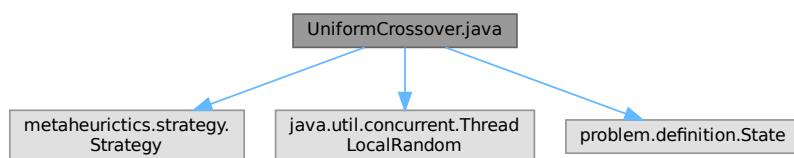
## 9.51 Referencia del archivo UniformCrossover.java

```

import metaheuristics.strategy.Strategy;
import java.util.concurrent.ThreadLocalRandom;
import problem.definition.State;

```

Gráfico de dependencias incluidas en UniformCrossover.java:



### Clases

- class [evolutionary\\_algorithms.complement.UniformCrossover](#)  
*UniformCrossover - complements uniform crossover strategy.*

### Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.52 UniformCrossover.java

[Ir a la documentación de este archivo.](#)

```

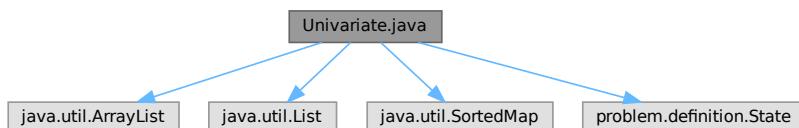
00001 package evolutionary_algorithms.complement;
00002
00003 import metaheuristics.strategy.Strategy;
00004 import java.util.concurrent.ThreadLocalRandom;
00005
00006 import problem.definition.State;
00007
00011 public class UniformCrossover extends Crossover {
00012
00019     @SuppressWarnings("squid:S2245")
00025     public int[] mascara(int length) {
00026         int[] mascara = new int[length];
00027         for (int i = 0; i < mascara.length; i++) {
00028             int value = ThreadLocalRandom.current().nextInt(2);
00029             mascara[i] = value;
00030         }
00031         return mascara;
00032     }
00033
00034     @Override
00042     public State crossover(State father1, State father2, double PC) {
00043         Object value;
00044         State state = (State) father1.getCopy();
00045         int[] mascara = mascara(father1.getCode().size());
00046         for (int k = 0; k < mascara.length; k++) {
00047             if(mascara[k] == 1){
00048                 value = father1.getCode().get(k);
00049                 state.getCode().set(k, value);
00050             }
00051             else{
00052                 if(mascara[k] == 0){
00053                     value = father2.getCode().get(k);
00054                     state.getCode().set(k, value);
00055                 }
00056             }
00057         }
00058         return state;
00059     }
00060 }
```

## 9.53 Referencia del archivo Univariate.java

```

import java.util.ArrayList;
import java.util.List;
import java.util.SortedMap;
import problem.definition.State;
```

Gráfico de dependencias incluidas en Univariate.java:



### Clases

- class [evolutionary\\_algorithms.complement.Univariate](#)  
*Univariate - applies univariate distribution strategy.*

## Paquetes

- package [evolutionary\\_algorithms.complement](#)

## 9.54 Univariate.java

[Ir a la documentación de este archivo.](#)

```

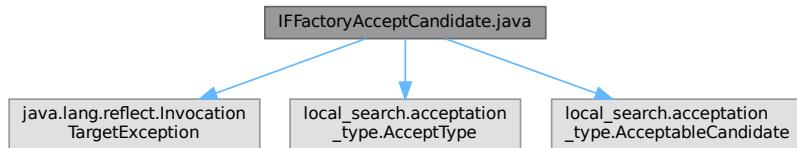
00001 package evolutionary_algorithms.complement;
00002
00003 import java.util.ArrayList;
00004 import java.util.List;
00005 import java.util.SortedMap;
00006
00007 import problem.definition.State;
00008
00009
00010
00014 public class Univariate extends Distribution {
00015
00016     @Override
00022     public List<Probability> distribution(List<State> fathers) {
00023
00024         List<Probability> ListProbability = new ArrayList<Probability>();
00025         int cantV = fathers.get(0).getCode().size();
00026         for (int i = 0; i < cantV; i++) {
00027             int[] values = new int[fathers.size()];
00028             //llenar los valores de cada variable
00029             for (int j = 0; j < fathers.size(); j++) {
00030                 values[j] = (Integer) fathers.get(j).getCode().get(i);
00031             }
00032             //para cada valor contar la cantidad de veces que se repite
00033             int k = 0;
00034             while (k < values.length) {
00035                 float count = 0;
00036                 if(values[k] != -1){
00037                     int l = k;
00038                     int temp = values[k];
00039                     while (l < values.length) {
00040                         if(temp == values[l]){
00041                             count++;
00042                             values[l] = -1;
00043                         }
00044                         l++;
00045                     }
00046                     Probability probability = new Probability();
00047                     float prob = count / values.length;
00048                     probability.setKey(i);
00049                     probability.setValue(temp);
00050                     probability.setProbability(prob);
00051                     ListProbability.add(probability);
00052                 }
00053                 k++;
00054             }
00055         }
00056         return ListProbability;
00057     }
00058
00065     public List<String> getListKey(SortedMap<String, Object> map) {
00066         List<String> listKey = new ArrayList<String>();
00067         String key = map.keySet().toString();
00068         String returnString = key.substring(1, key.length() - 1);
00069         returnString = returnString + ", ";
00070         int countKey = map.size();
00071         for (int j = 0; j < countKey; j++) {
00072             String r = returnString.substring(0, returnString.indexOf(','));
00073             returnString = returnString.substring(returnString.indexOf(',')+2);
00074             listKey.add(r);
00075         }
00076         return listKey;
00077     }
00078 }
```

## 9.55 Referencia del archivo IFFactoryAcceptCandidate.java

```
import java.lang.reflect.InvocationTargetException;
import local_search.acceptation_type.AcceptType;
```

```
import local_search.acceptation_type.AcceptableCandidate;
```

Gráfico de dependencias incluidas en IFFactoryAcceptCandidate.java:



## Clases

- interface [factory\\_interface.IFFactoryAcceptCandidate](#)  
*IFFactoryAcceptCandidate - Interface for creating acceptable candidates.*

## Paquetes

- package [factory\\_interface](#)  
`@(#) IFFactoryAcceptCandidate.java`

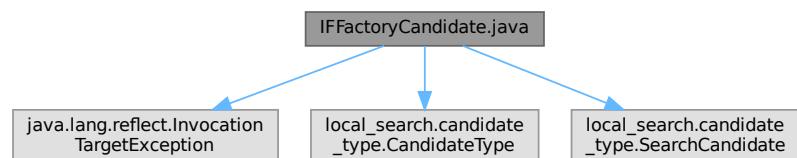
## 9.56 IFFactoryAcceptCandidate.java

[Ir a la documentación de este archivo.](#)

```
00001
00004
00005 package factory_interface;
00006
00007 import java.lang.reflect.InvocationTargetException;
00008
00009 import local_search.acceptation_type.AcceptType;
00010 import local_search.acceptation_type.AcceptableCandidate;
00011
00015 public interface IFFactoryAcceptCandidate{
00016     AcceptableCandidate createAcceptCandidate(AcceptType typeacceptation) throws
        IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
        IllegalAccessException, InvocationTargetException, NoSuchMethodException;
00017
00018 }
```

## 9.57 Referencia del archivo IFFactoryCandidate.java

```
import java.lang.reflect.InvocationTargetException;
import local_search.candidate_type.CandidateType;
import local_search.candidate_type.SearchCandidate;
Gráfico de dependencias incluidas en IFFactoryCandidate.java:
```



## Clases

- interface [factory\\_interface.IFFactoryCandidate](#)  
*IFFactoryCandidate* - Interface for creating search candidates.

## Paquetes

- package [factory\\_interface](#)  
*@(#)* [IFFactoryAcceptCandidate.java](#)

## 9.58 IFFactoryCandidate.java

[Ir a la documentación de este archivo.](#)

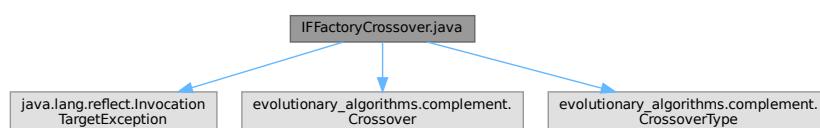
```

00001
00004
00005 package factory_interface;
00006
00007 import java.lang.reflect.InvocationTargetException;
00008
00009 import local_search.candidate_type.CandidateType;
00010 import local_search.candidate_type.SearchCandidate;
00011
00012
00013
00014
00018 public interface IFFactoryCandidate
00019 {
00020     SearchCandidate createSearchCandidate(CandidateType typeCandidate) throws
        IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
        IllegalAccessException, InvocationTargetException, NoSuchMethodException;
00021 }
```

## 9.59 Referencia del archivo IFFactoryCrossover.java

```

import java.lang.reflect.InvocationTargetException;
import evolutionary_algorithms.complement.Crossover;
import evolutionary_algorithms.complement.CrossoverType;
Gráfico de dependencias incluidas en IFFactoryCrossover.java:
```



## Clases

- interface [factory\\_interface.IFFactoryCrossover](#)  
*IFFactoryCrossover* - Interface for creating crossover operators.

## **Paquetes**

- package factory\_interface  
    @(#) IFFactoryAcceptCandidate.java

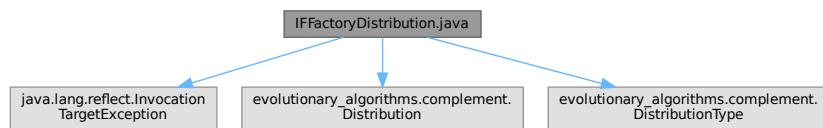
## 9.60 IFFactoryCrossover.java

[Ir a la documentación de este archivo.](#)

```
00001 package factory_interface;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005 import evolutionary_algorithms.complement.Crossover;
00006 import evolutionary_algorithms.complement.CrossoverType;
00007
00008
00009
00010
00014 public interface IFFactoryCrossover {
00015     Crossover createCrossover(CrossoverType CrossoverType) throws IllegalArgumentException,
SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
InvocationTargetException, NoSuchMethodException ;
00016 }
```

## 9.61 Referencia del archivo IFFactoryDistribution.java

```
import java.lang.reflect.InvocationTargetException;
import evolutionary_algorithms.complement.Distribution;
import evolutionary_algorithms.complement.DistributionType;
Gráfico de dependencias incluidas en IFFactoryDistribution.java:
```



## Clases

- interface `factory_interface.IFFactoryDistribution`  
*IFFactoryDistribution* - Interface for creating distribution strategies.

## **Paquetes**

- package `factory_interface`  
    @(#)*IFFactoryAcceptCandidate.java*

## 9.62 IFFactoryDistribution.java

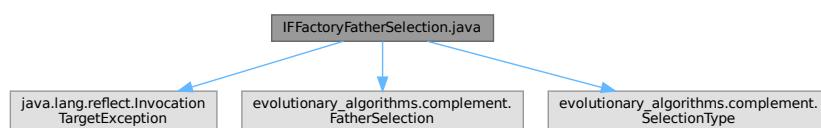
[Ir a la documentación de este archivo.](#)

```
00001 package factory_interface;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005 import evolutionary_algorithms.complement.Distribution;
00006 import evolutionary_algorithms.complement.DistributionType;
00007
00008
00009
00010
00014 public interface IFFactoryDistribution {
00015     Distribution createDistribution(DistributionType typedistribution) throws
00016         IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00017         IllegalAccessException, InvocationTargetException, NoSuchMethodException;
00016 }
```

## 9.63 Referencia del archivo IFFactoryFatherSelection.java

```
import java.lang.reflect.InvocationTargetException;
import evolutionary_algorithms.complement.FatherSelection;
import evolutionary_algorithms.complement.SelectionType;
```

Gráfico de dependencias incluidas en IFFactoryFatherSelection.java:



### Clases

- interface [factory\\_interface.IFFactoryFatherSelection](#)  
*IFFactoryFatherSelection - Interface for creating father selection strategies.*

### Paquetes

- package [factory\\_interface](#)  
*@(#)* [IFFactoryAcceptCandidate.java](#)

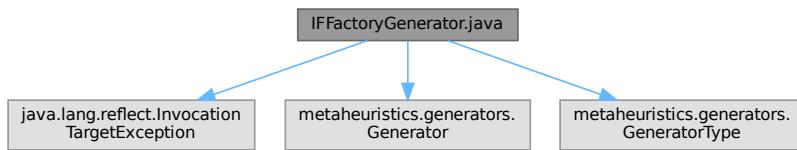
## 9.64 IFFactoryFatherSelection.java

[Ir a la documentación de este archivo.](#)

```
00001 package factory_interface;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005 import evolutionary_algorithms.complement.FatherSelection;
00006 import evolutionary_algorithms.complement.SelectionType;
00007
00008
00009
00010
00014 public interface IFFactoryFatherSelection {
00015     FatherSelection createSelectFather(SelectionType selectionType) throws IllegalArgumentException,
00016         SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
00017         InvocationTargetException, NoSuchMethodException ;
00017 }
```

## 9.65 Referencia del archivo IFFactoryGenerator.java

```
import java.lang.reflect.InvocationTargetException;
import metaheuristics.generators.Generator;
import metaheuristics.generators.GeneratorType;
Gráfico de dependencias incluidas en IFFactoryGenerator.java:
```



### Clases

- interface [factory\\_interface.IFFactoryGenerator](#)  
*IFFactoryGenerator - Interface for creating generator instances.*

### Paquetes

- package [factory\\_interface](#)  
`@(#)` [IFFactoryAcceptCandidate.java](#)

## 9.66 IFFactoryGenerator.java

[Ir a la documentación de este archivo.](#)

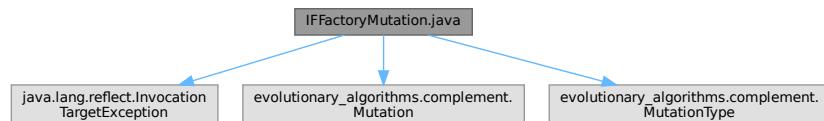
```
00001 package factory_interface;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005 import metaheuristics.generators.Generator;
00006 import metaheuristics.generators.GeneratorType;
00007
00011 public interface IFFactoryGenerator {
00012
00013     Generator createGenerator(GeneratorType Generatortype) throws IllegalArgumentException,
00014         SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
00015         InvocationTargetException, NoSuchMethodException ;
00014 }
```

## 9.67 Referencia del archivo IFFactoryMutation.java

```
import java.lang.reflect.InvocationTargetException;
import evolutionary_algorithms.complement.Mutation;
```

```
import evolutionary_algorithms.complement.MutationType;
```

Gráfico de dependencias incluidas en IFFactoryMutation.java:



## Clases

- interface [factory\\_interface.IFFactoryMutation](#)  
*IFFactoryMutation - Interface for creating mutation strategies.*

## Paquetes

- package [factory\\_interface](#)  
`@(#) IFFactoryAcceptCandidate.java`

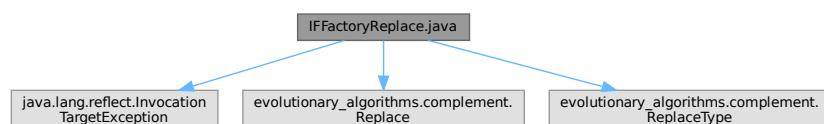
## 9.68 IFFactoryMutation.java

[Ir a la documentación de este archivo.](#)

```
00001 package factory_interface;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005 import evolutionary_algorithms.complement.Mutation;
00006 import evolutionary_algorithms.complement.MutationType;
00007
00008
00009
00010
00014 public interface IFFactoryMutation {
00015     Mutation createMutation(MutationType typeMutation) throws IllegalArgumentException,
00016         SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
00017         InvocationTargetException, NoSuchMethodException ;
00016 }
```

## 9.69 Referencia del archivo IFFactoryReplace.java

```
import java.lang.reflect.InvocationTargetException;
import evolutionary_algorithms.complement.Replace;
import evolutionary_algorithms.complement.ReplaceType;
Gráfico de dependencias incluidas en IFFactoryReplace.java:
```



## Clases

- interface `factory_interface.IFFactoryReplace`

*IFFactoryReplace - Interface for creating replacement strategies.*

## Paquetes

- package `factory_interface`  
`@(#)` *IFFactoryAcceptCandidate.java*

## 9.70 IFFactoryReplace.java

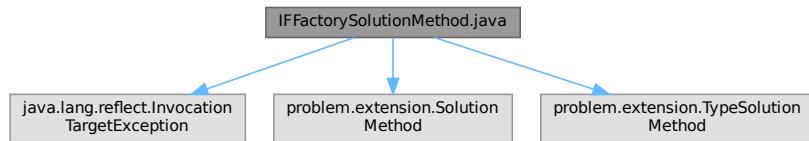
[Ir a la documentación de este archivo.](#)

```
00001 package factory_interface;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005 import evolutionary_algorithms.complement.Replace;
00006 import evolutionary_algorithms.complement.ReplaceType;
00007
00008
00009
00010
00014 public interface IFFactoryReplace {
00015     Replace createReplace(ReplaceType typereplace) throws IllegalArgumentException, SecurityException,
00016     ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00017     NoSuchMethodException ;
00016 }
```

## 9.71 Referencia del archivo IFFactorySolutionMethod.java

```
import java.lang.reflect.InvocationTargetException;
import problem.extension.SolutionMethod;
import problem.extension.TypeSolutionMethod;
```

Gráfico de dependencias incluidas en IFFactorySolutionMethod.java:



## Clases

- interface `factory_interface.IFFactorySolutionMethod`

*IFFactorySolutionMethod - Interface for creating solution methods.*

## Paquetes

- package `factory_interface`  
`@(#)` *IFFactoryAcceptCandidate.java*

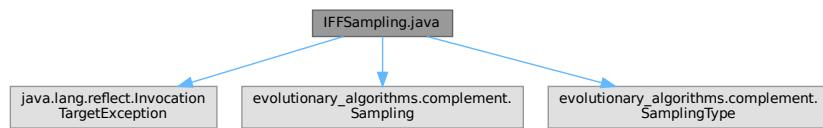
## 9.72 IFFactorySolutionMethod.java

[Ir a la documentación de este archivo.](#)

```
00001 package factory_interface;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005 import problem.extension.SolutionMethod;
00006 import problem.extension.TypeSolutionMethod;
00007
00011 public interface IFFactorySolutionMethod {
00012
00013     SolutionMethod createdSolutionMethod(TypeSolutionMethod method) throws IllegalArgumentException,
00014         SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
00015         InvocationTargetException, NoSuchMethodException ;
00014
00015 }
```

## 9.73 Referencia del archivo IFFSampling.java

```
import java.lang.reflect.InvocationTargetException;
import evolutionary_algorithms.complement.Sampling;
import evolutionary_algorithms.complement.SamplingType;
Gráfico de dependencias incluidas en IFFSampling.java:
```



### Clases

- interface [factory\\_interface.IFFSampling](#)  
*IFFSampling - Interface for creating sampling strategies.*

### Paquetes

- package [factory\\_interface](#)  
`@(#)` [IFFFactoryAcceptCandidate.java](#)

## 9.74 IFFSampling.java

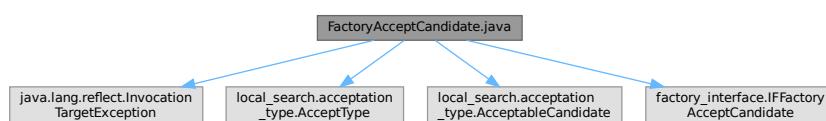
[Ir a la documentación de este archivo.](#)

```
00001 package factory_interface;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005 import evolutionary_algorithms.complement.Sampling;
00006 import evolutionary_algorithms.complement.SamplingType;
00007
00008
00009
00010
00014 public interface IFFSampling {
00015     Sampling createSampling(SamplingType typesampling) throws IllegalArgumentException,
00016         SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
00017         InvocationTargetException, NoSuchMethodException;
00016 }
```

## 9.75 Referencia del archivo FactoryAcceptCandidate.java

```
import java.lang.reflect.InvocationTargetException;
import local_search.acceptation_type.AcceptType;
import local_search.acceptation_type.AcceptableCandidate;
import factory_interface.IFFactoryAcceptCandidate;
```

Gráfico de dependencias incluidas en FactoryAcceptCandidate.java:



### Clases

- class [factory\\_method.FactoryAcceptCandidate](#)  
*FactoryAcceptCandidate - Interface for creating acceptable candidates.*

### Paquetes

- package [factory\\_method](#)  
`@(#) FactoryAcceptCandidate.java`

## 9.76 FactoryAcceptCandidate.java

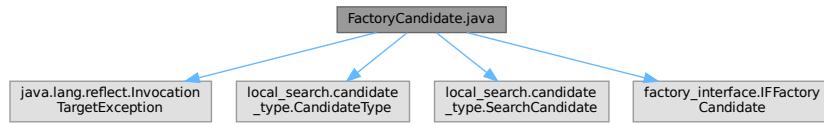
[Ir a la documentación de este archivo.](#)

```
00001
00004
00005 package factory_method;
00006
00007 import java.lang.reflect.InvocationTargetException;
00008
00009 import local_search.acceptation_type.AcceptType;
00010 import local_search.acceptation_type.AcceptableCandidate;
00011
00012
00013 import factory_interface.IFFactoryAcceptCandidate;
00014
00015
00016
00017
00021 public class FactoryAcceptCandidate implements IFFactoryAcceptCandidate{
00022     private AcceptableCandidate acceptCandidate;
00023
00029     public AcceptableCandidate createAcceptCandidate( AcceptType typeacceptation ) throws
        IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
        IllegalAccessException, InvocationTargetException, NoSuchMethodException{
00030         String className = "local_search.acceptation_type." + typeacceptation.toString();
00031         acceptCandidate = (AcceptableCandidate) FactoryLoader.getInstance(className);
00032         return acceptCandidate;
00033     }
00034 }
```

## 9.77 Referencia del archivo FactoryCandidate.java

```
import java.lang.reflect.InvocationTargetException;
import local_search.candidate_type.CandidateType;
import local_search.candidate_type.SearchCandidate;
import factory_interface.IFFactoryCandidate;
```

Gráfico de dependencias incluidas en FactoryCandidate.java:



### Clases

- class [factory\\_method.FactoryCandidate](#)  
*FactoryCandidate - Interface for creating search candidates.*

### Paquetes

- package [factory\\_method](#)  
*@(#)* [FactoryAcceptCandidate.java](#)

## 9.78 FactoryCandidate.java

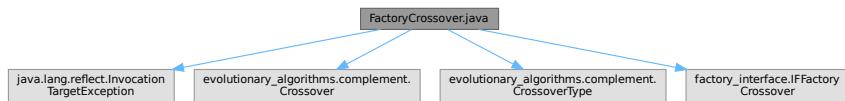
[Ir a la documentación de este archivo.](#)

```
00001
00004 package factory_method;
00005
00006 import java.lang.reflect.InvocationTargetException;
00007
00008 import local_search.candidate_type.CandidateType;
00009 import local_search.candidate_type.SearchCandidate;
00010
00011 import factory_interface.IFFactoryCandidate;
00012
00013
00014
00018 public class FactoryCandidate implements IFFactoryCandidate{
00019     private SearchCandidate searchcandidate;
00020
00026     public SearchCandidate createSearchCandidate(CandidateType typeCandidate) throws
        IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
        IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00027         String className = "local_search.candidate_type." + typeCandidate.toString();
00028         searchcandidate = (SearchCandidate) FactoryLoader.getInstance(className);
00029         return searchcandidate;
00030     }
00031 }
```

## 9.79 Referencia del archivo FactoryCrossover.java

```
import java.lang.reflect.InvocationTargetException;
import evolutionary_algorithms.complement.Crossover;
import evolutionary_algorithms.complement.CrossoverType;
import factory_interface.IFFactoryCrossover;
```

Gráfico de dependencias incluidas en FactoryCrossover.java:



### Clases

- class [factory\\_method.FactoryCrossover](#)  
*FactoryCrossover - Interface for creating crossover strategies.*

### Paquetes

- package [factory\\_method](#)  
`@(#) FactoryAcceptCandidate.java`

## 9.80 FactoryCrossover.java

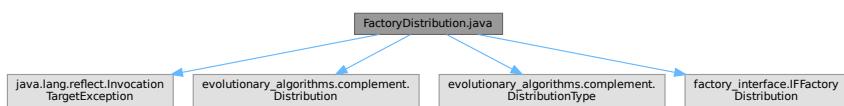
[Ir a la documentación de este archivo.](#)

```
00001 package factory_method;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005
00006 import evolutionary_algorithms.complement.Crossover;
00007 import evolutionary_algorithms.complement.CrossoverType;
00008 import factory_interface.IFFactoryCrossover;
00009
00010
00011
00012
00016 public class FactoryCrossover implements IFFactoryCrossover {
00017     private Crossover crossing;
00018
00024     public Crossover createCrossover(CrossoverType CrossoverType) throws IllegalArgumentException,
00025         SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
00026         InvocationTargetException, NoSuchMethodException {
00027
00028         String className = "evolutionary_algorithms.complement." + CrossoverType.toString();
00029         crossing = (Crossover) FactoryLoader.getInstance(className);
00030     }
00030 }
```

## 9.81 Referencia del archivo FactoryDistribution.java

```
import java.lang.reflect.InvocationTargetException;
import evolutionary_algorithms.complement.Distribution;
import evolutionary_algorithms.complement.DistributionType;
import factory_interface.IFFactoryDistribution;
```

Gráfico de dependencias incluidas en FactoryDistribution.java:



### Clases

- class [factory\\_method.FactoryDistribution](#)  
*FactoryDistribution - Interface for creating distribution strategies.*

### Paquetes

- package [factory\\_method](#)  
`@(#) FactoryAcceptCandidate.java`

## 9.82 FactoryDistribution.java

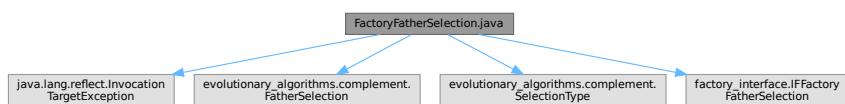
[Ir a la documentación de este archivo.](#)

```
00001 package factory_method;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005 import evolutionary_algorithms.complement.Distribution;
00006 import evolutionary_algorithms.complement.DistributionType;
00007 import factory_interface.IFFactoryDistribution;
00008
00012 public class FactoryDistribution implements IFFactoryDistribution {
00013     private Distribution distribution;
00014
00020     public Distribution createDistribution(DistributionType distributiontype) throws
00021         IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00022         IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00023
00024         String className = "evolutionary_algorithms.complement." + distributiontype.toString();
00025         distribution = (Distribution) FactoryLoader.getInstance(className);
00026         return distribution;
00027     }
00028 }
```

## 9.83 Referencia del archivo FactoryFatherSelection.java

```
import java.lang.reflect.InvocationTargetException;
import evolutionary_algorithms.complement.FatherSelection;
import evolutionary_algorithms.complement.SelectionType;
import factory_interface.IFFactoryFatherSelection;
```

Gráfico de dependencias incluidas en FactoryFatherSelection.java:



### Clases

- class [factory\\_method.FactoryFatherSelection](#)  
*FactoryFatherSelection - Interface for creating father selection strategies.*

### Paquetes

- package [factory\\_method](#)  
`@(#) FactoryAcceptCandidate.java`

## 9.84 FactoryFatherSelection.java

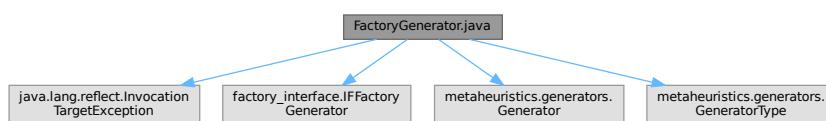
[Ir a la documentación de este archivo.](#)

```
00001 package factory_method;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005
00006 import evolutionary_algorithms.complement.FatherSelection;
00007 import evolutionary_algorithms.complement.SelectionType;
00008 import factory_interface.IFFactoryFatherSelection;
00009
00010
00011
00012
00016 public class FactoryFatherSelection implements IFFactoryFatherSelection{
00017     private FatherSelection selection;
00018
00024     public FatherSelection createSelectFather(SelectionType selectionType) throws
00025         IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00026         IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00027         String className = "evolutionary_algorithms.complement." + selectionType.toString();
00028         selection = (FatherSelection) FactoryLoader.getInstance(className);
00029     }
00029 }
```

## 9.85 Referencia del archivo FactoryGenerator.java

```
import java.lang.reflect.InvocationTargetException;
import factory_interface.IFFactoryGenerator;
import metaheuristics.generators.Generator;
import metaheuristics.generators.GeneratorType;
```

Gráfico de dependencias incluidas en FactoryGenerator.java:



### Clases

- class [factory\\_method.FactoryGenerator](#)  
*FactoryGenerator - Interface for creating generator strategies.*

### Paquetes

- package [factory\\_method](#)  
`@(#) FactoryAcceptCandidate.java`

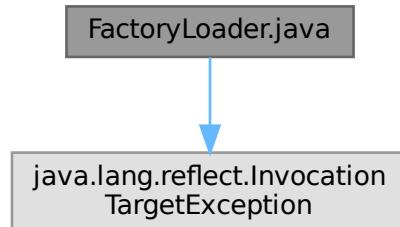
## 9.86 FactoryGenerator.java

[Ir a la documentación de este archivo.](#)

```
00001 package factory_method;
00002
00003
00004 import java.lang.reflect.InvocationTargetException;
00005
00006 import factory_interface.IFFactoryGenerator;
00007
00008 import metaheuristics.generators.Generator;
00009 import metaheuristics.generators.GeneratorType;
00010
00011
00012
00016 public class FactoryGenerator implements IFFactoryGenerator {
00017
00018     private Generator generator;
00019
00025     public Generator createGenerator(GeneratorType generatorType) throws IllegalArgumentException,
00026         SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
00027         InvocationTargetException, NoSuchMethodException {
00028
00029         String className = "metaheuristics.generators." + generatorType.toString();
00030         generator = (Generator) FactoryLoader.getInstance(className);
00031
00032     }
00033 }
```

## 9.87 Referencia del archivo FactoryLoader.java

import java.lang.reflect.InvocationTargetException;  
Gráfico de dependencias incluidas en FactoryLoader.java:



### Clases

- class [factory\\_method.FactoryLoader](#)  
*FactoryLoader - Class responsible for loading factory classes.*

### Paquetes

- package [factory\\_method](#)  
    [@\(#\) FactoryAcceptCandidate.java](#)

## 9.88 FactoryLoader.java

[Ir a la documentación de este archivo.](#)

```
00001 package factory_method;
00002 import java.lang.reflect.InvocationTargetException;
00003
00004
00005 public class FactoryLoader {
00006     public static Object getInstance(String className) throws ClassNotFoundException,
00007         IllegalArgumentException, SecurityException, InstantiationException, IllegalAccessException,
00008         InvocationTargetException, NoSuchMethodException{
00009         // Let exceptions propagate to caller instead of swallowing them (prevents null dereference)
00010         @SuppressWarnings("rawtypes")
00011         Class c = Class.forName(className);
00012         // use getDeclaredConstructor().newInstance() to avoid deprecated Class.newInstance()
00013         Object o = c.getDeclaredConstructor().newInstance();
00014         return o;
00015     }
00016 }
```

## 9.89 Referencia del archivo FactoryMutation.java

```
import java.lang.reflect.InvocationTargetException;
import evolutionary_algorithms.complement.Mutation;
import evolutionary_algorithms.complement.MutationType;
import factory_interface.IFFactoryMutation;
```

Gráfico de dependencias incluidas en FactoryMutation.java:



### Clases

- class [factory\\_method.FactoryMutation](#)

*FactoryMutation - Class responsible for creating mutation strategies.*

### Paquetes

- package [factory\\_method](#)  
`@(#) FactoryAcceptCandidate.java`

## 9.90 FactoryMutation.java

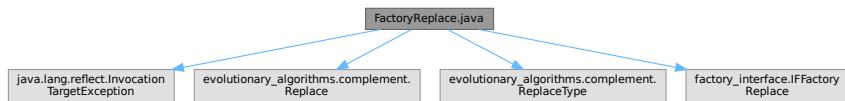
[Ir a la documentación de este archivo.](#)

```
00001 package factory_method;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005
00006 import evolutionary_algorithms.complement.Mutation;
00007 import evolutionary_algorithms.complement.MutationType;
00008 import factory_interface.IFFactoryMutation;
00009
00010
00011
00012
00016 public class FactoryMutation implements IFFactoryMutation {
00017     private Mutation mutation;
00018
00024     public Mutation createMutation(MutationType typeMutation) throws IllegalArgumentException,
00025         SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
00026         InvocationTargetException, NoSuchMethodException {
00027
00028         String className = "evolutionary_algorithms.complement." + typeMutation.toString();
00029         mutation = (Mutation) FactoryLoader.getInstance(className);
00030     }
}
```

## 9.91 Referencia del archivo FactoryReplace.java

```
import java.lang.reflect.InvocationTargetException;
import evolutionary_algorithms.complement.Replace;
import evolutionary_algorithms.complement.ReplaceType;
import factory_interface.IFFactoryReplace;
```

Gráfico de dependencias incluidas en FactoryReplace.java:



### Clases

- class [factory\\_method.FactoryReplace](#)

*FactoryReplace - Class responsible for creating replacement strategies.*

### Paquetes

- package [factory\\_method](#)  
`@(#) FactoryAcceptCandidate.java`

## 9.92 FactoryReplace.java

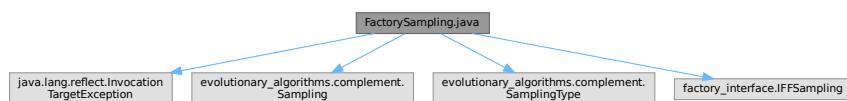
[Ir a la documentación de este archivo.](#)

```
00001 package factory_method;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005
00006 import evolutionary_algorithms.complement.Replace;
00007 import evolutionary_algorithms.complement.ReplaceType;
00008 import factory_interface.IFFactoryReplace;
00009
00010
00011
00012
00013
00017 public class FactoryReplace implements IFFactoryReplace {
00018
00019     private Replace replace;
00020
00026     public Replace createReplace( ReplaceType typereplace ) throws IllegalArgumentException,
SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
InvocationTargetException, NoSuchMethodException{
00027         String className = "evolutionary_algorithms.complement." + typereplace.toString();
00028         replace = (Replace) FactoryLoader.getInstance(className);
00029         return replace;
00030     }
00031 }
```

## 9.93 Referencia del archivo FactorySampling.java

```
import java.lang.reflect.InvocationTargetException;
import evolutionary_algorithms.complement.Sampling;
import evolutionary_algorithms.complement.SamplingType;
import factory_interface.IFFSampling;
```

Gráfico de dependencias incluidas en FactorySampling.java:



### Clases

- class [factory\\_method.FactorySampling](#)  
*FactorySampling - Class responsible for creating sampling strategies.*

### Paquetes

- package [factory\\_method](#)  
*@(#) FactoryAcceptCandidate.java*

## 9.94 FactorySampling.java

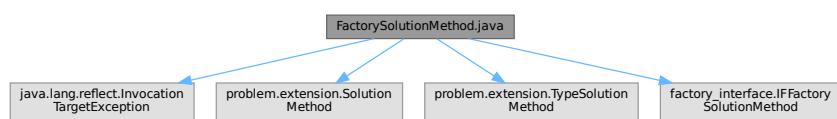
[Ir a la documentación de este archivo.](#)

```
00001 package factory_method;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005
00006 import evolutionary_algorithms.complement.Sampling;
00007 import evolutionary_algorithms.complement.SamplingType;
00008 import factory_interface.IFFSampling;
00009
00010
00011
00012
00016 public class FactorySampling implements IFFSampling {
00017     private Sampling sampling;
00023     public Sampling createSampling(SamplingType typesampling) throws IllegalArgumentException,
SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
InvocationTargetException, NoSuchMethodException {
00024         String className = "evolutionary_algorithms.complement." + typesampling.toString();
00026         sampling = (Sampling) FactoryLoader.getInstance(className);
00027         return sampling;
00028     }
00029 }
```

## 9.95 Referencia del archivo FactorySolutionMethod.java

```
import java.lang.reflect.InvocationTargetException;
import problem.extension.SolutionMethod;
import problem.extension.TypeSolutionMethod;
import factory_interface.IFFactorySolutionMethod;
```

Gráfico de dependencias incluidas en FactorySolutionMethod.java:



### Clases

- class [factory\\_method.FactorySolutionMethod](#)

*FactorySolutionMethod - Class responsible for creating solution method strategies.*

### Paquetes

- package [factory\\_method](#)
- @(#) FactoryAcceptCandidate.java*

## 9.96 FactorySolutionMethod.java

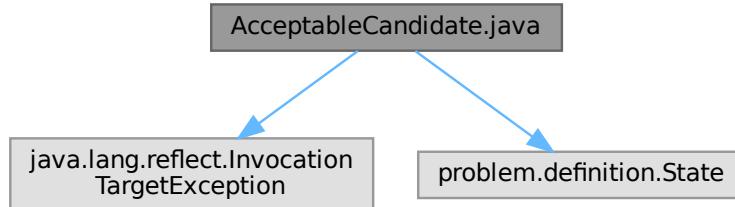
[Ir a la documentación de este archivo.](#)

```
00001 package factory_method;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005 import problem.extension.SolutionMethod;
00006 import problem.extension.TypeSolutionMethod;
00007 import factory_interface.IFFactorySolutionMethod;
00008
00012 public class FactorySolutionMethod implements IFFactorySolutionMethod {
00013
00014     private SolutionMethod solutionMethod;
00015
00016     @Override
00022     public SolutionMethod createdSolutionMethod(TypeSolutionMethod method) throws
IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00023         String className = "problem.extension." + method.toString();
00024         solutionMethod = (SolutionMethod) FactoryLoader.getInstance(className);
00025         return solutionMethod;
00026     }
00027
00028 }
```

## 9.97 Referencia del archivo AcceptableCandidate.java

```
import java.lang.reflect.InvocationTargetException;
import problem.definition.State;
```

Gráfico de dependencias incluidas en AcceptableCandidate.java:



### Clases

- class [local\\_search.acceptation\\_type.AcceptableCandidate](#)  
*AcceptableCandidate - Base type for acceptation strategies.*

### Paquetes

- package [local\\_search.acceptation\\_type](#)  
*@(#)* [AcceptAnyone.java](#)

## 9.98 AcceptableCandidate.java

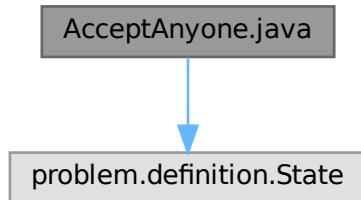
[Ir a la documentación de este archivo.](#)

```
00001 package local_search.acceptation_type;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004
00005 import problem.definition.State;
00006
00012 public abstract class AcceptableCandidate {
00013
00028     public abstract Boolean acceptCandidate(State stateCurrent, State stateCandidate) throws
        IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
        IllegalAccessException, InvocationTargetException, NoSuchMethodException ;
00029 }
```

## 9.99 Referencia del archivo AcceptAnyone.java

```
import problem.definition.State;
```

Gráfico de dependencias incluidas en AcceptAnyone.java:



### Clases

- class [local\\_search.acceptation\\_type.AcceptAnyone](#)  
*AcceptAnyone - Accept any candidate solution.*

### Paquetes

- package [local\\_search.acceptation\\_type](#)  
  *@(#)* [AcceptAnyone.java](#)

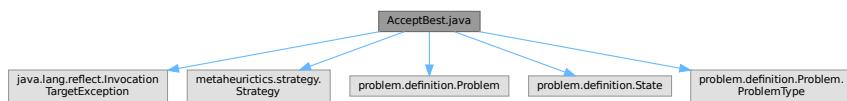
## 9.100 AcceptAnyone.java

[Ir a la documentación de este archivo.](#)

```
00001
00004 package local_search.acceptation_type;
00005
00006 import problem.definition.State;
00007
00008
00015 public class AcceptAnyone extends AcceptableCandidate{
00016
00017     @Override
00025     public Boolean acceptCandidate(State stateCurrent, State stateCandidate) {
00026         Boolean accept = true;
00027         return accept;
00028     }
00029
00030 }
```

## 9.101 Referencia del archivo AcceptBest.java

```
import java.lang.reflect.InvocationTargetException;
import metaheuristics.strategy.Strategy;
import problem.definition.Problem;
import problem.definition.State;
import problem.definition.Problem.ProblemType;
Gráfico de dependencias incluidas en AcceptBest.java:
```



### Clases

- class [local\\_search.acceptation\\_type.AcceptBest](#)  
`AcceptBest` - Accept the candidate only if it improves (or ties) the current state.

### Paquetes

- package [local\\_search.acceptation\\_type](#)  
`@(#)` [AcceptAnyone.java](#)

## 9.102 AcceptBest.java

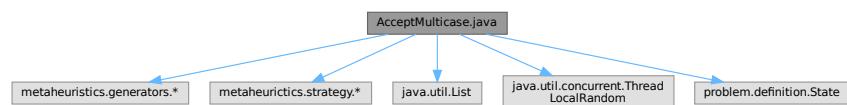
[Ir a la documentación de este archivo.](#)

```
00001
00004
00005 package local_search.acceptation_type;
00006
00007 import java.lang.reflect.InvocationTargetException;
00008
00009 import metaheuristics.strategy.Strategy;
00010 import problem.definition.Problem;
00011 import problem.definition.State;
00012 import problem.definition.Problem.ProblemType;
00013
00020 public class AcceptBest extends AcceptableCandidate {
00021
00022     @Override
00038     public Boolean acceptCandidate(State stateCurrent, State stateCandidate) throws
IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00039         Boolean accept = null;
00040         Problem problem = Strategy.getStrategy().getProblem();
00041         if(problem.getTypeProblem().equals(ProblemType.Maximizar)) {
00042             if (stateCandidate.getEvaluation().get(0) >= stateCurrent.getEvaluation().get(0)) {
00043                 accept = true;
00044             } else {
00045                 accept = false;
00046             }
00047         } else {
00048             if (stateCandidate.getEvaluation().get(0) <= stateCurrent.getEvaluation().get(0)) {
00049                 accept = true;
00050             } else {
00051                 accept = false;
00052             }
00053         }
00054     return accept;
00055 }
00056 }
```

## 9.103 Referencia del archivo AcceptMulticase.java

```
import metaheuristics.generators.*;
import metaheuristics.strategy.*;
import java.util.List;
import java.util.concurrent.ThreadLocalRandom;
import problem.definition.State;
```

Gráfico de dependencias incluidas en AcceptMulticase.java:



### Clases

- class [local\\_search.accepatation\\_type.AcceptMulticase](#)  
*AcceptMulticase - Multi-case acceptance strategy for multiobjective algorithms.*

### Paquetes

- package [local\\_search.accepatation\\_type](#)  
*@(#) AcceptAnyone.java*

## 9.104 AcceptMulticase.java

[Ir a la documentación de este archivo.](#)

```
00001 package local_search.accepatation_type;
00002
00003 import metaheuristics.generators.*;
00004 import metaheuristics.strategy.*;
00005
00006 import java.util.List;
00007 import java.util.concurrent.ThreadLocalRandom;
00008
00009 import problem.definition.State;
0010
0017 public class AcceptMulticase extends AcceptableCandidate {
0024     @SuppressWarnings("squid:S2245")
0025     @Override
0034     public Boolean acceptCandidate(State stateCurrent, State stateCandidate) {
0035         // TODO Auto-generated method stub
0036         Boolean accept = false;
0037         List<State> list = Strategy.getStrategy().listRefPoblacFinal;
0038
0039         if(list.size() == 0){
0040             list.add(stateCurrent.clone());
0041         }
0042         Double T = MultiCaseSimulatedAnnealing.getTinitial();
0043         double pAccept = 0;
0044         // use ThreadLocalRandom to avoid creating a new Random instance on each call
0045         Dominance dominance= new Dominance();
0046         //Verificando si la soluci n candidata domina a la soluci n actual
0047         //Si la soluci n candidata domina a la soluci n actual
0048         if(dominance.dominance(stateCandidate, stateCurrent) == true){
0049             //Se asigna como soluci n actual la soluci n candidata con probabilidad 1
0050             pAccept = 1;
0051         }
0052         else if(dominance.dominance(stateCandidate, stateCurrent)== false){
0053             if(dominanceCounter(stateCandidate, list) > 0){
```

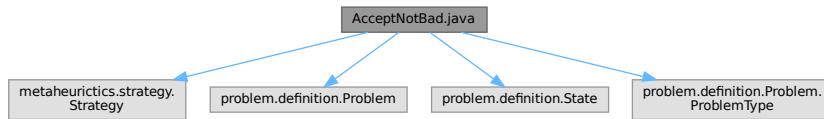
```

00054         pAccept = 1;
00055     }
00056     else if(dominanceRank(stateCandidate, list) == 0){
00057         pAccept = 1;
00058     }
00059     else if(dominanceRank(stateCandidate, list) < dominanceRank(stateCurrent, list)){
00060         pAccept = 1;
00061     }
00062     else if(dominanceRank(stateCandidate, list) == dominanceRank(stateCurrent, list)){
00063         //Calculando la probabilidad de aceptaci&on
00064         List<Double> evaluations = stateCurrent.getEvaluation();
00065         double total = 0;
00066         for (int i = 0; i < evaluations.size()-1; i++) {
00067             Double evalA = evaluations.get(i);
00068             Double evalB = stateCandidate.getEvaluation().get(i);
00069             if (evalA != 0 && evalB != 0) {
00070                 total += (evalA - evalB)/((evalA + evalB)/2);
00071             }
00072         }
00073         pAccept = Math.exp(-(1-total)/T);
00074     }
00075     else if (dominanceRank(stateCandidate, list) > dominanceRank(stateCurrent, list) &&
00076     dominanceRank(stateCurrent, list)!= 0){
00077         // avoid integer division -> compute ranks once and use floating point division
00078         int rankCandidate = dominanceRank(stateCandidate, list);
00079         int rankCurrent = dominanceRank(stateCurrent, list);
00080         float value = (rankCurrent == 0) ? 0f : ((float) rankCandidate) / ((float)
00081         rankCurrent);
00082         pAccept = Math.exp(-((double)value+1.0)/T);
00083     }
00084     else{
00085         //Calculando la probabilidad de aceptaci&on
00086         List<Double> evaluations = stateCurrent.getEvaluation();
00087         double total = 0;
00088         for (int i = 0; i < evaluations.size()-1; i++) {
00089             Double evalA = evaluations.get(i);
00090             Double evalB = stateCandidate.getEvaluation().get(i);
00091             if (evalA != 0 && evalB != 0) {
00092                 total += (evalA - evalB)/((evalA + evalB)/2);
00093             }
00094         }
00095     }
00096     //Generar un n&numero aleatorio
00097     if(ThreadLocalRandom.current().nextFloat() < pAccept){
00098         // Don't assign to the parameter reference (dead store). The caller should handle state
00099         updates.
00100         // Verificando que la soluci&nacute;n candidata domina a alguna de las soluciones
00101         accept = dominance.listDominance(stateCandidate, list);
00102     }
00103     return accept;
00104 }
00105
00106     private int dominanceCounter(State stateCandidate, List<State> list) { //chequea el n&numero de
00107     soluciones de Pareto que son dominados por la nueva soluci&nacute;n
00108     int counter = 0;
00109     for (int i = 0; i < list.size(); i++) {
00110         State solution = list.get(i);
00111         Dominance dominance = new Dominance();
00112         if(dominance.dominance(stateCandidate, solution) == true)
00113             counter++;
00114     }
00115     return counter;
00116 }
00117     private int dominanceRank(State stateCandidate, List<State> list) { //calculando el n&numero de
00118     soluciones en el conjunto de Pareto que dominan a la soluci&nacute;n
00119     int rank = 0;
00120     for (int i = 0; i < list.size(); i++) {
00121         State solution = list.get(i);
00122         Dominance dominance = new Dominance();
00123         if(dominance.dominance(solution, stateCandidate) == true){
00124             rank++;
00125         }
00126     }
00127     return rank;
00128 }
00129
00130 }
```

## 9.105 Referencia del archivo AcceptNotBad.java

```
import metaheuristics.strategy.Strategy;
import problem.definition.Problem;
import problem.definition.State;
import problem.definition.Problem.ProblemType;
```

Gráfico de dependencias incluidas en AcceptNotBad.java:



### Clases

- class [local\\_search.acceptation\\_type.AcceptNotBad](#)

*AcceptNotBad - Accept candidate if it is not worse than the current state.*

### Paquetes

- package [local\\_search.acceptation\\_type](#)
- @(#)* [AcceptAnyone.java](#)

## 9.106 AcceptNotBad.java

[Ir a la documentación de este archivo.](#)

```
00001
00004
00005 package local_search.acceptation_type;
00006
00007 import metaheuristics.strategy.Strategy;
00008 import problem.definition.Problem;
00009 import problem.definition.State;
00010 import problem.definition.Problem.ProblemType;
00011
00018 public class AcceptNotBad extends AcceptableCandidate{
00019
00020     @Override
00028     public Boolean acceptCandidate(State stateCurrent, State stateCandidate) {
00029         Boolean accept = null;
00030         Problem problem = Strategy.getStrategy().getProblem();
00031         for (int i = 0; i < problem.getFunction().size(); i++) {
00032             if (problem.getTypeProblem().equals(ProblemType.Maximizar)) {
00033                 if (stateCandidate.getEvaluation().get(0) >= stateCurrent.getEvaluation().get(0)) {
00034                     accept = true;
00035                 } else {
00036                     accept = false;
00037                 }
00038             } else {
00039                 if (stateCandidate.getEvaluation().get(0) <= stateCurrent.getEvaluation().get(0)) {
00040                     accept = true;
00041                 } else {
00042                     accept = false;
00043                 }
00044             }
00045         }
00046         return accept;
00047     }
}
```

## 9.107 Referencia del archivo AcceptNotBadT.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.concurrent.ThreadLocalRandom;
import metaheuristics.strategy.Strategy;
import metaheuristics.generators.SimulatedAnnealing;
import problem.definition.Problem;
import problem.definition.State;
import problem.definition.Problem.ProblemType;
Gráfico de dependencias incluidas en AcceptNotBadT.java:

```



### Clases

- class [local\\_search.acceptation\\_type.AcceptNotBadT](#)  
*AcceptNotBadT - Temperature based acceptation (simulated annealing style).*

### Paquetes

- package [local\\_search.acceptation\\_type](#)  
`@(#) AcceptAnyone.java`

## 9.108 AcceptNotBadT.java

[Ir a la documentación de este archivo.](#)

```

00001
00004
00005 package local_search.acceptation_type;
00006
00007 import java.lang.reflect.InvocationTargetException;
00008 import java.util.concurrent.ThreadLocalRandom;
00009
00010 import metaheuristics.strategy.Strategy;
00011 import metaheuristics.generators.SimulatedAnnealing;
00012 import problem.definition.Problem;
00013 import problem.definition.State;
00014 import problem.definition.Problem.ProblemType;
00015
00022 public class AcceptNotBadT extends AcceptableCandidate{
00023
00029     @SuppressWarnings("squid:S2245")
00030     @Override
00045     public Boolean acceptCandidate(State stateCurrent, State stateCandidate) throws
IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00046         Boolean accept = null;
00047         Problem problem = Strategy.getStrategy().getProblem();
00048         if (problem.getTypeProblem().equals(ProblemType.Maximizar)) {
00049             double result = (stateCandidate.getEvaluation().get(0) -
stateCurrent.getEvaluation().get(0)) / SimulatedAnnealing.getTinitial();
00050             double probaleatory = ThreadLocalRandom.current().nextDouble();
00051             double exp = Math.exp(result);
00052             if ((stateCandidate.getEvaluation().get(0) >= stateCurrent.getEvaluation().get(0))
00053                 || (probaleatory < exp))
00054                 accept = true;
00055             else
00056                 accept = false;
00057         } else {

```

```

00058         double result_min = (stateCandidate.getEvaluation().get(0) -
00059             stateCurrent.getEvaluation().get(0)) / SimulatedAnnealing.getTinitial();
00060             || (ThreadLocalRandom.current().nextDouble() < Math.exp(result_min)))
00061             accept = true;
00062         else
00063             accept = false;
00064     }
00065     return accept;
00066 }
00067 }

```

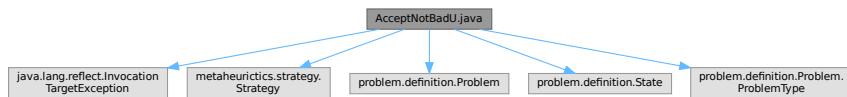
## 9.109 Referencia del archivo AcceptNotBadU.java

```

import java.lang.reflect.InvocationTargetException;
import metaheuristics.strategy.Strategy;
import problem.definition.Problem;
import problem.definition.State;
import problem.definition.Problem.ProblemType;

```

Gráfico de dependencias incluidas en AcceptNotBadU.java:



### Clases

- class [local\\_search.acceptation\\_type.AcceptNotBadU](#)  
*AcceptNotBadU - Threshold based acceptance strategy.*

### Paquetes

- package [local\\_search.acceptation\\_type](#)  
@(#) [AcceptAnyone.java](#)

## 9.110 AcceptNotBadU.java

[Ir a la documentación de este archivo.](#)

```

00001
00004
00005 package local_search.acceptation_type;
00006
00007 import java.lang.reflect.InvocationTargetException;
00008
00009 import metaheuristics.strategy.Strategy;
00010 import problem.definition.Problem;
00011 import problem.definition.State;
00012 import problem.definition.Problem.ProblemType;
00013
00021 public class AcceptNotBadU extends AcceptableCandidate{
00022
00023     @Override
00038     public Boolean acceptCandidate(State stateCurrent, State stateCandidate) throws
        IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
        IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00039         Boolean accept = null;

```

```

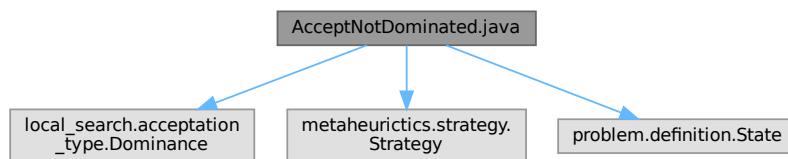
00040     Problem problem = Strategy.getStrategy().getProblem();
00041     if (problem.getTypeProblem().equals(ProblemType.Maximizar)) {
00042         Double result = stateCurrent.getEvaluation().get(0) -
00043             stateCandidate.getEvaluation().get(0);
00044         if (result < Strategy.getStrategy().getThreshold())
00045             accept = true;
00046         else
00047             accept = false;
00048     } else {
00049         Double result_min = stateCurrent.getEvaluation().get(0) -
00050             stateCandidate.getEvaluation().get(0);
00051         if (result_min > Strategy.getStrategy().getThreshold())
00052             accept = true;
00053         else
00054             accept = false;
00055     }
00056 }

```

## 9.111 Referencia del archivo AcceptNotDominated.java

import local\_search.acceptation\_type.Dominance;  
 import metaheuristics.strategy.Strategy;  
 import problem.definition.State;

Gráfico de dependencias incluidas en AcceptNotDominated.java:



### Clases

- class [local\\_search.acceptation\\_type.AcceptNotDominated](#)  
*AcceptNotDominated - Accept candidate if it is not dominated by current non-dominated set.*

### Paquetes

- package [local\\_search.acceptation\\_type](#)  
`@(#)` [AcceptAnyone.java](#)

## 9.112 AcceptNotDominated.java

[Ir a la documentación de este archivo.](#)

```

00001 package local_search.acceptation_type;
00002
00003
00004 import local_search.acceptation_type.Dominance;
00005 import metaheuristics.strategy.Strategy;
00006
00007 import problem.definition.State;
  
```

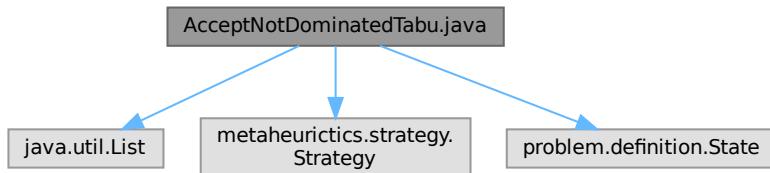
```

00008
00016 public class AcceptNotDominated extends AcceptableCandidate {
00017
00018     @Override
00028     public Boolean acceptCandidate(State stateCurrent, State stateCandidate) {
00029         Boolean accept = false;
00030         Dominance dominace = new Dominance();
00031
00032         if(Strategy.getStrategy().listRefPoblacFinal.size() == 0){
00033             Strategy.getStrategy().listRefPoblacFinal.add(stateCurrent.clone());
00034         }
00035         if(dominace.dominance(stateCurrent, stateCandidate)== false)
00036         {
00037             //Verificando si la solución candidata domina a alguna de las soluciones de la lista de
00038             //soluciones no dominadas
00039             //De ser así se eliminan de la lista y se adiciona la nueva solución en la lista
00040             //De lo contrario no se adiciona la solución candidata a la lista
00041             //Si fue insertada en la lista entonces la solución candidata se convierte en solución
00042             actual
00041             if(dominace.listDominance(stateCandidate, Strategy.getStrategy().listRefPoblacFinal) ==
00042                 true){
00043                 //Se pone la solución candidata como solución actual
00043                 accept = true;
00044             }
00045             else{
00046                 accept = false;
00047             }
00048         }
00049         return accept;
00050     }
00051 }
00052 }
```

## 9.113 Referencia del archivo AcceptNotDominatedTabu.java

```

import java.util.List;
import metaheuristics.strategy.Strategy;
import problem.definition.State;
Gráfico de dependencias incluidas en AcceptNotDominatedTabu.java:
```



### Clases

- class [local\\_search.acceptation\\_type.AcceptNotDominatedTabu](#)  
*AcceptNotDominatedTabu - Acceptation that integrates a tabu-like Pareto list.*

### Paquetes

- package [local\\_search.acceptation\\_type](#)  
*@(#)* [AcceptAnyone.java](#)

## 9.114 AcceptNotDominatedTabu.java

[Ir a la documentación de este archivo.](#)

```

00001 package local_search.acceptation_type;
00002
00003 import java.util.List;
00004
00005 import metaheuristics.strategy.Strategy;
00006
00007 import problem.definition.State;
00008
00016 public class AcceptNotDominatedTabu extends AcceptableCandidate {
00017
00018     @Override
00027     public Boolean acceptCandidate(State stateCurrent, State stateCandidate) {
00028         List<State> list = Strategy.getStrategy().listRefPoblacFinal;
00029
00030         Dominance dominance = new Dominance();
00031         if(list.size() == 0){
00032             list.add(stateCurrent.clone());
00033         }
00034         //Verificando si la solución candidata domina a alguna de las soluciones de la lista de
00035         //soluciones no dominadas
00036         //De ser así se eliminan de la lista y se adiciona la nueva solución en la lista
00037         //De lo contrario no se adiciona la solución candidata a la lista
00038         //Si fue insertada en la lista entonces la solución candidata se convierte en solución actual
00039         dominance.listDominance(stateCandidate, list);
00040         return true;
00041     }

```

## 9.115 Referencia del archivo AcceptType.java

### Clases

- enum [local\\_search.acceptation\\_type.AcceptType](#)

*AcceptType - Enumeration of the available acceptation strategies.*

### Paquetes

- package [local\\_search.acceptation\\_type](#)  
`@(#)` [AcceptAnyone.java](#)

## 9.116 AcceptType.java

[Ir a la documentación de este archivo.](#)

```

00001
00004
00005 package local_search.acceptation_type;
00006
00013 public enum AcceptType
00014 {
00015     AcceptBest, AcceptAnyone, AcceptNotBadT, AcceptNotBadU, AcceptNotDominated,
00016     AcceptNotDominatedTabu, AcceptNotBad, AcceptMulticase;
00017 }

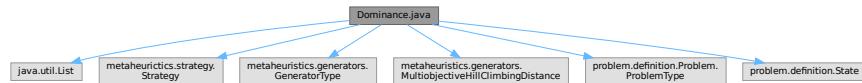
```

## 9.117 Referencia del archivo Dominance.java

```

import java.util.List;
import metaheuristics.strategy.Strategy;
import metaheuristics.generators.GeneratorType;
import metaheuristics.generators.MultiobjectiveHillClimbingDistance;
import problem.definition.Problem.ProblemType;
import problem.definition.State;
Gráfico de dependencias incluidas en Dominance.java:

```



### Clases

- class [local\\_search.acceptation\\_type.Dominance](#)  
*Dominance - Utilities for Pareto dominance comparisons.*

### Paquetes

- package [local\\_search.acceptation\\_type](#)  
*@(#)* [AcceptAnyone.java](#)

## 9.118 Dominance.java

[Ir a la documentación de este archivo.](#)

```

00001 package local_search.acceptation_type;
00002
00003 import java.util.List;
00004
00005 import metaheuristics.strategy.Strategy;
00006 import metaheuristics.generators.GeneratorType;
00007 import metaheuristics.generators.MultiobjectiveHillClimbingDistance;
00008 import problem.definition.Problem.ProblemType;
00009 import problem.definition.State;
0010
0018 public class Dominance {
0019
0020     //-----Metodos que se utilizan en los algoritmos
0021     //multiojetivo-----
0021     //Funcion que determina si la solucion X domina a alguna de las soluciones no dominadas de una
0021     //lista
0022     //Devuelve la lista actualizada y true si fue adicionada a la lista o false de lo contrario
0033     public boolean listDominance(State solutionX, List<State> list){
0034         boolean domain = false;
0035         for (int i = 0; i < list.size() && domain == false; i++) {
0036             //Si la soluci&on X domina a la soluci&n de la lista
0037             if(dominance(solutionX, list.get(i)) == true){
0038                 //Se elimina el elemento de la lista
0039                 list.remove(i);
0040                 if (i!=0) {
0041                     i--;
0042                 }
0043                 if
0044                     (Strategy.getStrategy().generator.getType().equals(GeneratorType.MultiobjectiveHillClimbingDistance)&&list.size()!=0)
0044                         MultiobjectiveHillClimbingDistance.distanceCalculateAdd(list);
0045                 }
0046             }

```

```

00047         if (list.size() > 0) {
00048             if(dominance(list.get(i), solutionX) == true) {
00049                 domain = true;
00050             }
00051         }
00052     }
00053 }
00054 //Si la soluci&on X no fue dominada
00055 if(domain == false){
00056     //Comprobando que la soluci&on no exista
00057     boolean found = false;
00058     for (int k = 0; k < list.size() && found == false; k++) {
00059         State element = list.get(k);
00060         found = solutionX.Comparator(element);
00061     }
00062 //Si la soluci&on no existe
00063 if(found == false){
00064     //Se guarda la soluci&on candidata en la lista de soluciones &optimas de Pareto
00065     list.add(solutionX.clone());
00066     if
00067         (Strategy.getStrategy().generator.getType().equals(GeneratorType.MultiobjectiveHillClimbingDistance))
00068     {
00069         MultiobjectiveHillClimbingDistance.distanceCalculateAdd(list);
00070     }
00071 }
00072 return false;
00073
00074 }
00075 }
00076
00077
00078 //Funcion que devuelve true si solutionX domina a solutionY
00079 public boolean dominance(State solutionX, State solutionY) {
00080     boolean dominance = false;
00081     int countBest = 0;
00082     int countEquals = 0;
00083     //Si solutionX domina a solutionY
00084     if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)) {
00085         //Recorriendo las evaluaciones de las funciones objetivo
00086         for (int i = 0; i < solutionX.getEvaluation().size(); i++) {
00087             if(solutionX.getEvaluation().get(i).floatValue() >
00088                 solutionY.getEvaluation().get(i).floatValue()){
00089                 countBest++;
00090             }
00091             if(solutionX.getEvaluation().get(i).floatValue() ==
00092                 solutionY.getEvaluation().get(i).floatValue()){
00093                 countEquals++;
00094             }
00095         }
00096     }
00097     else{
00098         //Recorriendo las evaluaciones de las funciones objetivo
00099         for (int i = 0; i < solutionX.getEvaluation().size(); i++) {
00100             if(solutionX.getEvaluation().get(i).floatValue() <
00101                 solutionY.getEvaluation().get(i).floatValue()){
00102                 countBest++;
00103             }
00104             if(solutionX.getEvaluation().get(i).floatValue() ==
00105                 solutionY.getEvaluation().get(i).floatValue()){
00106                 countEquals++;
00107             }
00108         }
00109     }
00110     if((countBest >= 1) && (countEquals + countBest == solutionX.getEvaluation().size())){
00111         dominance = true;
00112     }
00113 }
00114
00115 return dominance;
00116 }
00117
00118 }
00119 }

```

## 9.119 Referencia del archivo CandidateType.java

### Clases

- enum local\_search.candidate\_type.CandidateType

*CandidateType - enumeration of available candidate selection strategies.*

## Paquetes

- package local\_search.candidate\_type  
`@(#)` TypeCandidate.java

## 9.120 CandidateType.java

[Ir a la documentación de este archivo.](#)

```
00001
00004
00005 package local_search.candidate_type;
00006
00013 public enum CandidateType{
00014
00016     SmallerCandidate,
00018     GreaterCandidate,
00020     RandomCandidate,
00022     NotDominatedCandidate;
00023 }
```

## 9.121 Referencia del archivo CandidateValue.java

```
import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import problem.definition.State;
import local_search.complement.StrategyType;
import local_search.complement.TabuSolutions;
import metaheuristics.strategy.Strategy;
import factory_interface.IFFactoryCandidate;
import factory_method.FactoryCandidate;
Gráfico de dependencias incluidas en CandidateValue.java:
```



## Clases

- class local\_search.candidate\_type.CandidateValue  
*CandidateValue - Factory/wrapper for creating and using SearchCandidate selection strategies.*

## Paquetes

- package local\_search.candidate\_type  
`@(#)` TypeCandidate.java

## 9.122 CandidateValue.java

[Ir a la documentación de este archivo.](#)

```

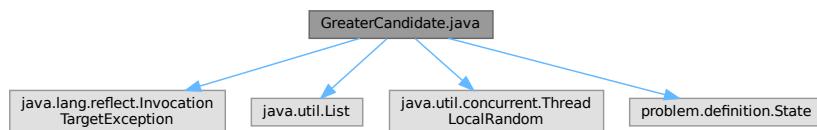
00001
00004
00005 package local_search.candidate_type;
00006
00007 import java.lang.reflect.InvocationTargetException;
00008 import java.util.ArrayList;
00009 import java.util.List;
00010
00011 import problem.definition.State;
00012
00013 import local_search.complement.StrategyType;
00014 import local_search.complement.TabuSolutions;
00015 import metaheuristics.strategy.Strategy;
00016
00017 //import ceis.grial.problem.Problem;
00018 import factory_interface.IFFactoryCandidate;
00019 import factory_method.FactoryCandidate;
00020
00029 public class CandidateValue {
00030
00031     private IFFactoryCandidate iffFactory;
00032
00033     // strategy and typecandidate fields were unused; removed to avoid SpotBugs unread-field warnings.
00035
00036     private TabuSolutions tabusolution;
00037
00038     private SearchCandidate searchcandidate;
00039
00040     public CandidateValue(){}
00041
00042     // Keep a no-arg constructor; parameterized constructor was removed because its fields were
00043     // unused.
00044
00052     public SearchCandidate newSearchCandidate(CandidateType typecandidate) throws
00053         IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00054         IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00055         iffFactory = new FactoryCandidate();
00056         searchcandidate = iffFactory.createSearchCandidate(typecandidate);
00057         return searchcandidate;
00058     }
00074     public State stateCandidate(State stateCurrent, CandidateType typeCandidate, StrategyType
00075         strategy, Integer operatornumber, List<State> neighborhood) throws IllegalArgumentException,
00076         SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
00077         InvocationTargetException, NoSuchMethodException{
00078         //Problem problem = ExecuteGenerator.getExecuteGenerator().getProblem();
00079         State stateCandidate;
00080         List<State> auxList = new ArrayList<State>();
00081         for (int i = 0; i < neighborhood.size(); i++) {
00082             auxList.add(neighborhood.get(i));
00083         }
00084         this.tabusolution = new TabuSolutions();
00085         if (strategy.equals(StrategyType.TABU)) {
00086             try {
00087                 auxList = this.tabusolution.filterNeighborhood(auxList);
00088             } catch (Exception e) {
00089                 Strategy strategys = Strategy.getStrategy();
00090                 if(strategys.getProblem()!=null){
00091                     neighborhood =
00092                         strategys.getProblem().getOperator().generatedNewState(neighborhood.get(0), operatornumber);
00093                 }
00094             SearchCandidate searchCand = newSearchCandidate(typeCandidate);
00095             stateCandidate = searchCand.stateSearch(auxList);
00096             return stateCandidate;
00097         }
00098
00104         public TabuSolutions getTabusolution() {
00105             return tabusolution;
00106         }
00107
00113         public void setTabusolution(TabuSolutions tabusolution) {
00114             this.tabusolution = tabusolution;
00115         }
00116     }

```

## 9.123 Referencia del archivo GreaterCandidate.java

```
import java.lang.reflect.InvocationTargetException;
import java.util.List;
import java.util.concurrent.ThreadLocalRandom;
import problem.definition.State;
```

Gráfico de dependencias incluidas en GreaterCandidate.java:



### Clases

- class [local\\_search.candidate\\_type.GreaterCandidate](#)  
*GreaterCandidate - choose the neighbor with the greatest evaluation.*

### Paquetes

- package [local\\_search.candidate\\_type](#)  
*@(#) TypeCandidate.java*

## 9.124 GreaterCandidate.java

[Ir a la documentación de este archivo.](#)

```
00001
00004
00005 package local_search.candidate_type;
00006
00007
00008 import java.lang.reflect.InvocationTargetException;
00009 import java.util.List;
00010 import java.util.concurrent.ThreadLocalRandom;
00011
00012 import problem.definition.State;
00013
00021 public class GreaterCandidate extends SearchCandidate {
00022
00028     @SuppressWarnings("squid:S2245")
00029     @Override
00037     public State stateSearch(List<State> listNeighborhood) throws IllegalArgumentException,
SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
InvocationTargetException, NoSuchMethodException {
00038         State stateGreater = null;
00039         if(listNeighborhood.size() > 1){
00040             double counter = 0;
00041             double currentCount = listNeighborhood.get(0).getEvaluation().get(0);
00042             for (int i = 1; i < listNeighborhood.size(); i++) {
00043                 counter = listNeighborhood.get(i).getEvaluation().get(0);
00044                 if (counter > currentCount) {
00045                     currentCount = counter;
00046                     stateGreater = listNeighborhood.get(i);
00047                 }
00048             }
00049         }
00050         if(stateGreater == null){
00051             // bound is listNeighborhood.size() - 1 (>=1 since size>1)
```

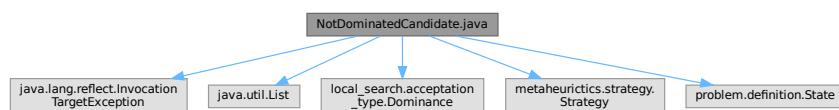
```

00052         int pos = ThreadLocalRandom.current().nextInt(listNeighborhood.size() - 1);
00053         stateGreater = listNeighborhood.get(pos);
00054     }
00055 }
00056 else stateGreater = listNeighborhood.get(0);
00057 return stateGreater;
00058 }
00059 }
```

## 9.125 Referencia del archivo NotDominatedCandidate.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.List;
import local_search.acceptation_type.Dominance;
import metaheuristics.strategy.Strategy;
import problem.definition.State;
Gráfico de dependencias incluidas en NotDominatedCandidate.java:
```



### Clases

- class [local\\_search.candidate\\_type.NotDominatedCandidate](#)  
*NotDominatedCandidate* - select a non-dominated neighbor for multi-objective.

### Paquetes

- package [local\\_search.candidate\\_type](#)  
*@(#)* TypeCandidate.java

## 9.126 NotDominatedCandidate.java

[Ir a la documentación de este archivo.](#)

```

00001 package local_search.candidate_type;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.List;
00005
00006 import local_search.acceptation_type.Dominance;
00007 import metaheuristics.strategy.Strategy;
00008
00009 import problem.definition.State;
00010
00017 public class NotDominatedCandidate extends SearchCandidate {
00018
00019     @Override
00027     public State stateSearch(List<State> listNeighborhood) throws IllegalArgumentException,
SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
InvocationTargetException, NoSuchMethodException {
00028         State state = new State();
00029         State stateA = listNeighborhood.get(0);
00030         boolean stop = false;
00031         if(listNeighborhood.size() == 1){
```

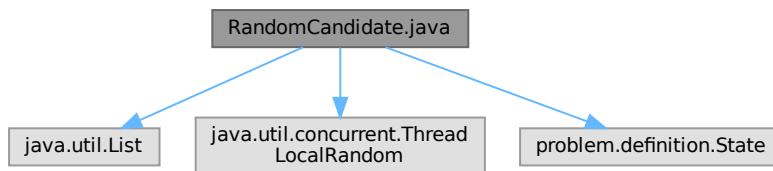
```

00032         stop = true;
00033         state = stateA;
00034     }
00035     else {
00036         Strategy.getStrategy().getProblem().Evaluate(stateA);
00037         State stateB;
00038         Dominance dominance = new Dominance();
00039         for (int i = 1; i < listNeighborhood.size(); i++) {
00040             while(stop == false){
00041                 stateB = listNeighborhood.get(i);
00042                 Strategy.getStrategy().getProblem().Evaluate(stateB);
00043                 if(dominance.dominance(stateB, stateA) == true){
00044                     stateA = stateB;
00045                 }else{
00046                     stop = true;
00047                     state = stateA;
00048                 }
00049             }
00050         }
00051     }
00052     return state;
00053 }
00054 }
00055 }
```

## 9.127 Referencia del archivo RandomCandidate.java

```

import java.util.List;
import java.util.concurrent.ThreadLocalRandom;
import problem.definition.State;
Gráfico de dependencias incluidas en RandomCandidate.java:
```



### Clases

- class [local\\_search.candidate\\_type.RandomCandidate](#)  
*RandomCandidate - select a neighbor at random.*

### Paquetes

- package [local\\_search.candidate\\_type](#)  
*@(#) TypeCandidate.java*

## 9.128 RandomCandidate.java

[Ir a la documentación de este archivo.](#)

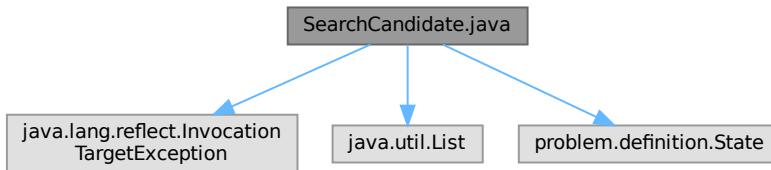
```

00001
00004
00005 package local_search.candidate_type;
00006
00007 import java.util.List;
00008 import java.util.concurrent.ThreadLocalRandom;
00009
00010 import problem.definition.State;
00011
00018 public class RandomCandidate extends SearchCandidate {
00025     @SuppressWarnings("squid:S2245")
00026     @Override
00033     public State stateSearch(List<State> listNeighborhood) {
00034         int size = listNeighborhood.size();
00035         int pos = (size > 0) ? ThreadLocalRandom.current().nextInt(size) : 0;
00036         State stateAleatory = listNeighborhood.get(pos);
00037         return stateAleatory;
00038     }
00039 }
```

## 9.129 Referencia del archivo SearchCandidate.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.List;
import problem.definition.State;
Gráfico de dependencias incluidas en SearchCandidate.java:
```



### Clases

- class [local\\_search.candidate\\_type.SearchCandidate](#)  
*SearchCandidate - abstract base class for candidate selection strategies.*

### Paquetes

- package [local\\_search.candidate\\_type](#)  
*@(#)* *TypeCandidate.java*

## 9.130 SearchCandidate.java

[Ir a la documentación de este archivo.](#)

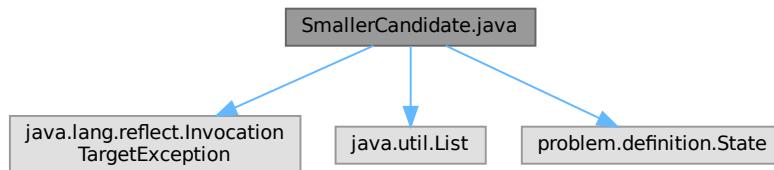
```

00001
00004
00005 package local_search.candidate_type;
00006
00007 import java.lang.reflect.InvocationTargetException;
00008 import java.util.List;
00009
00010 import problem.definition.State;
00011
00012
00019 public abstract class SearchCandidate {
00020
00028     public abstract State stateSearch(List<State> listNeighborhood) throws IllegalArgumentException,
SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
InvocationTargetException, NoSuchMethodException ;
00029 }
```

## 9.131 Referencia del archivo SmallerCandidate.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.List;
import problem.definition.State;
Gráfico de dependencias incluidas en SmallerCandidate.java:
```



### Clases

- class [local\\_search.candidate\\_type.SmallerCandidate](#)  
*SmallerCandidate - choose the neighbor with the smallest evaluation.*

### Paquetes

- package [local\\_search.candidate\\_type](#)  
`@(#)` *TypeCandidate.java*

## 9.132 SmallerCandidate.java

[Ir a la documentación de este archivo.](#)

```

00001
00004
00005 package local_search.candidate_type;
00006
00007
00008 import java.lang.reflect.InvocationTargetException;
00009 import java.util.List;
00010
00011 import problem.definition.State;
00012
00019 public class SmallerCandidate extends SearchCandidate {
00020
00021     @Override
00029     public State stateSearch(List<State> listNeighborhood) throws IllegalArgumentException,
SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
InvocationTargetException, NoSuchMethodException {
00030         State stateSmaller = null;
00031         if(listNeighborhood.size() > 1){
00032             double counter = 0;
00033             double currentCount = listNeighborhood.get(0).getEvaluation().get(0);
00034             for (int i = 1; i < listNeighborhood.size(); i++) {
00035                 counter = listNeighborhood.get(i).getEvaluation().get(0);
00036                 if (counter < currentCount) {
00037                     currentCount = counter;
00038                     stateSmaller = listNeighborhood.get(i);
00039                 }
00040                 counter = 0;
00041             }
00042         }
00043         else stateSmaller = listNeighborhood.get(0);
00044     return stateSmaller;
00045 }
00046 }
```

## 9.133 Referencia del archivo StopExecute.java

### Clases

- class [local\\_search.complement.StopExecute](#)  
*StopExecute* - termination predicate for local search loops.

### Paquetes

- package [local\\_search.complement](#)  
*@(#)* [Strategy.java](#)

## 9.134 StopExecute.java

[Ir a la documentación de este archivo.](#)

```

00001 package local_search.complement;
00002
00003
00010 public class StopExecute {
00011
00020     public Boolean stopIterations(int countIterationsCurrent, int countmaxIterations) {
00021         if (countIterationsCurrent < countmaxIterations) {
00022             return false;
00023         } else {
00024             return true;
00025         }
00026     }
00027 }
```

## 9.135 Referencia del archivo StrategyType.java

### Clases

- enum local\_search.complement.StrategyType

*StrategyType - enumeration of local search strategy modes.*

### Paquetes

- package local\_search.complement

*@(#)* Strategy.java

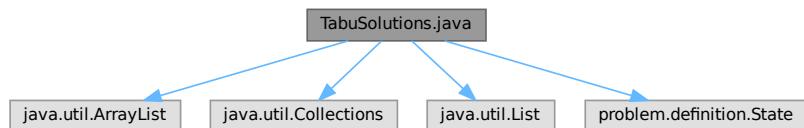
## 9.136 StrategyType.java

[Ir a la documentación de este archivo.](#)

```
00001
00004
00005 package local_search.complement;
00006
00013 public enum StrategyType
00014 {
00016     TABU,
00018     NORMAL;
00019 }
```

## 9.137 Referencia del archivo TabuSolutions.java

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import problem.definition.State;
Gráfico de dependencias incluidas en TabuSolutions.java:
```



### Clases

- class local\_search.complement.TabuSolutions

*TabuSolutions - helper that manages a tabu list and filters neighborhoods.*

### Paquetes

- package local\_search.complement

*@(#)* Strategy.java

## 9.138 TabuSolutions.java

[Ir a la documentación de este archivo.](#)

```

00001 package local_search.complement;
00002
00003 import java.util.ArrayList;
00004 import java.util.Collections;
00005 import java.util.List;
00006
00007 import problem.definition.State;
00008
00016 public class TabuSolutions {
00017
00019     public static final List<State> listTabu = Collections.synchronizedList(new ArrayList<State>());
00020
00022     public static final int maxelements = 0;
00023
00036     public List<State> filterNeighborhood(List<State> listNeighborhood) throws Exception {
00037         List<State> listFiltrate;
00038         // If there are entries in the tabu list, remove matching neighbors
00039         if (!listTabu.isEmpty()) {
00040             for (int i = listNeighborhood.size() - 1; i >= 0 ; i--) {
00041                 int count_tabu = 0;
00042                 while (listTabu.size() > count_tabu) {
00043                     if (listNeighborhood.get(i).equals(listTabu.get(count_tabu))) {
00044                         listNeighborhood.remove(i);
00045                     }
00046                     count_tabu++;
00047                 }
00048             }
00049             listFiltrate = listNeighborhood;
00050             if (listFiltrate.isEmpty()) {
00051                 throw new Exception();
00052             }
00053         } else {
00054             listFiltrate = listNeighborhood;
00055         }
00056         return listFiltrate;
00057     }
00058 }
```

## 9.139 Referencia del archivo UpdateParameter.java

```

import factory_interface.IFFactoryGenerator;
import factory_method.FactoryGenerator;
import java.lang.reflect.InvocationTargetException;
import metaheuristics.strategy.Strategy;
import metaheuristics.generators.DistributionEstimationAlgorithm;
import metaheuristics.generators.EvolutionStrategies;
import metaheuristics.generators.GeneratorType;
import metaheuristics.generators.GeneticAlgorithm;
import metaheuristics.generators.ParticleSwarmOptimization;
Gráfico de dependencias incluidas en UpdateParameter.java:
```



### Clases

- class `local_search.complement.UpdateParameter`

*UpdateParameter* - utility that updates generator parameters during search.

## Paquetes

- package local\_search.complement  
*@(#)* Strategy.java

## 9.140 UpdateParameter.java

[Ir a la documentación de este archivo.](#)

```

00001 package local_search.complement;
00002
00003 import factory_interface.IFFactoryGenerator;
00004 import factory_method.FactoryGenerator;
00005
00006 import java.lang.reflect.InvocationTargetException;
00007
00008 import metaheuristics.strategy.Strategy;
00009 import metaheuristics.generators.DistributionEstimationAlgorithm;
00010 import metaheuristics.generators.EvolutionStrategies;
00011 import metaheuristics.generators.GeneratorType;
00012 import metaheuristics.generators.GeneticAlgorithm;
00013 import metaheuristics.generators.ParticleSwarmOptimization;
00014
00015
00023 public class UpdateParameter {
00024
00025     private static IFFactoryGenerator iffFactoryGenerator;
00026
00039     public static Integer updateParameter(Integer countIterationsCurrent) throws
IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
IllegalAccessException, InvocationTargetException, NoSuchMethodException {//HashMap<String, Object>
map,
00040         countIterationsCurrent = countIterationsCurrent + 1;
//        Here update parameter for update and change generator.
00042         if(countIterationsCurrent.equals(GeneticAlgorithm.countRef - 1)){
00043             iffFactoryGenerator = new FactoryGenerator();
00044             Strategy.getStrategy().generator =
iffFactoryGenerator.createGenerator(GeneratorType.GeneticAlgorithm);
00045         }
00046         else{
00047             if(countIterationsCurrent.equals(EvolutionStrategies.countRef - 1)){
00048                 iffFactoryGenerator = new FactoryGenerator();
00049                 Strategy.getStrategy().generator =
iffFactoryGenerator.createGenerator(GeneratorType.EvolutionStrategies);
00050             }
00051             if(countIterationsCurrent.equals(DistributionEstimationAlgorithm.countRef - 1)){
00052                 iffFactoryGenerator = new FactoryGenerator();
00053                 Strategy.getStrategy().generator =
iffFactoryGenerator.createGenerator(GeneratorType.DistributionEstimationAlgorithm);
00054             }
00055             if(countIterationsCurrent.equals(ParticleSwarmOptimization.getCountRef() - 1)){
00056                 iffFactoryGenerator = new FactoryGenerator();
00057                 Strategy.getStrategy().generator =
iffFactoryGenerator.createGenerator(GeneratorType.ParticleSwarmOptimization);
00058             }
00059         }
00060         return countIterationsCurrent;
00061     }
00062 }
00063
00064
00065

```

## 9.141 Referencia del archivo Strategy.java

```

import java.lang.reflect.InvocationTargetException;
import java.security.AccessController;
import java.security.PrivilegedAction;
import java.util.ArrayList;
import java.util.List;
import java.util.SortedMap;

```

```

import java.util.TreeMap;
import factory_interface.IFFactoryGenerator;
import factory_method.FactoryGenerator;
import problem.definition.Problem;
import problem.definition.State;
import problem.definition.Problem.ProblemType;
import local_search.acceptation_type.Dominance;
import local_search.complement.StopExecute;
import local_search.complement.UpdateParameter;
import metaheuristics.generators.DistributionEstimationAlgorithm;
import metaheuristics.generators.EvolutionStrategies;
import metaheuristics.generators.Generator;
import metaheuristics.generators.GeneratorType;
import metaheuristics.generators.GeneticAlgorithm;
import metaheuristics.generators.MultiGenerator;
import metaheuristics.generators.ParticleSwarmOptimization;
import metaheuristics.generators.RandomSearch;
Gráfico de dependencias incluidas en Strategy.java:

```



## Clases

- class [metaheuristics.strategy.Strategy](#)  
*Strategy* - .

## Paquetes

- package [metaheuristics.strategy](#)

## 9.142 Strategy.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.strategy;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.security.AccessController;
00005 import java.security.PrivilegedAction;
00006 import java.util.ArrayList;
00007 import java.util.List;
00008 import java.util.SortedMap;
00009 import java.util.TreeMap;
00010
00011 import factory_interface.IFFactoryGenerator;
00012 import factory_method.FactoryGenerator;
00013
00014 import problem.definition.Problem;
00015 import problem.definition.State;
00016 import problem.definition.Problem.ProblemType;
00017
00018 import local_search.acceptation_type.Dominance;
00019 import local_search.complement.StopExecute;
00020 import local_search.complement.UpdateParameter;
00021 import metaheuristics.generators.DistributionEstimationAlgorithm;
00022 import metaheuristics.generators.EvolutionStrategies;
00023 import metaheuristics.generators.Generator;
00024 import metaheuristics.generators.GeneratorType;
00025 import metaheuristics.generators.GeneticAlgorithm;
00026 import metaheuristics.generators.MultiGenerator;
00027 import metaheuristics.generators.ParticleSwarmOptimization;
00028 import metaheuristics.generators.RandomSearch;

```

```

00029 public class Strategy {
00030
00031     private static Strategy strategy = null;
00032     private State bestState;
00033     private Problem problem;
00034     public SortedMap<GeneratorType, Generator> mapGenerators;
00035     private StopExecute stopexecute;
00036     private UpdateParameter updateparameter;
00037     private IFFactoryGenerator iffFactoryGenerator;
00038     private int countCurrent;
00039     private int countMax;
00040     public Generator generator;
00041     public double threshold;
00042
00043
00044
00045
00046
00047     public List<State> listStates; //lista de todos los estados generados en cada iteracion
00048     public List<State> listBest; //lista de la mejor solucion en cada iteracion
00049     public List<State> listRefPoblacFinal = new ArrayList<State> (); //lista de soluciones no
00050     dominadas
00051     public Dominance notDominated;
00052
00053
00054     public boolean saveListStates; //guardar lista de estados generados
00055     public boolean saveListBestStates; // guardar lista con los mejores estados encontrados en cada
00056     iteracion
00057     public boolean saveFrenetParetoMonoObjetivo; //guardar lista de soluciones no dominadas de una
00058     ejecucion
00059     public boolean calculateTime; // calcular tiempo de ejecucion de un algoritmo
00060
00061
00062
00063     //calculo del Tiempo inicial y final
00064     long initialTime;
00065     long finalTime;
00066     long timeExecute;
00067
00068
00069
00070     public float[] listOfflineError = new float[100]; // para almacenar la metrica offlinePerformance
00071     public int countPeriodChange = 0; // cantidad de iteraciones antes de un cambio
00072     public int countChange = 0;
00073     private int countPeriodo; // contador para saber cada cuantas iteraciones tiene que guardar en la
00074     lista listCountBetterGender de cada generador
00075     private int periodo; //contador para controlar el periodo que esta guardando
00076
00077
00078
00079     private Strategy(){
00080         super();
00081     }
00082     public static Strategy getStrategy() {
00083         if (strategy == null) {
00084             strategy = new Strategy();
00085         }
00086         return strategy;
00087     }
00088
00089     public void executeStrategy (int countmaxIterations, int countIterationsChange, int
00090     operatornumber, GeneratorType generatorType) throws IllegalArgumentException, SecurityException,
00091     ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00092     NoSuchMethodException{
00093         // si se quiere calcular el tiempo de ejecucion del un algoritmo
00094         if(calculateTime == true){
00095             initialTime = System.currentTimeMillis();
00096         }
00097         this.countMax = countmaxIterations; // max cantidad de iteraciones
00098         //generar estado inicial de la estrategia
00099         Generator randomInitial = new RandomSearch();
00100         State initialState = randomInitial.generate(operatornumber);
00101         problem.Evaluate(initialState); //evaluar ese estado
00102         initialState.setTypeGenerator(generatorType);
00103         getProblem().setState(initialState);
00104         //si se va a salvar la lista de estados generados, adicionar el estado
00105         if(saveListStates == true){
00106             listStates = new ArrayList<State>(); //list de estados generados
00107             listStates.add(initialState);
00108         }
00109         //si se va a salvar la lista de mejores soluciones encontradas en cada iteracion
00110         if(saveListBestStates == true){
00111             listBest = new ArrayList<State>(); // list de mejores estados encontrados
00112             listBest.add(initialState);
00113         }
00114         if(saveFrenetParetoMonoObjetivo == true){
00115             notDominated = new Dominance();
00116         }
00117         // crear el generador a ejecutar
00118         generator = newGenerator(generatorType);
00119         generator.setInitialReference(initialState);
00120         bestState = initialState;
00121         countCurrent = 0;
00122         listRefPoblacFinal = new ArrayList<State>();
00123         MultiGenerator multiGenerator = null;
00124
00125

```

```

00126     countPeriodChange = countIterationsChange;
00127     countChange = countIterationsChange;
00128     countPeriodo = countIterationsChange / 10; //cantidad de iteraciones de un periodo
00129     //verificar que es portafolio e inicializar los generadores del portafolio
00130     if(generatorType.equals(GeneratorType.MultiGenerator)){
00131         initializeGenerators();
00132         MultiGenerator.initializeGenerators();
00133         MultiGenerator.listGeneratedDP.clear();
00134         // create a clone of the MultiGenerator using reflection to access protected clone()
00135         try {
00136             final java.lang.reflect.Method cloneMethod =
00137                 generator.getClass().getDeclaredMethod("clone");
00138             // Grant privileged permission for reflective access to non-public clone()
00139             AccessController.doPrivileged(new PrivilegedAction<Void>() {
00140                 public Void run() {
00141                     cloneMethod.setAccessible(true);
00142                     return null;
00143                 }
00144             });
00145             multiGenerator = (MultiGenerator) cloneMethod.invoke(generator);
00146         } catch (Exception e) {
00147             throw new RuntimeException("Failed to clone MultiGenerator", e);
00148         }
00149     }
00150     else initialize(); //crea el mapa de generadores
00151     update(countCurrent);
00152
00153     float sumMax = 0; // suma acumulativa para almacenar la evaluacion de la mejor solucion
00154     encontrada y calcular el OfflinePerformance
00155     int countOff = 0; // variable par contar los OfflinePerformance que se van salvando en el
00156     arreglo
00157     //ciclo de ejecucion del algoritmo
00158     while (!stopexecute.stopIterations(countCurrent, countmaxIterations)){
00159         //si se detecta un cambio
00160         if(countCurrent == countChange){
00161             //calcular offlinePerformance
00162             calculateOfflinePerformance(sumMax, countOff);
00163             countOff++;
00164             sumMax = 0;
00165             countIterationsChange = countIterationsChange + countPeriodChange; // actualizar la
00166             cantidad de iteraciones
00167             //actualizar la referencia luego de un cambio
00168             updateRef(generatorType);
00169             countChange = countChange + countPeriodChange;
00170             //generar un nuevo candidato en la iteracion, dependiendo del generador
00171             State stateCandidate = null;
00172             if(generatorType.equals(GeneratorType.MultiGenerator)){
00173                 if(countPeriodo == countCurrent){
00174                     updateCountGender();
00175                     countPeriodo = countPeriodo + countPeriodChange / 10;
00176                     periodo = 0;
00177                     MultiGenerator.activeGenerator.countBetterGender = 0;
00178                 }
00179                 updateWeight(); //actualizar el peso de los generadores si se reinician cuando
00180                 ocurre un cambio
00181                 //generar el estado candidato de la iteracion
00182                 stateCandidate = multiGenerator.generate(operatornumber);
00183                 problem.Evaluate(stateCandidate);
00184                 stateCandidate.setEvaluation(stateCandidate.getEvaluation());
00185                 stateCandidate.setNumber(countCurrent);
00186                 stateCandidate.setTypeGenerator(generatorType);
00187                 multiGenerator.updateReference(stateCandidate, countCurrent);
00188             }
00189             else {
00190                 stateCandidate = generator.generate(operatornumber);
00191                 problem.Evaluate(stateCandidate);
00192                 stateCandidate.setEvaluation(stateCandidate.getEvaluation());
00193                 stateCandidate.setNumber(countCurrent);
00194                 stateCandidate.setTypeGenerator(generatorType);
00195                 generator.updateReference(stateCandidate, countCurrent);
00196                 if(saveListStates == true){
00197                     listStates.add(stateCandidate);
00198                 }
00199             }
00200             //actualizar el mejor estado encontrado solo tiene sentido para algoritmos
00201             monoobjetivos
00202             //actualizar el mejor estado encontrado solo tiene sentido para algoritmos
00203             monoobjetivos
00204             if ((getProblem().getTypeProblem().equals(ProblemType.Maximizar)) &&
00205                 bestState.getEvaluation().get(bestState.getEvaluation().size() - 1) <
00206                 stateCandidate.getEvaluation().get(bestState.getEvaluation().size() - 1)) {
00207                 bestState = stateCandidate;
00208             }
00209             if ((problem.getTypeProblem().equals(ProblemType.Minimizar)) &&
00210                 bestState.getEvaluation().get(bestState.getEvaluation().size() - 1) >

```

```

00207         stateCandidate.getEvaluation().get(bestState.getEvaluation().size() - 1)) {
00208             bestState = stateCandidate;
00209         }
00210         // System.out.println("Evaluacion: " + bestState.getEvaluation());
00211         if(saveListBestStates == true){
00212             listBest.add(bestState);
00213         }
00214         sumMax = (float) (sumMax + bestState.getEvaluation().get(0));
00215     }
00216     // no ha ocurrido un cambio
00217     else {
00218         State stateCandidate = null;
00219         if(generatorType.equals(GeneratorType.MultiGenerator)){
00220             if(countPeriodo == countCurrent){
00221                 updateCountGender();
00222                 countPeriodo = countPeriodo + countPeriodChange / 10;
00223                 periodo++;
00224                 MultiGenerator.activeGenerator.countBetterGender = 0;
00225             }
00226             stateCandidate = multiGenerator.generate(operatornumber);
00227             problem.Evaluate(stateCandidate);
00228             stateCandidate.setEvaluation(stateCandidate.getEvaluation());
00229             stateCandidate.setNumber(countCurrent);
00230             stateCandidate.setTypeGenerator(generatorType);
00231             multiGenerator.updateReference(stateCandidate, countCurrent);
00232         }
00233         else {
00234             //generar estado candidato y evaluar si es aceptado o no
00235             stateCandidate = generator.generate(operatornumber);
00236             problem.Evaluate(stateCandidate);
00237             stateCandidate.setEvaluation(stateCandidate.getEvaluation());
00238             stateCandidate.setNumber(countCurrent);
00239             stateCandidate.setTypeGenerator(generatorType);
00240             generator.updateReference(stateCandidate, countCurrent); // actualizar la
referencia del estado
00241             if(saveListStates == true){
00242                 listStates.add(stateCandidate);
00243             }
00244             if(saveFreneParetoMonoObjetivo == true){
00245                 notDominated.listDominance(stateCandidate, listRefPoblacFinal);
00246             }
00247         }
00248         countCurrent = UpdateParameter.updateParameter(countCurrent);
00249         //actualizar el mejor estado encontrado solo tiene sentido para algoritmos
monoobjetivos
00250         if ((getProblem().getTypeProblem().equals(ProblemType.Maximizar)) &&
bestState.getEvaluation().get(bestState.getEvaluation().size() - 1) <
stateCandidate.getEvaluation().get(bestState.getEvaluation().size() - 1)) {
00251             bestState = stateCandidate;
00252         }
00253         if ((problem.getTypeProblem().equals(ProblemType.Minimizar)) &&
bestState.getEvaluation().get(bestState.getEvaluation().size() - 1) >
stateCandidate.getEvaluation().get(bestState.getEvaluation().size() - 1)) {
00254             bestState = stateCandidate;
00255         }
00256         // System.out.println("Evaluacion: " + bestState.getEvaluation());
00257         if(saveListBestStates == true){
00258             listBest.add(bestState);
00259         }
00260         sumMax = (float) (sumMax + bestState.getEvaluation().get(0));
00261     }
00262 //     System.out.println("Iteracion: " + countCurrent);
00263 }
00264 //calcular tiempo final
00265 if(calculateTime == true){
00266     finalTime = System.currentTimeMillis();
00267     timeExecute = finalTime - initialTime;
00268     System.out.println("El tiempo de ejecucion: " + timeExecute);
00269 }
00270 if(generatorType.equals(GeneratorType.MultiGenerator)){
00271     listBest = new ArrayList<State>(multiGenerator.getReferenceList());
00272     //calcular offlinePerformance
00273     calculateOfflinePerformance(sumMax, countOff);
00274     if(countPeriodo == countCurrent){
00275         updateCountGender();
00276     }
00277 }
00278 else{
00279     listBest = new ArrayList<State>(generator.getReferenceList());
00280     calculateOfflinePerformance(sumMax, countOff);
00281 }
00282 }
00283
00284 public void updateCountGender(){ // actualizar la cantidad de mejoras y cantidad de veces que se
uso un generador en un periodo dado
00285     for (int i = 0; i < MultiGenerator.getListGenerators().length; i++) {

```

```

00286             if(!MultiGenerator.getListGenerators() [i].getType().equals(GeneratorType.MultiGenerator))
00287         ) {
00288             MultiGenerator.getListGenerators() [i].getListCountGender() [periodo] =
00289                 MultiGenerator.getListGenerators() [i].countGender +
00290                 MultiGenerator.getListGenerators() [i].getListCountGender() [periodo];
00291             MultiGenerator.getListGenerators() [i].getListCountBetterGender() [periodo] =
00292                 MultiGenerator.getListGenerators() [i].countBetterGender +
00293                 MultiGenerator.getListGenerators() [i].getListCountBetterGender() [periodo];
00294             MultiGenerator.getListGenerators() [i].countGender = 0;
00295             MultiGenerator.getListGenerators() [i].countBetterGender = 0;
00296         }
00297     }
00298 
00299     public void updateWeight() {
00300         for (int i = 0; i < MultiGenerator.getListGenerators().length; i++) {
00301             if(!MultiGenerator.getListGenerators() [i].getType().equals(GeneratorType.MultiGenerator)){
00302                 MultiGenerator.getListGenerators() [i].setWeight((float) 50.0);
00303             }
00304         }
00305     }
00306 
00307     public void update(Integer countIterationsCurrent) throws IllegalArgumentException,
00308         SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
00309         InvocationTargetException, NoSuchMethodException { //HashMap<String, Object> map,
00310         //      Here update parameter for update and change generator.
00311         if(countIterationsCurrent.equals(GeneticAlgorithm.countRef - 1)){
00312             ifFactoryGenerator = new FactoryGenerator();
00313             Strategy.getStrategy().generator =
00314                 ifFactoryGenerator.createGenerator(GeneratorType.GeneticAlgorithm);
00315         }
00316         if(countIterationsCurrent.equals(EvolutionStrategies.countRef - 1)){
00317             ifFactoryGenerator = new FactoryGenerator();
00318             Strategy.getStrategy().generator =
00319                 ifFactoryGenerator.createGenerator(GeneratorType.EvolutionStrategies);
00320         }
00321         if(countIterationsCurrent.equals(DistributionEstimationAlgorithm.countRef - 1)){
00322             ifFactoryGenerator = new FactoryGenerator();
00323             Strategy.getStrategy().generator =
00324                 ifFactoryGenerator.createGenerator(GeneratorType.DistributionEstimationAlgorithm);
00325         }
00326     }
00327 
00328     public Generator newGenerator(GeneratorType Generatortype) throws IllegalArgumentException,
00329         SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
00330         InvocationTargetException, NoSuchMethodException {
00331         ifFactoryGenerator = new FactoryGenerator();
00332         Generator generator = ifFactoryGenerator.createGenerator(Generatortype);
00333         return generator;
00334     }
00335 
00336     public State getBestState() {
00337         return bestState;
00338     }
00339 
00340     public void setBestState(State besState) {
00341         this.bestState = besState;
00342     }
00343 
00344     public StopExecute getStopexecute() {
00345         return stopexecute;
00346     }
00347 
00348     public int getCountMax() {
00349         return countMax;
00350     }
00351 
00352     public void setCountMax(int countMax) {
00353         this.countMax = countMax;
00354     }
00355 
00356     public void setStopexecute(StopExecute stopexecute) {
00357         this.stopexecute = stopexecute;
00358     }
00359 
00360     public UpdateParameter getUpdateparameter() {
00361         return updateparameter;
00362     }
00363 
00364     public void setUpdateparameter(UpdateParameter updateparameter) {
00365         this.updateparameter = updateparameter;
00366     }

```

```
00400
00405     public Problem getProblem() {
00406         return problem;
00407     }
00408
00413     public void setProblem(Problem problem) {
00414         this.problem = problem;
00415     }
00416
00421     public ArrayList<String> getListKey() {
00422         ArrayList<String> listKeys = new ArrayList<String>();
00423         String key = mapGenerators.keySet().toString();
00424         String returnString = key.substring(1, key.length() - 1);
00425         returnString = returnString + ", ";
00426         int countKey = mapGenerators.size();
00427         for (int i = 0; i < countKey; i++) {
00428             String r = returnString.substring(0, returnString.indexOf(',') );
00429             returnString = returnString.substring(returnString.indexOf(',') + 2);
00430             listKeys.add(r);
00431         }
00432         return listKeys;
00433     }
00434
00438     public void initializeGenerators()throws IllegalArgumentException, SecurityException,
00439     ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00440     NoSuchMethodException {
00441         List<GeneratorType> listType = new ArrayList<GeneratorType>();
00442         this.mapGenerators = new TreeMap<GeneratorType, Generator>();
00443         GeneratorType type[] = GeneratorType.values();
00444         for (GeneratorType generator : type) {
00445             listType.add(generator);
00446         }
00447         for (int i = 0; i < listType.size(); i++) {
00448             Generator generator = newGenerator(listType.get(i));
00449             mapGenerators.put(listType.get(i), generator);
00450         }
00451     }
00454     public void initialize()throws IllegalArgumentException, SecurityException,
00455     ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00456     NoSuchMethodException {
00457         List<GeneratorType> listType = new ArrayList<GeneratorType>();
00458         this.mapGenerators = new TreeMap<GeneratorType, Generator>();
00459         GeneratorType type[] = GeneratorType.values();
00460         for (GeneratorType generator : type) {
00461             listType.add(generator);
00462         }
00463         for (int i = 0; i < listType.size(); i++) {
00464             Generator generator = newGenerator(listType.get(i));
00465             mapGenerators.put(listType.get(i), generator);
00466         }
00467     }
00471     public int getCountCurrent() {
00472         return countCurrent;
00473     }
00474
00479     public void setCountCurrent(int countCurrent) {
00480         this.countCurrent = countCurrent;
00481     }
00482
00483     public static void destroyExecute() {
00484         strategy = null;
00485         RandomSearch.listStateReference.clear();
00486     }
00487
00492     public double getThreshold() {
00493         return threshold;
00494     }
00495
00500     public void setThreshold(double threshold) {
00501         this.threshold = threshold;
00502     }
00503
00509     public void calculateOffLinePerformance(float sumMax, int countOff) {
00510         float off = sumMax / countPeriodChange;
00511         listOfflineError[countOff] = off;
00512     }
00513
00518     public void updateRef(GeneratorType generatorType){
00519         if(generatorType.equals(GeneratorType.MultiGenerator)){
00520             updateRefMultiG();
00521             bestState = MultiGenerator.listStateReference.get(
00522                 MultiGenerator.listStateReference.size() - 1);
00523         } else{
00524             updateRefGenerator(generatorType);
00525         }
00526     }
00527 }
```

```

00525         bestState = generator.getReference();
00526     }
00527 }
00531 public void updateRefMultiG() {
00532     for (int i = 0; i < MultiGenerator.getListGenerators().length; i++) {
00533         updateRefGenerator(MultiGenerator.getListGenerators()[i]);
00534     }
00535 }
00540 public void updateRefGenerator(Generator generator) {
00541     if(generator.getType().equals(GeneratorType.HillClimbing) ||
00542         generator.getType().equals(GeneratorType.TabuSearch) ||
00543         generator.getType().equals(GeneratorType.RandomSearch) ||
00544         generator.getType().equals(GeneratorType.SimulatedAnnealing)){
00545         double evaluation =
00546             getProblem().getFunction().get(0).Evaluation(generator.getReference());
00547             generator.getReference().getEvaluation().set(0, evaluation);
00548         if(generator.getType().equals(GeneratorType.Geneticalgorithm) ||
00549             generator.getType().equals(GeneratorType.DistributionEstimationAlgorithm) ||
00550             generator.getType().equals(GeneratorType.EvolutionStrategies)){
00551             for (int j = 0; j < generator.getReferenceList().size(); j++) {
00552                 double evaluation =
00553                     getProblem().getFunction().get(0).Evaluation(generator.getReferenceList().get(j));
00554                     generator.getReferenceList().get(j).getEvaluation().set(0, evaluation);
00555             }
00556         }
00557     }
00558 }
00559 }
```

## 9.143 Referencia del archivo DistributionEstimationAlgorithm.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import java.util.Arrays;
import metaheuristics.strategy.Strategy;
import problem.definition.State;
import problem.definition.Problem.ProblemType;
import evolutionary_algorithms.complement.DistributionType;
import evolutionary_algorithms.complement.FatherSelection;
import evolutionary_algorithms.complement.Replace;
import evolutionary_algorithms.complement.ReplaceType;
import evolutionary_algorithms.complement.Sampling;
import evolutionary_algorithms.complement.SamplingType;
import evolutionary_algorithms.complement.SelectionType;
import factory_interface.IFFSampling;
import factory_interface.IFFFactoryFatherSelection;
import factory_interface.IFFFactoryReplace;
import factory_method.FactoryFatherSelection;
import factory_method.FactoryReplace;
import factory_method.FactorySampling;
```

Gráfico de dependencias incluidas en DistributionEstimationAlgorithm.java:



### Clases

- class [metaheuristics.generators.DistributionEstimationAlgorithm](#)

*DistributionEstimationAlgorithm - class for distribution estimation algorithms.*

## Paquetes

- package metaheuristics.generators

## 9.144 DistributionEstimationAlgorithm.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.List;
00006 import java.util.Arrays;
00007
00008 import metaheuristics.strategy.Strategy;
00009
00010 import problem.definition.State;
00011 import problem.definition.Problem.ProblemType;
00012
00013 import evolutionary_algorithms.complement.DistributionType;
00014 import evolutionary_algorithms.complement.FatherSelection;
00015 import evolutionary_algorithms.complement.Replace;
00016 import evolutionary_algorithms.complement.ReplaceType;
00017 import evolutionary_algorithms.complement.Sampling;
00018 import evolutionary_algorithms.complement.SamplingType;
00019 import evolutionary_algorithms.complement.SelectionType;
00020 import factory_interface.IFFSampling;
00021 import factory_interface.IFFactoryFatherSelection;
00022 import factory_interface.IFFactoryReplace;
00023 import factory_method.FactoryFatherSelection;
00024 import factory_method.FactoryReplace;
00025 import factory_method.FactorySampling;
00026
00027 public class DistributionEstimationAlgorithm extends Generator {
00028
00029     private State stateReferenceDA;
00030     private List<State> referenceList = new ArrayList<State>();
00031     private static final List<State> sonList = new ArrayList<State>();
00032     private IFFactoryFatherSelection iffatherselection;
00033     private IFFSampling ifffsampling;
00034     private IFFactoryReplace iffreplace;
00035     private DistributionType distributionType;
00036     private SamplingType Samplingtype;
00037
00038 //    private ReplaceType replaceType;
00039 //    public static final ReplaceType replaceType = ReplaceType.GenerationalReplace;
00040 //    public static final SelectionType selectionType = SelectionType.TruncationSelection;
00041
00042     private GeneratorType generatorType;
00043 //    private ProblemState candidate;
00044     public static final int truncation = 0;
00045     public static final int countRef = 0;
00046     private float weight;
00047
00048 //    //problemas dinamicos: use counters declared in superclass `Generator`
00049 //    private int[] listCountBetterGender = new int[10];
00050 //    private int[] listCountGender = new int[10];
00051 //    private float[] listTrace = new float[1200000];
00052
00053
00054
00055
00056
00057     public DistributionEstimationAlgorithm() {
00058         super();
00059         this.referenceList = getListStateRef(); // llamada al método que devuelve la lista.
00060         this.generatorType = GeneratorType.DistributionEstimationAlgorithm;
00061         this.distributionType = DistributionType.Univariate;
00062         this.Samplingtype = SamplingType.ProbabilisticSampling;
00063         this.weight = 50;
00064         listTrace[0] = weight;
00065         listCountBetterGender[0] = 0;
00066         listCountGender[0] = 0;
00067     }
00068
00069
00070
00071
00072     public State maxValue(List<State> listInd) {
00073         State state = new State(listInd.get(0));
00074         double max = state.getEvaluation().get(0);
00075         for (int i = 1; i < listInd.size(); i++) {
00076             if (listInd.get(i).getEvaluation().get(0) > max) {
00077                 max = listInd.get(i).getEvaluation().get(0);
00078                 state = new State(listInd.get(i));
00079             }
00080         }
00081     }
00082
00083
00084 }
```

```

00085         }
00086         return state;
00087     }
00088
00089     @Override
00090     public State generate(Integer operatornumber) throws IllegalArgumentException, SecurityException,
00091     ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00092     NoSuchMethodException {
00093
00094         List<State> fathers = getfathersList();
00095         iffsampling = new FactorySampling();
00096         Sampling samplingG = iffsampling.createSampling(Samplingtype);
00097         List<State> ind = samplingG.sampling(fathers, operatornumber);
00098         State candidate = null;
00099         if(ind.size() > 1){
00100             for (int i = 0; i < ind.size(); i++) {
00101                 double evaluation =
00102                     Strategy.getStrategy().getProblem().getFunction().get(0).Evaluation(ind.get(i));
00103                 ArrayList<Double> listEval = new ArrayList<Double>();
00104                 listEval.add(evaluation);
00105                 ind.get(0).setEvaluation(listEval);
00106             }
00107             candidate = maxValue(ind);
00108         }
00109     } else{
00110         candidate = ind.get(0);
00111     }
00112
00113     }
00114
00115     return candidate;
00116 }
00117
00118     @Override
00119     public State getReference() {
00120         stateReferenceDA = referenceList.get(0);
00121         if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00122             for (int i = 1; i < referenceList.size(); i++) {
00123                 if(stateReferenceDA.getEvaluation().get(0) <
00124                     referenceList.get(i).getEvaluation().get(0))
00125                     stateReferenceDA = referenceList.get(i);
00126             }
00127         } else{
00128             for (int i = 1; i < referenceList.size(); i++) {
00129                 if(stateReferenceDA.getEvaluation().get(0) >
00130                     referenceList.get(i).getEvaluation().get(0))
00131                     stateReferenceDA = referenceList.get(i);
00132             }
00133         }
00134         return stateReferenceDA;
00135     }
00136
00137     @Override
00138     public List<State> getReferenceList() {
00139         List<State> ReferenceList = new ArrayList<State>();
00140         for (int i = 0; i < referenceList.size(); i++) {
00141             State value = referenceList.get(i);
00142             ReferenceList.add( value);
00143         }
00144         return ReferenceList;
00145     }
00146
00147     @Override
00148     public GeneratorType getType() {
00149         return this.generatorType;
00150     }
00151
00152     @Override
00153     public void setInitialReference(State stateInitialRef) {
00154         this.stateReferenceDA = stateInitialRef;
00155     }
00156
00157     @Override
00158     public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
00159     IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00160     IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00161         iffreplace = new FactoryReplace();
00162         Replace replace = iffreplace.createReplace(replaceType);
00163         referenceList = replace.replace(stateCandidate, referenceList);
00164     }
00165
00166     public List<State> getListStateRef(){
00167         Boolean found = false;
00168         List<String> key = Strategy.getStrategy().getListKey();
00169         int count = 0;
00170         while((found.equals(false)) && (Strategy.getStrategy().mapGenerators.size() > count)){
00171             if(key.get(count).equals(GeneratorType.DistributionEstimationAlgorithm.toString())){
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830

```

```
00195         GeneratorType keyGenerator = GeneratorType.valueOf(String.valueOf(key.get(count)));
00196         DistributionEstimationAlgorithm generator =
00197             (DistributionEstimationAlgorithm)Strategy.getStrategy().mapGenerators.get(keyGenerator);
00198             if(generator.getListReference().isEmpty()){
00199                 referenceList.addAll(RandomSearch.listStateReference);
00200             }
00201             else{
00202                 referenceList = generator.getListReference();
00203             }
00204             found = true;
00205         }
00206         count++;
00207     }
00208 }
00209
00210     public List<State> getListReference() {
00211         return referenceList;
00212     }
00213
00214     public void setListReference(List<State> listReference) {
00215         referenceList = listReference;
00216     }
00217
00218     public GeneratorType getGeneratorType() {
00219         return generatorType;
00220     }
00221
00222     public void setGeneratorType(GeneratorType generatorType) {
00223         this.generatorType = generatorType;
00224     }
00225
00226     public List<State> getfathersList() throws IllegalArgumentException, SecurityException,
00227         ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00228         NoSuchMethodException {
00229         List<State> refList = new ArrayList<State>(this.referenceList);
00230         Iffatherselection = new FactoryFatherSelection();
00231         FatherSelection selection = iffatherselection.createSelectFather(selectionType);
00232         List<State> fathers = selection.selection(refList, truncation);
00233         return fathers;
00234     }
00235
00236     @Override
00237     public List<State> getSonList() {
00238         // return a defensive copy to avoid exposing internal mutable list
00239         return new ArrayList<State>(sonList);
00240     }
00241
00242     public boolean awardUpdateREF(State stateCandidate) {
00243         boolean find = false;
00244         int i = 0;
00245         while (find == false && i < this.referenceList.size()) {
00246             if(stateCandidate.equals(this.referenceList.get(i)))
00247                 find = true;
00248             else i++;
00249         }
00250         return find;
00251     }
00252
00253     @Override
00254     public float getWeight() {
00255         // TODO Auto-generated method stub
00256         return 0;
00257     }
00258
00259     @Override
00260     public void setWeight(float weight) {
00261         // TODO Auto-generated method stub
00262     }
00263
00264     public DistributionType getDistributionType() {
00265         return distributionType;
00266     }
00267
00268     public void setDistributionType(DistributionType distributionType) {
00269         this.distributionType = distributionType;
00270     }
00271
00272     @Override
00273     public int[] getListCountBetterGender() {
00274         // TODO Auto-generated method stub
00275         return (this.listCountBetterGender == null) ? new int[0] :
00276             Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
00277     }
00278
00279     @Override
```

```

00331     public int[] getListCountGender() {
00332         // TODO Auto-generated method stub
00333         return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00334             this.listCountGender.length);
00335     }
00336     @Override
00337     public float[] getTrace() {
00338         // TODO Auto-generated method stub
00339         return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00340             this.listTrace.length);
00341     }
00342 }
00343
00344 }
```

## 9.145 Referencia del archivo EvolutionStrategies.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import metaheuristics.strategy.Strategy;
import java.util.concurrent.ThreadLocalRandom;
import java.util.Arrays;
import problem.definition.State;
import problem.definition.Problem.Type;
import evolutionary_algorithms.complement.FatherSelection;
import evolutionary_algorithms.complement.Mutation;
import evolutionary_algorithms.complement.MutationType;
import evolutionary_algorithms.complement.Replace;
import evolutionary_algorithms.complement.ReplaceType;
import evolutionary_algorithms.complement.SelectionType;
import factory_interface.IFFactoryFatherSelection;
import factory_interface.IFFactoryMutation;
import factory_interface.IFFactoryReplace;
import factory_method.FactoryFatherSelection;
import factory_method.FactoryMutation;
import factory_method.FactoryReplace;
```

Gráfico de dependencias incluidas en EvolutionStrategies.java:



### Clases

- class [metaheuristics.generators.EvolutionStrategies](#)

*EvolutionStrategies - class that implements the Evolution Strategies generator.*

### Paquetes

- package [metaheuristics.generators](#)

## 9.146 EvolutionStrategies.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.List;
00006
00007 import metaheuristics.strategy.Strategy;
00008 import java.util.concurrent.ThreadLocalRandom;
00009 import java.util.Arrays;
00010
00011 import problem.definition.State;
00012 import problem.definition.Problem.Type;
00013 import evolutionary_algorithms.complement.FatherSelection;
00014 import evolutionary_algorithms.complement.Mutation;
00015 import evolutionary_algorithms.complement.Mutation.Type;
00016 import evolutionary_algorithms.complement.Replace;
00017 import evolutionary_algorithms.complement.Replace.Type;
00018 import evolutionary_algorithms.complement.Selection.Type;
00019 import factory_interface.IFFactoryFatherSelection;
00020 import factory_interface.IFFactoryMutation;
00021 import factory_interface.IFFactoryReplace;
00022 import factory_method.FactoryFatherSelection;
00023 import factory_method.FactoryMutation;
00024 import factory_method.FactoryReplace;
00025
00029 public class EvolutionStrategies extends Generator {
00030
00031     private State stateReferenceES;
00032     private List<State> listStateReference = new ArrayList<State>();
00033     private IFFactoryFatherSelection iffatherselection;
00034     private IFFactoryMutation iffactorymutation;
00035     private IFFactoryReplace iffreplace;
00036     private GeneratorType generatorType;
00037     public static final double PM = 0.0;
00038     public static final Mutation.Type mutationType = Mutation.Type.OnePointMutation;
00039     public static final Replace.Type replaceType = Replace.Type.GenerationalReplace;
00040     public static final Selection.Type selectionType = Selection.Type.TruncationSelection;
00041     public static final int countRef = 0;
00042     public static final int truncation = 0;
00043     private float weight = 50;
00044
00045     //problemas dinamicos
00046     //problemas dinamicos: use instance fields from Generator (countGender, countBetterGender)
00047     private int[] listCountBetterGender = new int[10];
00048     private int[] listCountGender = new int[10];
00049     private float[] listTrace = new float[1200000];
00050
00054     public EvolutionStrategies() {
00055         super();
00056         this.listStateReference = getListStateRef();
00057         this.generatorType = GeneratorType.EvolutionStrategies;
00058         this.weight = 50;
00059         listTrace[0] = this.weight;
00060         listCountBetterGender[0] = 0;
00061         listCountGender[0] = 0;
00062     }
00069     @SuppressWarnings("squid:S2245")
00070     @Override
00076     public State generate(Integer operatornumber) throws IllegalArgumentException, SecurityException,
00077             ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00078             NoSuchMethodException {
00078
00079         iffatherselection = new FactoryFatherSelection();
00080         FatherSelection selection = iffatherselection.createSelectFather(selectionType);
00081         List<State> fathers = selection.selection(this.listStateReference, truncation);
00082         int pos1 = (fathers.size() > 0) ? ThreadLocalRandom.current().nextInt(fathers.size()) : 0;
00083         State candidate = (State) Strategy.getStrategy().getProblem().getState().getCopy();
00084         candidate.setCode(new ArrayList<Object>(fathers.get(pos1).getCode()));
00085         candidate.setEvaluation(fathers.get(pos1).getEvaluation());
00086         candidate.setNumber(fathers.get(pos1).getNumber());
00087         candidate.setTypeGenerator(fathers.get(pos1).getTypeGenerator());
00088
00089         //*****mutacion*****
00090         iffactorymutation = new FactoryMutation();
00091         Mutation mutation = iffactorymutation.createMutation(mutationType);
00092         candidate = mutation.mutation(candidate, PM);
00093         //list.add(candidate);
00094         return candidate;
00095     }
00096     @Override
00101     public State getReference() {

```

```

00102     if (listStateReference == null || listStateReference.isEmpty()) {
00103         return null;
00104     }
00105     stateReferenceES = listStateReference.get(0);
00106     if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00107         for (int i = 1; i < listStateReference.size(); i++) {
00108             if(stateReferenceES.getEvaluation().get(0) <
00109                 listStateReference.get(i).getEvaluation().get(0))
00110                 stateReferenceES = listStateReference.get(i);
00111         }
00112     } else{
00113         for (int i = 1; i < listStateReference.size(); i++) {
00114             if(stateReferenceES.getEvaluation().get(0) >
00115                 listStateReference.get(i).getEvaluation().get(0))
00116                 stateReferenceES = listStateReference.get(i);
00117         }
00118     }
00119     return (stateReferenceES == null) ? null : new State(stateReferenceES);
00120 }
00121
00122 public void setStateRef(State stateRef) {
00123     this.stateReferenceES = (stateRef == null) ? null : new State(stateRef);
00124 }
00125
00126 @Override
00127 public GeneratorType getType() {
00128     return this.generatorType;
00129 }
00130
00131 @Override
00132 public void setInitialReference(State stateInitialRef) {
00133     this.stateReferenceES = (stateInitialRef == null) ? null : new State(stateInitialRef);
00134 }
00135
00136 @Override
00137 public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
00138     IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00139     IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00140     ifReplace = new FactoryReplace();
00141     Replace replace = ifReplace.createReplace(replaceType);
00142     listStateReference = replace.replace(stateCandidate, listStateReference);
00143 }
00144
00145 public List<State> getListStateRef(){
00146     Boolean found = false;
00147     List<String> key = Strategy.getStrategy().getListKey();
00148     int count = 0;
00149     while((found.equals(false)) && (Strategy.getStrategy().mapGenerators.size() > count)){
00150         if(key.get(count).equals(GeneratorType.EvolutionStrategies.toString())){
00151             GeneratorType keyGenerator = GeneratorType.valueOf(String.valueOf(key.get(count)));
00152             EvolutionStrategies generator = (EvolutionStrategies)
00153                 Strategy.getStrategy().mapGenerators.get(keyGenerator);
00154             if(generator.getListStateReference().isEmpty()){
00155                 listStateReference.addAll(RandomSearch.listStateReference);
00156             }
00157             else{
00158                 listStateReference = generator.getListStateReference();
00159             }
00160             found = true;
00161         }
00162         count++;
00163     }
00164     return (listStateReference == null) ? new ArrayList<State>() : new
00165     ArrayList<State>(listStateReference);
00166 }
00167
00168 public List<State> getListStateReference() {
00169     return (listStateReference == null) ? new ArrayList<State>() : new
00170     ArrayList<State>(listStateReference);
00171 }
00172
00173 public void setListStateReference(List<State> listStateReference) {
00174     this.listStateReference = (listStateReference == null) ? new ArrayList<State>() : new
00175     ArrayList<State>(listStateReference);
00176 }
00177
00178 public GeneratorType getTypeGenerator() {
00179     return generatorType;
00180 }
00181
00182 public void setTypeGenerator(GeneratorType generatorType) {
00183     this.generatorType = generatorType;
00184 }
00185
00186 @Override

```

```

00222     public List<State> getReferenceList() {
00223         List<State> ReferenceList = new ArrayList<State>();
00224         for (int i = 0; i < listStateReference.size(); i++) {
00225             State value = listStateReference.get(i);
00226             ReferenceList.add(value);
00227         }
00228         return new ArrayList<State>(ReferenceList);
00229     }
00230
00231     @Override
00232     public List<State> getSonList() {
00233         // TODO Auto-generated method stub
00234         return null;
00235     }
00236
00237     @Override
00238     public boolean awardUpdateREF(State stateCandidate) {
00239         // TODO Auto-generated method stub
00240         return false;
00241     }
00242
00243     @Override
00244     public float getWeight() {
00245         // TODO Auto-generated method stub
00246         return this.weight;
00247     }
00248
00249     @Override
00250     public void setWeight(float weight) {
00251         // TODO Auto-generated method stub
00252         this.weight = weight;
00253     }
00254
00255     @Override
00256     public int[] getListCountBetterGender() {
00257         // TODO Auto-generated method stub
00258         return (this.listCountBetterGender == null) ? new int[0] :
00259             Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
00260     }
00261
00262     @Override
00263     public int[] getListCountGender() {
00264         // TODO Auto-generated method stub
00265         return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00266             this.listCountGender.length);
00267     }
00268
00269     @Override
00270     public float[] getTrace() {
00271         // TODO Auto-generated method stub
00272         return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00273             this.listTrace.length);
00274     }
00275
00276     @Override
00277     public void setTrace(float[] trace) {
00278         // TODO Auto-generated method stub
00279         this.listTrace = trace;
00280     }
00281
00282     @Override
00283     public void setListCountGender(int[] listCountGender) {
00284         // TODO Auto-generated method stub
00285         this.listCountGender = listCountGender;
00286     }
00287
00288     @Override
00289     public void setListTrace(float[] listTrace) {
00290         // TODO Auto-generated method stub
00291         this.listTrace = listTrace;
00292     }
00293
00294     @Override
00295     public void setListCountBetterGender(int[] listCountBetterGender) {
00296         // TODO Auto-generated method stub
00297         this.listCountBetterGender = listCountBetterGender;
00298     }
00299
00300     @Override
00301     public void setReferenceList(List<State> referenceList) {
00302         this.referenceList = referenceList;
00303     }

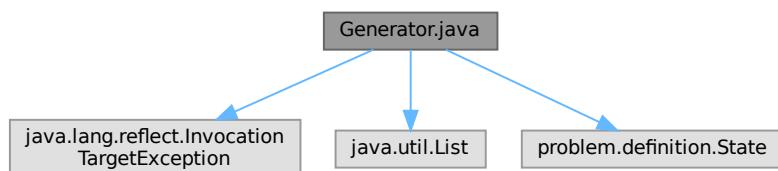
```

## 9.147 Referencia del archivo Generator.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.List;
import problem.definition.State;
Gráfico de dependencias incluidas en Generator.java:

```



## Clases

- class [metaheuristics.generators.Generator](#)

*Generator - abstract base class for solution generators in metaheuristic algorithms.*

## Paquetes

- package [metaheuristics.generators](#)

## 9.148 Generator.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003
00004 import java.lang.reflect.InvocationTargetException;
00005 import java.util.List;
00006
00007 import problem.definition.State;
00008
00009
00013 public abstract class Generator {
00014
00015     public abstract State generate(Integer operatornumber) throws IllegalArgumentException,
SecurityException, ClassNotFoundException, InstantiationException, IllegalAccessException,
InvocationTargetException, NoSuchMethodException;
00016
00017     public abstract void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
IllegalAccessException, InvocationTargetException, NoSuchMethodException;
00018
00019     public abstract State getReference();
00020
00021     public abstract void setInitialReference (State stateInitialRef);
00022
00023     public abstract GeneratorType getType ();
00024
00025     public abstract List<State> getReferenceList ();
00026
00027     public abstract List<State> getSonList ();
00028
00029     public abstract boolean awardUpdateREF(State stateCandidate);
00030
00031     public abstract void setWeight(float weight);
00032
00033     public abstract float getWeight();
00034
00035
00036     public abstract float[] getTrace();
00037     public int countGender;
00038     public int countBetterGender;
00039     public abstract int[] getListCountBetterGender();
00040     public abstract int[] getListCountGender();
00041
00042
00043 }
```

## 9.149 Referencia del archivo GeneratorType.java

## Clases

- enum [metaheuristics.generators.GeneratorType](#)

*GeneratorType - enumeration of different types of solution generators used in metaheuristic algorithms.*

## Paquetes

- package [metaheuristics.generators](#)

## 9.150 GeneratorType.java

[Ir a la documentación de este archivo.](#)

```
00001 package metaheuristics.generators;
00002
00006 public enum GeneratorType {
00007     HillClimbing, TabuSearch, SimulatedAnnealing, RandomSearch, LimitThreshold, HillClimbingRestart,
00008     //un punto
00009     GeneticAlgorithm, EvolutionStrategies, DistributionEstimationAlgorithm, ParticleSwarmOptimization,
00010     //poblaciones de puntos
00011     MultiGenerator,
00012     MultiobjectiveTabuSearch, MultiobjectiveStochasticHillClimbing, MultiCaseSimulatedAnnealing,
00013     MultiobjectiveHillClimbingRestart, MultiobjectiveHillClimbingDistance; //multiobjetivos
00011 }
```

## 9.151 Referencia del archivo GeneticAlgorithm.java

```
import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import metaheuristics.strategy.Strategy;
import java.util.concurrent.ThreadLocalRandom;
import java.util.Arrays;
import problem.definition.State;
import problem.definition.Problem.ProblemType;
import evolutionary_algorithms.complement.Crossover;
import evolutionary_algorithms.complement.CrossoverType;
import evolutionary_algorithms.complement.FatherSelection;
import evolutionary_algorithms.complement.Mutation;
import evolutionary_algorithms.complement.MutationType;
import evolutionary_algorithms.complement.Replace;
import evolutionary_algorithms.complement.ReplaceType;
import evolutionary_algorithms.complement.SelectionType;
import factory_interface.IFFactoryCrossover;
import factory_interface.IFFactoryFatherSelection;
import factory_interface.IFFactoryMutation;
import factory_interface.IFFactoryReplace;
import factory_method.FactoryCrossover;
import factory_method.FactoryFatherSelection;
import factory_method.FactoryMutation;
import factory_method.FactoryReplace;
```

Gráfico de dependencias incluidas en GeneticAlgorithm.java:



## Clases

- class [metaheuristics.generators.GeneticAlgorithm](#)
- GeneticAlgorithm - descripción (añade detalles).*

## Paquetes

- package [metaheuristics.generators](#)

## 9.152 GeneticAlgorithm.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.List;
00006
00007 import metaheuristics.strategy.Strategy;
00008 import java.util.concurrent.ThreadLocalRandom;
00009 import java.util.Arrays;
00010
00011 import problem.definition.State;
00012 import problem.definition.Problem.ProblemType;
00013
00014 import evolutionary_algorithms.complement.Crossover;
00015 import evolutionary_algorithms.complement.CrossoverType;
00016 import evolutionary_algorithms.complement.FatherSelection;
00017 import evolutionary_algorithms.complement.Mutation;
00018 import evolutionary_algorithms.complement.MutationType;
00019 import evolutionary_algorithms.complement.Replace;
00020 import evolutionary_algorithms.complement.ReplaceType;
00021 import evolutionary_algorithms.complement.SelectionType;
00022 import factory_interface.IFFactoryCrossover;
00023 import factory_interface.IFFactoryFatherSelection;
00024 import factory_interface.IFFactoryMutation;
00025 import factory_interface.IFFactoryReplace;
00026 import factory_method.FactoryCrossover;
00027 import factory_method.FactoryFatherSelection;
00028 import factory_method.FactoryMutation;
00029 import factory_method.FactoryReplace;
00030
00031
00032
00033
00034 public class GeneticAlgorithm extends Generator {
00035
00036     private State stateReferenceGA;
00037     private List<State> listState = new ArrayList<State>();
00038     private IFFactoryFatherSelection iffatherselection;
00039     private IFFactoryCrossover iffactorycrossover;
00040     private IFFactoryMutation iffactorymutation;
00041     private IFFactoryReplace iffreplace;
00042
00043
00044 //    private SelectionType selectionType;
00045 //    private CrossoverType crossoverType;
00046 //    private MutationType mutationType;
00047 //    private ReplaceType replaceType;
00048     private GeneratorType generatorType;
00049     public static final MutationType mutationType = MutationType.OnePointMutation;
00050     public static final CrossoverType crossoverType = CrossoverType.UniformCrossover;
00051     public static final ReplaceType replaceType = ReplaceType.GenerationalReplace;
00052     public static final SelectionType selectionType = SelectionType.TruncationSelection;
00053     public static final double PC = 0.6;
00054     public static final double PM = 0.01;
00055     public static final int countRef = 0;
00056     public static final int truncation = 0;
00057     private float weight;
00058
00059     //problemas dinamicos
00060     //problemas dinamicos: use instance fields from Generator (countGender, countBetterGender)
00061     private int[] listCountBetterGender = new int[10];
00062     private int[] listCountGender = new int[10];
00063     private float[] listTrace = new float[1200000];
00064
00065
00066
00067
00068     public GeneticAlgorithm() {
00069         super();
00070         this.listState = getListStateRef(); // llamada al método que devuelve la lista.
00071         this.generatorType = GeneratorType.GeneticAlgorithm;
00072         this.weight = 50;
00073         listTrace[0] = this.weight;
00074         listCountBetterGender[0] = 0;
00075         listCountGender[0] = 0;
00076     }
00077     @SuppressWarnings("squid:S2245")
00078     @Override

```

```

00090     public State generate(Integer operatornumber) throws IllegalArgumentException, SecurityException,
00091         ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00092         NoSuchMethodException {
00093
00094         //*****selection*****
00095         List<State> refList = new ArrayList<State>(this.listState);
00096         iffatherselection = new FactoryFatherSelection();
00097         FatherSelection selection = iffatherselection.createSelectFather(selectionType);
00098         List<State> fathers = selection.selection(refList, truncation);
00099         int pos1 = (fathers.size() > 0) ? ThreadLocalRandom.current().nextInt(fathers.size()) : 0;
00100        int pos2 = (fathers.size() > 0) ? ThreadLocalRandom.current().nextInt(fathers.size()) : 0;
00101
00102        State auxState1 = (State) Strategy.getStrategy().getProblem().getState().getCopy();
00103        auxState1.setCode(new ArrayList<Object>(fathers.get(pos1).getCode()));
00104        auxState1.setEvaluation(fathers.get(pos1).getEvaluation());
00105        auxState1.setNumber(fathers.get(pos1).getNumber());
00106        auxState1.setTypeGenerator(fathers.get(pos1).getTypeGenerator());
00107
00108        State auxState2 = (State) Strategy.getStrategy().getProblem().getState().getCopy();
00109        auxState2.setCode(new ArrayList<Object>(fathers.get(pos2).getCode()));
00110        auxState2.setEvaluation(fathers.get(pos2).getEvaluation());
00111        auxState2.setNumber(fathers.get(pos2).getNumber());
00112        auxState2.setTypeGenerator(fathers.get(pos2).getTypeGenerator());
00113
00114        //*****cruzamiento*****
00115        iffactorycrossover = new FactoryCrossover();
00116        Crossover crossover = iffactorycrossover.createCrossover(crossoverType);
00117        auxState1 = crossover.crossover(auxState1, auxState2, PC);
00118
00119        //*****mutacion*****
00120        iffactorymutation = new FactoryMutation();
00121        Mutation mutation = iffactorymutation.createMutation(mutationType);
00122        auxState1 = mutation.mutation(auxState1, PM);
00123        //list.add(auxState1);
00124
00125    }
00126
00127    @Override
00128    public State getReference() {
00129        if (listState == null || listState.isEmpty()) {
00130            return null;
00131        }
00132        stateReferenceGA = listState.get(0);
00133        if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00134            for (int i = 1; i < listState.size(); i++) {
00135                if(stateReferenceGA.getEvaluation().get(0) < listState.get(i).getEvaluation().get(0))
00136                    stateReferenceGA = listState.get(i);
00137            }
00138        } else{
00139            for (int i = 1; i < listState.size(); i++) {
00140                if(stateReferenceGA.getEvaluation().get(0) > listState.get(i).getEvaluation().get(0))
00141                    stateReferenceGA = listState.get(i);
00142            }
00143        }
00144        return (stateReferenceGA == null) ? null : new State(stateReferenceGA);
00145    }
00146
00147    public void setStateRef(State stateRef) {
00148        this.stateReferenceGA = (stateRef == null) ? null : new State(stateRef);
00149    }
00150
00151    @Override
00152    public void setInitialReference(State stateInitialRef) {
00153        this.stateReferenceGA = (stateInitialRef == null) ? null : new State(stateInitialRef);
00154    }
00155
00156    @Override
00157    public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
00158        IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00159        IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00160        iffreplace = new FactoryReplace();
00161        Replace replace = iffreplace.createReplace(replaceType);
00162        listState = replace.replace(stateCandidate, listState);
00163    }
00164
00165    public List<State> getListState() {
00166        return (listState == null) ? new ArrayList<State>() : new ArrayList<State>(listState);
00167    }
00168
00169    public void setListState(List<State> listState) {
00170        this.listState = (listState == null) ? new ArrayList<State>() : new
00171        ArrayList<State>(listState);
00172    }
00173
00174    public List<State> getListStateRef() {

```

```

00201     Boolean found = false;
00202     List<String> key = Strategy.getStrategy().getListKey();
00203     int count = 0;
00204
00205     while((found.equals(false)) && (Strategy.getStrategy().mapGenerators.size() > count)){
00206         if(key.get(count).equals(GeneratorType.GeneticAlgorithm.toString())){
00207             GeneratorType keyGenerator = GeneratorType.valueOf(String.valueOf(key.get(count)));
00208             GeneticAlgorithm generator = (GeneticAlgorithm)
00209                 Strategy.getStrategy().mapGenerators.get(keyGenerator);
00210             if(generator.getListState().isEmpty()){
00211                 listState.addAll(RandomSearch.listStateReference);
00212             }
00213             else{
00214                 listState = generator.getListState();
00215             }
00216             found = true;
00217         }
00218         count++;
00219     }
00220     return (listState == null) ? new ArrayList<State>() : new ArrayList<State>(listState);
00221 }
00222
00223     public GeneratorType getGeneratorType() {
00224         return generatorType;
00225     }
00226
00227     public void setGeneratorType(GeneratorType generatorType) {
00228         this.generatorType = generatorType;
00229     }
00230
00231     @Override
00232     public GeneratorType getType() {
00233         return this.generatorType;
00234     }
00235
00236     @Override
00237     public List<State> getReferenceList() {
00238         List<State> ReferenceList = new ArrayList<State>();
00239         for (int i = 0; i < listState.size(); i++) {
00240             State value = listState.get(i);
00241             ReferenceList.add(value);
00242         }
00243         return new ArrayList<State>(ReferenceList);
00244     }
00245
00246     @Override
00247     public List<State> getSonList() {
00248         // TODO Auto-generated method stub
00249         return null;
00250     }
00251
00252     @Override
00253     public boolean awardUpdateREF(State stateCandidate) {
00254         // TODO Auto-generated method stub
00255         return false;
00256     }
00257
00258     @Override
00259     public float getWeight() {
00260         // TODO Auto-generated method stub
00261         return this.weight;
00262     }
00263
00264     @Override
00265     public void setWeight(float weight) {
00266         // TODO Auto-generated method stub
00267         this.weight = weight;
00268     }
00269
00270     @Override
00271     public int[] getListCountBetterGender() {
00272         // TODO Auto-generated method stub
00273         return (this.listCountBetterGender == null) ? new int[0] :
00274             Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
00275     }
00276
00277     @Override
00278     public int[] getListCountGender() {
00279         // TODO Auto-generated method stub
00280         return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00281             this.listCountGender.length);
00282     }
00283
00284     @Override
00285     public float[] getTrace() {
00286         // TODO Auto-generated method stub
00287     }

```

```

00330     return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00331         this.listTrace.length);
00332
00333 }

```

## 9.153 Referencia del archivo HillClimbing.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import local_search.acceptation_type.AcceptType;
import local_search.acceptation_type.AcceptableCandidate;
import local_search.candidate_type.CandidateType;
import local_search.candidate_type.CandidateValue;
import local_search.complement.StrategyType;
import metaheuristics.strategy.Strategy;
import problem.definition.State;
import problem.definition.ProblemType;
import factory_interface.IFFactoryAcceptCandidate;
import factory_method.FactoryAcceptCandidate;

```

Gráfico de dependencias incluidas en HillClimbing.java:



### Clases

- class [metaheuristics.generators.HillClimbing](#)  
*HillClimbing* - class that implements the Hill Climbing metaheuristic.

### Paquetes

- package [metaheuristics.generators](#)

## 9.154 HillClimbing.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.List;
00006
00007
00008 import local_search.acceptation_type.AcceptType;
00009 import local_search.acceptation_type.AcceptableCandidate;
00010 import local_search.candidate_type.CandidateType;
00011 import local_search.candidate_type.CandidateValue;
00012 import local_search.complement.StrategyType;
00013 import metaheuristics.strategy.Strategy;
00014
00015 import problem.definition.State;
00016 import problem.definition.ProblemType;
00017
00018

```

```

00019 import factory_interface.IFFactoryAcceptCandidate;
00020 import factory_method.FactoryAcceptCandidate;
00021
00022
00023
00027 public class HillClimbing extends Generator{
00028
00029     protected CandidateValue candidatevalue;
00030     protected AcceptType typeAcceptation;
00031     protected StrategyType strategy;
00032     protected CandidateType typeCandidate;
00033     protected State stateReferenceHC;
00034     protected IFFactoryAcceptCandidate ifacceptCandidate;
00035     protected GeneratorType generatortype;
00036     protected List<State> listStateReference = new ArrayList<State>();
00037     protected float weight;
00038
00039     //problemas dinamicos
00040     //problemas dinamicos: use instance fields from Generator (countGender, countBetterGender)
00041     private int[] listCountBetterGender = new int[10];
00042     private int[] listCountGender = new int[10];
00043     private float[] listTrace = new float[1200000];
00044
00048     public HillClimbing() {
00049         super();
00050         this.typeAcceptation = AcceptType.AcceptBest;
00051         this.strategy = StrategyType.NORMAL;
00052         if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)) {
00053             this.typeCandidate = CandidateType.GreaterCandidate;
00054         }
00055         else{
00056             this.typeCandidate = CandidateType.SmallerCandidate;
00057         }
00058         this.candidatevalue = new CandidateValue();
00059         this.generatortype = GeneratorType.HillClimbing;
00060         this.weight = 50;
00061         listTrace[0] = this.weight;
00062         listCountBetterGender[0] = 0;
00063         listCountGender[0] = 0;
00064     }
00065
00066     @Override
00072     public State generate(Integer operatornumber) throws IllegalArgumentException, SecurityException,
        ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
        NoSuchMethodException {
00073         List<State> neighborhood =
        Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC, operatornumber);
00074         State statecandidate = candidatevalue.stateCandidate(stateReferenceHC, typeCandidate,
        strategy, operatornumber, neighborhood);
00075
00076         return statecandidate;
00077     }
00078
00079     @Override
00085     public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
        IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
        IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00086         ifacceptCandidate = new FactoryAcceptCandidate();
00087         AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00088         Boolean accept = candidate.acceptCandidate(stateReferenceHC, stateCandidate);
00089         if(accept.equals(true))
00090             stateReferenceHC = stateCandidate;
00091     }
00092
00093     @Override
00098     public List<State> getReferenceList() {
00099         if (stateReferenceHC != null) {
00100             // keep internal list updated but do not expose it directly
00101             if (listStateReference.isEmpty() || listStateReference.get(listStateReference.size() - 1)
00102                 != stateReferenceHC) {
00103                 listStateReference.add(stateReferenceHC);
00104             }
00105             return new ArrayList<State>(listStateReference);
00106         }
00107
00108     @Override
00113     public State getReference() {
00114         return (stateReferenceHC == null) ? null : new State(stateReferenceHC);
00115     }
00116
00121     public void setStateRef(State stateRef) {
00122         this.stateReferenceHC = (stateRef == null) ? null : new State(stateRef);
00123     }
00124
00125     @Override
00130     public void setInitialReference(State stateInitialRef) {

```

```
00131     this.stateReferenceHC = (stateInitialRef == null) ? null : new State(stateInitialRef);
00132 }
00133
00138     public GeneratorType getGeneratorType() {
00139         return generatortype;
00140     }
00141
00146     public void setGeneratorType(GeneratorType generatortype) {
00147         this.generatortype = generatortype;
00148     }
00149
00150     @Override
00155     public GeneratorType getType() {
00156         return this.generatortype;
00157     }
00158
00159     @Override
00164     public List<State> getSonList() {
00165         // TODO Auto-generated method stub
00166         return null;
00167     }
00168
00173     public void setTypeCandidate(CandidateType typeCandidate) {
00174         this.typeCandidate = typeCandidate;
00175     }
00176
00177     @Override
00183     public boolean awardUpdateREF(State stateCandidate) {
00184         // TODO Auto-generated method stub
00185         return false;
00186     }
00187
00188     @Override
00193     public float getWeight() {
00194         // TODO Auto-generated method stub
00195         return 0;
00196     }
00197
00198     @Override
00203     public void setWeight(float weight) {
00204         // TODO Auto-generated method stub
00205     }
00206
00207
00208     @Override
00213     public int[] getListCountBetterGender() {
00214         // TODO Auto-generated method stub
00215         return (this.listCountBetterGender == null) ? new int[0] :
00216             java.util.Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
00217     }
00218
00223     public int[] getListCountGender() {
00224         // TODO Auto-generated method stub
00225         return (this.listCountGender == null) ? new int[0] :
00226             java.util.Arrays.copyOf(this.listCountGender, this.listCountGender.length);
00227     }
00228
00233     public float[] getTrace() {
00234         // TODO Auto-generated method stub
00235         return (this.listTrace == null) ? new float[0] : java.util.Arrays.copyOf(this.listTrace,
00236             this.listTrace.length);
00236     }
00237 }
```

## 9.155 Referencia del archivo HillClimbingRestart.java

```
import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import factory_interface.IFFactoryAcceptCandidate;
import factory_method.FactoryAcceptCandidate;
import local_search.acceptation_type.AcceptType;
import local_search.acceptation_type.AcceptableCandidate;
import local_search.candidate_type.CandidateType;
import local_search.candidate_type.CandidateValue;
```

```

import local_search.complement.StrategyType;
import metaheuristics.strategy.Strategy;
import problem.definition.State;
import problem.definition.Problem.ProblemType;
Gráfico de dependencias incluidas en HillClimbingRestart.java:

```



## Clases

- class [metaheuristics.generators.HillClimbingRestart](#)

*HillClimbingRestart - class that implements the Hill Climbing Restart metaheuristic.*

## Paquetes

- package [metaheuristics.generators](#)

## 9.156 HillClimbingRestart.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.List;
00006
00007 import factory_interface.IFFactoryAcceptCandidate;
00008 import factory_method.FactoryAcceptCandidate;
00009
00010 import local_search.acceptation_type.AcceptType;
00011 import local_search.acceptation_type.AcceptableCandidate;
00012 import local_search.candidate_type.CandidateType;
00013 import local_search.candidate_type.CandidateValue;
00014 import local_search.complement.StrategyType;
00015 import metaheuristics.strategy.Strategy;
00016
00017 import problem.definition.State;
00018 import problem.definition.Problem.ProblemType;
00019
00023 public class HillClimbingRestart extends Generator{
00024
00025     int count = 0;
00026     int countCurrent;
00027     private List<State> listRef = new ArrayList<State>();
00028     protected CandidateValue candidatevalue;
00029     protected AcceptType typeAcceptation;
00030     protected StrategyType strategy;
00031     protected CandidateType typeCandidate;
00032     protected State stateReferenceHC;
00033     protected IFFactoryAcceptCandidate ifacceptCandidate;
00034     protected GeneratorType generatortype;
00035     protected List<State> listStateReference = new ArrayList<State>();
00036     protected float weight;
00037
00038     //problemas dinamicos
00039     //problemas dinamicos: use instance fields from Generator (countGender, countBetterGender)
00040     private int[] listCountBetterGender = new int[10];
00041     private int[] listCountGender = new int[10];
00042     private float[] listTrace = new float[12000000];
00043
00047     public HillClimbingRestart() {
00048         super();
00049         countCurrent = count;
00050         this.typeAcceptation = AcceptType.AcceptBest;
00051         this.strategy = StrategyType.NORMAL;
00052         if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)) {
```

```

00053         this.typeCandidate = CandidateType.GreaterCandidate;
00054     }
00055     else{
00056         this.typeCandidate = CandidateType.SmallerCandidate;
00057     }
00058     this.candidatevalue = new CandidateValue();
00059     this.generatortype = GeneratorType.HillClimbing;
00060     this.weight = 50;
00061     listTrace[0] = this.weight;
00062     listCountBetterGender[0] = 0;
00063     listCountGender[0] = 0;
00064 }
00065
00066
00067
00073     public State generate (Integer operatornumber) throws IllegalArgumentException, SecurityException,
00074     ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00075     NoSuchMethodException {
00076         State statecandidate;
00077         if(count == Strategy.getStrategy().getCountCurrent()){
00078             State stateR = new State(stateReferenceHC);
00079             listRef.add(stateR);
00080             stateReferenceHC =
00081             Strategy.getStrategy().getProblem().getOperator().generateRandomState(1).get(0);
00082             Strategy.getStrategy().getProblem().Evaluate(stateReferenceHC);
00083             count = count + countCurrent;
00084         }
00085         List<State> neighborhood =
00086         Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC, operatornumber);
00087         statecandidate = candidatevalue.stateCandidate(stateReferenceHC, typeCandidate, strategy,
00088         operatornumber, neighborhood);
00089         return statecandidate;
00090     }
00091
00092     @Override
00093     public void updateReference(State stateCandidate,
00094         Integer countIterationsCurrent) throws IllegalArgumentException,
00095         SecurityException, ClassNotFoundException, InstantiationException,
00096         IllegalAccessException, InvocationTargetException,
00097         NoSuchMethodException {
00098         // TODO Auto-generated method stub
00099         ifacceptCandidate = new FactoryAcceptCandidate();
00100         AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00101         Boolean accept = candidate.acceptCandidate(stateReferenceHC, stateCandidate);
00102         if(accept.equals(true))
00103             stateReferenceHC = stateCandidate;
00104     }
00105
00106
00107
00108     @Override
00109     public List<State> getReferenceList() {
00110         if (stateReferenceHC != null) {
00111             if (listStateReference.isEmpty() || listStateReference.get(listStateReference.size() - 1)
00112             != stateReferenceHC) {
00113                 listStateReference.add(stateReferenceHC);
00114             }
00115         }
00116
00117     @Override
00118     public State getReference() {
00119         return (stateReferenceHC == null) ? null : new State(stateReferenceHC);
00120     }
00121
00122
00123     public void setStateRef(State stateRef) {
00124         this.stateReferenceHC = (stateRef == null) ? null : new State(stateRef);
00125     }
00126
00127
00128     @Override
00129     public void setInitialReference(State stateInitialRef) {
00130         this.stateReferenceHC = (stateInitialRef == null) ? null : new State(stateInitialRef);
00131     }
00132
00133
00134     @Override
00135     public GeneratorType getGeneratorType() {
00136         return generatortype;
00137     }
00138
00139
00140     public void setGeneratorType(GeneratorType generatortype) {
00141         this.generatortype = generatortype;
00142     }
00143
00144
00145     public GeneratorType getType() {
00146         return this.generatortype;
00147     }
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166

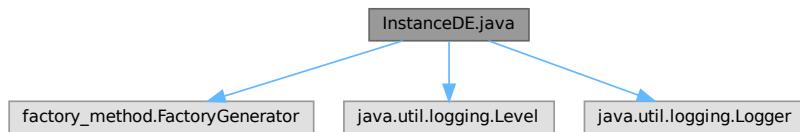
```

```

00167
00168     @Override
00169     public List<State> getSonList() {
00170         // TODO Auto-generated method stub
00171         return null;
00172     }
00173
00174     public void setTypeCandidate(CandidateType typeCandidate) {
00175         this.typeCandidate = typeCandidate;
00176     }
00177
00178     @Override
00179     public boolean awardUpdateREF(State stateCandidate) {
00180         // TODO Auto-generated method stub
00181         return false;
00182     }
00183
00184     @Override
00185     public float getWeight() {
00186         // TODO Auto-generated method stub
00187         return 0;
00188     }
00189
00190     @Override
00191     public void setWeight(float weight) {
00192         // TODO Auto-generated method stub
00193     }
00194
00195     @Override
00196     public int[] getListCountBetterGender() {
00197         // TODO Auto-generated method stub
00198         return (this.listCountBetterGender == null) ? new int[0] :
00199             java.util.Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
00200     }
00201
00202     @Override
00203     public int[] getListCountGender() {
00204         // TODO Auto-generated method stub
00205         return (this.listCountGender == null) ? new int[0] :
00206             java.util.Arrays.copyOf(this.listCountGender, this.listCountGender.length);
00207     }
00208
00209     @Override
00210     public float[] getTrace() {
00211         // TODO Auto-generated method stub
00212         return (this.listTrace == null) ? new float[0] : java.util.Arrays.copyOf(this.listTrace,
00213             this.listTrace.length);
00214     }
00215 }
00216 }
```

## 9.157 Referencia del archivo InstanceDE.java

import factory\_method.FactoryGenerator;  
 import java.util.logging.Level;  
 import java.util.logging.Logger;  
 Gráfico de dependencias incluidas en InstanceDE.java:



## Clases

- class [metaheuristics.generators.InstanceDE](#)

*InstanceDE* - class that implements the Runnable interface to create an instance of the Distribution Estimation Algorithm generator.

## Paquetes

- package [metaheuristics.generators](#)

## 9.158 InstanceDE.java

[Ir a la documentación de este archivo.](#)

```

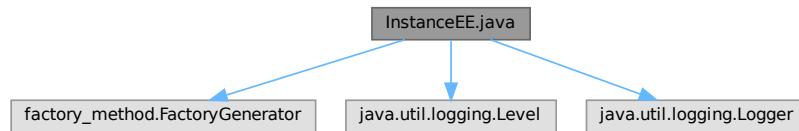
00001 package metaheuristics.generators;
00002
00003 import factory_method.FactoryGenerator;
00004 import java.util.logging.Level;
00005 import java.util.logging.Logger;
00006
00007
00012 public class InstanceDE implements Runnable {
00013
00014     private static final Logger LOGGER = Logger.getLogger(InstanceDE.class.getName());
00015
00016     private boolean terminate = false;
00017
00021     public void run() {
00022         FactoryGenerator iffFactoryGenerator = new FactoryGenerator();
00023         Generator generatorDE = null;
00024         try {
00025             generatorDE =
00026                 iffFactoryGenerator.createGenerator(GeneratorType.DistributionEstimationAlgorithm);
00027         } catch (Exception e) {
00028             // Log exception instead of printing stack trace so debug info is available
00029             // but the code is safe for production.
00030             LOGGER.log(Level.SEVERE, "Failed to create DistributionEstimationAlgorithm generator", e);
00031         }
00032         boolean find = false;
00033         int i = 0;
00034         while (find == false) {
00035             if(MultiGenerator.getListGenerators()[i].getType().equals(GeneratorType.DistributionEstimationAlgorithm)) {
00036                 MultiGenerator.getListGenerators()[i] = generatorDE;
00037             }
00038             else i++;
00039         }
00040         terminate = true;
00041     }
00042
00047     public boolean isTerminate() {
00048         return terminate;
00049     }
00050
00055     public void setTerminate(boolean terminate) {
00056         this.terminate = terminate;
00057     }
00058 }
```

## 9.159 Referencia del archivo InstanceEE.java

```

import factory_method.FactoryGenerator;
import java.util.logging.Level;
```

```
import java.util.logging.Logger;
Gráfico de dependencias incluidas en InstanceEE.java:
```



## Clases

- class [metaheuristics.generators.InstanceEE](#)

*InstanceEE - class that implements the Runnable interface to create an instance of.*

## Paquetes

- package [metaheuristics.generators](#)

## 9.160 InstanceEE.java

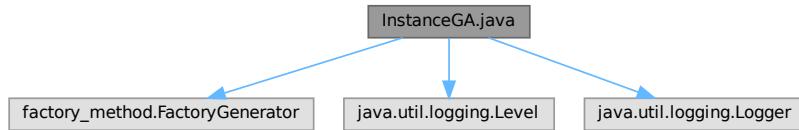
[Ir a la documentación de este archivo.](#)

```
00001 package metaheuristics.generators;
00002
00003 import factory_method.FactoryGenerator;
00004 import java.util.logging.Level;
00005 import java.util.logging.Logger;
00006
00010 public class InstanceEE implements Runnable {
00011
00012     private static final Logger LOGGER = Logger.getLogger(InstanceEE.class.getName());
00013
00014     private boolean terminate = false;
00015
00019     public void run() {
00020         FactoryGenerator ifFactoryGenerator = new FactoryGenerator();
00021         Generator generatoreEe = null;
00022         try {
00023             generatoreEe = ifFactoryGenerator.createGenerator(GeneratorType.EvolutionStrategies);
00024         } catch (Exception e) {
00025             // Log exception instead of printing stack trace to avoid debug output in production
00026             LOGGER.log(Level.SEVERE, "Failed to create EvolutionStrategies generator", e);
00027         }
00028         boolean find = false;
00029         int i = 0;
00030         while (find == false) {
00031             if(MultiGenerator.getListGenerators()[i].getType().equals(GeneratorType.EvolutionStrategies)){
00032                 MultiGenerator.getListGenerators()[i] = generatoreEe;
00033                 find = true;
00034             }
00035             else i++;
00036         }
00037         terminate = true;
00038     }
00039
00044     public boolean isTerminate() {
00045         return terminate;
00046     }
00047
00052     public void setTerminate(boolean terminate) {
00053         this.terminate = terminate;
00054     }
00055
00056 }
```

## 9.161 Referencia del archivo InstanceGA.java

```
import factory_method.FactoryGenerator;
import java.util.logging.Level;
import java.util.logging.Logger;
```

Gráfico de dependencias incluidas en InstanceGA.java:



### Clases

- class [metaheuristics.generators.InstanceGA](#)

*InstanceGA - class that implements the Runnable interface to create an instance of the Genetic Algorithm generator.*

### Paquetes

- package [metaheuristics.generators](#)

## 9.162 InstanceGA.java

[Ir a la documentación de este archivo.](#)

```
00001 package metaheuristics.generators;
00002
00003 import factory_method.FactoryGenerator;
00004 import java.util.logging.Level;
00005 import java.util.logging.Logger;
00006
00007
00012 public class InstanceGA implements Runnable {
00013
00014     private static final Logger LOGGER = Logger.getLogger(InstanceGA.class.getName());
00015
00016     private boolean terminate = false;
00017
00021     public void run() {
00022         FactoryGenerator iffFactoryGenerator = new FactoryGenerator();
00023         Generator generatorGA = null;
00024         try {
00025             generatorGA = iffFactoryGenerator.createGenerator(GeneratorType.GeneticAlgorithm);
00026         } catch (Exception e) {
00027             LOGGER.log(Level.SEVERE, "Failed to create GeneticAlgorithm generator", e);
00028         }
00029         boolean find = false;
00030         int i = 0;
00031         while (find == false) {
00032
00033             if(MultiGenerator.getListGenerators()[i].getType().equals(GeneratorType.GeneticAlgorithm)) {
00034                 MultiGenerator.getListGenerators()[i] = generatorGA;
00035             }
00036             else i++;
00037         }
00038         terminate = true;
00039     }
00040 }
```

```

00045     public boolean isTerminate() {
00046         return terminate;
00047     }
00048
00053     public void setTerminate(boolean terminate) {
00054         this.terminate = terminate;
00055     }
00056
00057 }
```

## 9.163 Referencia del archivo LimitRoulette.java

### Clases

- class [metaheuristics.generators.LimitRoulette](#)  
*LimitRoulette* - class that implements the Limit Roulette structure.

### Paquetes

- package [metaheuristics.generators](#)

## 9.164 LimitRoulette.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00006 public class LimitRoulette {
00007
00008     private float limitLow;
00009     private float limitHigh;
0010     private Generator generator;
0011
0016     public Generator getGenerator() {
0017         return generator;
0018     }
0023     public void setGenerator(Generator generator) {
0024         this.generator = generator;
0025     }
0030     public float getLimitHigh() {
0031         return limitHigh;
0032     }
0037     public void setLimitHigh(float limitHigh) {
0038         this.limitHigh = limitHigh;
0039     }
0044     public float getLimitLow() {
0045         return limitLow;
0046     }
0051     public void setLimitLow(float limitLow) {
0052         this.limitLow = limitLow;
0053     }
0054
0055 }
```

## 9.165 Referencia del archivo LimitThreshold.java

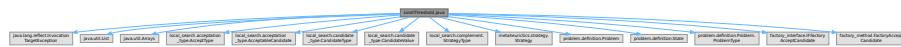
```

import java.lang.reflect.InvocationTargetException;
import java.util.List;
import java.util.Arrays;
import local_search.acceptation_type.AcceptType;
import local_search.acceptation_type.AcceptableCandidate;
```

```

import local_search.candidate_type.CandidateType;
import local_search.candidate_type.CandidateValue;
import local_search.complement.StrategyType;
import metaheuristics.strategy.Strategy;
import problem.definition.Problem;
import problem.definition.State;
import problem.definition.Problem.ProblemType;
import factory_interface.IFFactoryAcceptCandidate;
import factory_method.FactoryAcceptCandidate;
Gráfico de dependencias incluidas en LimitThreshold.java:

```



## Clases

- class [metaheuristics.generators.LimitThreshold](#)

*LimitThreshold - class that implements the Limit Threshold metaheuristic.*

## Paquetes

- package [metaheuristics.generators](#)

## 9.166 LimitThreshold.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.List;
00005 import java.util.Arrays;
00006
00007
00008 import local_search.acceptation_type.AcceptType;
00009 import local_search.acceptation_type.AcceptableCandidate;
00010 import local_search.candidate_type.CandidateType;
00011 import local_search.candidate_type.CandidateValue;
00012 import local_search.complement.StrategyType;
00013 import metaheuristics.strategy.Strategy;
00014
00015 import problem.definition.Problem;
00016 import problem.definition.State;
00017 import problem.definition.Problem.ProblemType;
00018
00019
00020
00021 import factory_interface.IFFactoryAcceptCandidate;
00022 import factory_method.FactoryAcceptCandidate;
00023
00024
00025 public class LimitThreshold extends Generator{
00026
00027     private CandidateValue candidatevalue;
00028     private AcceptType typeAcceptation;
00029     private StrategyType strategy;
00030     private CandidateType typeCandidate;
00031     private State stateReferenceLT;
00032     private IFFactoryAcceptCandidate ifacceptCandidate;
00033     private GeneratorType typeGenerator;
00034     private float weight;
00035
00036     //problemas dinamicos
00037     //problemas dinamicos: use instance fields from Generator (countGender, countBetterGender)
00038
00039     private int[] listCountBetterGender = new int[10];
00040
00041

```

```

00042     private int[] listCountGender = new int[10];
00043     private float[] listTrace = new float[1200000];
00044
00045     public GeneratorType getTypeGenerator() {
00046         return typeGenerator;
00047     }
00048
00049     public void setTypeGenerator(GeneratorType typeGenerator) {
00050         this.typeGenerator = typeGenerator;
00051     }
00052
00053     public LimitThreshold() {
00054         super();
00055         this.typeAcceptation = AcceptType.AcceptNotBadU;
00056         this.strategy = StrategyType.NORMAL;
00057
00058         Problem problem = Strategy.getStrategy().getProblem();
00059
00060         if(problem.getTypeProblem().equals(ProblemType.Maximizar)) {
00061             this.typeCandidate = CandidateType.GreaterCandidate;
00062         }
00063         else{
00064             this.typeCandidate = CandidateType.SmallerCandidate;
00065         }
00066
00067         this.candidatevalue = new CandidateValue();
00068         this.typeGenerator = GeneratorType.LimitThreshold;
00069         this.weight = (float) 50.0;
00070         listTrace[0] = weight;
00071         listCountBetterGender[0] = 0;
00072         listCountGender[0] = 0;
00073
00074     }
00075
00076     @Override
00077     public State generate(Integer operatornumber) throws IllegalArgumentException, SecurityException,
00078     ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00079     NoSuchMethodException {
00080         List<State> neighborhood =
00081             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceLT, operatornumber);
00082         State statecandidate = candidatevalue.stateCandidate(stateReferenceLT, typeCandidate,
00083             strategy, operatornumber, neighborhood);
00084         return statecandidate;
00085     }
00086
00087     @Override
00088     public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
00089     IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00090     IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00091         ifacceptCandidate = new FactoryAcceptCandidate();
00092         AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00093         Boolean accept = candidate.acceptCandidate(stateReferenceLT, stateCandidate);
00094         if(accept.equals(true)){
00095             stateReferenceLT = stateCandidate;
00096         }
00097     }
00098
00099     @Override
00100     public State getReference() {
00101         return stateReferenceLT;
00102     }
00103
00104     public void setStateRef(State stateRef) {
00105         this.stateReferenceLT = stateRef;
00106     }
00107
00108     @Override
00109     public void setInitialReference(State stateInitialRef) {
00110         this.stateReferenceLT = stateInitialRef;
00111     }
00112
00113
00114     @Override
00115     public GeneratorType getType() {
00116         return this.typeGenerator;
00117     }
00118
00119     @Override
00120     public List<State> getReferenceList() {
00121         return null;
00122     }
00123
00124     @Override
00125     public List<State> getSonList() {
00126         // TODO Auto-generated method stub
00127         return null;
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167

```

```

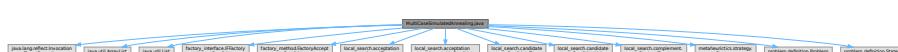
00168      }
00169
00170  public void setTypeCandidate(CandidateType typeCandidate) {
00171      this.typeCandidate = typeCandidate;
00172  }
00173
00174  @Override
00175  public boolean awardUpdateREF(State stateCandidate) {
00176      // TODO Auto-generated method stub
00177      return false;
00178  }
00179
00180  public float getWeight() {
00181      // TODO Auto-generated method stub
00182      return this.weight;
00183  }
00184
00185  @Override
00186  public void setWeight(float weight) {
00187      // TODO Auto-generated method stub
00188      this.weight = weight;
00189  }
00190
00191  @Override
00192  public int[] getListCountBetterGender() {
00193      // TODO Auto-generated method stub
00194      return (this.listCountBetterGender == null) ? new int[0] :
00195          Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
00196  }
00197
00198  @Override
00199  public int[] getListCountGender() {
00200      // TODO Auto-generated method stub
00201      return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00202          this.listCountGender.length);
00203  }
00204
00205  @Override
00206  public float[] getTrace() {
00207      // TODO Auto-generated method stub
00208      return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00209          this.listTrace.length);
00210  }
00211
00212  }
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239 }
```

## 9.167 Referencia del archivo MultiCaseSimulatedAnnealing.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import factory_interface.IFFactoryAcceptCandidate;
import factory_method.FactoryAcceptCandidate;
import local_search.acceptation_type.AcceptType;
import local_search.acceptation_type.AcceptableCandidate;
import local_search.candidate_type.CandidateType;
import local_search.candidate_type.CandidateValue;
import local_search.complement.StrategyType;
import metaheuristics.strategy.Strategy;
import problem.definition.Problem;
import problem.definition.State;
```

Gráfico de dependencias incluidas en MultiCaseSimulatedAnnealing.java:



### Clases

- class [metaheuristics.generators.MultiCaseSimulatedAnnealing](#)

[MultiCaseSimulatedAnnealing](#) - class that implements the Multi-Case Simulated Annealing metaheuristic.

## Paquetes

- package [metaheuristics.generators](#)

## 9.168 MultiCaseSimulatedAnnealing.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.List;
00006
00007 import factory_interface.IFFactoryAcceptCandidate;
00008 import factory_method.FactoryAcceptCandidate;
00009 import local_search.acceptation_type.AcceptType;
00010 import local_search.acceptation_type.AcceptableCandidate;
00011 import local_search.candidate_type.CandidateType;
00012 import local_search.candidate_type.CandidateValue;
00013 import local_search.complement.StrategyType;
00014 import metaheuristics.strategy.Strategy;
00015 import problem.definition.Problem;
00016 import problem.definition.State;
00017
00018
00019 public class MultiCaseSimulatedAnnealing extends Generator {
00020
00021     private CandidateValue candidatevalue;
00022     private AcceptType typeAcceptation;
00023     private StrategyType strategy;
00024     private CandidateType typeCandidate;
00025     private State stateReferenceSA;
00026     private IFFactoryAcceptCandidate ifacceptCandidate;
00027     // Cooling factor: fixed constant for update rule. Set a conservative default.
00028     public static final double alpha = 0.93;
00029     // Make initial temperature private and provide accessors so it can be protected/validated.
00030     private static Double tinitial = 250.0;
00031     public static final Double tfinal = 41.66;
00032     static int countIterationsT;
00033     private int countRept;
00034     private GeneratorType typeGenerator;
00035     private List<State> listStateReference = new ArrayList<State>();
00036     private float weight;
00037     private List<Float> listTrace = new ArrayList<Float>();
00038
00039     public GeneratorType getTypeGenerator() {
00040         return typeGenerator;
00041     }
00042
00043     public void setTypeGenerator(GeneratorType typeGenerator) {
00044         this.typeGenerator = typeGenerator;
00045     }
00046
00047     // Accessors for tinitial to avoid exposing a mutable static field directly.
00048     public static Double getTinitial() {
00049         return tinitial;
00050     }
00051     public static void setTinitial(Double t) {
00052         tinitial = t;
00053     }
00054
00055     public MultiCaseSimulatedAnnealing() {
00056         super();
00057         this.typeAcceptation = AcceptType.AcceptMulticase;
00058         this.strategy = StrategyType.NORMAL;
00059         this.typeCandidate = CandidateType.RandomCandidate;
00060         this.candidatevalue = new CandidateValue();
00061         this.typeGenerator = GeneratorType.MultiCaseSimulatedAnnealing;
00062         this.weight = 50;
00063         listTrace.add(weight);
00064     }
00065
00066     @Override

```

```
00090     public State generate(Integer operatornumber) throws IllegalArgumentException, SecurityException,
00091     ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00092     NoSuchMethodException {
00093         Problem problem = Strategy.getStrategy().getProblem();
00094         List<State> neighborhood = problem.getOperator().generatedNewState(stateReferenceSA,
00095         operatornumber);
00096         State statecandidate = candidatevalue.stateCandidate(stateReferenceSA, typeCandidate,
00097         strategy, operatornumber, neighborhood);
00098         return statecandidate;
00099     }
00100
00101     @Override
00102     public State getReference() {
00103         return stateReferenceSA;
00104     }
00105
00106     public void setStateRef(State stateRef) {
00107         this.stateReferenceSA = stateRef;
00108     }
00109
00110     @Override
00111     public void setInitialReference(State stateInitialRef) {
00112         this.stateReferenceSA = stateInitialRef;
00113     }
00114
00115     @Override
00116     public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
00117     IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00118     IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00119         countRept = MultiCaseSimulatedAnnealing.getCountIterationsT();
00120         ifacceptCandidate = new FactoryAcceptCandidate();
00121         AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00122         Boolean accept = candidate.acceptCandidate(stateReferenceSA, stateCandidate);
00123         if(accept.equals(true))
00124             stateReferenceSA = stateCandidate.clone();
00125         if(countIterationsCurrent.equals(MultiCaseSimulatedAnnealing.getCountIterationsT()))
00126             // update via accessors to avoid writing static fields inside an instance method
00127             MultiCaseSimulatedAnnealing.setTinitial(MultiCaseSimulatedAnnealing.getTinitial() *
00128             alpha);
00129             //Variante Fast MOSA
00130             System.out.println("La T:" + MultiCaseSimulatedAnnealing.getTinitial());
00131
00132         MultiCaseSimulatedAnnealing.setCountIterationsT(MultiCaseSimulatedAnnealing.getCountIterationsT() +
00133         countRept);
00134             System.out.println("La Cant es: " + MultiCaseSimulatedAnnealing.getCountIterationsT());
00135
00136         }
00137         getReferenceList();
00138
00139     }
00140
00141     public static int getCountIterationsT() {
00142         return countIterationsT;
00143     }
00144
00145     public static void setCountIterationsT(int c) {
00146         countIterationsT = c;
00147     }
00148
00149     @Override
00150     public GeneratorType getType() {
00151         return this.typeGenerator;
00152     }
00153
00154     @Override
00155     public List<State> getReferenceList() {
00156         listStateReference.add(stateReferenceSA.clone());
00157         return listStateReference;
00158     }
00159
00160     @Override
00161     public List<State> getSonList() {
00162         // TODO Auto-generated method stub
00163         return null;
00164     }
00165
00166     @Override
00167     public boolean awardUpdateREF(State stateCandidate) {
00168         // TODO Auto-generated method stub
00169         return false;
00170     }
00171
00172     @Override
00173     public float getWeight() {
00174         // TODO Auto-generated method stub
00175         return 0;
00176     }
00177
00178     @Override
```

```

00218     public void setWeight(float weight) {
00219         // TODO Auto-generated method stub
00220     }
00221
00222     @Override
00223     public int[] getListCountBetterGender() {
00224         // TODO Auto-generated method stub
00225         return new int[0];
00226     }
00227
00228     @Override
00229     public int[] getListCountGender() {
00230         // TODO Auto-generated method stub
00231         return new int[0];
00232     }
00233
00234     @Override
00235     public float[] getTrace() {
00236         // TODO Auto-generated method stub
00237         if (this.listTrace == null) return new float[0];
00238         float[] arr = new float[this.listTrace.size()];
00239         for (int i = 0; i < this.listTrace.size(); i++) {
00240             Float v = this.listTrace.get(i);
00241             arr[i] = (v == null) ? 0f : v.floatValue();
00242         }
00243         return arr;
00244     }
00245 }
00246
00247 }
00248
00249 }
```

## 9.169 Referencia del archivo MultiGenerator.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.concurrent.ThreadLocalRandom;
import java.util.logging.Level;
import java.util.logging.Logger;
import factory_method.FactoryGenerator;
import metaheuristics.strategy.Strategy;
import problem.definition.Problem.ProblemType;
import problem.definition.State;
Gráfico de dependencias incluidas en MultiGenerator.java:
```



### Clases

- class [metaheuristics.generators.MultiGenerator](#)  
*MultiGenerator - class that implements the Multi-Generator metaheuristic.*

### Paquetes

- package [metaheuristics.generators](#)

## 9.170 MultiGenerator.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.Arrays;
00006 import java.util.Collections;
00007 import java.util.List;
00008 import java.util.concurrent.ThreadLocalRandom;
00009 import java.util.logging.Level;
00010 import java.util.logging.Logger;
00011
00012 import factory_method.FactoryGenerator;
00013
00014 import metaheuristics.strategy.Strategy;
00015
00016 import problem.definition.Problem.ProblemType;
00017 import problem.definition.State;
00018
00019 public class MultiGenerator extends Generator implements Cloneable {
00020
00021     private GeneratorType generatortype;
00022     private static Generator[] listGenerators = new Generator[GeneratorType.values().length];
00023     public static final List<State> listGeneratedPP = Collections.synchronizedList(new
00024         ArrayList<State>());
00025     public static volatile Generator activeGenerator;
00026     public static final List<State> listStateReference = Collections.synchronizedList(new
00027         ArrayList<State>());
00028     private static final Logger LOGGER = Logger.getLogger(MultiGenerator.class.getName());
00029
00030     public void setGeneratorType(GeneratorType generatortype) {
00031         this.generatortype = generatortype;
00032     }
00033
00034     public MultiGenerator() {
00035         super();
00036         this.generatortype = GeneratorType.MultiGenerator;
00037     }
00038
00039     public static void destroyMultiGenerator() {
00040         listGeneratedPP.clear();
00041         //listGenerators.clear();
00042         listStateReference.clear();
00043         activeGenerator = null;
00044         listGenerators = null;
00045     }
00046
00047     public static void initializeListGenerator() throws IllegalArgumentException, SecurityException,
00048             ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00049             NoSuchMethodException {
00050         listGenerators = new Generator[4];
00051         Generator generator1 = new HillClimbing();
00052         Generator generator2 = new EvolutionStrategies();
00053         Generator generator3 = new LimitThreshold();
00054         Generator generator4 = new GeneticAlgorithm();
00055         listGenerators[0] = generator1;
00056         listGenerators[1] = generator2;
00057         listGenerators[2] = generator3;
00058         listGenerators[3] = generator4;
00059     }
00060
00061     public static void initializeGenerators() throws IllegalArgumentException, SecurityException,
00062             ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00063             NoSuchMethodException {
00064         initializeListGenerator();
00065         State stateREF = new State(Strategy.getStrategy().getProblem().getState());
00066         listStateReference.add(stateREF);
00067         for (int i = 0; i < listGenerators.length; i++) {
00068             if (((listGenerators[i].getType()).equals(GeneratorType.HillClimbing)) ||
00069                 (listGenerators[i].getType()).equals(GeneratorType.RandomSearch)) ||
00070                 (listGenerators[i].getType()).equals(GeneratorType.TabuSearch)) ||
00071                 (listGenerators[i].getType()).equals(GeneratorType.SimulatedAnnealing) ||
00072                 (listGenerators[i].getType()).equals(GeneratorType.LimitThreshold))){
00073                 listGenerators[i].setInitialReference(stateREF);
00074             }
00075         }
00076         createInstanceGeneratorsBPP();
00077         Strategy.getStrategy().listStates = MultiGenerator.getListGeneratedPP();
00078
00079         FactoryGenerator iffFactoryGeneratorEE = new FactoryGenerator();
00080         Generator generatorEE =
00081             iffFactoryGeneratorEE.createGenerator(GeneratorType.EvolutionStrategies);

```

```

00105
00106     FactoryGenerator iffFactoryGeneratorGA = new FactoryGenerator();
00107     Generator generatorGA = iffFactoryGeneratorGA.createGenerator(GeneratorType.GeneticAlgorithm);
00108
00109     FactoryGenerator iffFactoryGeneratorEDA = new FactoryGenerator();
00110     Generator generatorEDA =
00111         iffFactoryGeneratorEDA.createGenerator(GeneratorType.DistributionEstimationAlgorithm);
00112
00113     for (int i = 0; i < MultiGenerator.getListGenerators().length; i++) {
00114
00115         if(MultiGenerator.getListGenerators()[i].getType().equals(GeneratorType.EvolutionStrategies)){
00116             MultiGenerator.getListGenerators()[i] = generatorEE;
00117         }
00118
00119         if(MultiGenerator.getListGenerators()[i].getType().equals(GeneratorType.GeneticAlgorithm)){
00120             MultiGenerator.getListGenerators()[i] = generatorGA;
00121         }
00122
00123     }
00124
00125
00126     public static void createInstanceGeneratorsBPP() {
00127         Generator generator = new RandomSearch();
00128
00129         int j = 0;
00130         while (j < EvolutionStrategies.countRef){
00131             State stateCandidate;
00132             try {
00133                 stateCandidate = generator.generate(1);
00134                 Strategy.getStrategy().getProblem().Evaluate(stateCandidate);
00135                 stateCandidate.setNumber(j);
00136                 stateCandidate.setTypeGenerator(generator.getType());
00137                 listGeneratedPP.add(stateCandidate);
00138             } catch (Exception e) {
00139                 // Log exception instead of printing stack trace to avoid debug output in production
00140                 LOGGER.log(Level.SEVERE, "Failed to create state candidate in
00141 createInstanceGeneratorsBPP", e);
00142             }
00143             j++;
00144         }
00145     }
00146
00147 }
00148
00149     private static List<State> getListGeneratedPP() {
00150         return listGeneratedPP;
00151     }
00152
00153     public static Generator[] getListGenerators() {
00154         // return a defensive copy to avoid exposing internal static array
00155         return (listGenerators == null) ? null : Arrays.copyOf(listGenerators, listGenerators.length);
00156     }
00157     public static void setListGenerators(Generator[] listGenerators) {
00158         // store a defensive copy to avoid keeping a reference to caller's mutable array
00159         MultiGenerator.listGenerators = (listGenerators == null) ? null :
00160             Arrays.copyOf(listGenerators, listGenerators.length);
00161     }
00162     public static Generator getActiveGenerator() {
00163         return activeGenerator;
00164     }
00165     public static void setActiveGenerator(Generator activeGenerator) {
00166         MultiGenerator.activeGenerator = activeGenerator;
00167     }
00168     public static void setListGeneratedPP(List<State> newListGeneratedPP) {
00169         synchronized (listGeneratedPP) {
00170             listGeneratedPP.clear();
00171             if (newListGeneratedPP != null) {
00172                 listGeneratedPP.addAll(newListGeneratedPP);
00173             }
00174         }
00175     }
00176
00177     @Override
00178     public State generate(Integer operatornumber)
00179         throws IllegalArgumentException, SecurityException,
00180             ClassNotFoundException, InstantiationException,
00181             IllegalAccessException, InvocationTargetException,
00182             NoSuchMethodException {
00183
00184         // TODO Auto-generated method stub
00185         Strategy.getStrategy().generator = roulette();
00186         MultiGenerator.setActiveGenerator(Strategy.getStrategy().generator);
00187         activeGenerator.countGender++;
00188         State state = Strategy.getStrategy().generator.generate(1);
00189         return state;
00190     }

```

```

00225
00226     @Override
00227     public State getReference() {
00228         // TODO Auto-generated method stub
00229         return null;
00230     }
00231
00232
00233
00234
00235
00236     @Override
00237     public List<State> getReferenceList() {
00238         // TODO Auto-generated method stub
00239         return listStateReference;
00240     }
00241
00242
00243
00244
00245
00246     @Override
00247     public List<State> getSonList() {
00248         // TODO Auto-generated method stub
00249         return null;
00250     }
00251
00252
00253
00254
00255
00256     @Override
00257     public GeneratorType getType() {
00258         return this.generatortype;
00259     }
00260
00261
00262
00263
00264
00265     @Override
00266     public void setInitialReference(State stateInitialRef) {
00267         // TODO Auto-generated method stub
00268     }
00269
00270
00271
00272
00273
00274
00275
00276     @Override
00277     public void updateReference(State stateCandidate,
00278         Integer countIterationsCurrent) throws IllegalArgumentException,
00279         SecurityException, ClassNotFoundException, InstantiationException,
00280         IllegalAccessException, InvocationTargetException,
00281         NoSuchMethodException {
00282         // TODO Auto-generated method stub
00283         updateWeight(stateCandidate);
00284         tournament(stateCandidate, countIterationsCurrent);
00285     }
00286
00287
00288
00289     public void updateWeight(State stateCandidate) {
00290         boolean search = searchState(stateCandidate); // premio por calidad.
00291         if(search == false)
00292             updateAwardImp();
00293         else updateAwardSC();
00294     }
00295
00296
00297
00298     public boolean searchState(State stateCandidate) {
00299         if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00300             if(stateCandidate.getEvaluation().get(0) >
00301                 Strategy.getStrategy().getBestState().getEvaluation().get(0)){
00302                 if(stateCandidate.getEvaluation().get(0) >
00303                     Strategy.getStrategy().getBestState().getEvaluation().get(0))
00304                     activeGenerator.countBetterGender++;
00305                 return true;
00306             }
00307             else return false;
00308         }
00309         else {
00310             if(stateCandidate.getEvaluation().get(0) <
00311                 Strategy.getStrategy().getBestState().getEvaluation().get(0)){
00312                 if(stateCandidate.getEvaluation().get(0) <
00313                     Strategy.getStrategy().getBestState().getEvaluation().get(0))
00314                     activeGenerator.countBetterGender++;
00315                 return true;
00316             }
00317             else return false;
00318         }
00319
00320
00321     }
00322
00323
00324     @Override
00325     public float getWeight() {
00326         // TODO Auto-generated method stub
00327         return 0;
00328     }
00329
00330
00331
00332
00333     public Generator roulette() {
00334         float totalWeight = 0;
00335         for (int i = 0; i < listGenerators.length; i++) {
00336             totalWeight = listGenerators[i].getWeight() + totalWeight;
00337         }
00338         List<Float> listProb = new ArrayList<Float>();
00339         for (int i = 0; i < listGenerators.length; i++) {
00340             float probF = listGenerators[i].getWeight() / totalWeight;
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
01940
01941
01942
01943
01944
01945
01946
01947
01948
01949
01950
01951
01952
01953
01954
01955
01956
01957
01958
01959
01960
01961
01962
01963
01964
01965
01966
01967
01968
01969
01970
01971
01972
01973
01974
01975
01976
01977
01978
01979
01980
01981
01982
01983
01984
01985
01986
01987
01988

```

```

00345         listProb.add(probF);
00346     }
00347     List<LimitRoulette> listLimit = new ArrayList<LimitRoulette>();
00348     float limitHigh = 0;
00349     float limitLow = 0;
00350     for (int i = 0; i < listProb.size(); i++) {
00351         LimitRoulette limitRoulette = new LimitRoulette();
00352         limitHigh = listProb.get(i) + limitHigh;
00353         limitRoulette.setLimitHigh(limitHigh);
00354         limitRoulette.setLimitLow(limitLow);
00355         limitLow = limitHigh;
00356         limitRoulette.setGenerator(listGenerators[i]);
00357         listLimit.add(limitRoulette);
00358     }
00359     // Uses ThreadLocalRandom for algorithmic (non-cryptographic) randomness.
00360     // This RNG chooses a generator according to roulette weights and is not
00361     // used for security-sensitive purposes. Suppress Sonar hotspot S2245.
00362     @SuppressWarnings("squid:S2245")
00363     float numbAleatory = ThreadLocalRandom.current().nextFloat();
00364     boolean find = false;
00365     int i = 0;
00366     while ((find == false) && (i < listLimit.size())){
00367         if((listLimit.get(i).getLimitLow() <= numbAleatory) && (numbAleatory <=
00368             listLimit.get(i).getLimitHigh())){
00369             find = true;
00370         } else i++;
00371     }
00372     if (find) {
00373         return listLimit.get(i).getGenerator();
00374     }
00375     else return listLimit.get(listLimit.size() - 1).getGenerator();
00376 }
00377
00378 @Override
00384 public boolean awardUpdateREF(State stateCandidate) {
00385     // TODO Auto-generated method stub
00386     return false;
00387 }
00388
00389 @SuppressWarnings("static-access")
00393 public void updateAwardSC() {
00394     float weightLast = activeGenerator.getWeight();
00395     float weightUpdate = (float) (weightLast * (1 - 0.1) + 10);
00396     activeGenerator.setWeight(weightUpdate);
00397     for (int i = 0; i < listGenerators.length; i++) {
00398         if(listGenerators[i].equals(activeGenerator))
00399             activeGenerator.getTrace()[Strategy.getStrategy().getCountCurrent()] = weightUpdate;
00400         else{
00401             if(!listGenerators[i].getType().equals(GeneratorType.MultiGenerator)){
00402                 float trace = listGenerators[i].getWeight();
00403                 listGenerators[i].getTrace() [Strategy.getStrategy().getCountCurrent()] = trace;
00404             }
00405         }
00406     }
00407 }
00408
00409 @SuppressWarnings("static-access")
00413 public void updateAwardImp() {
00414     float weightLast = activeGenerator.getWeight();
00415     float weightUpdate = (float) (weightLast * (1 - 0.1));
00416     activeGenerator.setWeight(weightUpdate);
00417     for (int i = 0; i < listGenerators.length; i++) {
00418         if(listGenerators[i].equals(activeGenerator))
00419             activeGenerator.getTrace()[Strategy.getStrategy().getCountCurrent()] = weightUpdate;
00420         else{
00421             if(!listGenerators[i].getType().equals(GeneratorType.MultiGenerator)){
00422                 float trace = listGenerators[i].getWeight();
00423                 listGenerators[i].getTrace() [Strategy.getStrategy().getCountCurrent()] = trace;
00424             }
00425         }
00426     }
00427 }
00428
00429 @Override
00434 public void setWeight(float weight) {
00435     // TODO Auto-generated method stub
00436
00437 }
00438 @Override
00443 public float[] getTrace() {
00444     // TODO Auto-generated method stub
00445     // MultiGenerator does not maintain a single trace array; return empty array to avoid nulls
00446     return new float[0];
00447 }
00448
00449 @SuppressWarnings("static-access")

```

```

00455     public void tournament(State stateCandidate, Integer countIterationsCurrent) throws
00456         IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00457         IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00458             State stateTem = new State(stateCandidate);
00459             for (int i = 0; i < MultiGenerator.getListGenerators().length; i++) {
00460                 if(!listGenerators[i].getGeneratorType().equals(GeneratorType.MultiGenerator))
00461                     MultiGenerator.getListGenerators()[i].updateReference(stateTem,
00462                         countIterationsCurrent);
00463             }
00464         }
00465     @Override
00466     public Object clone() {
00467         try{
00468             return super.clone();
00469         } catch (CloneNotSupportedException e) {
00470             return new MultiGenerator();
00471         }
00472     }
00473
00474     @Override
00475     public int[] getListCountBetterGender() {
00476         // TODO Auto-generated method stub
00477         // MultiGenerator does not maintain these arrays itself; return empty array to avoid nulls
00478         return new int[0];
00479     }
00480
00481     @Override
00482     public int[] getListCountGender() {
00483         // TODO Auto-generated method stub
00484         // MultiGenerator does not maintain these arrays itself; return empty array to avoid nulls
00485         return new int[0];
00486     }
00487 }
00488 }
```

## 9.171 Referencia del archivo MultiobjectiveHillClimbingDistance.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import factory_interface.IFFactoryAcceptCandidate;
import factory_method.FactoryAcceptCandidate;
import problem.definition.State;
import local_search.acceptation_type.AcceptType;
import local_search.acceptation_type.AcceptableCandidate;
import local_search.candidate_type.CandidateType;
import local_search.candidate_type.CandidateValue;
import local_search.complement.StrategyType;
import metaheuristics.strategy.Strategy;
```

Gráfico de dependencias incluidas en MultiobjectiveHillClimbingDistance.java:



### Clases

- class [metaheuristics.generators.MultiobjectiveHillClimbingDistance](#)

*MultiobjectiveHillClimbingDistance - class that implements the Multiobjective Hill Climbing Distance metaheuristic.*

### Paquetes

- package [metaheuristics.generators](#)

## 9.172 MultiobjectiveHillClimbingDistance.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003
00004 import java.lang.reflect.InvocationTargetException;
00005 import java.util.ArrayList;
00006 import java.util.Iterator;
00007 import java.util.List;
00008 import factory_interface.IFFactoryAcceptCandidate;
00009 import factory_method.FactoryAcceptCandidate;
0010
0011 import problem.definition.State;
0012 import local_search.acceptation_type.AcceptType;
0013 import local_search.acceptation_type.AcceptableCandidate;
0014 import local_search.candidate_type.CandidateType;
0015 import local_search.candidate_type.CandidateValue;
0016 import local_search.complement.StrategyType;
0017 import metaheuristics.strategy.Strategy;
0018
0022 public class MultiobjectiveHillClimbingDistance extends Generator{
0023
0024     protected CandidateValue candidatevalue;
0025     protected AcceptType typeAcceptation;
0026     protected StrategyType strategy;
0027     protected CandidateType typeCandidate;
0028     protected State stateReferenceHC;
0029     protected IFFactoryAcceptCandidate ifacceptCandidate;
0030     protected GeneratorType generatortype;
0031     protected List<State> listStateReference = new ArrayList<State>();
0032     protected float weight;
0033     protected List<Float> listTrace = new ArrayList<Float>();
0034     private List<State> visitedState = new ArrayList<State>();
0035     private static final int sizeNeighbors = 10;
0036
0040     public static int getSizeNeighbors() {
0041         return sizeNeighbors;
0042     }
0043     //Lista que contiene las distancias de cada soluci&on del frente de Pareto estimado
0044     static List<Double> distanceSolution = new ArrayList<Double>();
0045
0046
0050     public MultiobjectiveHillClimbingDistance() {
0051         super();
0052         this.typeAcceptation = AcceptType.AcceptNotDominated;
0053         this.strategy = StrategyType.NORMAL;
0054         this.typeCandidate = CandidateType.NotDominatedCandidate;
0055         this.candidatevalue = new CandidateValue();
0056         this.generatortype = GeneratorType.MultiobjectiveHillClimbingDistance;
0057         this.stateReferenceHC = new State();
0058         this.weight = 50;
0059         listTrace.add(weight);
0060     }
0061
0062     @Override
0068     public State generate(Integer operatornumber) throws IllegalArgumentException, SecurityException,
0069     ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
0070     NoSuchMethodException {
0071         List<State> neighborhood =
0072             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC, operatornumber);
0073         State statecandidate = candidatevalue.stateCandidate(stateReferenceHC, typeCandidate,
0074             strategy, operatornumber, neighborhood);
0075         return statecandidate;
0076     }
0086     @Override
0087     public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
0088     IllegalArgumentException, SecurityException, ClassNotFoundException,
0089     InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException {
0090         //Agregando la primera soluci&n a la lista de soluciones no dominadas
0091         if(Strategy.getStrategy().listRefPoblacFinal.size() == 0){
0092             Strategy.getStrategy().listRefPoblacFinal.add(stateReferenceHC.clone());
0093             distanceSolution.add(Double.valueOf(0.0));
0094         }
0095         ifacceptCandidate = new FactoryAcceptCandidate();
0096         AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
0097         State lastState =
0098             Strategy.getStrategy().listRefPoblacFinal.get(Strategy.getStrategy().listRefPoblacFinal.size()-1);
0099         List<State> neighborhood =
0100             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC,
0101             getSizeNeighbors());
0102         int i= 0;
0103         Boolean accept = candidate.acceptCandidate(lastState, stateCandidate.clone());

```

```

00101         if(accept.equals(true)){
00102             stateReferenceHC = stateCandidate.clone();
00103             visitedState = new ArrayList<State>();
00104         }
00105
00106         else{
00107
00108             boolean stop = false;
00109             while (i < neighborhood.size()&& stop==false) {
00110                 if (contain(neighborhood.get(i))==false) {
00111                     stateReferenceHC = solutionMoreDistance(Strategy.getStrategy().listRefPoblacFinal,
00112                         distanceSolution);
00113                     visitedState.add(stateReferenceHC);
00114                     stop=true;
00115                     lastState=stateReferenceHC.clone();
00116                 }
00117                 i++;
00118             }
00119             int coutrestart=0;
00120             while (stop == false && coutrestart < getSizeNeighbors() && accept==false) {
00121                 stateCandidate =
00122                     Strategy.getStrategy().getProblem().getOperator().generateRandomState(1).get(0);
00123                 if (contain(stateCandidate)==false) {
00124                     Strategy.getStrategy().getProblem().Evaluate(stateCandidate);
00125                     visitedState.add(stateCandidate);
00126                     stop=true;
00127                     coutrestart++;
00128                     accept = candidate.acceptCandidate(lastState, stateCandidate.clone());
00129                 }
00130                 if(accept.equals(true)){
00131                     stateReferenceHC = stateCandidate.clone();
00132                     visitedState = new ArrayList<State>();
00133                     //tomar xc q pertenesca a la vecindad de xa
00134                 }
00135             }
00136             getReferenceList();
00137         }
00138     }
00139
00140     private State solutionMoreDistance(List<State> state, List<Double> distanceSolution) {
00141         Double max = (double) -1;
00142         int pos = -1;
00143         Double[] distance = distanceSolution.toArray(new Double[distanceSolution.size()]);
00144         State[] solutions = state.toArray(new State[state.size()]);
00145         for (int i = 0; i < distance.length; i++) {
00146             Double dist = distance[i];
00147             if(dist > max){
00148                 max = dist;
00149                 pos = i;
00150             }
00151         }
00152         if(pos != -1)
00153             return solutions[pos];
00154         else
00155             return null;
00156     }
00157
00158     @Override
00159     public List<State> getReferenceList() {
00160         listStateReference.add(stateReferenceHC.clone());
00161         return listStateReference;
00162     }
00163
00164     @Override
00165     public State getReference() {
00166         return stateReferenceHC;
00167     }
00168
00169     public void setStateRef(State stateRef) {
00170         this.stateReferenceHC = stateRef;
00171     }
00172
00173     @Override
00174     public void setInitialReference(State stateInitialRef) {
00175         this.stateReferenceHC = stateInitialRef;
00176     }
00177
00178     public GeneratorType getGeneratorType() {
00179         return generatortype;
00180     }
00181
00182     public void setGeneratorType(GeneratorType generatortype) {
00183         this.generatortype = generatortype;
00184     }
00185
00186     @Override
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
012210
012211
012212
012213
012214
012215
012216
012217
012218
012219
012220
012221
012222
012223
012224
012225
012226
012227
012228
012229
0122210
0122211
0122212
0122213
0122214
0122215
0122216
0122217
0122218
0122219
0122220
0122221
0122222
0122223
0122224
0122225
0122226
0122227
0122228
0122229
01222210
01222211
01222212
01222213
01222214
01222215
01222216
01222217
01222218
01222219
01222220
01222221
01222222
01222223
01222224
01222225
01222226
01222227
01222228
01222229
012222210
012222211
012222212
012222213
012222214
012222215
012222216
012222217
012222218
012222219
012222220
012222221
012222222
012222223
012222224
012222225
012222226
012222227
012222228
012222229
0122222210
0122222211
0122222212
0122222213
0122222214
0122222215
0122222216
0122222217
0122222218
0122222219
0122222220
0122222221
0122222222
0122222223
0122222224
0122222225
0122222226
0122222227
0122222228
0122222229
01222222210
01222222211
01222222212
01222222213
01222222214
01222222215
01222222216
01222222217
01222222218
01222222219
01222222220
01222222221
01222222222
01222222223
01222222224
01222222225
01222222226
01222222227
01222222228
01222222229
012222222210
012222222211
012222222212
012222222213
012222222214
012222222215
012222222216
012222222217
012222222218
012222222219
012222222220
012222222221
012222222222
012222222223
012222222224
012222222225
012222222226
012222222227
012222222228
012222222229
0122222222210
0122222222211
0122222222212
0122222222213
0122222222214
0122222222215
0122222222216
0122222222217
0122222222218
0122222222219
0122222222220
0122222222221
0122222222222
0122222222223
0122222222224
0122222222225
0122222222226
0122222222227
0122222222228
0122222222229
01222222222210
01222222222211
01222222222212
01222222222213
01222222222214
01222222222215
01222222222216
01222222222217
01222222222218
01222222222219
01222222222220
01222222222221
01222222222222
01222222222223
01222222222224
01222222222225
01222222222226
01222222222227
01222222222228
01222222222229
012222222222210
012222222222211
012222222222212
012222222222213
012222222222214
012222222222215
012222222222216
012222222222217
012222222222218
012222222222219
012222222222220
012222222222221
012222222222222
012222222222223
012222222222224
012222222222225
012222222222226
012222222222227
012222222222228
012222222222229
0122222222222210
0122222222222211
0122222222222212
0122222222222213
0122222222222214
0122222222222215
0122222222222216
0122222222222217
0122222222222218
0122222222222219
0122222222222220
0122222222222221
0122222222222222
0122222222222223
0122222222222224
0122222222222225
0122222222222226
0122222222222227
0122222222222228
0122222222222229
01222222222222210
01222222222222211
01222222222222212
01222222222222213
01222222222222214
01222222222222215
01222222222222216
01222222222222217
01222222222222218
01222222222222219
01222222222222220
01222222222222221
01222222222222222
01222222222222223
01222222222222224
01222222222222225
01222222222222226
01222222222222227
01222222222222228
01222222222222229
012222222222222210
012222222222222211
012222222222222212
012222222222222213
012222222222222214
012222222222222215
012222222222222216
012222222222222217
012222222222222218
012222222222222219
012222222222222220
012222222222222221
012222222222222222
012222222222222223
012222222222222224
012222222222222225
012222222222222226
012222222222222227
012222222222222228
012222222222222229
0122222222222222210
0122222222222222211
0122222222222222212
0122222222222222213
0122222222222222214
0122222222222222215
0122222222222222216
0122222222222222217
0122222222222222218
0122222222222222219
0122222222222222220
0122222222222222221
0122222222222222222
0122222222222222223
0122222222222222224
0122222222222222225
0122222222222222226
0122222222222222227
0122222222222222228
0122222222222222229
01222222222222222
```

```
00220     public GeneratorType getType() {
00221         return this.generatortype;
00222     }
00223
00224     @Override
00225     public List<State> getSonList() {
00226         // TODO Auto-generated method stub
00227         return null;
00228     }
00229
00230     public static List<Double> distanceCalculateAdd(List<State> solution) {
00231         State[] solutions = solution.toArray(new State[solution.size()]);
00232         Double distance = 0.0;
00233         List<Double>listDist=new ArrayList<Double>();
00234         State lastSolution = solution.get(solution.size()-1);
00235         //Actualizando las distancias de todos los elementos excepto el nuevo insertando
00236         for (int k = 0; k < solutions.length-1; k++) {
00237             State solA = solutions[k];
00238             distance = solA.Distance(lastSolution);
00239             listDist.add(distanceSolution.get(k)+distance);
00240             //Actualizando la distancia del ultimo elemento (elemento insertado) respecto al resto de los
00241             //elementos
00242             distanceSolution.set(k, distanceSolution.get(k) + distance);
00243         }
00244         distance = 0.0;
00245         //Calculando la distancia del ultimo elemento (elemento insertado) respecto al resto de los
00246         //elementos
00247         if (solutions.length==1) {
00248             return distanceSolution;
00249         }
00250
00251         }else {
00252
00253             for (int l = 0; l < solutions.length-1; l++) {
00254                 State solB = solutions[l];
00255                 distance += lastSolution.Distance(solB);
00256             }
00257             listDist.add(distance);
00258             //distanceSolution.add(distance);
00259             distanceSolution=listDist;
00260
00261             return distanceSolution;
00262         }
00263     }
00264 }
00265
00266
00267     private boolean contain(State state){
00268         boolean found = false;
00269         for (Iterator<State> iter = visitedState.iterator(); iter.hasNext();) {
00270             State element = (State) iter.next();
00271             if(element.Comparator(state)){
00272                 found = true;
00273             }
00274         }
00275         return found;
00276     }
00277
00278
00279     @Override
00280     public boolean awardUpdateREF(State stateCandidate) {
00281         // TODO Auto-generated method stub
00282         return false;
00283     }
00284
00285
00286     @Override
00287     public float getWeight() {
00288         // TODO Auto-generated method stub
00289         return 0;
00290     }
00291
00292
00293     @Override
00294     public void setWeight(float weight) {
00295         // TODO Auto-generated method stub
00296     }
00297
00298
00299     @Override
00300     public int[] getListCountBetterGender() {
00301         // This generator doesn't maintain listCount arrays; return empty array to avoid nulls
00302         return new int[0];
00303     }
00304
00305
00306     @Override
00307     public int[] getListCountGender() {
00308         // This generator doesn't maintain listCount arrays; return empty array to avoid nulls
00309         return new int[0];
00310     }
00311
00312
00313     @Override
00314     public int[] getListCountGender() {
00315         // This generator doesn't maintain listCount arrays; return empty array to avoid nulls
00316         return new int[0];
00317     }
00318
00319
00320     @Override
00321     public int[] getListCountGender() {
00322         // This generator doesn't maintain listCount arrays; return empty array to avoid nulls
00323         return new int[0];
00324     }
00325
00326
00327     @Override
00328     public int[] getListCountGender() {
00329         // This generator doesn't maintain listCount arrays; return empty array to avoid nulls
00330         return new int[0];
00331     }
00332
00333
00334     @Override
```

```

00340     public float[] getTrace() {
00341         // TODO Auto-generated method stub
00342         if (this.listTrace == null) return new float[0];
00343         float[] arr = new float[this.listTrace.size()];
00344         for (int i = 0; i < this.listTrace.size(); i++) {
00345             Float v = this.listTrace.get(i);
00346             arr[i] = (v == null) ? 0f : v.floatValue();
00347         }
00348         return arr;
00349     }
00350 }

```

## 9.173 Referencia del archivo MultiobjectiveHillClimbingRestart.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import factory_interface.IFFactoryAcceptCandidate;
import factory_method.FactoryAcceptCandidate;
import problem.definition.State;
import local_search.acceptation_type.AcceptType;
import local_search.acceptation_type.AcceptableCandidate;
import local_search.candidate_type.CandidateType;
import local_search.candidate_type.CandidateValue;
import local_search.complement.StrategyType;
import metaheuristics.strategy.Strategy;

```

Gráfico de dependencias incluidas en MultiobjectiveHillClimbingRestart.java:



### Clases

- class [metaheuristics.generators.MultiobjectiveHillClimbingRestart](#)

*MultiobjectiveHillClimbingRestart - class that implements the Multiobjective Hill Climbing with Restart metaheuristic.*

### Paquetes

- package [metaheuristics.generators](#)

## 9.174 MultiobjectiveHillClimbingRestart.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.Iterator;
00006 import java.util.List;
00007
00008 import factory_interface.IFFactoryAcceptCandidate;
00009 import factory_method.FactoryAcceptCandidate;
0010
0011 import problem.definition.State;
0012

```

```

00013 import local_search.acceptation_type.AcceptType;
00014 import local_search.acceptation_type.AcceptableCandidate;
00015 import local_search.candidate_type.CandidateType;
00016 import local_search.candidate_type.CandidateValue;
00017 import local_search.complement.StrategyType;
00018 import metaheuristics.strategy.Strategy;
00019
00020
00021
00022
00023 public class MultiobjectiveHillClimbingRestart extends Generator{
00024
00025     protected CandidateValue candidatevalue;
00026     protected AcceptType typeAcceptation;
00027     protected StrategyType strategy;
00028     protected CandidateType typeCandidate;
00029     protected State stateReferenceHC;
00030     protected IFFactoryAcceptCandidate ifacceptCandidate;
00031     protected GeneratorType generatorType;
00032     protected List<State> listStateReference = new ArrayList<State>();
00033     protected float weight;
00034     protected List<Float> listTrace = new ArrayList<Float>();
00035     private List<State> visitedState = new ArrayList<State>();
00036     private static final int sizeNeighbors = 10;
00037
00038     public static int getSizeNeighbors() {
00039         return sizeNeighbors;
00040     }
00041
00042
00043     public MultiobjectiveHillClimbingRestart() {
00044         super();
00045         this.typeAcceptation = AcceptType.AcceptNotDominated;
00046         this.strategy = StrategyType.NORMAL;
00047         //Problem problem = Strategy.getStrategy().getProblem();
00048         this.typeCandidate = CandidateType.NotDominatedCandidate;
00049         this.candidatevalue = new CandidateValue();
00050         this.generatorType = GeneratorType.MultiobjectiveHillClimbingRestart;
00051         this.stateReferenceHC = new State();
00052         this.weight = 50;
00053         listTrace.add(weight);
00054     }
00055
00056     @Override
00057     public State generate(Integer operatorNumber) throws IllegalArgumentException, SecurityException,
00058     ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00059     NoSuchMethodException {
00060         List<State> neighborhood =
00061             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC, operatorNumber);
00062         State statecandidate = candidatevalue.stateCandidate(stateReferenceHC, typeCandidate,
00063             strategy, operatorNumber, neighborhood);
00064         return statecandidate;
00065     }
00066
00067     @Override
00068     public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
00069     IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00070     IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00071         //Agregando la primera solución a la lista de soluciones no dominadas
00072
00073         if(Strategy.getStrategy().listRefPoblacFinal.size() == 0){
00074             Strategy.getStrategy().listRefPoblacFinal.add(stateReferenceHC.clone());
00075         }
00076
00077         ifacceptCandidate = new FactoryAcceptCandidate();
00078         AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00079         State lastState =
00080             Strategy.getStrategy().listRefPoblacFinal.get(Strategy.getStrategy().listRefPoblacFinal.size()-1);
00081         List<State> neighborhood =
00082             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC,
00083                 getSizeNeighbors());
00084         int i = 0;
00085
00086         Boolean accept = candidate.acceptCandidate(lastState, stateCandidate.clone());
00087
00088         if(accept.equals(true)){
00089             stateReferenceHC = stateCandidate.clone();
00090             visitedState = new ArrayList<State>();
00091             //tomar xc q pertenezca a la vecindad de xa
00092         }
00093         else{
00094             boolean stop = false;
00095             while (i < neighborhood.size() && stop==false) {
00096                 if (contain(neighborhood.get(i))==false) {
00097                     stateCandidate = neighborhood.get(i);
00098                     Strategy.getStrategy().getProblem().Evaluate(stateCandidate);
00099                     visitedState.add(stateCandidate);
00100                 }
00101             }
00102         }
00103     }
00104
00105     boolean contain(State s) {
00106         for (State state : visitedState) {
00107             if (state.equals(s)) {
00108                 return true;
00109             }
00110         }
00111         return false;
00112     }
00113 }

```

```

00110             accept = candidate.acceptCandidate(lastState, stateCandidate.clone());
00111             stop=true;
00112         }
00113         i++;
00114     }
00115     while (stop == false) {
00116         stateCandidate =
00117             Strategy.getStrategy().getProblem().getOperator().generateRandomState(1).get(0);
00118         if (contain(stateCandidate)==false) {
00119             Strategy.getStrategy().getProblem().Evaluate(stateCandidate);
00120             stop=true;
00121             accept = candidate.acceptCandidate(lastState, stateCandidate.clone());
00122         }
00123     }
00124     if(accept.equals(true)){
00125         stateReferenceHC = stateCandidate.clone();
00126         visitedState = new ArrayList<State>();
00127         //tomar xc q pertenesca a la vecindad de xa
00128     }
00129 }
00130
00131     getReferenceList();
00132 }
00133
00134
00135 @Override
00140 public List<State> getReferenceList() {
00141     listStateReference.add(stateReferenceHC.clone());
00142     return listStateReference;
00143 }
00144
00145 @Override
00150 public State getReference() {
00151     return stateReferenceHC;
00152 }
00153
00158 public void setStateRef(State stateRef) {
00159     this.stateReferenceHC = stateRef;
00160 }
00161
00162 @Override
00167 public void setInitialReference(State stateInitialRef) {
00168     this.stateReferenceHC = stateInitialRef;
00169 }
00170
00175 public GeneratorType getGeneratorType() {
00176     return generatortype;
00177 }
00178
00183 public void setGeneratorType(GeneratorType generatortype) {
00184     this.generatortype = generatortype;
00185 }
00186
00187 @Override
00192 public GeneratorType getType() {
00193     return this.generatortype;
00194 }
00195
00196 @Override
00201 public List<State> getSonList() {
00202     // TODO Auto-generated method stub
00203     return null;
00204 }
00205
00211 private boolean contain(State state){
00212     boolean found = false;
00213     for (Iterator<State> iter = visitedState.iterator(); iter.hasNext();) {
00214         State element = (State) iter.next();
00215         if(element.Comparator(state)==true){
00216             found = true;
00217         }
00218     }
00219     return found;
00220 }
00221
00222 @Override
00228 public boolean awardUpdateREF(State stateCandidate) {
00229     // TODO Auto-generated method stub
00230     return false;
00231 }
00232
00233
00234 @Override
00239 public float getWeight() {
00240     // TODO Auto-generated method stub
00241     return 0;

```

```

00242     }
00243
00244     @Override
00245     public void setWeight(float weight) {
00246         // TODO Auto-generated method stub
00247     }
00248
00249     @Override
00250     public float[] getTrace() {
00251         // TODO Auto-generated method stub
00252         if (this.listTrace == null) return new float[0];
00253         float[] arr = new float[this.listTrace.size()];
00254         for (int i = 0; i < this.listTrace.size(); i++) {
00255             Float v = this.listTrace.get(i);
00256             arr[i] = (v == null) ? 0f : v.floatValue();
00257         }
00258         return arr;
00259     }
00260
00261     @Override
00262     public int[] getListCountBetterGender() {
00263         // This generator doesn't maintain listCount arrays; return empty array to avoid nulls
00264         return new int[0];
00265     }
00266
00267     @Override
00268     public int[] getListCountGender() {
00269         // This generator doesn't maintain listCount arrays; return empty array to avoid nulls
00270         return new int[0];
00271     }
00272 }
00273
00274 }
```

## 9.175 Referencia del archivo MultiobjectiveStochasticHillClimbing.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import factory_interface.IFFactoryAcceptCandidate;
import factory_method.FactoryAcceptCandidate;
import local_search.acceptation_type.AcceptType;
import local_search.acceptation_type.AcceptableCandidate;
import local_search.candidate_type.CandidateType;
import local_search.candidate_type.CandidateValue;
import local_search.complement.StrategyType;
import metaheuristics.strategy.Strategy;
import problem.definition.State;
```

Gráfico de dependencias incluidas en MultiobjectiveStochasticHillClimbing.java:



### Clases

- class [metaheuristics.generators.MultiobjectiveStochasticHillClimbing](#)

*MultiobjectiveStochasticHillClimbing - class that implements the Multiobjective Stochastic Hill Climbing metaheuristic.*

### Paquetes

- package [metaheuristics.generators](#)

## 9.176 MultiobjectiveStochasticHillClimbing.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.List;
00006
00007 import factory_interface.IFFactoryAcceptCandidate;
00008 import factory_method.FactoryAcceptCandidate;
00009 import local_search.acceptation_type.AcceptType;
00010 import local_search.acceptation_type.AcceptableCandidate;
00011 import local_search.candidate_type.CandidateType;
00012 import local_search.candidate_type.CandidateValue;
00013 import local_search.complement.StrategyType;
00014 import metaheuristics.strategy.Strategy;
00015 import problem.definition.State;
00016
00017
00021 public class MultiobjectiveStochasticHillClimbing extends Generator{
00022
00023     protected CandidateValue candidatevalue;
00024     protected AcceptType typeAcceptation;
00025     protected StrategyType strategy;
00026     protected CandidateType typeCandidate;
00027     protected State stateReferenceHC;
00028     protected IFFactoryAcceptCandidate ifacceptCandidate;
00029     protected GeneratorType generatorType;
00030     protected List<State> listStateReference = new ArrayList<State>();
00031     protected float weight;
00032     protected List<Float> listTrace = new ArrayList<Float>();
00033
00037     public MultiobjectiveStochasticHillClimbing() {
00038         super();
00039         this.typeAcceptation = AcceptType.AcceptNotDominated;
00040         this.strategy = StrategyType.NORMAL;
00041         this.typeCandidate = CandidateType.NotDominatedCandidate;
00042         this.candidatevalue = new CandidateValue();
00043         this.generatorType = GeneratorType.MultiobjectiveStochasticHillClimbing;
00044         this.stateReferenceHC = new State();
00045         this.weight = 50;
00046         listTrace.add(weight);
00047     }
00048
00049     @Override
00055     public State generate(Integer operatornumber) throws IllegalArgumentException, SecurityException,
        ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
        NoSuchMethodException {
00056         List<State> neighborhood =
        Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceHC, operatornumber);
00057         State statecandidate = candidatevalue.stateCandidate(stateReferenceHC, typeCandidate,
        strategy, operatornumber, neighborhood);
00058         return statecandidate;
00059     }
00060
00061     @Override
00067     public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
        IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
        IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00068         ifacceptCandidate = new FactoryAcceptCandidate();
00069         AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00070         Boolean accept = candidate.acceptCandidate(stateReferenceHC, stateCandidate);
00071         if(accept.equals(true))
00072             stateReferenceHC = stateCandidate.clone();
00073         getReferenceList();
00074     }
00075
00081     @Override
00081     public List<State> getReferenceList() {
00082         listStateReference.add( stateReferenceHC.clone());
00083         return listStateReference;
00084     }
00085
00086     @Override
00091     public State getReference() {
00092         return stateReferenceHC;
00093     }
00094
00099     public void setStateRef(State stateRef) {
00100         this.stateReferenceHC = stateRef;
00101     }
00102
00103     @Override
00108     public void setInitialReference(State stateInitialRef) {

```

```

00109     this.stateReferenceHC = stateInitialRef;
00110 }
00111
00116 public GeneratorType getGeneratorType() {
00117     return generatortype;
00118 }
00119
00124 public void setGeneratorType(GeneratorType generatortype) {
00125     this.generatortype = generatortype;
00126 }
00127
00128 @Override
00133 public GeneratorType getType() {
00134     return this.generatortype;
00135 }
00136
00137 @Override
00142 public List<State> getSonList() {
00143     // TODO Auto-generated method stub
00144     return null;
00145 }
00146
00147 @Override
00153 public boolean awardUpdateREF(State stateCandidate) {
00154     // TODO Auto-generated method stub
00155     return false;
00156 }
00157
00158 @Override
00163 public float getWeight() {
00164     // TODO Auto-generated method stub
00165     return 0;
00166 }
00167
00168 @Override
00173 public void setWeight(float weight) {
00174     // TODO Auto-generated method stub
00175 }
00176
00177
00178 @Override
00183 public float[] getTrace() {
00184     // TODO Auto-generated method stub
00185     return new float[0];
00186 }
00187
00188 @Override
00193 public int[] getListCountBetterGender() {
00194     // TODO Auto-generated method stub
00195     return new int[0];
00196 }
00197
00198 @Override
00203 public int[] getListCountGender() {
00204     // TODO Auto-generated method stub
00205     return new int[0];
00206 }
00207 }

```

## 9.177 Referencia del archivo MultiobjectiveTabuSearch.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import factory_interface.IFFactoryAcceptCandidate;
import factory_method.FactoryAcceptCandidate;
import local_search.acceptation_type.AcceptType;
import local_search.acceptation_type.AcceptableCandidate;
import local_search.candidate_type.CandidateType;
import local_search.candidate_type.CandidateValue;
import local_search.complement.StrategyType;
import local_search.complement.TabuSolutions;
import metaheuristics.strategy.Strategy;
import problem.definition.Problem;

```

```
import problem.definition.State;
Gráfico de dependencias incluidas en MultiobjectiveTabuSearch.java:
```



## Clases

- class [metaheuristics.generators.MultiobjectiveTabuSearch](#)

*MultiobjectiveTabuSearch - class that implements the Multiobjective Tabu Search metaheuristic.*

## Paquetes

- package [metaheuristics.generators](#)

## 9.178 MultiobjectiveTabuSearch.java

[Ir a la documentación de este archivo.](#)

```
00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.List;
00006
00007 import factory_interface.IFFactoryAcceptCandidate;
00008 import factory_method.FactoryAcceptCandidate;
00009 import local_search.acceptation_type.AcceptType;
00010 import local_search.acceptation_type.AcceptableCandidate;
00011 import local_search.candidate_type.CandidateType;
00012 import local_search.candidate_type.CandidateValue;
00013 import local_search.complement.StrategyType;
00014 import local_search.complement.TabuSolutions;
00015 import metaheuristics.strategy.Strategy;
00016 import problem.definition.Problem;
00017 import problem.definition.State;
00018
00019
00020
00021
00025 public class MultiobjectiveTabuSearch extends Generator {
00026
00027     private CandidateValue candidatevalue;
00028     private AcceptType typeAcceptation;
00029     private StrategyType strategy;
00030     private CandidateType typeCandidate;
00031     private State stateReferenceTS;
00032     private IFFactoryAcceptCandidate ifacceptCandidate;
00033     private GeneratorType typeGenerator;
00034     private List<State> listStateReference = new ArrayList<State>();
00035     private float weight;
00036     private List<Float> listTrace = new ArrayList<Float>();
00037
00042     public State getStateReferenceTS() {
00043         return stateReferenceTS;
00044     }
00045
00050     public void setStateReferenceTS(State stateReferenceTS) {
00051         this.stateReferenceTS = stateReferenceTS;
00052     }
00053
00058     public GeneratorType getTypeGenerator() {
00059         return typeGenerator;
00060     }
00061
00066     public void setTypeGenerator(GeneratorType typeGenerator) {
00067         this.typeGenerator = typeGenerator;
00068     }
00069 }
```

```

00073     public MultiobjectiveTabuSearch() {
00074         super();
00075         this.typeAcceptation = AcceptType.AcceptNotDominatedTabu;
00076         this.strategy = StrategyType.TABU;
00077         // Use problem API directly when needed; avoid unused local variable
00078         this.typeCandidate = CandidateType.RandomCandidate;
00079         this.candidatevalue = new CandidateValue();
00080         this.typeGenerator = GeneratorType.MultiobjectiveTabuSearch;
00081         this.weight = 50;
00082         listTrace.add(weight);
00083     }
00084
00085     @Override
00091     public State generate(Integer operatornumber) throws IllegalArgumentException, SecurityException,
00092         ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00093         NoSuchMethodException {
00092         //Devuelve la lista de soluciones no dominadas de todos los vecinos posibles de
00093         stateReferenceTS que no se encuentran en la lista Tabu
00093         List<State> neighborhood =
00094             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceTS, operatornumber);
00094         //Se escoge uno aleatoriamente como vecino con RandomCandidate
00095         State statecandidate = candidatevalue.stateCandidate(stateReferenceTS, typeCandidate,
00095             strategy, operatornumber, neighborhood);
00096         return statecandidate;
00097     }
00098
00099     @Override
00105     public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
00105         IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00105         IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00106         ifacceptCandidate = new FactoryAcceptCandidate();
00107         AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00108         Boolean accept = candidate.acceptCandidate(stateReferenceTS, stateCandidate);
00109         if(accept.equals(true))
00110             stateReferenceTS = stateCandidate;
00111
00112         if (strategy.equals(StrategyType.TABU) && accept.equals(true)) {
00113             if (TabuSolutions.listTabu.size() < TabuSolutions.maxelements) {
00114                 Boolean find = false;
00115                 int count = 0;
00116                 while ((TabuSolutions.listTabu.size() > count) && (find.equals(false))) {
00117                     if (TabuSolutions.listTabu.get(count).equals(stateCandidate)) {
00118                         find = true;
00119                     }
00120                     count++;
00121                 }
00122                 if (find.equals(false)) {
00123                     TabuSolutions.listTabu.add(stateCandidate);
00124                 }
00125             } else {
00126                 TabuSolutions.listTabu.remove(0);
00127                 Boolean find = false;
00128                 int count = 0;
00129                 while (TabuSolutions.listTabu.size() > count && find.equals(false)) {
00130                     if (TabuSolutions.listTabu.get(count).equals(stateCandidate)) {
00131                         find = true;
00132                     }
00133                     count++;
00134                 }
00135                 if (find.equals(false)) {
00136                     TabuSolutions.listTabu.add(stateCandidate);
00137                 }
00138             }
00139         }
00140         getReferenceList();
00141     }
00142
00143     @Override
00148     public GeneratorType getType() {
00149         return this.typeGenerator;
00150     }
00151
00152     @Override
00157     public List<State> getReferenceList() {
00158         listStateReference.add(stateReferenceTS);
00159         return listStateReference;
00160     }
00161
00162     @Override
00167     public State getReference() {
00168         return stateReferenceTS;
00169     }
00170
00171     @Override
00176     public void setInitialReference(State stateInitialRef) {
00177         this.stateReferenceTS = stateInitialRef;
00178     }

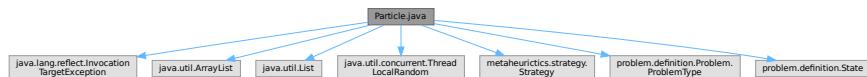
```

```
00179
00184     public void setStateRef(State stateRef) {
00185         this.stateReferenceTS = stateRef;
00186     }
00187
00188     @Override
00189     public List<State> getSonList() {
00190         // TODO Auto-generated method stub
00191         return null;
00192     }
00193
00194     public void setTypeCandidate(CandidateType typeCandidate) {
00195         this.typeCandidate = typeCandidate;
00196     }
00197
00198     @Override
00199     public boolean awardUpdateREF(State stateCandidate) {
00200         // TODO Auto-generated method stub
00201         return false;
00202     }
00203
00204     public float getWeight() {
00205         // TODO Auto-generated method stub
00206         return this.weight;
00207     }
00208
00209     @Override
00210     public void setWeight(float weight) {
00211         // TODO Auto-generated method stub
00212         this.weight = weight;
00213     }
00214
00215     @Override
00216     public int[] getListCountBetterGender() {
00217         // TODO Auto-generated method stub
00218         // This generator doesn't maintain listCount arrays; return empty array to avoid nulls
00219         return new int[0];
00220     }
00221
00222     @Override
00223     public int[] getListCountGender() {
00224         // TODO Auto-generated method stub
00225         // This generator doesn't maintain listCount arrays; return empty array to avoid nulls
00226         return new int[0];
00227     }
00228
00229     @Override
00230     public float[] getTrace() {
00231         // TODO Auto-generated method stub
00232         if (this.listTrace == null) return new float[0];
00233         float[] arr = new float[this.listTrace.size()];
00234         for (int i = 0; i < this.listTrace.size(); i++) {
00235             Float v = this.listTrace.get(i);
00236             arr[i] = (v == null) ? 0f : v.floatValue();
00237         }
00238         return arr;
00239     }
00240
00241 }
00242
00243 }
```

## 9.179 Referencia del archivo Particle.java

```
import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.ThreadLocalRandom;
import metaheuristics.strategy.Strategy;
import problem.definition.Problem.ProblemType;
import problem.definition.State;
```

Gráfico de dependencias incluidas en Particle.java:



## Clases

- class [metaheuristics.generators.Particle](#)

*Particle - class that represents a particle in the [Particle](#) Swarm Optimization algorithm.*

## Paquetes

- package [metaheuristics.generators](#)

## 9.180 Particle.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.List;
00006 import java.util.concurrent.ThreadLocalRandom;
00007
00008 import metaheuristics.strategy.Strategy;
00009
00010 import problem.definition.Problem.ProblemType;
00011 import problem.definition.State;
00012
00016 public class Particle extends Generator {
00017
00018     private State statePBest;
00019     private State stateActual;
00020     private ArrayList<Object> velocity;
00021
00022
00026     public Particle() {
00027         super();
00028         this.stateActual = new State();
00029         this.statePBest = new State();
00030         this.velocity = new ArrayList<Object>();
00031     }
00032
00039     public Particle(State statePBest, State stateActual, ArrayList<Object> velocity) {
00040         super();
00041         this.statePBest = statePBest;
00042         this.stateActual = stateActual;
00043         this.velocity = velocity;
00044     }
00045
00050     public ArrayList<Object> getVelocity() {
00051         return velocity;
00052     }
00053
00058     public void setVelocity(ArrayList<Object> velocity) {
00059         this.velocity = velocity;
00060     }
00061
00066     public State getStatePBest() {
00067         return statePBest;
00068     }
00069
00074     public void setStatePBest(State statePBest) {
00075         this.statePBest = statePBest;
00076     }
00077
  
```

```

00082     public State getStateActual() {
00083         return stateActual;
00084     }
00085
00090     public void setStateActual(State stateActual) {
00091         this.stateActual = stateActual;
00092     }
00105     @Override
00106     public State generate(Integer operatornumber)
00107         throws IllegalArgumentException, SecurityException,
00108             ClassNotFoundException, InstantiationException,
00109             IllegalAccessException, InvocationTargetException,
00110             NoSuchMethodException {
00111         // TODO Auto-generated method stub
00112
00113         ArrayList<Object> actualVelocity = UpdateVelocity();
00114         ArrayList<Object> newCode = UpdateCode(actualVelocity);
00115         this.velocity = actualVelocity;
00116         this.stateActual.setCode(newCode);
00117         return null;
00118     }
00119
00120
00126     @SuppressWarnings("squid:S2245")
00127     private ArrayList<Object> UpdateVelocity(){ // actualizar velocidad
00128         double w = ParticleSwarmOptimization.wmax - ((ParticleSwarmOptimization.wmax -
00129             ParticleSwarmOptimization.wmin) / Strategy.getStrategy().getCountMax()) *
00130             ParticleSwarmOptimization.getCountCurrentIterPSO(); //CALCULO DE LA INERCIA
00131         double rand1 = ThreadLocalRandom.current().nextDouble();
00130         double rand2 = ThreadLocalRandom.current().nextDouble();
00131         double inertia, cognitive, social;
00132         int learning = ParticleSwarmOptimization.learning1 + ParticleSwarmOptimization.learning2; //
00133         ratios de aprendizaje cognitivo y social
00134         ParticleSwarmOptimization.constriction = 2/(Math.abs(2 - learning-Math.sqrt((learning *
00135             learning)- 4 * learning))); // Factor de costriccion
00136         ArrayList<Object> actualVelocity = new ArrayList<Object>();
00137         if(velocity.isEmpty()){
00138             for (int i = 0; i < Strategy.getStrategy().getProblem().getState().getCode().size(); i++) {
00139                 velocity.add(0.0);
00140             }
00141             // recorre el vector velocidad y lo actualiza
00142             for (int i = 0; i < Strategy.getStrategy().getProblem().getState().getCode().size(); i++) {
00143                 // cumulo donde se encuentra la particula
00144                 int swarm = ParticleSwarmOptimization.getCountParticle() /
00145                     ParticleSwarmOptimization.getCountParticleBySwarm();
00146                 inertia = w * (Double)velocity.get(i);
00147                 if(ParticleSwarmOptimization.binary == true){
00148                     cognitive = (Double)(ParticleSwarmOptimization.learning1 * rand1 *
00149                         ((Integer)(this.statePBest.getCode().get(i)) - (Integer)(stateActual.getCode().get(i))));
00150                     social = (Double)(ParticleSwarmOptimization.learning2 * rand2 * (((Integer)((State)
00151                         ParticleSwarmOptimization.lBest[swarm]).getCode().get(i)) -
00152                         (Integer)(stateActual.getCode().get(i)))));
00153                     cognitive = (Double)(ParticleSwarmOptimization.learning1 * rand1 *
00154                         ((Double)(this.statePBest.getCode().get(i)) - (Double)(stateActual.getCode().get(i))));
00155                     social = (Double)(ParticleSwarmOptimization.learning2 * rand2 * (((Double)((State)
00156                         ParticleSwarmOptimization.lBest[swarm]).getCode().get(i)) -
00157                         (Double)(stateActual.getCode().get(i)))));
00158                     }
00159                     actualVelocity.add(ParticleSwarmOptimization.constriction*(inertia + cognitive + social));
00160                 }
00161
00163     @SuppressWarnings("squid:S2245")
00164     private ArrayList<Object> UpdateCode(ArrayList<Object> actualVelocity) { // CALCULO DE LA NUEA
00165         POSICION DE LA PARTICULA
00166         ArrayList<Object> newCode = new ArrayList<Object>();
00167         //poner la condicion de si se esta trabajando con valores continuos o binarios
00168         if(ParticleSwarmOptimization.binary == false){
00169             for (int i = 0; i < stateActual.getCode().size(); i++) {
00170                 newCode.add( (Double)(stateActual.getCode().get(i)) +
00171                     (Double)(actualVelocity.get(i)) );
00172             }
00173         } else{
00174             codificacion binaria
00175             ArrayList<Object> binaryCode = new ArrayList<Object>();
00176             for (int i = 0; i < stateActual.getCode().size(); i++){
00177                 double rand = ThreadLocalRandom.current().nextDouble();
00178                 double s = 1/(1 + 1.72 * (Double)(actualVelocity.get(i))); //
00179                 if (rand < s){
00180                     binaryCode.add(1);
00181                 }
00182             }
00183         }
00184     }

```

```
00180         }
00181         else{
00182             binaryCode.add(0);
00183         }
00184     }
00185     return binaryCode;
00186 }
00187 }
00188 }
00189
00202 @Override
00203 public void updateReference(State stateCandidate,
00204     Integer countIterationsCurrent) throws IllegalArgumentException,
00205     SecurityException, ClassNotFoundException, InstantiationException,
00206     IllegalAccessException, InvocationTargetException,
00207     NoSuchMethodException {
00208     // TODO Auto-generated method stub
00209     if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00210         if(stateActual.getEvaluation().get(0) > statePBest.getEvaluation().get(0)){
00211             statePBest.setCode(new ArrayList<Object>(stateActual.getCode()));
00212             statePBest.setEvaluation(stateActual.getEvaluation());
00213         }
00214     }
00215     else{
00216         if(stateCandidate.getEvaluation().get(0) < statePBest.getEvaluation().get(0)){
00217             statePBest.setCode(new ArrayList<Object>(stateCandidate.getCode()));
00218             statePBest.setEvaluation(stateCandidate.getEvaluation());
00219         }
00220     }
00221 }
00223
00224 @Override
00225 public State getReference() {
00226     // TODO Auto-generated method stub
00227     return null;
00228 }
00229
00230 @Override
00231 public void setInitialReference(State stateInitialRef) {
00232     // TODO Auto-generated method stub
00233 }
00234
00235 @Override
00236 public GeneratorType getType() {
00237     // TODO Auto-generated method stub
00238     return null;
00239 }
00240
00241 @Override
00242 public List<State> getReferenceList() {
00243     // TODO Auto-generated method stub
00244     return null;
00245 }
00246
00247 @Override
00248 public List<State> getSonList() {
00249     // TODO Auto-generated method stub
00250     return null;
00251 }
00252
00253
00254 @Override
00255 public boolean awardUpdateREF(State stateCandidate) {
00256     // TODO Auto-generated method stub
00257     return false;
00258 }
00259
00260 @Override
00261 public void setWeight(float weight) {
00262     // TODO Auto-generated method stub
00263 }
00264
00265 @Override
00266 public float getWeight() {
00267     // TODO Auto-generated method stub
00268     return 0;
00269 }
00270
00271 @Override
00272 public float[] getTrace() {
00273     // TODO Auto-generated method stub
00274     return new float[0];
00275 }
00276
00277 @Override
```

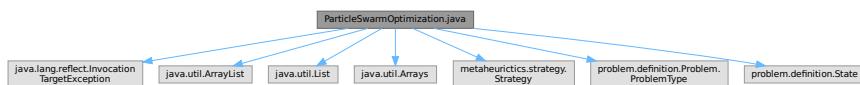
```

00320     public int[] getListCountBetterGender() {
00321         // TODO Auto-generated method stub
00322         return new int[0];
00323     }
00324
00325     @Override
00326     public int[] getListCountGender() {
00327         // TODO Auto-generated method stub
00328         return new int[0];
00329     }
00330 }
00331
00332 }
```

## 9.181 Referencia del archivo ParticleSwarmOptimization.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import java.util.Arrays;
import metaheuristics.strategy.Strategy;
import problem.definition.Problem.ProblemType;
import problem.definition.State;
Gráfico de dependencias incluidas en ParticleSwarmOptimization.java:
```



### Clases

- class [metaheuristics.generators.ParticleSwarmOptimization](#)

*ParticleSwarmOptimization - class that implements the Particle Swarm Optimization metaheuristic.*

### Paquetes

- package [metaheuristics.generators](#)

## 9.182 ParticleSwarmOptimization.java

[Ir a la documentación de este archivo.](#)

```

00001
00002 package metaheuristics.generators;
00003
00004 import java.lang.reflect.InvocationTargetException;
00005 import java.util.ArrayList;
00006 import java.util.List;
00007 import java.util.Arrays;
00008
00009
00010 import metaheuristics.strategy.Strategy;
00011
00012
00013 import problem.definition.Problem.ProblemType;
00014 import problem.definition.State;
00015
00019 public class ParticleSwarmOptimization extends Generator {
00020 }
```

```

00021     private State stateReferencePSO;
00022     private List<State> listStateReference = new ArrayList<State>();
00023     private List<Particle> listParticle = new ArrayList<Particle> ();
00024     private GeneratorType generatorType;
00025     static int countParticle = 0; // CANTIDAD DE PARTICULAS QUE SE HAN MOVIDO EN CADA CUMULO
00026     // Make these configurable (not final). Default to 0 (no swarms) to preserve existing test
00027     expectations.
00028     // Declaring them non-final prevents static analysis from treating loops as dead code while
00029     // keeping
00030     // the runtime default behavior (0) the same as before.
00031     // non-public backing field for number of swarms; use accessors to read/write
00032     private static int coutSwarm = 0; // CANTIDAD DE CUMULOS
00033     public static int getCountSwarm() {
00034         return coutSwarm;
00035     }
00036     public static void setCoutSwarm(int v) {
00037         if (v < 0) throw new IllegalArgumentException("coutSwarm must be >= 0");
00038         coutSwarm = v;
00039         // keep dependent countRef in sync
00040         setCountRef(coutSwarm * countParticleBySwarm);
00041     }
00042     // CANTIDAD DE PARTICULAS POR CUMULO - make non-public and provide accessors
00043     private static int countParticleBySwarm = 0;
00044     public static int getCountParticleBySwarm() {
00045         return countParticleBySwarm;
00046     }
00047     public static void setCountParticleBySwarm(int v) {
00048         if (v < 0) throw new IllegalArgumentException("countParticleBySwarm must be >= 0");
00049         countParticleBySwarm = v;
00050         // keep dependent countRef in sync
00051         setCountRef(coutSwarm * countParticleBySwarm);
00052     }
00053     private static int countRef = coutSwarm * countParticleBySwarm; // CANTIDAD DE
00054     PARTICULAS TOTAL = coutSwarm * countParticleBySwarm
00055     private float weight = 50;
00056     public static final double wmax = 0.9;
00057     public static final double wmin = 0.2;
00058     public static final int learning1 = 2, learning2 = 2;
00059     static double constriction;
00060     public static final boolean binary = false;
00061     static State[] lBest;
00062     static State gBest;
00063     static int countCurrentIterPSO;
00064
00065     // Accessors to avoid direct writes to static fields from instance methods.
00066     // Defensive copies are used to avoid exposing internal mutable state.
00067     public static State[] getLBest() {
00068         if (lBest == null) return new State[0];
00069         State[] copy = new State[lBest.length];
00070         for (int i = 0; i < lBest.length; i++) {
00071             copy[i] = (lBest[i] == null) ? null : lBest[i].clone();
00072         }
00073         return copy;
00074     }
00075     public static void setLBest(State[] arr) {
00076         if (arr == null) {
00077             lBest = null;
00078             return;
00079         }
00080         lBest = new State[arr.length];
00081         for (int i = 0; i < arr.length; i++) {
00082             lBest[i] = (arr[i] == null) ? null : arr[i].clone();
00083         }
00084     }
00085     public static void setLBestAt(int idx, State s) {
00086         if (lBest == null) return;
00087         if (idx < 0 || idx >= lBest.length) return;
00088         lBest[idx] = (s == null) ? null : s.clone();
00089     }
00090     public static State getGBest() { return gBest; }
00091     public static void setGBest(State s) { gBest = s; }
00092     public static int getCountCurrentIterPSO() { return countCurrentIterPSO; }
00093     public static void setCountCurrentIterPSO(int v) { countCurrentIterPSO = v; }
00094     public static int getCountParticle() { return countParticle; }
00095     public static void setCountParticle(int v) { countParticle = v; }
00096     //problemas dinamicos
00097     //problemas dinamicos: use instance fields from Generator (countGender, countBetterGender)
00098     private int[] listCountBetterGender = new int[10];
00099     private int[] listCountGender = new int[10];
00100     private float[] listTrace = new float[1200000];
00101
00102     public ParticleSwarmOptimization(){
00103         super();
00104         this.setListParticle(getListStateRef());
00105         this.generatorType = GeneratorType.ParticleSwarmOptimization;
00106         this.weight = 50;
00107         setLBest(new State[coutSwarm]);
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137

```

```

00138     if(!listParticle.isEmpty()){
00139         setCountCurrentIterPSO(1);
00140         inicialiceLBest();
00141         setGBest(gBestInicial());
00142     }
00143     setCountParticle(0);
00144     listTrace[0] = this.weight;
00145     listCountBetterGender[0] = 0;
00146     listCountGender[0] = 0;
00147 }
00148
00149 @Override
00150 public State generate(Integer operatornumber) throws IllegalArgumentException, SecurityException,
00151 ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00152 NoSuchMethodException { // PSO
00153     if (getCountParticle() >= countRef) {
00154         setCountParticle(0);
00155     }
00156     // Advance the current particle and return its active state.
00157     listParticle.get(getCountParticle()).generate(1);
00158     return listParticle.get(getCountParticle()).getStateActual();
00159 }
00160
00161
00162
00163
00164
00165
00166
00167 public void inicialiceLBest (){
00168     for (int j = 0; j < coutSwarm; j++) {
00169         // pick initial reference from the particle's personal best
00170         State reference = listParticle.get(getCountParticle()).getStatePBest();
00171         int iterator = countParticleBySwarm + getCountParticle();
00172         if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)) {
00173             for (int i = countParticle; i < iterator; i++) {
00174                 if (listParticle.get(i).getStatePBest().getEvaluation().get(0) >
00175                     reference.getEvaluation().get(0))
00176                     reference = listParticle.get(i).getStatePBest();
00177                     setCountParticle(getCountParticle() + 1);
00178             }
00179         }
00180         else{
00181             for (int i = countParticle; i < iterator; i++) {
00182                 if (listParticle.get(i).getStatePBest().getEvaluation().get(0) <
00183                     reference.getEvaluation().get(0))
00184                     reference = listParticle.get(i).getStatePBest();
00185                     setCountParticle(getCountParticle() + 1);
00186             }
00187         }
00188         lBest[j] = reference;
00189     }
00190 }
00191
00192
00193
00194
00195
00196
00197
00198 @Override
00199 public State getReference() {
00200     return null;
00201 }
00202
00203 private List<Particle> getListStateRef() {
00204     Boolean found = false;
00205     List<String> key = Strategy.getStrategy().getListKey();
00206     int count = 0;
00207     if(RandomSearch.listStateReference.size() == 0){
00208         return this.setListParticle(new ArrayList<Particle>());
00209     }
00210     while((found.equals(false)) && (Strategy.getStrategy().mapGenerators.size() > count)){
00211         //recorrer la lista de generadores, hasta que encuentre el PSO
00212         if(key.get(count).equals(GeneratorType.ParticleSwarmOptimization.toString())){
00213             //creo el generador PSO, y si su lista de particulas esta vacia entonces es la primera
00214             vez que lo estoy creando, y cada estado lo convierto en particulas
00215             GeneratorType keyGenerator = GeneratorType.valueOf(String.valueOf(key.get(count)));
00216             ParticleSwarmOptimization generator = (ParticleSwarmOptimization)
00217                 Strategy.getStrategy().mapGenerators.get(keyGenerator);
00218             if(generator.getListParticle().isEmpty()){
00219                 //convertir los estados en particulas
00220                 for (int j = 0; j < RandomSearch.listStateReference.size(); j++) {
00221                     //si el estado es creado con el generador RandomSearch entonces lo convierto
00222                     en particula
00223                     if(getListParticle().size() != countRef){
00224                         ArrayList<Object> velocity = new ArrayList<Object>();
00225                         State stateAct = (State) RandomSearch.listStateReference.get(j).getCopy();
00226                         stateAct.setCode(new
00227                             ArrayList<Object>(RandomSearch.listStateReference.get(j).getCode()));
00228                         stateAct.setEvaluation(RandomSearch.listStateReference.get(j).getEvaluation());
00229                         State statePBest = (State)
00230                             RandomSearch.listStateReference.get(j).getCopy();
00231                         statePBest.setCode(new
00232                             ArrayList<Object>(RandomSearch.listStateReference.get(j).getCode()));
00233                     }
00234                 }
00235             }
00236         }
00237     }
00238 }
00239
00240
00241
00242
00243
00244

```

```

00245     statePBest.setEvaluation(RandomSearch.listStateReference.get(j).getEvaluation());
00246
00247             Particle particle = new Particle(stateAct, statePBest, velocity);
00248             getListParticle().add(particle);
00249         }
00250     }
00251     else{
00252         setListParticle(generator.getListStateReference());
00253     }
00254     found = true;
00255   }
00256   count++;
00257 }
00258 return getListParticle();
00259 }
00260 }
00261
00262
00263 public State getStateReferencePSO() {
00264     return stateReferencePSO;
00265 }
00266
00267 public void setStateReferencePSO(State stateReferencePSO) {
00268     this.stateReferencePSO = stateReferencePSO;
00269 }
00270
00271
00272 public List<Particle> getListStateReference() {
00273     return this.getListParticle();
00274 }
00275
00276
00277 public void setListStateReference(List<State> listStateReference) {
00278     this.listStateReference = listStateReference;
00279 }
00280
00281
00282 public List<Particle> getListParticle() {
00283     return listParticle;
00284 }
00285
00286
00287 public List<Particle> setListParticle(List<Particle> listParticle) {
00288     this.listParticle = listParticle;
00289     return listParticle;
00290 }
00291
00292
00293 public GeneratorType getGeneratorType() {
00294     return generatorType;
00295 }
00296
00297
00298 public void setGeneratorType(GeneratorType generatorType) {
00299     this.generatorType = generatorType;
00300 }
00301
00302
00303 public static int getCountRef() {
00304     return countRef;
00305 }
00306
00307
00308 public static void setCountRef(int countRef) {
00309     ParticleSwarmOptimization.countRef = countRef;
00310 }
00311
00312
00313 /**
00314 * ****
00315 */
00316 @Override
00317 public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
00318     IllegalArgumentException, SecurityException, ClassNotFoundException,
00319     InstantiationException, IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00320     Particle particle = listParticle.get(countParticle);
00321     int swarm = countParticle/countParticleBySwarm;
00322     if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00323         if ((lBest[swarm]).getEvaluation().get(0) <
00324             particle.getStatePBest().getEvaluation().get(0)){
00325             setLBestAt(swarm, particle.getStatePBest());
00326             if(lBest[swarm].getEvaluation().get(0) >
00327                 getReferenceList().get(getReferenceList().size() - 1).getEvaluation().get(0)){
00328                 State tmp = new State();
00329                 tmp.setCode(new ArrayList<Object>(lBest[swarm].getCode()));
00330                 tmp.setEvaluation(lBest[swarm].getEvaluation());
00331                 tmp.setTypeGenerator(lBest[swarm].getTypeGenerator());
00332                 setGBest(tmp);
00333             }
00334         }
00335     }
00336     else {
00337         particle.updateReference(stateCandidate, countIterationsCurrent);
00338         if ((lBest[swarm]).getEvaluation().get(0) >
00339             particle.getStatePBest().getEvaluation().get(0)){
00340             setLBestAt(swarm, particle.getStatePBest());
00341         }
00342     }
00343 }
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382

```

```
00383             if(lBest[swarm].getEvaluation().get(0) <
00384                 getReferenceList().get(referenceList().size() - 1).getEvaluation().get(0)){
00385                 State tmp = new State();
00386                 tmp.setCode(new ArrayList<Object>(lBest[swarm].getCode()));
00387                 tmp.setEvaluation(lBest[swarm].getEvaluation());
00388                 tmp.setTypeGenerator(lBest[swarm].getTypeGenerator());
00389                 setGBest(tmp);
00390             }
00391         }
00392     listStateReference.add(getGBest());
00393     setCountParticle(getCountParticle() + 1);
00394     setCountCurrentIterPSO(getCountCurrentIterPSO() + 1);
00395   }
00396
00403   public State gBestInicial (){
00404     State stateBest = lBest[0];
00405     for (int i = 1; i < lBest.length; i++) {
00406       if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00407           if (lBest[i].getEvaluation().get(0) > stateBest.getEvaluation().get(0)){
00408               stateBest = lBest[i];
00409           }
00410       }
00411       else{
00412           if (lBest[i].getEvaluation().get(0) < stateBest.getEvaluation().get(0)){
00413               stateBest = lBest[i];
00414           }
00415       }
00416     }
00417     return stateBest;
00418   }
00419
00420   @Override
00428   public void setInitialReference(State stateInitialRef) {
00429     // Intentionally left blank for PSO (no single initial reference)
00430   }
00431
00432   @Override
00438   public GeneratorType getType() {
00439     return this.generatorType;
00440   }
00441
00442   @Override
00448   public List<State> getReferenceList() {
00449     return new ArrayList<State>(this.listStateReference);
00450   }
00451
00452   @Override
00460   public List<State> getSonList() {
00461     return null;
00462   }
00463
00464   @Override
00475   public boolean awardUpdateREF(State stateCandidate) {
00476     return false;
00477   }
00478
00479   @Override
00485   public void setWeight(float weight) {
00486     this.weight = weight;
00487   }
00488
00489   @Override
00495   public float getWeight() {
00496     return this.weight;
00497   }
00498
00499   @Override
00506   public int[] getListCountBetterGender() {
00507     return (this.listCountBetterGender == null) ? new int[0] :
00508       Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
00509   }
00510
00516   @Override
00516   public int[] getListCountGender() {
00517     return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00518       this.listCountGender.length);
00519   }
00520
00527   @Override
00527   public float[] getTrace() {
00528     return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00529       this.listTrace.length);
00530
00531
00532 }
```

```
00533 }
```

## 9.183 Referencia del archivo RandomSearch.java

```
import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Arrays;
import local_search.acceptation_type.AcceptType;
import local_search.acceptation_type.AcceptableCandidate;
import local_search.candidate_type.CandidateType;
import local_search.candidate_type.CandidateValue;
import local_search.complement.StrategyType;
import metaheuristics.strategy.Strategy;
import problem.definition.State;
import factory_interface.IFFactoryAcceptCandidate;
import factory_method.FactoryAcceptCandidate;
Gráfico de dependencias incluidas en RandomSearch.java:
```



### Clases

- class [metaheuristics.generators.RandomSearch](#)  
*RandomSearch - class that implements the Random Search metaheuristic.*

### Paquetes

- package [metaheuristics.generators](#)

## 9.184 RandomSearch.java

[Ir a la documentación de este archivo.](#)

```
00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.Collections;
00006 import java.util.List;
00007 import java.util.Arrays;
00008
00009 import local_search.acceptation_type.AcceptType;
00010 import local_search.acceptation_type.AcceptableCandidate;
00011 import local_search.candidate_type.CandidateType;
00012 import local_search.candidate_type.CandidateValue;
00013 import local_search.complement.StrategyType;
00014 import metaheuristics.strategy.Strategy;
00015
00016 import problem.definition.State;
00017
00018 import factory_interface.IFFactoryAcceptCandidate;
00019 import factory_method.FactoryAcceptCandidate;
00020
00024 public class RandomSearch extends Generator {
```

```

00025     private CandidateValue candidatevalue;
00026     private AcceptType typeAcceptation;
00027     private StrategyType strategy;
00028     private CandidateType typeCandidate;
00029     private State stateReferenceRS;
00030     private IFFactoryAcceptCandidate ifacceptCandidate;
00031     private GeneratorType typeGenerator;
00032
00033     private float weight;
00034     //para acceder desde los algoritmos basados en poblaciones de puntos
00035     public static final List<State> listStateReference = Collections.synchronizedList(new
00036         ArrayList<State>());
00037
00038     //problemas dinamicos: use instance fields from Generator (countGender, countBetterGender)
00039     private int[] listCountBetterGender = new int[10];
00040     private int[] listCountGender = new int[10];
00041     private float[] listTrace = new float[1200000];
00042
00043
00044     public RandomSearch() {
00045         super();
00046         this.typeAcceptation = AcceptType.AcceptBest;
00047         this.strategy = StrategyType.NORMAL;
00048         this.typeCandidate = CandidateType.RandomCandidate;
00049         this.candidatevalue = new CandidateValue();
00050         this.typeGenerator = GeneratorType.RandomSearch;
00051         this.weight = 50;
00052         listTrace[0] = this.weight;
00053         listCountBetterGender[0] = 0;
00054         listCountGender[0] = 0;
00055         listStateReference.clear();
00056     }
00057
00058
00059     @Override
00060     public State generate(Integer operatornumber) throws IllegalArgumentException, SecurityException,
00061     ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00062     NoSuchMethodException {
00063         List<State> neighborhood =
00064             Strategy.getStrategy().getProblem().getOperator().generateRandomState(operatornumber);
00065         State statecandidate = candidatevalue.stateCandidate(stateReferenceRS, typeCandidate,
00066             strategy, operatornumber, neighborhood);
00067         if(GeneticAlgorithm.countRef != 0 || EvolutionStrategies.countRef != 0 ||
00068             DistributionEstimationAlgorithm.countRef != 0 || ParticleSwarmOptimization.getCountRef() != 0)
00069             listStateReference.add(statecandidate);
00070         return statecandidate;
00071     }
00072
00073
00074     @Override
00075     public State getReference() {
00076         return stateReferenceRS;
00077     }
00078
00079
00080     @Override
00081     public void setInitialReference(State stateInitialRef) {
00082         this.stateReferenceRS = stateInitialRef;
00083     }
00084
00085
00086     @Override
00087     public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
00088         IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00089         IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00090         ifacceptCandidate = new FactoryAcceptCandidate();
00091         AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00092         Boolean accept = candidate.acceptCandidate(stateReferenceRS, stateCandidate);
00093         if(accept.equals(true))
00094             stateReferenceRS = stateCandidate;
00095     }
00096
00097
00098     @Override
00099     public GeneratorType getType() {
00100         return this.typeGenerator;
00101     }
00102
00103
00104     @Override
00105     public GeneratorType getTypeGenerator() {
00106         return typeGenerator;
00107     }
00108
00109
00110     public void setTypeGenerator(GeneratorType typeGenerator) {
00111         this.typeGenerator = typeGenerator;
00112     }
00113
00114
00115     @Override
00116     public List<State> getReferenceList() {
00117         listStateReference.add(stateReferenceRS);
00118         return listStateReference;
00119     }
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140

```

```

00141
00142     @Override
00143     public List<State> getSonList() {
00144         // TODO Auto-generated method stub
00145         return null;
00146     }
00147
00148     @Override
00149     public boolean awardUpdateREF(State stateCandidate) {
00150         // TODO Auto-generated method stub
00151         return false;
00152     }
00153
00154     @Override
00155     public float getWeight() {
00156         // TODO Auto-generated method stub
00157         return this.weight;
00158     }
00159
00160     @Override
00161     public void setWeight(float weight) {
00162         // TODO Auto-generated method stub
00163         this.weight = weight;
00164     }
00165
00166     @Override
00167     public int[] getListCountBetterGender() {
00168         // TODO Auto-generated method stub
00169         return (this.listCountBetterGender == null) ? new int[0] :
00170             Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
00171     }
00172
00173     @Override
00174     public int[] getListCountGender() {
00175         // TODO Auto-generated method stub
00176         return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00177             this.listCountGender.length);
00178     }
00179
00180     @Override
00181     public float[] getTrace() {
00182         // TODO Auto-generated method stub
00183         return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00184             this.listTrace.length);
00185     }
00186
00187     @Override
00188     public void accept(IFactoryAcceptCandidate candidate) {
00189         // TODO Auto-generated method stub
00190         candidate.accept();
00191     }
00192
00193     @Override
00194     public void accept(FactoryAcceptCandidate candidate) {
00195         // TODO Auto-generated method stub
00196         candidate.accept();
00197     }
00198
00199     @Override
00200     public void accept(StrategyType strategy) {
00201         // TODO Auto-generated method stub
00202         strategy.accept();
00203     }
00204
00205     @Override
00206     public void accept(ProblemDefinition.State state) {
00207         // TODO Auto-generated method stub
00208         state.accept();
00209     }
00210
00211     @Override
00212     public void accept(LocalSearchAcceptationType acceptationType) {
00213         // TODO Auto-generated method stub
00214     }

```

## 9.185 Referencia del archivo SimulatedAnnealing.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import java.util.Arrays;
import local_search.acceptation_type.AcceptType;
import local_search.acceptation_type.AcceptableCandidate;
import local_search.candidate_type.CandidateType;
import local_search.candidate_type.CandidateValue;
import local_search.complement.StrATEGY;
import metaheuristics.strategy.StrATEGY;
import problem.definition.State;
import factory_interface.IFFactoryAcceptCandidate;
import factory_method.FactoryAcceptCandidate;
Gráfico de dependencias incluidas en SimulatedAnnealing.java:

```



## Clases

- class [metaheuristics.generators.SimulatedAnnealing](#)

*SimulatedAnnealing - class that implements the Simulated Annealing metaheuristic.*

## Paquetes

- package [metaheuristics.generators](#)

## 9.186 SimulatedAnnealing.java

[Ir a la documentación de este archivo.](#)

```
00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.List;
00006 import java.util.Arrays;
00007
00008 import local_search.acceptation_type.AcceptType;
00009 import local_search.acceptation_type.AcceptableCandidate;
00010 import local_search.candidate_type.CandidateType;
00011 import local_search.candidate_type.CandidateValue;
00012 import local_search.complement.StrategyType;
00013 import metaheuristics.strategy.Strategy;
00014
00015 import problem.definition.State;
00016
00017 import factory_interface.IFFactoryAcceptCandidate;
00018 import factory_method.FactoryAcceptCandidate;
00019
00020 public class SimulatedAnnealing extends Generator {
00021
00022     private CandidateValue candidatevalue;
00023     private AcceptType typeAcceptation;
00024     private StrategyType strategy;
00025     private CandidateType typeCandidate;
00026     private State stateReferenceSA;
00027     private IFFactoryAcceptCandidate ifacceptCandidate;
00028     // cooling factor - fixed constant
00029     public static final double alpha = 0.93;
00030     // default temperature parameters (can be adjusted per run)
00031     public static final Double tinitial = 250.0;
00032     public static final Double tfinal = 41.66;
00033     static int countIterationsT = 50;
00034
00035     public static int getCountIterations() {
00036         return countIterationsT;
00037     }
00038
00039     public static void setCountIterations(int c) {
00040         countIterationsT = c;
00041     }
00042
00043     private int countRept;
00044     private GeneratorType typeGenerator;
00045     private List<State> listStateReference = new ArrayList<State>();
00046     private float weight;
00047
00048     //problemas dinamicos
00049     //problemas dinamicos: use instance fields from Generator (countGender, countBetterGender)
00050     private int[] listCountBetterGender = new int[10];
00051     private int[] listCountGender = new int[10];
00052     private float[] listTrace = new float[1200000];
00053
00054     public GeneratorType getTypeGenerator() {
00055         return typeGenerator;
00056     }
00057
00058     public void setTypeGenerator(GeneratorType typeGenerator) {
00059         this.typeGenerator = typeGenerator;
00060     }
00061
00062     public static Double getTinitial() {
00063         return tinitial;
00064     }
```

```

00086     }
00087
00091     public static void setTinitial(Double t) {
00092         tinitial = t;
00093     }
00094
00098     public SimulatedAnnealing() {
00099
00100         super();
00101         this.typeAcceptation = AcceptType.AcceptNotBadT;
00102         this.strategy = StrategyType.NORMAL;
00103         this.typeCandidate = CandidateType.RandomCandidate;
00104         this.candidatevalue = new CandidateValue();
00105         this.typeGenerator = GeneratorType.SimulatedAnnealing;
00106         this.weight = 50;
00107         listTrace[0] = this.weight;
00108         listCountBetterGender[0] = 0;
00109         listCountGender[0] = 0;
00110     }
00111
00112     @Override
00113     public State generate(Integer operatornumber) throws IllegalArgumentException, SecurityException,
00114     ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00115     NoSuchMethodException {
00116         List<State> neighborhood =
00117             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceSA, operatornumber);
00118         State statecandidate = candidatevalue.stateCandidate(stateReferenceSA, typeCandidate,
00119             strategy, operatornumber, neighborhood);
00120         return statecandidate;
00121     }
00122
00123
00124     @Override
00125     public State getReference() {
00126         return stateReferenceSA;
00127     }
00128
00129     public void setStateRef(State stateRef) {
00130         this.stateReferenceSA = stateRef;
00131     }
00132
00133
00137     public void setInitialReference(State stateInitialRef) {
00138         this.stateReferenceSA = stateInitialRef;
00139     }
00140
00141     @Override
00142     public void setInitialReference(State stateInitialRef) {
00143         this.stateReferenceSA = stateInitialRef;
00144     }
00145
00146     @Override
00147     public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
00148         IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00149         IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00150
00151         countRept = countIterationsT;
00152         ifacceptCandidate = new FactoryAcceptCandidate();
00153         AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00154         Boolean accept = candidate.acceptCandidate(stateReferenceSA, stateCandidate);
00155         if(accept.equals(true))
00156             stateReferenceSA = stateCandidate;
00157         if(countIterationsCurrent.equals(getCountIterationsT())){
00158             // use static setters to avoid writing the static field from an instance method
00159             SimulatedAnnealing.setTinitial(SimulatedAnnealing.getTinitial() * alpha);
00160             SimulatedAnnealing.setCountIterationsT(SimulatedAnnealing.getCountIterationsT() +
00161             countRept);
00162         }
00163     }
00164
00165
00166
00167
00168
00169
00170     @Override
00171     public GeneratorType getType() {
00172         return this.typeGenerator;
00173     }
00174
00175
00176
00177
00178
00179     @Override
00180     public List<State> getReferenceList() {
00181         listStateReference.add(stateReferenceSA);
00182         return new ArrayList<State>(listStateReference);
00183     }
00184
00185
00186
00187
00188
00189     @Override
00190     public List<State> getSonList() {
00191         // TODO Auto-generated method stub
00192         return null;
00193     }
00194
00195
00196
00197
00198
00199     @Override
00200     public boolean awardUpdateREF(State stateCandidate) {
00201         // TODO Auto-generated method stub
00202         return false;
00203     }
00204
00205
00206
00207
00208
00209
00210

```

```

00211     @Override
00216     public float getWeight() {
00217         // TODO Auto-generated method stub
00218         return this.weight;
00219     }
00220
00221     @Override
00226     public void setWeight(float weight) {
00227         // TODO Auto-generated method stub
00228         this.weight = weight;
00229     }
00230
00231     @Override
00236     public int[] getListCountBetterGender() {
00237         // TODO Auto-generated method stub
00238         return (this.listCountBetterGender == null) ? new int[0] :
00239             Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
00240     }
00241
00246     @Override
00246     public int[] getListCountGender() {
00247         // TODO Auto-generated method stub
00248         return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00249             this.listCountGender.length);
00250     }
00251
00256     @Override
00256     public float[] getTrace() {
00257         // TODO Auto-generated method stub
00258         return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00259             this.listTrace.length);
00260     }
00261 }

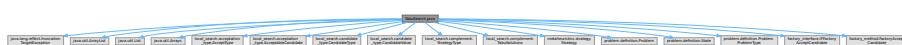
```

## 9.187 Referencia del archivo TabuSearch.java

```

import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import java.util.List;
import java.util.Arrays;
import local_search.acceptation_type.AcceptType;
import local_search.acceptation_type.AcceptableCandidate;
import local_search.candidate_type.CandidateType;
import local_search.candidate_type.CandidateValue;
import local_search.complement.StrategyType;
import local_search.complement.TabuSolutions;
import metaheuristics.strategy.Strategy;
import problem.definition.Problem;
import problem.definition.State;
import problem.definition.Problem.Type;
import factory_interface.IFFactoryAcceptCandidate;
import factory_method.FactoryAcceptCandidate;
Gráfico de dependencias incluidas en TabuSearch.java:

```



### Clases

- class [metaheuristics.generators.TabuSearch](#)

*TabuSearch - class that implements the Tabu Search metaheuristic.*

## Paquetes

- package [metaheuristics.generators](#)

## 9.188 TabuSearch.java

[Ir a la documentación de este archivo.](#)

```

00001 package metaheuristics.generators;
00002
00003 import java.lang.reflect.InvocationTargetException;
00004 import java.util.ArrayList;
00005 import java.util.List;
00006 import java.util.Arrays;
00007
00008 import local_search.acceptation_type.AcceptType;
00009 import local_search.acceptation_type.AcceptableCandidate;
00010 import local_search.candidate_type.CandidateType;
00011 import local_search.candidate_type.CandidateValue;
00012 import local_search.complement.StrategyType;
00013 import local_search.complement.TabuSolutions;
00014 import metaheuristics.strategy.Strategy;
00015
00016 import problem.definition.Problem;
00017 import problem.definition.State;
00018 import problem.definition.Problem.ProblemType;
00019
00020
00021
00022 import factory_interface.IFFactoryAcceptCandidate;
00023 import factory_method.FactoryAcceptCandidate;
00024
00025 public class TabuSearch extends Generator {
00026
00027     private CandidateValue candidatevalue;
00028     private AcceptType typeAcceptation;
00029     private StrategyType strategy;
00030     private CandidateType typeCandidate;
00031     private State stateReferenceTS;
00032     private IFFactoryAcceptCandidate ifacceptCandidate;
00033     private GeneratorType typeGenerator;
00034     private List<State> listStateReference = new ArrayList<State>();
00035     private float weight;
00036
00037     //problemas dinamicos: use instance fields from Generator (countGender, countBetterGender)
00038     private int[] listCountBetterGender = new int[10];
00039     private int[] listCountGender = new int[10];
00040     private float[] listTrace = new float[1200000];
00041
00042
00043
00044
00045
00046     public GeneratorType getTypeGenerator() {
00047         return typeGenerator;
00048     }
00049
00050     public void setTypeGenerator(GeneratorType typeGenerator) {
00051         this.typeGenerator = typeGenerator;
00052     }
00053
00054
00055     public TabuSearch() {
00056         super();
00057         this.typeAcceptation = AcceptType.AcceptAnyone;
00058         this.strategy = StrategyType.TABU;
00059
00060
00061
00062         Problem problem = Strategy.getStrategy().getProblem();
00063
00064         if(problem.getTypeProblem().equals(ProblemType.Maximizar)) {
00065             this.typeCandidate = CandidateType.GreaterCandidate;
00066         }
00067         else{
00068             this.typeCandidate = CandidateType.SmallerCandidate;
00069         }
00070
00071         this.candidatevalue = new CandidateValue();
00072         this.typeGenerator = GeneratorType.TabuSearch;
00073         this.weight = 50;
00074         listTrace[0] = this.weight;
00075         listCountBetterGender[0] = 0;
00076         listCountGender[0] = 0;
00077     }
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087 }
```

```
00088
00089     @Override
00090     public State generate(Integer operatornumber) throws IllegalArgumentException, SecurityException,
00091         ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00092         NoSuchMethodException {
00093         List<State> neighborhood =
00094             Strategy.getStrategy().getProblem().getOperator().generatedNewState(stateReferenceTS, operatornumber);
00095         State statecandidate = candidatevalue.stateCandidate(stateReferenceTS, typeCandidate,
00096             strategy, operatornumber, neighborhood);
00097         return statecandidate;
00098     }
00100
00101     @Override
00102     public State getReference() {
00103         return stateReferenceTS;
00104     }
00105
00106     @Override
00107     public void setInitialReference(State stateInitialRef) {
00108         this.stateReferenceTS = stateInitialRef;
00109     }
00110
00111     @Override
00112     public void setStateRef(State stateRef) {
00113         this.stateReferenceTS = stateRef;
00114     }
00115
00116     @Override
00117     public void updateReference(State stateCandidate, Integer countIterationsCurrent) throws
00118         IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00119         IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00120         ifacceptCandidate = new FactoryAcceptCandidate();
00121         AcceptableCandidate candidate = ifacceptCandidate.createAcceptCandidate(typeAcceptation);
00122         Boolean accept = candidate.acceptCandidate(stateReferenceTS, stateCandidate);
00123         if(accept.equals(true))
00124             stateReferenceTS = stateCandidate;
00125
00126         if (strategy.equals(StrategyType.TABU) && accept.equals(true)) {
00127             if (TabuSolutions.listTabu.size() < TabuSolutions.maxelements) {
00128                 Boolean find = false;
00129                 int count = 0;
00130                 while ((TabuSolutions.listTabu.size() > count) && (find.equals(false))) {
00131                     if ((TabuSolutions.listTabu.get(count).Comparator(stateCandidate))==true) {
00132                         find = true;
00133                     }
00134                     count++;
00135                 }
00136                 if (find.equals(false)) {
00137                     TabuSolutions.listTabu.add(stateCandidate);
00138                 }
00139             } else {
00140                 TabuSolutions.listTabu.remove(0);
00141                 Boolean find = false;
00142                 int count = 0;
00143                 while (TabuSolutions.listTabu.size() > count && find.equals(false)) {
00144                     if ((TabuSolutions.listTabu.get(count).Comparator(stateCandidate))==true) {
00145                         find = true;
00146                     }
00147                     count++;
00148                 }
00149                 if (find.equals(false)) {
00150                     TabuSolutions.listTabu.add(stateCandidate);
00151                 }
00152             }
00153         }
00154     }
00155
00156     @Override
00157     public GeneratorType getType() {
00158         return this.typeGenerator;
00159     }
00160
00161     @Override
00162     public List<State> getReferenceList() {
00163         listStateReference.add(stateReferenceTS);
00164         return new ArrayList<State>(listStateReference);
00165     }
00166
00167 }
00168
00169
00170     @Override
00171     public GeneratorType getType() {
00172         return this.typeGenerator;
00173     }
00174
00175     @Override
00176     public List<State> getSonList() {
00177         // TODO Auto-generated method stub
00178         return null;
00179     }
00180
00181     @Override
00182     public void setTypeCandidate(CandidateType typeCandidate) {
00183         this.typeCandidate = typeCandidate;
00184     }
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
```

```

00207     @Override
00213     public boolean awardUpdateREF(State stateCandidate) {
00214         // TODO Auto-generated method stub
00215         return false;
00216     }
00217
00222     public float getWeight() {
00223         // TODO Auto-generated method stub
00224         return this.weight;
00225     }
00226
00227     @Override
00232     public void setWeight(float weight) {
00233         // TODO Auto-generated method stub
00234         this.weight = weight;
00235     }
00236
00237     @Override
00242     public int[] getListCountBetterGender() {
00243         // TODO Auto-generated method stub
00244         return (this.listCountBetterGender == null) ? new int[0] :
00245             Arrays.copyOf(this.listCountBetterGender, this.listCountBetterGender.length);
00246     }
00247
00252     @Override
00255     public int[] getListCountGender() {
00256         // TODO Auto-generated method stub
00257         return (this.listCountGender == null) ? new int[0] : Arrays.copyOf(this.listCountGender,
00258             this.listCountGender.length);
00259     }
00260
00265     @Override
00266     public float[] getTrace() {
00267         // TODO Auto-generated method stub
00268         return (this.listTrace == null) ? new float[0] : Arrays.copyOf(this.listTrace,
00269             this.listTrace.length);
00270     }

```

## 9.189 Referencia del archivo Codification.java

### Clases

- class [problem.definition.Codification](#)  
*Codification.*

### Paquetes

- package [problem.definition](#)

## 9.190 Codification.java

[Ir a la documentación de este archivo.](#)

```

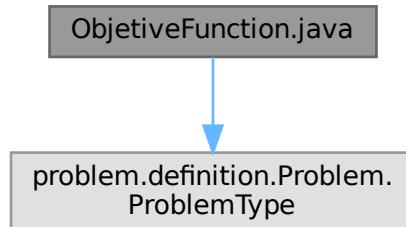
00001 package problem.definition;
00002
00011 public abstract class Codification {
00012
00013     public abstract boolean validState(State state);
00014     // public abstract int getVariableDomain(int variable);
00015     public abstract Object getVariableAleatoryValue(int key);
00016     public abstract int getAleatoryKey ();
00017     public abstract int getVariableCount();
00018
00019 }

```

## 9.191 Referencia del archivo ObjetiveFunction.java

```
import problem.definition.Problem.ProblemType;
```

Gráfico de dependencias incluidas en ObjetiveFunction.java:



### Clases

- class [problem.definition.ObjetiveFunction](#)  
*ObjetiveFunction.*

### Paquetes

- package [problem.definition](#)

## 9.192 ObjetiveFunction.java

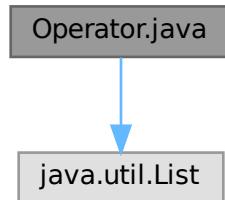
[Ir a la documentación de este archivo.](#)

```
00001 package problem.definition;
00002
00003 import problem.definition.Problem.ProblemType;
00004
00012 public abstract class ObjetiveFunction {
00013
00014     private ProblemType typeProblem;
00015     private float weight;
00016
00022     public float getWeight() {
00023         return weight;
00024     }
00025
00031     public void setWeight(float weight) {
00032         this.weight = weight;
00033     }
00034
00040     public ProblemType getTypeProblem() {
00041         return typeProblem;
00042     }
00043
00049     public void setTypeProblem(ProblemType typeProblem) {
00050         this.typeProblem = typeProblem;
00051     }
00052
00059     public abstract Double Evaluation(State state);
00060 }
```

## 9.193 Referencia del archivo Operator.java

```
import java.util.List;
```

Gráfico de dependencias incluidas en Operator.java:



### Clases

- class [problem.definition.Operator](#)  
*Operator.*

### Paquetes

- package [problem.definition](#)

## 9.194 Operator.java

[Ir a la documentación de este archivo.](#)

```
00001 package problem.definition;
00002
00003 import java.util.List;
00004
00011 public abstract class Operator {
00012
00020     public abstract List<State> generatedNewState(State stateCurrent, Integer operatornumber);
00021
00028     public abstract List<State> generateRandomState (Integer operatornumber);
00029
00030 }
00031
```

## 9.195 Referencia del archivo Problem.java

```
import java.lang.reflect.InvocationTargetException;
import java.util.ArrayList;
import problem.extension.SolutionMethod;
import problem.extension.TypeSolutionMethod;
import factory_interface.IFFactorySolutionMethod;
```

```
import factory_method.FactorySolutionMethod;
```

Gráfico de dependencias incluidas en Problem.java:



## Clases

- class [problem.definition.Problem](#)  
*Problem.*
- enum [problem.definition.Problem.ProblemType](#)  
*ProblemType - descripción (añade detalles).*

## Paquetes

- package [problem.definition](#)

## 9.196 Problem.java

Ir a la documentación de este archivo.

```
00001 package problem.definition;
00002 import java.lang.reflect.InvocationTargetException;
00003 import java.util.ArrayList;
00004
00005 import problem.extension.SolutionMethod;
00006 import problem.extension.TypeSolutionMethod;
00007
00008
00009 import factory_interface.IFFactorySolutionMethod;
00010 import factory_method.FactorySolutionMethod;
00011
00012
00024 public class Problem {
00025
00029     public enum ProblemType {Maximizar,Minimizar;}
00030
00031     private ArrayList<ObjetiveFunction> function;
00032     private State state;
00033     private ProblemType typeProblem;
00034     private Codification codification;
00035     private Operator operator;
00036     private int possibleValue;
00037     private TypeSolutionMethod typeSolutionMethod;
00038     private IFFactorySolutionMethod factorySolutionMethod;
00039
00047     public Problem() {
00048         super();
00049     }
00050
00057     public ArrayList<ObjetiveFunction> getFunction() {
00058         return (function == null) ? new ArrayList<ObjetiveFunction>() : new
00059             ArrayList<ObjetiveFunction>(function);
00060     }
00068     public void setFunction(ArrayList<ObjetiveFunction> function) {
00069         this.function = (function == null) ? new ArrayList<ObjetiveFunction>() : new
00070             ArrayList<ObjetiveFunction>(function);
00071     }
00079     public State getState() {
00080         return (state == null) ? null : new State(state);
00081     }
00082 }
```

```

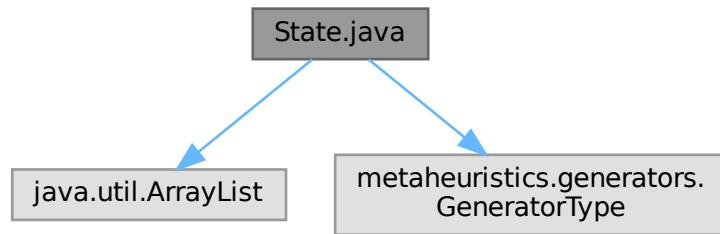
00088     public void setState(State state) {
00089         this.state = (state == null) ? null : new State(state);
00090     }
00091
00097     public ProblemType getTypeProblem() {
00098         return typeProblem;
00099     }
00105     public void setTypeProblem(ProblemType typeProblem) {
00106         this.typeProblem = typeProblem;
00107     }
00108
00114     public Codification getCodification() {
00115         return codification;
00116     }
00122     public void setCodification(Codification codification) {
00123         this.codification = codification;
00124     }
00125
00131     public Operator getOperator() {
00132         return operator;
00133     }
00134
00140     public void setOperator(Operator operator) {
00141         this.operator = operator;
00142     }
00143
00149     public int getPossibleValue() {
00150         return possibleValue;
00151     }
00152
00158     public void setPossibleValue(int possibleValue) {
00159         this.possibleValue = possibleValue;
00160     }
00161
00171     public void Evaluate(State state) throws IllegalArgumentException, SecurityException,
00172         ClassNotFoundException, InstantiationException, IllegalAccessException, InvocationTargetException,
00173         NoSuchMethodException {
00174         double eval = 0;
00175         ArrayList<Double> evaluation = new ArrayList<Double>(this.function.size());
00176         if (typeSolutionMethod == null) {
00177             eval= function.get(0).Evaluation(state);
00178             evaluation.add(evaluation.size(), eval);
00179             state.setEvaluation(evaluation);
00180         } else {
00181             SolutionMethod method = newSolutionMethod(typeSolutionMethod);
00182             method.evaluationState(state);
00183         }
00184     }
00190     public TypeSolutionMethod getTypeSolutionMethod() {
00191         return typeSolutionMethod;
00192     }
00198     public void setTypeSolutionMethod(TypeSolutionMethod typeSolutionMethod) {
00199         this.typeSolutionMethod = typeSolutionMethod;
00200     }
00206     public IFFactorySolutionMethod getFactorySolutionMethod() {
00207         return factorySolutionMethod;
00208     }
00209     public void setFactorySolutionMethod(
00210         IFFactorySolutionMethod factorySolutionMethod) {
00211         this.factorySolutionMethod = factorySolutionMethod;
00212     }
00213
00221     public SolutionMethod newSolutionMethod(TypeSolutionMethod typeSolutionMethod) throws
00222         IllegalArgumentException, SecurityException, ClassNotFoundException, InstantiationException,
00223         IllegalAccessException, InvocationTargetException, NoSuchMethodException {
00224         factorySolutionMethod = new FactorySolutionMethod();
00225         SolutionMethod solutionMethod =
00226             factorySolutionMethod.createdSolutionMethod(typeSolutionMethod);
00227         return solutionMethod;
00228     }
00229
00230 }
```

## 9.197 Referencia del archivo State.java

```

import java.util.ArrayList;
import metaheuristics.generators.GeneratorType;
```

Gráfico de dependencias incluidas en State.java:



## Clases

- class [problem.definition.State](#)  
*State.*

## Paquetes

- package [problem.definition](#)

## 9.198 State.java

[Ir a la documentación de este archivo.](#)

```
00001 package problem.definition;
00002
00003 import java.util.ArrayList;
00004
00005 import metaheuristics.generators.GeneratorType;
00006
00016 public class State implements Cloneable {
00017
00018     protected GeneratorType typeGenerator;
00019     protected ArrayList<Double> evaluation;
00020     protected int number;
00021     protected ArrayList<Object> code;
00022
00030     public State(State ps) {
00031         if (ps == null) {
00032             this.code = new ArrayList<Object>();
00033             this.evaluation = null;
00034             this.number = 0;
00035             this.typeGenerator = null;
00036             return;
00037         }
00038         typeGenerator = ps.getTypeGenerator();
00039         ArrayList<Double> eval = ps.getEvaluation();
00040         this.evaluation = (eval == null) ? null : new ArrayList<Double>(eval);
00041         number = ps.getNumber();
00042         ArrayList<Object> c = ps.getCode();
00043         this.code = (c == null) ? new ArrayList<Object>() : new ArrayList<Object>(c);
00044     }
00045
00051     public State(ArrayList<Object> code) {
00052         super();
00053         this.code = (code == null) ? new ArrayList<Object>() : new ArrayList<Object>(code);
00054     }
00059     public State() {
```

```

00060     code=new ArrayList<Object>();
00061 }
00062
00063 public ArrayList<Object> getCode() {
00064     return (code == null) ? new ArrayList<Object>() : new ArrayList<Object>(code);
00065 }
00066
00067 public void setCode(ArrayList<Object> listCode) {
00068     this.code = (listCode == null) ? new ArrayList<Object>() : new ArrayList<Object>(listCode);
00069 }
00070
00071
00072 public GeneratorType getTypeGenerator() {
00073     return typeGenerator;
00074 }
00075
00076 public void setTypeGenerator(GeneratorType typeGenerator) {
00077     this.typeGenerator = typeGenerator;
00078 }
00079
00080
00081
00082
00083
00084 public ArrayList<Double> getEvaluation() {
00085     return (evaluation == null) ? null : new ArrayList<Double>(evaluation);
00086 }
00087
00088
00089
00090
00091
00092
00093
00094 public void setEvaluation(ArrayList<Double> evaluation) {
00095     this.evaluation = (evaluation == null) ? null : new ArrayList<Double>(evaluation);
00096 }
00097
00098
00099
00100
00101
00102
00103
00104 public int getNumber() {
00105     return number;
00106 }
00107
00108 public void setNumber(int number) {
00109     this.number = number;
00110 }
00111
00112 @Override
00113 public State clone(){
00114     try {
00115         State s = (State) super.clone();
00116         // deep-copy mutable collections
00117         s.code = (this.code == null) ? new ArrayList<Object>() : new ArrayList<Object>(this.code);
00118         s.evaluation = (this.evaluation == null) ? null : new ArrayList<Double>(this.evaluation);
00119         // primitive and enum-like fields are safe to copy by assignment
00120         return s;
00121     } catch (CloneNotSupportedException e) {
00122         // fallback to copy constructor
00123         return new State(this);
00124     }
00125 }
00126
00127
00128 public Object getCopy(){
00129     return new State(this);
00130 }
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194 }

```

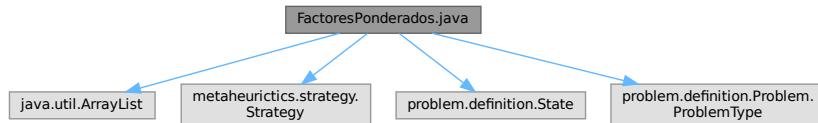
## 9.199 Referencia del archivo FactoresPonderados.java

```

import java.util.ArrayList;
import metaheuristics.strategy.Strategy;
import problem.definition.State;
import problem.definition.ProblemType;

```

Gráfico de dependencias incluidas en FactoresPonderados.java:



## Clases

- class [problem.extension.FactoresPonderados](#)  
*FactoresPonderados.*

## Paquetes

- package [problem.extension](#)

## 9.200 FactoresPonderados.java

[Ir a la documentación de este archivo.](#)

```

00001 package problem.extension;
00002
00003 import java.util.ArrayList;
00004
00005 import metaheuristics.strategy.Strategy;
00006
00007 import problem.definition.State;
00008 import problem.definition.Problem.ProblemType;
00009
00016 public class FactoresPonderados extends SolutionMethod {
00017
00018     @Override
00029     public void evaluationState(State state) {
00030         // TODO Auto-generated method stub
00031         double eval = 0;
00032         double tempWeight = 0;
00033         ArrayList<Double> evaluation = new
00034             ArrayList<Double>(Strategy.getStrategy().getProblem().getFunction().size());
00035
00036         for (int i = 0; i < Strategy.getStrategy().getProblem().getFunction().size(); i++) {
00037             tempWeight = 0;
00038             if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar)){
00039                 if(Strategy.getStrategy().getProblem().getFunction().get(i).getTypeProblem().equals(ProblemType.Maximizar)){
00040                     tempWeight =
00041                         Strategy.getStrategy().getProblem().getFunction().get(i).Evaluation(state);
00041                     tempWeight = tempWeight *
00042                         Strategy.getStrategy().getProblem().getFunction().get(i).getWeight();
00042                 }
00043                 else{
00044                     tempWeight = 1 -
00044                         Strategy.getStrategy().getProblem().getFunction().get(i).Evaluation(state);
00045                     tempWeight = tempWeight *
00045                         Strategy.getStrategy().getProblem().getFunction().get(i).getWeight();
00046                 }
00047             }
00048             else{
00049                 if(Strategy.getStrategy().getProblem().getFunction().get(i).getTypeProblem().equals(ProblemType.Maximizar)){
00050                     tempWeight = 1 -
00050                         Strategy.getStrategy().getProblem().getFunction().get(i).Evaluation(state);
00051                     tempWeight = tempWeight *
00051                         Strategy.getStrategy().getProblem().getFunction().get(i).getWeight();
00051                 }
00052             }
00053         }
00054     }
00055 }

```

```

00052         }
00053     else{
00054         tempWeight =
00055         Strategy.getStrategy().getProblem().getFunction().get(i).Evaluation(state);
00056         tempWeight = tempWeight *
00057         Strategy.getStrategy().getProblem().getFunction().get(i).getWeight();
00058     }
00059     eval += tempWeight;
00060 evaluation.add(evaluation.size(), eval);
00061 state.setEvaluation(evaluation);
00062
00063 }
00064
00065 }

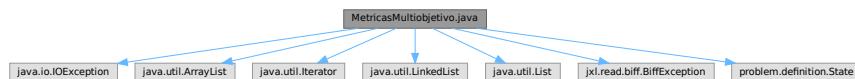
```

## 9.201 Referencia del archivo MetricasMultiobjetivo.java

```

import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import jxl.read.biff.BiffException;
import problem.definition.State;
Gráfico de dependencias incluidas en MetricasMultiobjetivo.java:

```



### Clases

- class [problem.extension.MetricasMultiobjetivo](#)  
*MetricasMultiobjetivo.*

### Paquetes

- package [problem.extension](#)

## 9.202 MetricasMultiobjetivo.java

[Ir a la documentación de este archivo.](#)

```

00001 package problem.extension;
00002
00003 import java.io.IOException;
00004 import java.util.ArrayList;
00005 import java.util.Iterator;
00006 import java.util.LinkedList;
00007 import java.util.List;
00008
00009 import jxl.read.biff.BiffException;
00010 import problem.definition.State;
00011
00018 public class MetricasMultiobjetivo {
00019

```

```

00020 // % de soluciones q no son miembros del frente de pareto verdadero
00021     public double TasaError(List<State> solutionsFPcurrent, List<State> solutionsFPtrue) throws
00022         BiffException, IOException{
00023             float tasaError = 0;
00024             for (int i = 0; i < solutionsFPcurrent.size() ; i++) { // frente de pareto actual
00025                 State solutionVO = solutionsFPcurrent.get(i);
00026                 if(!Contains(solutionVO, solutionsFPtrue)){ // no esta en el frente de pareto verdadero
00027                     tasaError++;
00028                 }
00029             }
00030             double total = tasaError/solutionsFPcurrent.size();
00031             //System.out.println(solutionsFP.size() + "/" + solutions.size() + "/" + total);
00032             return total;
00033         }
00034     }
00035
00036
00037
00038
00039
00040
00041
00042 // % Indica qué tan lejos están los elementos del frente de Pareto actual respecto al
00043 // frente de Pareto verdadero
00044     public double DistanciaGeneracional(List<State> solutionsFPcurrent, List<State> solutionsFPtrue)
00045         throws BiffException, IOException{
00046         float min = 1000;
00047         float distancia = 0;
00048         float distanciaGeneracional = 0;
00049         for (int i = 0; i < solutionsFPcurrent.size();i++) {
00050             State solutionVO = solutionsFPcurrent.get(i);
00051             //Calculando la distancia euclídea entre solutionVO y el miembro más cercano del frente
00052             de Pareto verdadero
00053             min = 1000;
00054             for (int j = 0; j < solutionsFPtrue.size();j++) {
00055                 for (int j2 = 0; j2 < solutionVO.getEvaluation().size(); j2++) {
00056                     State solutionFPV = solutionsFPtrue.get(j);
00057                     // porq elevar la distancia al cuadrado
00058                     distancia += (solutionVO.getEvaluation().get(j) -
00059                         solutionFPV.getEvaluation().get(j2))*
00060                             (solutionVO.getEvaluation().get(j2) -
00061                             solutionFPV.getEvaluation().get(j2)); //ceros si el argumento es el cero, 1.0 si el argumento es mayor
00062                             que el cero, -1.0 si el argumento está menos del cero
00063                     }
00064                     if(distancia < min){
00065                         min = distancia;
00066                     }
00067                 }
00068                 distanciaGeneracional += min;
00069             }
00070             double total = Math.sqrt(distanciaGeneracional)/solutionsFPcurrent.size();
00071             //System.out.println(total);
00072             return total;
00073         }
00074     }
00075
00076
00077
00078
00079     public double Dispersion(ArrayList<State> solutions) throws BiffException, IOException{
00080         //Soluciones obtenidas con la ejecución del algoritmo X
00081         LinkedList<Float> distancias = new LinkedList<Float>();
00082         float distancia = 0;
00083         float min = 1000;
00084         for (Iterator<State> iter = solutions.iterator(); iter.hasNext();) {
00085             State solutionVO = (State) iter.next();
00086             min = 1000;
00087             for (Iterator<State> iterator = solutions.iterator(); iterator.hasNext();) {
00088                 State solVO = (State) iterator.next();
00089                 for (int i = 0; i < solutionVO.getEvaluation().size(); i++) {
00090                     if(!solutionVO.getEvaluation().equals(solVO.getEvaluation())){
00091                         distancia += (solutionVO.getEvaluation().get(i)-
00092                             solVO.getEvaluation().get(i));
00093                     }
00094                     if(distancia < min){
00095                         min = distancia;
00096                     }
00097                 }
00098                 distancias.add(Float.valueOf(min));
00099             }
00100             //Calculando las media de las distancias
00101             float sum = 0;
00102             for (Iterator<Float> iter = distancias.iterator(); iter.hasNext();) {
00103                 Float dist = (Float) iter.next();
00104                 sum += dist;
00105             }
00106             float media = sum/distancias.size();
00107             float sumDistancias = 0;
00108             for (Iterator<Float> iter = distancias.iterator(); iter.hasNext();) {
00109                 Float dist = (Float) iter.next();
00110                 sumDistancias += Math.pow((media - dist),2);
00111             }
00112             //Calculando la dispersion
00113             double dispersion = 0;
00114             if(solutions.size() > 1){
00115                 dispersion = Math.sqrt((1.0/(solutions.size()-1))*sumDistancias);
00116             }
00117             //System.out.println(dispersion);
00118
00119
00120
00121
00122
00123
00124
00125

```

```

00126     return dispersion;
00127 }
00135 private boolean Contains(State solA, List<State> solutions){
00136     int i = 0;
00137     boolean result = false;
00138     while(i<solutions.size()&& result==false){
00139         if(solutions.get(i).getEvaluation().equals(solA.getEvaluation()))
00140             result=true;
00141         else
00142             i++;
00143     }
00144     return result;
00145 }
00152 public double CalcularMin(ArrayList<Double> allMetrics){
00153     double min = 1000;
00154     for (Iterator<Double> iter = allMetrics.iterator(); iter.hasNext();) {
00155         double element = (Double) iter.next();
00156         if(element < min){
00157             min = element;
00158         }
00159     }
00160     return min;
00161 }
00162
00169 public double CalcularMax(ArrayList<Double> allMetrics){
00170     double max = 0;
00171     for (Iterator<Double> iter = allMetrics.iterator(); iter.hasNext();) {
00172         double element = (Double) iter.next();
00173         if(element > max){
00174             max = element;
00175         }
00176     }
00177     return max;
00178 }
00185 public double CalcularMedia(ArrayList<Double> allMetrics){
00186     double sum = 0;
00187     for (Iterator<Double> iter = allMetrics.iterator(); iter.hasNext();) {
00188         double element = (Double) iter.next();
00189         sum = sum + element;
00190     }
00191     double media = sum/allMetrics.size();
00192     return media;
00193 }
00194 }

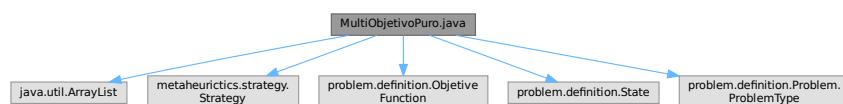
```

## 9.203 Referencia del archivo MultiObjetivoPuro.java

```

import java.util.ArrayList;
import metaheuristics.strategy.Strategy;
import problem.definition.ObjetiveFunction;
import problem.definition.State;
import problem.definition.Problem.ProblemType;
Gráfico de dependencias incluidas en MultiObjetivoPuro.java:

```



### Clases

- class [problem.extension.MultiObjetivoPuro](#)  
*MultiObjetivoPuro.*

### Paquetes

- package [problem.extension](#)

## 9.204 MultiObjetivoPuro.java

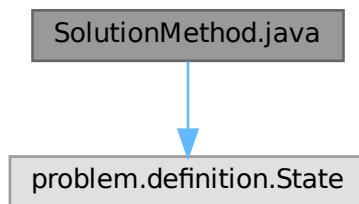
[Ir a la documentación de este archivo.](#)

```

00001 package problem.extension;
00002
00003 import java.util.ArrayList;
00004
00005 import metaheuristics.strategy.Strategy;
00006
00007 import problem.definition.ObjetiveFunction;
00008 import problem.definition.State;
00009 import problem.definition.Problem.ProblemType;
00010
00011 public class MultiObjetivoPuro extends SolutionMethod {
00012
00013     @Override
00014     public void evaluationState(State state) {
00015         // TODO Auto-generated method stub
00016         double tempEval = -1;
00017         ArrayList<Double> evaluation = new
00018             ArrayList<Double>(Strategy.getStrategy().getProblem().getFunction().size());
00019         for (int i = 0; i < Strategy.getStrategy().getProblem().getFunction().size(); i++)
00020         {
00021             ObjetiveFunction objfunction =
00022                 (ObjetiveFunction)Strategy.getStrategy().getProblem().getFunction().get(i);
00023             if(Strategy.getStrategy().getProblem().getTypeProblem().equals(ProblemType.Maximizar))
00024             {
00025                 if(objfunction.getTypeProblem().equals(ProblemType.Maximizar))
00026                 {
00027                     tempEval = objfunction.Evaluation(state);
00028                 }
00029                 else{
00030                     tempEval = 1-objfunction.Evaluation(state);
00031                 }
00032             }
00033             else{
00034                 if(objfunction.getTypeProblem().equals(ProblemType.Maximizar))
00035                 {
00036                     tempEval = 1-objfunction.Evaluation(state);
00037                 }
00038                 else{
00039                     tempEval = objfunction.Evaluation(state);
00040                 }
00041             }
00042             evaluation.add(tempEval);
00043         }
00044         //evaluation.add( (double) -1);
00045         state.setEvaluation(evaluation);
00046     }
00047 }
00048 }
```

## 9.205 Referencia del archivo SolutionMethod.java

import problem.definition.State;  
Gráfico de dependencias incluidas en SolutionMethod.java:



## Clases

- class problem.extension.SolutionMethod  
*SolutionMethod.*

## Paquetes

- package problem.extension

## 9.206 SolutionMethod.java

[Ir a la documentación de este archivo.](#)

```
00001 package problem.extension;
00002
00003 import problem.definition.State;
00004
00012 public abstract class SolutionMethod {
00013
00020     public abstract void evaluationState(State state);
00021 }
```

## 9.207 Referencia del archivo TypeSolutionMethod.java

## Clases

- enum problem.extension.TypeSolutionMethod  
*TypeSolutionMethod.*

## Paquetes

- package problem.extension

## 9.208 TypeSolutionMethod.java

[Ir a la documentación de este archivo.](#)

```
00001 package problem.extension;
00002
00009 public enum TypeSolutionMethod {
00010
00011     FactoresPonderados, MultiObjetivoPuro; //OrdenamientoLexicografico,
00012 }
```

## 9.209 Referencia del archivo MutationOperator.java

```
import java.util.ArrayList;
import java.util.List;
import metaheuristics.strategy.Strategy;
import problem.definition.Operator;
import problem.definition.State;
```

Gráfico de dependencias incluidas en MutationOperator.java:



### Clases

- class [problem\\_operators.MutationOperator](#)  
*MutationOperator - defines a mutation operator for generating new states.*

### Paquetes

- package [problem\\_operators](#)

## 9.210 MutationOperator.java

[Ir a la documentación de este archivo.](#)

```
00001 package problem_operators;
00002
00003 import java.util.ArrayList;
00004 import java.util.List;
00005
00006 import metaheuristics.strategy.Strategy;
00007
00008 import problem.definition.Operator;
00009 import problem.definition.State;
00010
00011
00015 public class MutationOperator extends Operator {
00016
00032     public List<State> generatedNewState(final State stateCurrent,
00033             final Integer operatornumber) {
00034         List<State> listNeigborhood = new ArrayList<State>();
00035         for (int i = 0; i < operatornumber; i++) {
00036             int key = Strategy.getStrategy().getProblem()
00037                 .getCodification()
00038                 .getAleatoryKey();
00039             Object candidate = Strategy.getStrategy().getProblem()
00040                 .getCodification()
00041                 .getVariableAleatoryValue(key);
00042             State state = (State) stateCurrent.getCopy();
00043             state.getCode().set(key, candidate);
00044             listNeigborhood.add(state);
00045         }
00046         return listNeigborhood;
00047     }
00048
00064     @Override
00070     public List<State> generateRandomState(final Integer operatornumber) {
00071         // TODO Auto-generated method stub
00072         return null;
00073     }
00074
00075 }
```

