HNCO

0.7

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 hnco Namespace Reference

top-level HNCO namespace

Namespaces

- algorithm

  Algorithms.
- exception

  Exceptions.
- function

  Functions to be maximized.
- neighborhood

  Neighborhoods for local search.
- random

  Pseudo random numbers.

Classes

- class AffineMap

  Affine map.
- class HypercubeIterator

  Hypercube iterator.
- class Iterator

  Iterator over bit vectors.
- class LinearMap

  Linear map.
- class Map

  Map.
- class MapComposition

  Map composition.
- class Permutation

  Permutation.
- class Translation

  Translation.

Types and functions related to bit matrices

- typedef std::vector< bit_vector_t > bit_matrix_t

    Bit matrix.
- void bm_display (const bit_matrix_t &M, std::ostream &stream)

    Display bit matrix.
- bool bm_is_valid (const bit_matrix_t &M)

    Check whether a bit matrix is valid.
- size_t bm_num_rows (const bit_matrix_t &M)

    Number of rows.
- size_t bm_num_columns (const bit_matrix_t &M)

    Number of columns.
- bool bm_is_square (const bit_matrix_t &M)

    Check whether the matrix is a square matrix.
- bool bm_is_identity (const bit_matrix_t &M)

    Check whether the matrix is the identity matrix.
- bool bm_is_upper_triangular (const bit_matrix_t &M)

    Check whether the matrix is upper triangular.
- void bm_resize (bit_matrix_t &M, std::size_t num_rows, std::size_t num_columns)

    Resize a bit matrix.
- void bm_resize (bit_matrix_t &M, std::size_t num_rows)

    Resize a bit matrix and make it a square matrix.
- void bm_clear (bit_matrix_t &M)

    Clear bit matrix.
- void bm_identity (bit_matrix_t &M)

    Set the matrix to the identity matrix.
- void bm_random (bit_matrix_t &M)

    Sample a random bit matrix.
- void bm_swap_rows (bit_matrix_t &M, std::size_t i, std::size_t j)

    Swap two rows.
- void bm_add_rows (bit_matrix_t &M, std::size_t i, std::size_t j)

    Add two rows.
- bool bm_solve (bit_matrix_t &A, bit_vector_t &b)

    Solve a linear system.
- bool bm_solve_upper_triangular (bit_matrix_t &A, bit_vector_t &b)

    Solve a linear system in upper triangular form.
- bool bm_invert (bit_matrix_t &M, bit_matrix_t &N)

    Invert a bit matrix.
- void bm_multiply (const bit_matrix_t &M, const bit_vector_t &x, bit_vector_t &y)

    Multiply a bit matrix and a bit vector.
- void bm_transpose (const bit_matrix_t &M, bit_matrix_t &N)

    Transpose.

Types and functions related to bit vectors

- typedef char bit_t

    Bit.
- typedef std::vector< bit_t > bit_vector_t

    Bit vector.
- typedef std::pair< bit_vector_t, double > point_value_t

Type to represent point value pairs.

- bit_t bit_flip (bit_t b)

  Flip bit.
- void bv_display (const bit_vector_t &v, std::ostream &stream)

  Display bit vector.
- bool bv_is_valid (const bit_vector_t &x)

  Check whether the bit vector is valid.
- bool bv_is_zero (const bit_vector_t &x)

  Check whether the bit vector is zero.
- int bv_hamming_weight (const bit_vector_t &x)

  Hamming weight.
- int bv_hamming_distance (const bit_vector_t &x, const bit_vector_t &y)

  Hamming distance between two bit vectors.
- bit_t bv_dot_product (const bit_vector_t &x, const bit_vector_t &y)

  Dot product.
- void bv_clear (bit_vector_t &x)

  Clear bit vector.
- void bv_flip (bit_vector_t &x, std::size_t i)

  Flip a single bit.
- void bv_flip (bit_vector_t &x, const bit_vector_t &mask)

  Flip many bits.
- void bv_random (bit_vector_t &x)

  Sample a random bit vector.
- void bv_random (bit_vector_t &x, int k)

  Sample a random bit vector with given Hamming weight.
- void bv_add (const bit_vector_t &src, bit_vector_t &dest)

  Add two bit vectors.
- void bv_add (const bit_vector_t &x, const bit_vector_t &y, bit_vector_t &dest)

  Add two bit vectors.

Types and functions related to permutations

- typedef std::vector< std::size_t > permutation_t

  Permutation type.
- bool perm_is_valid (const permutation_t &permutation)

  Check that a vector represents a permutation.
- void perm_random (permutation_t &s)

  Sample a random permutation.

Types and functions related to sparse bit matrices

- typedef std::vector< sparse_bit_vector_t > sparse_bit_matrix_t

  Sparse bit matrix.
- void sbm_display (const sparse_bit_matrix_t &sbm, std::ostream &stream)

  Display sparse bit matrix.
- void bm_to_sbm (const bit_matrix_t &bm, sparse_bit_matrix_t &sbm)

  Convert a bit matrix to a sparse bit matrix.
- void sbm_multiply (const sparse_bit_matrix_t &M, const bit_vector_t &x, bit_vector_t &y)

  Multiply a sparse bit matrix and a bit vector.

Types and functions related to sparse bit vectors

- typedef std::vector< std::size_t > sparse_bit_vector_t

  Sparse bit vector.
- void bv_flip (bit_vector_t &x, const sparse_bit_vector_t &sbv)

  Flip many bits.
- void sbv_display (const sparse_bit_vector_t &v, std::ostream &stream)

  Display sparse bit vector.
- void bv_to_sbv (const bit_vector_t &bv, sparse_bit_vector_t &sbv)

  Convert a bit vector to a sparse bit vector.

4.1.1 Detailed Description

top-level HNCO namespace

4.1.2 Typedef Documentation

4.1.2.1 bit_t

```
typedef char bit_t
```

Bit.

A single bit is represented by a char and the values 0 for false and 1 for true.

Definition at line 52 of file bit-vector.hh.

4.1.2.2 sparse_bit_matrix_t

```
typedef std::vector<sparse_bit_vector_t> sparse_bit_matrix_t
```

Sparse bit matrix.

A sparse bit matrix is represented as an array of sparse bit vectors. It knows its number of row, not its number of columns.

Definition at line 45 of file sparse-bit-matrix.hh.

4.1.2.3 sparse_bit_vector_t

```
typedef std::vector<std::size_t> sparse_bit_vector_t
```

Sparse bit vector.

A sparse bit vector is represented as an array containing the indices of its non-zero components. The indices must be sorted in ascending order.

A sparse bit vector does not know the dimension of the space it belongs to.

Definition at line 47 of file sparse-bit-vector.hh.

4.1.3 Function Documentation

4.1.3.1 bm_add_rows()

```
void bm_add_rows (
             bit_matrix_t & M,
             std::size_t i,
             std::size_t j )
```

Add two rows.

Row i is added to row j.

Definition at line 94 of file bit-matrix.cc.

4.1.3.2 bm_identity()

```
void bm_identity (
             bit_matrix_t & M )
```

Set the matrix to the identity matrix.

Precondition

bm_is_square(M)

Definition at line 29 of file bit-matrix.cc.

4.1.3.3 bm_invert()

```
bool bm_invert (
             bit_matrix_t & M,
             bit_matrix_t & N )
```

Invert a bit matrix.

Parameters

M	input matrix
N	inverse matrix

Precondition

bm_is_square(M)
bm_is_square(N)

Returns

true if M is invertible

Warning

M is modified by the function. Provided that M is invertible, after returning from the function, M is the identity matrix and N is the computed inverse matrix.

Definition at line 153 of file bit-matrix.cc.

4.1.3.4 bm_multiply()

```
void bm_multiply (
             const bit_matrix_t & M,
             const bit_vector_t & x,
             bit_vector_t & y )
```

Multiply a bit matrix and a bit vector.

The result is y = Mx.

Definition at line 195 of file bit-matrix.cc.

4.1.3.5 bm_solve()

```
bool bm_solve (
             bit_matrix_t & A,
             bit_vector_t & b )
```

Solve a linear system.

Solve the linear equation Ax = b.

Parameters

A	Matrix
b	Right hand side

Precondition

bm_is_square(A)
bm_num_rows(A) == b.size()

Returns

true if the system has a unique solution

Warning

Both A and b are modified by the function. Provided that A is invertible, after returning from the function, A is the identity matrix and b is the unique solution to the linear equation.

Definition at line 103 of file bit-matrix.cc.

4.1.3.6 bm_solve_upper_triangular()

```
bool bm_solve_upper_triangular (
            bit_matrix_t & A,
            bit_vector_t & b )
```

Solve a linear system in upper triangular form.

Solve the linear equation Ax = b.

Parameters

A	Upper triangular matrix
b	Right hand side

Precondition

bm_is_square(A)
bm_num_rows(A) == b.size()
bm_is_upper_triangular(A)

Returns

true if the system has a unique solution

Warning

> Both A and b are modified by the function. Provided that A is invertible, after returning from the function, A is the identity matrix and b is the unique solution to the linear equation.

Definition at line 134 of file bit-matrix.cc.

4.1.3.7 sbm_multiply()

```
void hnco::sbm_multiply (
            const sparse_bit_matrix_t & M,
            const bit_vector_t & x,
            bit_vector_t & y )  [inline]
```

Multiply a sparse bit matrix and a bit vector.

The result is y = Mx.

Definition at line 68 of file sparse-bit-matrix.hh.

4.2 hnco::algorithm Namespace Reference

Algorithms.

Namespaces

- bm_pbil

  Boltzmann machine PBIL.
- hea

  Herding evolutionary algorithm.

Classes

- class Algorithm

  Abstract search algorithm.
- class CompactGa

  Compact genetic algorithm.
- class CompleteSearch

  Complete search.
- class GeneticAlgorithm

  Genetic algorithm.
- class IterativeAlgorithm

  Iterative search.
- class Mmas

  Max-min ant system.
- class MuCommaLambdaEa

  (mu,lambda) EA

- class MuPlusLambdaEa

  (mu+lambda) EA
- class NpsPbil

  Population-based incremental learning with negative and positive selection.
- class OnePlusOneEa

  (1+1) EA.
- class Pbil

  Population-based incremental learning.
- class Population

  Population.
- class PvAlgorithm

  Probability vector algorithm.
- class RandomLocalSearch

  Random local search.
- class RandomSearch

  Random search.
- class Restart

  Restart.
- class SimulatedAnnealing

  Simulated annealing.
- class SteepestAscentHillClimbing

  Steepest ascent hill climbing.
- class TournamentSelection

  Population with tournament selection.
- class Umda

  Univariate marginal distribution algorithm.

Functions

- template<class T >
  bool matrix_is_symmetric (const std::vector< std::vector< T > > &A)

  Check for symmetric matrix.
- template<class T >
  bool matrix_has_diagonal (const std::vector< std::vector< T > > &A, T x)

  Check for diagonal elements.
- template<class T >
  bool matrix_has_range (const std::vector< std::vector< T > > &A, T inf, T sup)

  Check for element range.
- template<class T >
  bool matrix_has_dominant_diagonal (const std::vector< std::vector< T > > &A)

  Check for element range.
- template<class T >
  T square (T x)

  Generic square function.
- double logistic (double x)

  Logistic function (sigmoid)

Type and functions related to probability vectors

- typedef std::vector< double > pv_t

  Probability vector type.
- double pv_entropy (const pv_t &pv)

  Entropy of a probability vector.
- void pv_sample (const pv_t &pv, bit_vector_t &x)

  Sample a bit vector.
- void pv_uniform (pv_t &pv)

  Probability vector of the uniform distribution.
- void pv_init (pv_t &pv)

  Initialize.
- void pv_add (pv_t &pv, const bit_vector_t &x)

  Accumulate a bit vector.
- void pv_add (pv_t &pv, const bit_vector_t &x, double weight)

  Accumulate a bit vector.
- void pv_average (pv_t &pv, int count)

  Average.
- void pv_update (pv_t &pv, double rate, const bit_vector_t &x)

  Update a probability vector toward a bit vector.
- void pv_update (pv_t &pv, double rate, const std::vector< double > &x)

  Update a probability vector toward a probability vector.
- void pv_update (pv_t &pv, double rate, const std::vector< double > &x, const std::vector< double > &y)

  Update a probability vector toward a probability vector and away from another one.
- void pv_bound (pv_t &pv, double lower_bound, double upper_bound)

  Bound the components of a probability vector.

4.2.1 Detailed Description

Algorithms.

4.3 hnco::algorithm::bm_pbil Namespace Reference

Boltzmann machine PBIL.

Classes

- class BmPbil

  Boltzmann machine PBIL.
- class Model

  Model of a Boltzmann machine.
- class ModelParameters

  Parameters of a Boltzmann machine.

4.3.1 Detailed Description

Boltzmann machine PBIL.

4.4 hnco::algorithm::hea Namespace Reference

Herding evolutionary algorithm.

Classes

- class BinaryHerding

  Herding with binary variables.
- struct BinaryMoment

  Moment for binary variables.
- class Hea

  Herding evolutionary algorithm.
- class SpinHerding

  Herding with spin variables.
- struct SpinMoment

  Moment for spin variables.

4.4.1 Detailed Description

Herding evolutionary algorithm.

4.5 hnco::exception Namespace Reference

Exceptions.

Classes

- class Error

  Error.
- class Exception

  Basic exception.
- class LastEvaluation

  Last evaluation.
- class LocalMaximum

  Local maximum.
- class MaximumReached

  Maximum reached.
- class PointValueException

  Point-value exception.
- class TargetReached

  target reached

4.5.1 Detailed Description

Exceptions.

4.6 hnco::function Namespace Reference

Functions to be maximized.

Classes

- class AdditiveGaussianNoise

  Additive Gaussian Noise.
- class Cache

  Cache.
- class CallCounter

  Call counter.
- class DeceptiveJump

  Deceptive jump.
- class EqualProducts

  Equal products.
- class Factorization

  Factorization.
- class FourPeaks

  Four Peaks.
- class Function

  Function.
- class FunctionController

  Function controller.
- class FunctionDecorator

  Function decorator.
- class FunctionMapComposition

  Composition of a function and a map.
- class FunctionModifier

  Function modifier.
- class FunctionPlugin

  Function plugin.
- class Hiff

  Hierarchical if and only if.
- class Jump

  Jump.
- class Labs

  Low autocorrelation binary sequences.
- class LeadingOnes

  Leading ones.
- class LinearFunction

  Linear function.
- class LongPath

  Long path.
- class MaxSat

  MAX-SAT.
- class Needle

  Needle in a haystack.
- class Negation

Negation.

- class NkLandscape

  NK landscape.
- class OnBudgetFunction

  CallCounter with a limited number of evaluations.
- class OneMax

  OneMax.
- class Plateau

  Plateau.
- class ProgressTracker

  ProgressTracker.
- class Qubo

  Quadratic unconstrained binary optimization.
- class Ridge

  Ridge.
- class SinusSummationCancellation

  Summation cancellation with sinus.
- class SixPeaks

  Six Peaks.
- class StopOnMaximum

  Stop on maximum.
- class StopOnTarget

  Stop on target.
- class SummationCancellation

  Summation cancellation.
- class Trap

  Trap.
- class WalshExpansion

  Walsh expansion.
- class WalshExpansion1

  Walsh expansion of degree 1.
- class WalshExpansion2

  Walsh expansion of degree 2.

Functions

- std::ostream & operator<< (std::ostream &stream, const ProgressTracker::Event &event)

  Insert formatted output.

4.6.1 Detailed Description

Functions to be maximized.

4.7 hnco::neighborhood Namespace Reference

Neighborhoods for local search.

Classes

- class BernoulliProcess

  Bernoulli process.
- class HammingBall

  Hamming ball.
- class HammingBallIterator

  Hamming ball neighborhood iterator.
- class HammingSphere

  Hamming sphere.
- class MultiBitFlip

  Multi bit flip.
- class Neighborhood

  Neighborhood.
- class NeighborhoodIterator

  Neighborhood iterator.
- class SingleBitFlip

  One bit neighborhood.
- class SingleBitFlipIterator

  Single bit flip neighborhood iterator.

4.7.1 Detailed Description

Neighborhoods for local search.

There are two unrelated kinds of neighborhoods, those for random local search and those for exhaustive local search.

4.8 hnco::random Namespace Reference

Pseudo random numbers.

Classes

- struct Random

  Random numbers.

4.8.1 Detailed Description

Pseudo random numbers.

Chapter 5

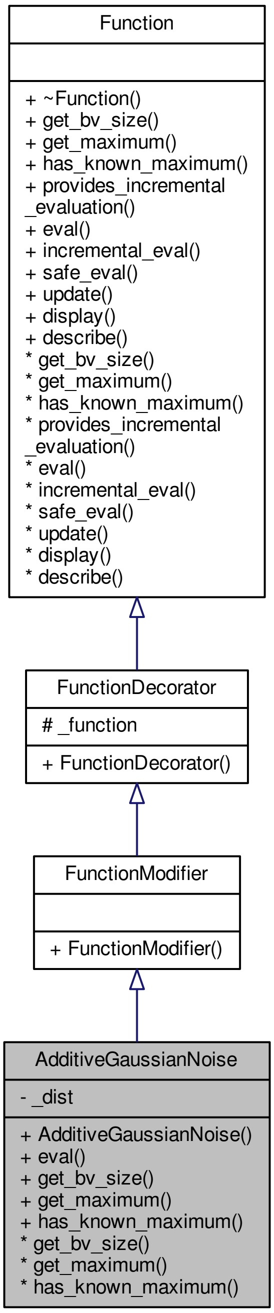
Class Documentation

5.1 AdditiveGaussianNoise Class Reference

Additive Gaussian Noise.

```
#include <hnco/functions/decorators/function-modifier.hh>
```

Inheritance diagram for AdditiveGaussianNoise:



Public Member Functions

- AdditiveGaussianNoise (Function ∗function, double stddev)

  Constructor.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.

Information about the function

- size_t get_bv_size ()

  Get bit vector size.
- double get_maximum ()

  Get the global maximum.
- bool has_known_maximum ()

  Check for a known maximum.

Private Attributes

- std::normal_distribution< double > _dist

  Normal distribution.

Additional Inherited Members

5.1.1 Detailed Description

Additive Gaussian Noise.

Definition at line 166 of file function-modifier.hh.

5.1.2 Member Function Documentation

5.1.2.1 get_maximum()

```
double get_maximum ( )  [inline], [virtual]
```

Get the global maximum.

Exceptions

Error	

Reimplemented from Function.

Definition at line 188 of file function-modifier.hh.

5.1.2.2 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

false

Reimplemented from Function.

Definition at line 192 of file function-modifier.hh.

The documentation for this class was generated from the following files:

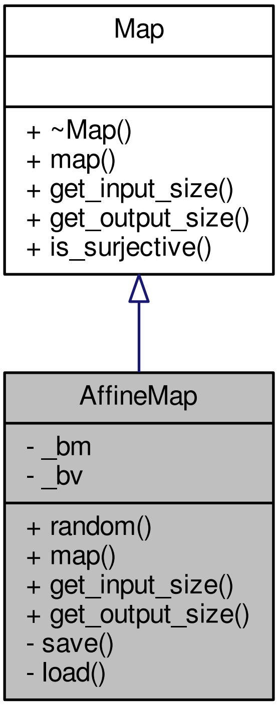
- lib/hnco/functions/decorators/function-modifier.hh
- lib/hnco/functions/decorators/function-modifier.cc

5.2 AffineMap Class Reference

Affine map.

```
#include <hnco/map.hh>
```

Inheritance diagram for AffineMap:



Public Member Functions

- void random (int n, int m)

  Random instance.
- void map (const bit_vector_t &input, bit_vector_t &output)

  Map.
- size_t get_input_size ()

  Get input size.
- size_t get_output_size ()

  Get output size.

Private Member Functions

- template<class Archive >
  void save (Archive &ar, const unsigned int version) const

  Save.
- template<class Archive >
  void load (Archive &ar, const unsigned int version)

  Load.

Private Attributes

- bit_matrix_t _bm

  Bit matrix.
- bit_vector_t _bv

  Translation vector.

Friends

- class **boost::serialization::access**

5.2.1 Detailed Description

Affine map.

An affine map f from Z_2^m to Z_2^n is defined by $f(x) = Ax+b$, where A is an n x m bit matrix and b is an n-dimensional bit vector.

Warning

> The class does not reimplement the member function is_surjective hence a linear map is always considered not surjective.

Definition at line 258 of file map.hh.

The documentation for this class was generated from the following files:

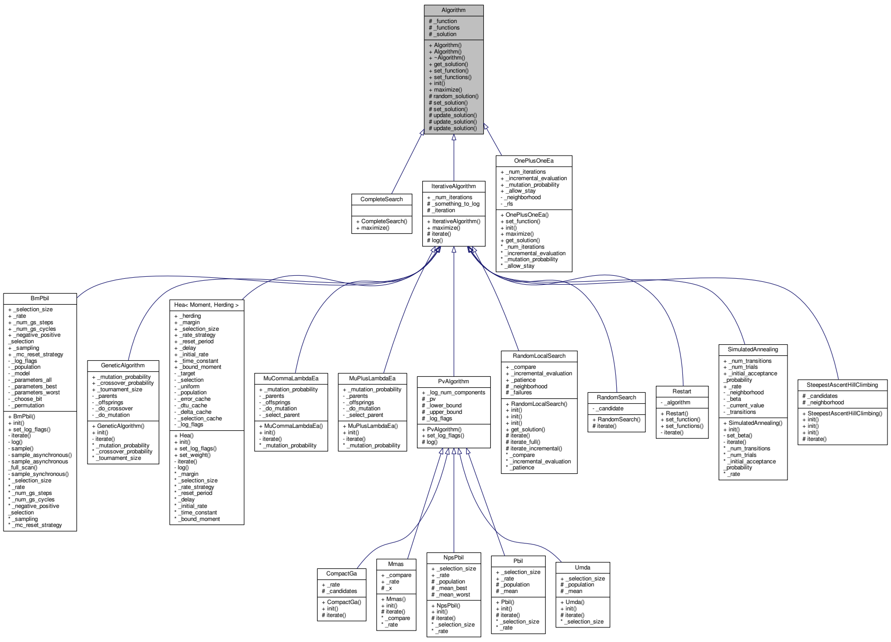
- lib/hnco/map.hh
- lib/hnco/map.cc

5.3 Algorithm Class Reference

Abstract search algorithm.

```
#include <hnco/algorithms/algorithm.hh>
```

Inheritance diagram for Algorithm:



Public Member Functions

- Algorithm ()

  Constructor.
- Algorithm (int n)

  Constructor.
- virtual ∼Algorithm ()

  Destructor.
- virtual const point_value_t & get_solution ()

  Solution.
- virtual void set_function (function::Function ∗function)

  Set function.
- virtual void set_functions (const std::vector< function::Function ∗> functions)

  Set functions.
- virtual void init ()

  Initialization.
- virtual void maximize ()=0

  Maximize.

Protected Member Functions

- virtual void random_solution ()
  Random solution.
- virtual void set_solution (const bit_vector_t &x, double value)
  Set solution.
- virtual void set_solution (const bit_vector_t &x)
  Set solution.
- virtual void update_solution (const bit_vector_t &x, double value)
  Update solution (strict)
- virtual void update_solution (const point_value_t &pv)
  Update solution (strict)
- virtual void update_solution (const bit_vector_t &x)
  Update solution (strict)

Protected Attributes

- function::Function ∗ _function
  Function.
- std::vector< function::Function ∗ > _functions
  Functions.
- point_value_t _solution
  Solution.

5.3.1 Detailed Description

Abstract search algorithm.

All algorithms maximize some given function, sometimes called a fitness function or an objective function.

Definition at line 36 of file algorithm.hh.

5.3.2 Member Data Documentation

5.3.2.1 _functions

```
std::vector<function::Function *> _functions  [protected]
```

Functions.

Each thread has its own function.

Definition at line 47 of file algorithm.hh.

The documentation for this class was generated from the following files:

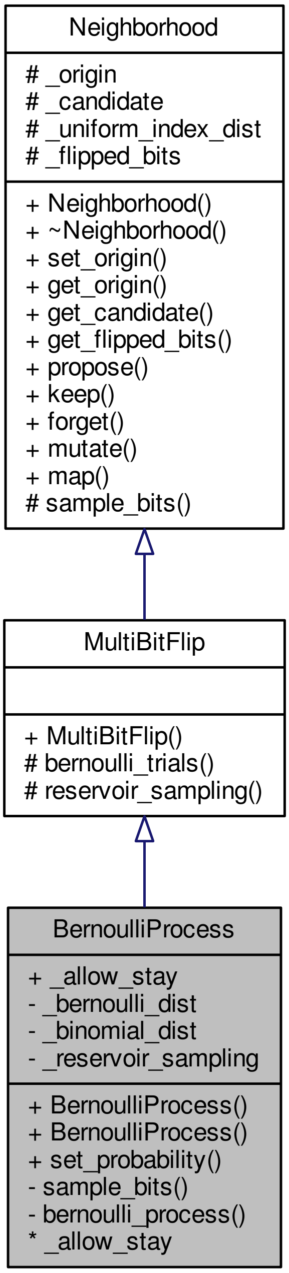
- lib/hnco/algorithms/algorithm.hh
- lib/hnco/algorithms/algorithm.cc

5.4 BernoulliProcess Class Reference

Bernoulli process.

```
#include <hnco/neighborhoods/neighborhood.hh>
```

Inheritance diagram for BernoulliProcess:



Public Member Functions

- BernoulliProcess (int n)

  Constructor.
- BernoulliProcess (int n, double p)

  Constructor.
- void set_probability (double p)

  Set probability.

Public Attributes

Parameters

- bool _allow_stay = false

  Allow stay.

Private Member Functions

- void sample_bits ()

  Sample bits.
- void bernoulli_process ()

  Bernoulli process.

Private Attributes

- std::bernoulli_distribution _bernoulli_dist

  Bernoulli distribution (biased coin)
- std::binomial_distribution$<$ int $>$ _binomial_dist

  Binomial distribution.
- bool _reservoir_sampling = false

  Reservoir sampling.

Additional Inherited Members

5.4.1 Detailed Description

Bernoulli process.

Each component of the origin bit vector is flipped with some fixed probability. If no component has been flipped at the end, the process is started all over again. Thus the number of flipped bits follows a pseudo binomial law.

Definition at line 220 of file neighborhood.hh.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 BernoulliProcess() [1/2]

```
BernoulliProcess (
                int n )   [inline]
```

Constructor.

Parameters

n	Size of bit vectors

The Bernoulli probability is set to 1 / n.

Definition at line 246 of file neighborhood.hh.

5.4.2.2 BernoulliProcess() [2/2]

```
BernoulliProcess (
            int n,
            double p )  [inline]
```

Constructor.

Parameters

n	Size of bit vectors
p	Bernoulli probability

Definition at line 256 of file neighborhood.hh.

5.4.3 Member Function Documentation

5.4.3.1 set_probability()

```
void set_probability (
            double p )  [inline]
```

Set probability.

Sets _reservoir_sampling to true if E(X) $<$ sqrt(n), where X is a random variable with a binomial distribution B(n, p), that is if np $<$ sqrt(n) or p $<$ 1 / sqrt(n).

Definition at line 267 of file neighborhood.hh.

5.4.4 Member Data Documentation

5.4.4.1 _allow_stay

```
bool _allow_stay = false
```

Allow stay.

In case no mutation occurs allow the current bit vector to stay unchanged.

Definition at line 283 of file neighborhood.hh.

The documentation for this class was generated from the following files:

- lib/hnco/neighborhoods/neighborhood.hh
- lib/hnco/neighborhoods/neighborhood.cc

5.5 BinaryHerding Class Reference

Herding with binary variables.

```
#include <hnco/algorithms/hea/herding-binary.hh>
```

Public Types

- enum { DYNAMICS_MINIMIZE_NORM, DYNAMICS_MAXIMIZE_INNER_PRODUCT }

Public Member Functions

- BinaryHerding (int n)

  Constructor.
- void init ()

  Initialization.
- double error (const BinaryMoment &target)

  Compute the error.
- double delta (const BinaryMoment &target)

  Compute the norm of delta.
- void sample (const BinaryMoment &target, bit_vector_t &x)

  Sample a bit vector.

Public Attributes

Parameters

- bool _randomize_bit_order = false

  Randomize bit order.
- int _dynamics = DYNAMICS_MINIMIZE_NORM

  Dynamics.
- double _weight = 1

  Weight of second order moments.

Protected Member Functions

- void compute_delta (const BinaryMoment &target)

  Compute delta.
- void sample_minimize_norm (const BinaryMoment &target, bit_vector_t &x)

  Sample a bit vector.
- void sample_maximize_inner_product (const BinaryMoment &target, bit_vector_t &x)

  Sample a bit vector.

Protected Attributes

- BinaryMoment _count

  Counter moment.
- BinaryMoment _delta

  Delta moment.
- permutation_t _permutation

  Permutation.
- std::uniform_int_distribution< int > _choose_bit

  Choose bit.
- int _time

  Time.

5.5.1 Detailed Description

Herding with binary variables.

Definition at line 38 of file herding-binary.hh.

5.5.2 Member Enumeration Documentation

5.5.2.1 anonymous enum

```
anonymous enum
```

Enumerator

DYNAMICS_MINIMIZE_NORM	Dynamics defined as minimization of a norm.
DYNAMICS_MAXIMIZE_INNER_PRODUCT	Dynamics defined as maximization of an inner product.

Definition at line 69 of file herding-binary.hh.

The documentation for this class was generated from the following files:

- lib/hnco/algorithms/hea/herding-binary.hh
- lib/hnco/algorithms/hea/herding-binary.cc

5.6 BinaryMoment Struct Reference

Moment for binary variables.

```
#include <hnco/algorithms/hea/moment-binary.hh>
```

Public Member Functions

- BinaryMoment (int n)

  Constructor.
- void uniform ()

  Set the moment to that of the uniform distribution.
- void init ()

  Initialize.
- void add (const bit_vector_t &x)

  Accumulate a bit vector.
- void average (int count)

  Compute average.
- void update (const BinaryMoment &p, double rate)

  Update moment.
- void bound (double margin)

  Bound moment.
- double distance (const BinaryMoment &p) const

  Distance.
- double norm_2 () const

  Compute the norm 2.
- double diameter () const

  Compute the diameter.
- size_t size () const

  Size.

Public Attributes

- std::vector< std::vector< double > > _moment

  Moment.
- double _weight = 1

  Weight of second order moments.

5.6.1 Detailed Description

Moment for binary variables.

Definition at line 37 of file moment-binary.hh.

The documentation for this struct was generated from the following files:

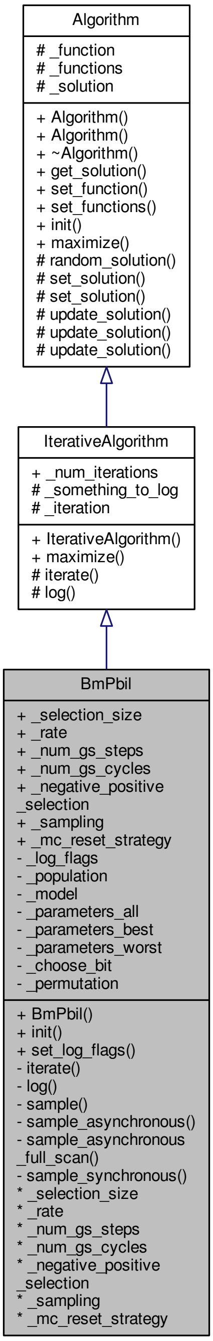
- lib/hnco/algorithms/hea/moment-binary.hh
- lib/hnco/algorithms/hea/moment-binary.cc

5.7 BmPbil Class Reference

Boltzmann machine PBIL.

```
#include <hnco/algorithms/bm-pbil/bm-pbil.hh>
```

Inheritance diagram for BmPbil:



Public Types

- enum { LOG_NORM_INFINITE, LOG_NORM_L1, **LAST_LOG** }
- enum { SAMPLING_ASYNCHRONOUS, SAMPLING_ASYNCHRONOUS_FULL_SCAN, SAMPLING_SY←NCHRONOUS }
- enum { RESET_NO_RESET, RESET_ITERATION, RESET_BIT_VECTOR }
- typedef std::bitset< LAST_LOG > **log_flags_t**

Public Member Functions

- BmPbil (int n, int population_size)

  Constructor.
- void init ()

  Initialization.
- void set_log_flags (const log_flags_t &lf)

  Set log flags.

Public Attributes

Parameters

- int _selection_size = 1

  Selection size (number of selected individuals in the population)
- double _rate = 1e-3

  Learning rate.
- int _num_gs_steps = 100

  Number of gibbs sampler steps.
- int _num_gs_cycles = 1

  Number of gibbs sampler cycles.
- bool _negative_positive_selection = false

  Negative and positive selection.
- int _sampling = SAMPLING_ASYNCHRONOUS

  Sampling mode.
- int _mc_reset_strategy = RESET_NO_RESET

  MC reset strategy.

Private Member Functions

- void iterate ()

  Single iteration.
- void log ()

  Log.
- void sample (bit_vector_t &x)

  Sample a bit vector.
- void sample_asynchronous ()

  Asynchronous sampling.
- void sample_asynchronous_full_scan ()

  Asynchronous sampling with full scan.
- void sample_synchronous ()

  Synchronous sampling.

Private Attributes

- log_flags_t _log_flags

    Log flags.
- Population _population

    Population.
- Model _model

    Model.
- ModelParameters _parameters_all

    Parameters averaged over all individuals.
- ModelParameters _parameters_best

    Parameters averaged over selected individuals.
- ModelParameters _parameters_worst

    Parameters averaged over negatively selected individuals.
- std::uniform_int_distribution< size_t > _choose_bit

    Uniform distribution on bit_vector_t components.
- permutation_t _permutation

    Permutation.

Additional Inherited Members

5.7.1 Detailed Description

Boltzmann machine PBIL.

Definition at line 40 of file bm-pbil.hh.

5.7.2 Member Enumeration Documentation

5.7.2.1 anonymous enum

```
anonymous enum
```

Enumerator

LOG_NORM_INFINITE	Log infinite norm of the model parameters.
LOG_NORM_L1	Log 1-norm of the model parameters.

Definition at line 45 of file bm-pbil.hh.

5.7.2.2 anonymous enum

```
anonymous enum
```

Enumerator

SAMPLING_ASYNCHRONOUS	Asynchronous sampling. A single component of the internal state is randomly selected then updated by Gibbs sampling. This step is repeated _num_gs_steps times.
SAMPLING_ASYNCHRONOUS_FULL_SCAN	Asynchronous sampling with full scan. To sample a new bit vector, a random permutation is sampled and all components of the internal state are updated by Gibbs sampling in the order defined by the permutation.
SAMPLING_SYNCHRONOUS	Synchronous sampling. The full internal state is updated in one step from the probability vector made of the very marginal probabilities used in Gibbs sampling.

Definition at line 55 of file bm-pbil.hh.

5.7.2.3 anonymous enum

```
anonymous enum
```

Enumerator

RESET_NO_RESET	No reset.
RESET_ITERATION	Reset MC at the beginning of each iteration.
RESET_BIT_VECTOR	Reset MC before sampling each bit vector.

Definition at line 82 of file bm-pbil.hh.

The documentation for this class was generated from the following files:

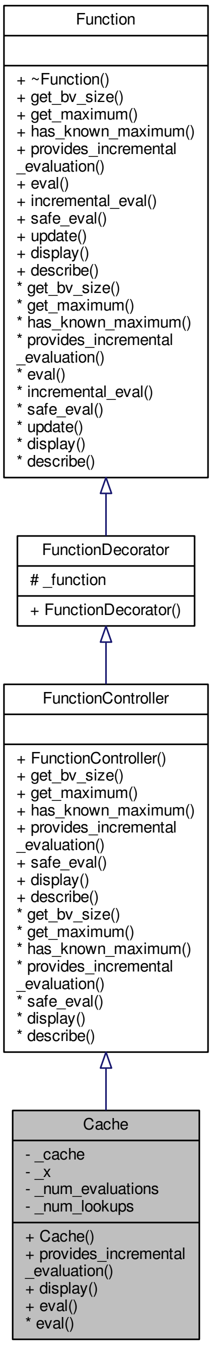
- lib/hnco/algorithms/bm-pbil/bm-pbil.hh
- lib/hnco/algorithms/bm-pbil/bm-pbil.cc

5.8 Cache Class Reference

Cache.

```
#include <hnco/functions/decorators/function-controller.hh>
```

Inheritance diagram for Cache:



Public Member Functions

- Cache (Function ∗function)

    Constructor.

- bool provides_incremental_evaluation ()

    Check whether the function provides incremental evaluation.

- void display (std::ostream &stream)

Display.

Evaluation

- double eval (const bit_vector_t &)

  Evaluate a bit vector.

Private Attributes

- std::unordered_map< std::vector< bool >, double > _cache

  Database of past evaluations.
- std::vector< bool > _x

  STL bit vector.
- int _num_evaluations

  Evaluation counter.
- int _num_lookups

  Lookup counter.

Additional Inherited Members

5.8.1 Detailed Description

Cache.

This is a naive approach, in particular with respect to time complexity. Moreover, there is no control on the size of the database.

There is no default hash function for std::vector<char> hence the need to first copy a bit_vector_t into a std↩ ::vector<bool>, for which such a function exists, before inserting it or checking its existence in the map.

Definition at line 344 of file function-controller.hh.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Cache()

```
Cache (
            Function * function )   [inline]
```

Constructor.

Parameters

function	Decorated function

Definition at line 363 of file function-controller.hh.

5.8.3 Member Function Documentation

5.8.3.1 provides_incremental_evaluation()

```
bool provides_incremental_evaluation ( )  [inline], [virtual]
```

Check whether the function provides incremental evaluation.

Returns

false

Reimplemented from FunctionController.

Definition at line 372 of file function-controller.hh.

The documentation for this class was generated from the following files:

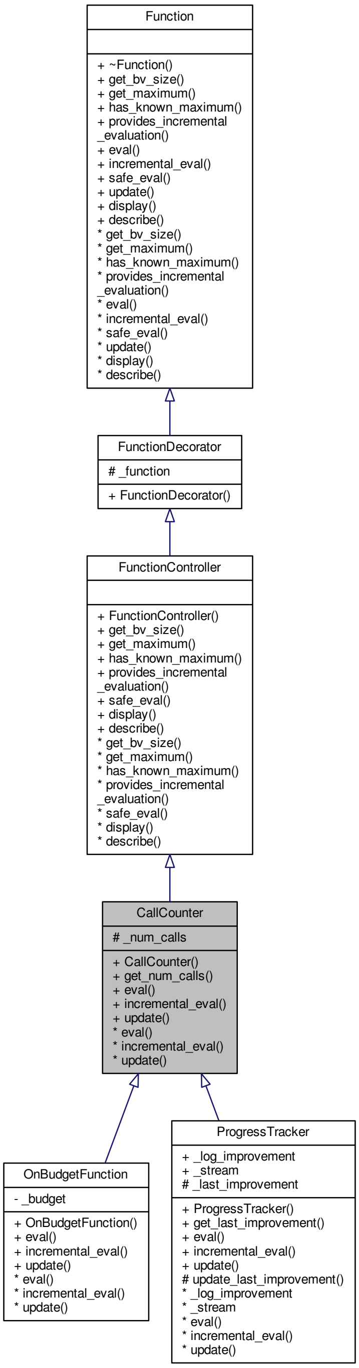
- lib/hnco/functions/decorators/function-controller.hh
- lib/hnco/functions/decorators/function-controller.cc

5.9 CallCounter Class Reference

Call counter.

```
#include <hnco/functions/decorators/function-controller.hh>
```

Inheritance diagram for CallCounter:



Public Member Functions

- CallCounter (Function ∗function)

    Constructor.

- int get_num_calls ()

    Get the number of calls.

Evaluation

- double eval (const bit_vector_t &)
  Evaluate a bit vector.
- double incremental_eval (const bit_vector_t &x, double value, const hnco::sparse_bit_vector_t &flipped↩_bits)
  Incremental evaluation.
- void update (const bit_vector_t &x, double value)
  Update after a safe evaluation.

Protected Attributes

- int _num_calls
  Number of calls.

5.9.1 Detailed Description

Call counter.

Definition at line 170 of file function-controller.hh.

The documentation for this class was generated from the following files:

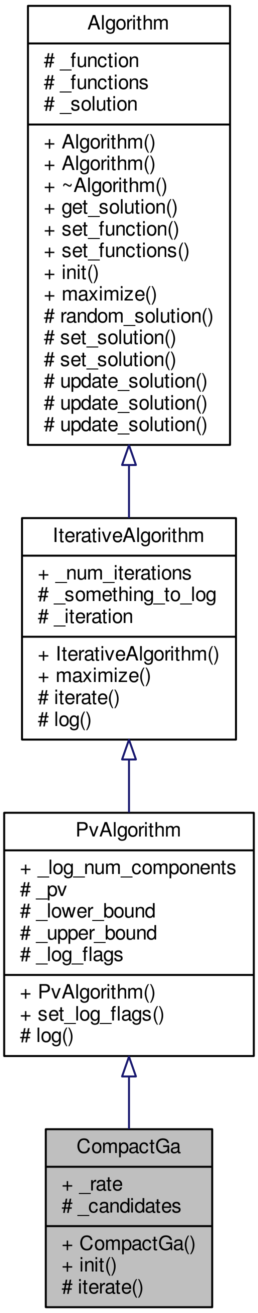
- lib/hnco/functions/decorators/function-controller.hh
- lib/hnco/functions/decorators/function-controller.cc

5.10 CompactGa Class Reference

Compact genetic algorithm.

```
#include <hnco/algorithms/pv/compact-ga.hh>
```

Inheritance diagram for CompactGa:



Public Member Functions

- CompactGa (int n)

    Constructor.

- void init ()

    Initialization.

Public Attributes

- double _rate = 1e-3

  Learning rate.

Protected Member Functions

- void iterate ()

  Single iteration.

Protected Attributes

- std::vector< bit_vector_t > _candidates

  Candidates.

Additional Inherited Members

5.10.1 Detailed Description

Compact genetic algorithm.

Definition at line 34 of file compact-ga.hh.

The documentation for this class was generated from the following files:

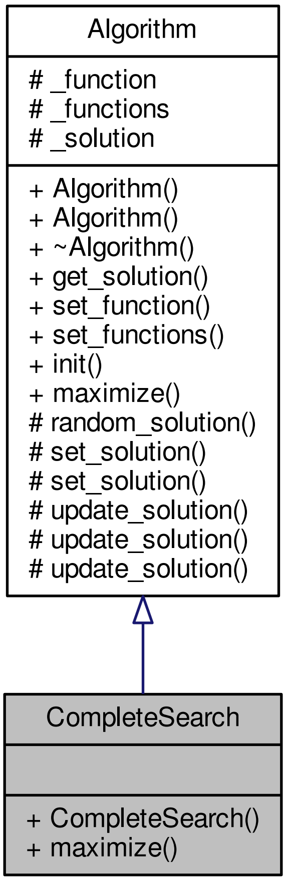
- lib/hnco/algorithms/pv/compact-ga.hh
- lib/hnco/algorithms/pv/compact-ga.cc

5.11 CompleteSearch Class Reference

Complete search.

```
#include <hnco/algorithms/complete-search.hh>
```

Inheritance diagram for CompleteSearch:



Public Member Functions

- CompleteSearch (int n)

  Constructor.
- void maximize ()

  Maximize.

Additional Inherited Members

5.11.1 Detailed Description

Complete search.

Definition at line 34 of file complete-search.hh.

The documentation for this class was generated from the following files:

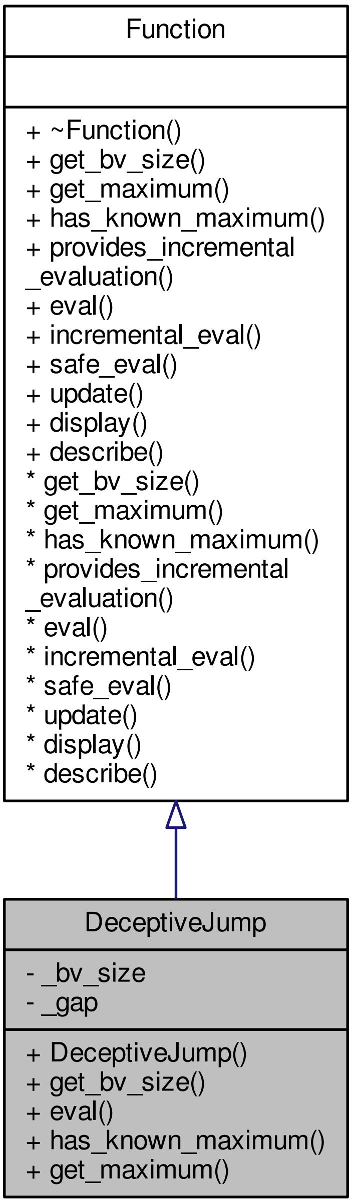
- lib/hnco/algorithms/complete-search.hh
- lib/hnco/algorithms/complete-search.cc

5.12 DeceptiveJump Class Reference

Deceptive jump.

`#include <hnco/functions/jump.hh>`

Inheritance diagram for DeceptiveJump:



Public Member Functions

- DeceptiveJump (int bv_size, int gap)

Constructor.

- size_t get_bv_size ()

  Get bit vector size.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.
- bool has_known_maximum ()

  Check for a known maximum.
- double get_maximum ()

  Get the global maximum.

Private Attributes

- size_t _bv_size

  Bit vector size.
- int _gap

  Gap.

5.12.1 Detailed Description

Deceptive jump.

This is a jump function with a deceptive gap as defined in "Analyzing evolutionary algorithms" by Thomas Jansen, where it is called Jump_k. Algorithms in the neighborhood of the maximizer (which is the all one bit vector) are taken away from it.

Definition at line 68 of file jump.hh.

5.12.2 Member Function Documentation

5.12.2.1 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true

Reimplemented from Function.

Definition at line 90 of file jump.hh.

The documentation for this class was generated from the following files:

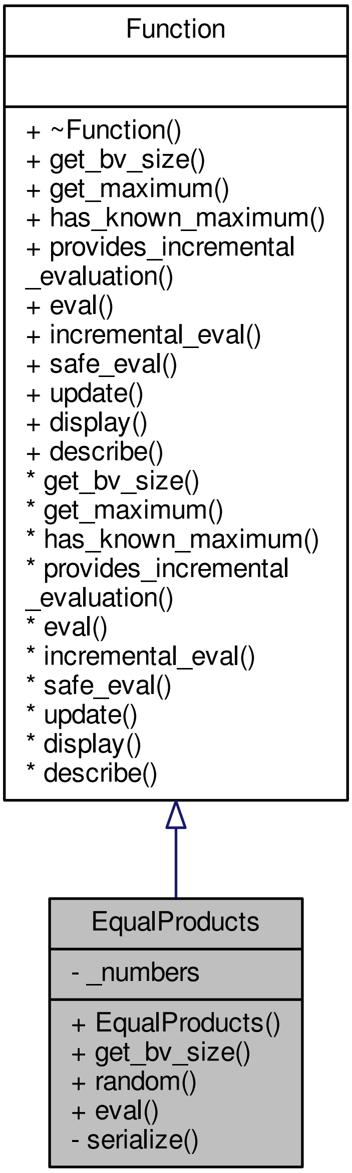
- lib/hnco/functions/jump.hh
- lib/hnco/functions/jump.cc

5.13 EqualProducts Class Reference

Equal products.

```
#include <hnco/functions/equal-products.hh>
```

Inheritance diagram for EqualProducts:



Public Member Functions

- EqualProducts ()

Constructor.

- size_t get_bv_size ()

  Get bit vector size.

- void random (int n, double upper_bound)

  Random instance.

- double eval (const bit_vector_t &)

  Evaluate a bit vector.

Private Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serialize.

Private Attributes

- std::vector< double > _numbers

  Numbers.

Friends

- class **boost::serialization::access**

5.13.1 Detailed Description

Equal products.

Partition a finite set of positive numbers into two subsets such that the product of numbers in the first subset is the closest to the product of numbers in the second subset. This is equivalent to the partition problem applied to the logarithms of the given numbers.

The function computes the negation of the distance between the product of numbers corresponding to ones in the bit vector and the product of those corresponding to zeros. The negation is a consequence of the fact that algorithms in HNCO maximize rather than minimize a function.

Source: technical report by Larranaga et al.

Definition at line 55 of file equal-products.hh.

5.13.2 Member Function Documentation

5.13.2.1 random()

```
void random (
            int n,
            double upper_bound )
```

Random instance.

Parameters

n	Size of bit vector
upper_bound	Upper bound of numbers

Definition at line 33 of file equal-products.cc.

The documentation for this class was generated from the following files:

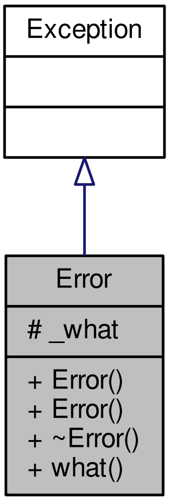
- lib/hnco/functions/equal-products.hh
- lib/hnco/functions/equal-products.cc

5.14 Error Class Reference

Error.

```
#include <hnco/exception.hh>
```

Inheritance diagram for Error:



Public Member Functions

- Error ()

  Constructor.
- Error (const std::string &s)

  Constructor.
- virtual ∼Error ()

  Destructor.
- virtual const char ∗ what () const

  Get message.

Protected Attributes

- std::string _what

  Message.

5.14.1 Detailed Description

Error.

Definition at line 83 of file exception.hh.

The documentation for this class was generated from the following file:

- lib/hnco/exception.hh

5.15 ProgressTracker::Event Struct Reference

Event.

```
#include <hnco/functions/decorators/function-controller.hh>
```

Public Attributes

- int time

  Time.
- double value

  Value.

5.15.1 Detailed Description

Event.

Definition at line 218 of file function-controller.hh.

The documentation for this struct was generated from the following file:

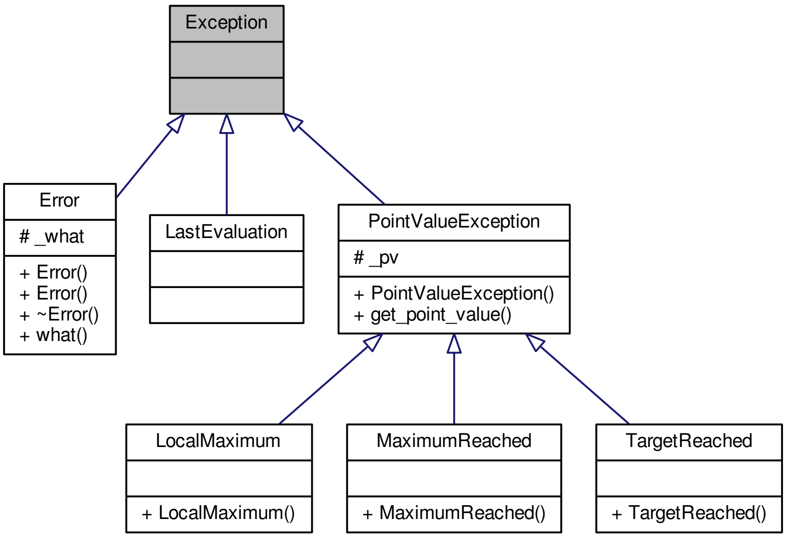
- lib/hnco/functions/decorators/function-controller.hh

5.16 Exception Class Reference

Basic exception.

```
#include <hnco/exception.hh>
```

Inheritance diagram for Exception:



5.16.1 Detailed Description

Basic exception.

Definition at line 35 of file exception.hh.

The documentation for this class was generated from the following file:

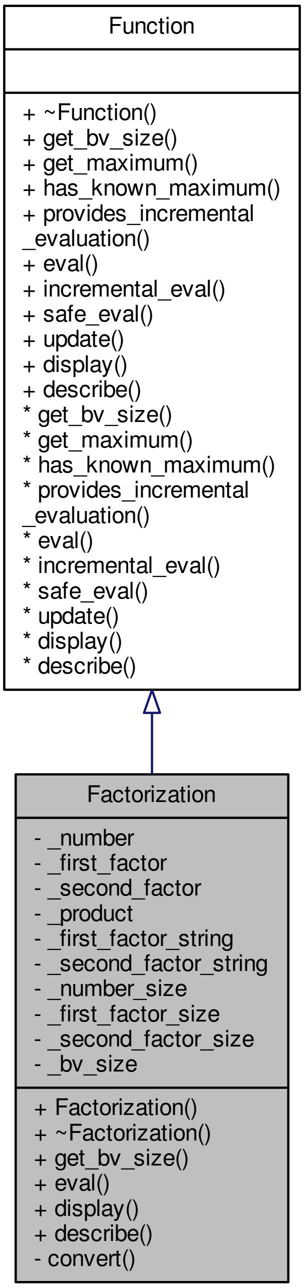
- lib/hnco/exception.hh

5.17 Factorization Class Reference

Factorization.

```
#include <hnco/functions/factorization.hh>
```

Inheritance diagram for Factorization:



Public Member Functions

- Factorization (std::string path)

  Constructor.
- ∼Factorization ()

  Destructor.
- size_t get_bv_size ()

Get bit vector size.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.
- void display (std::ostream &stream)

  Display.
- void describe (const bit_vector_t &x, std::ostream &stream)

  Describe a bit vector.

Private Member Functions

- void convert (const bit_vector_t &x)

  Convert a bit vector into two numbers.

Private Attributes

- mpz_t _number

  Number to factorize.
- mpz_t _first_factor

  First factor.
- mpz_t _second_factor

  Second factor.
- mpz_t _product

  Product.
- std::string _first_factor_string

  First factor in binary form.
- std::string _second_factor_string

  Secon factor in binary form.
- size_t _number_size

  Number size in bits.
- size_t _first_factor_size

  First factor size in bits.
- size_t _second_factor_size

  Second factor size in bits.
- size_t _bv_size

  Bit vector size.

5.17.1 Detailed Description

Factorization.

Definition at line 18 of file factorization.hh.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 Factorization()

```
Factorization (
            std::string path )
```

Constructor.

Parameters

path	Path to a file containing a number to factorize

Warning

The file is a text file which contains exactly one natural number written in base 10 without any space.

Definition at line 16 of file factorization.cc.

The documentation for this class was generated from the following files:

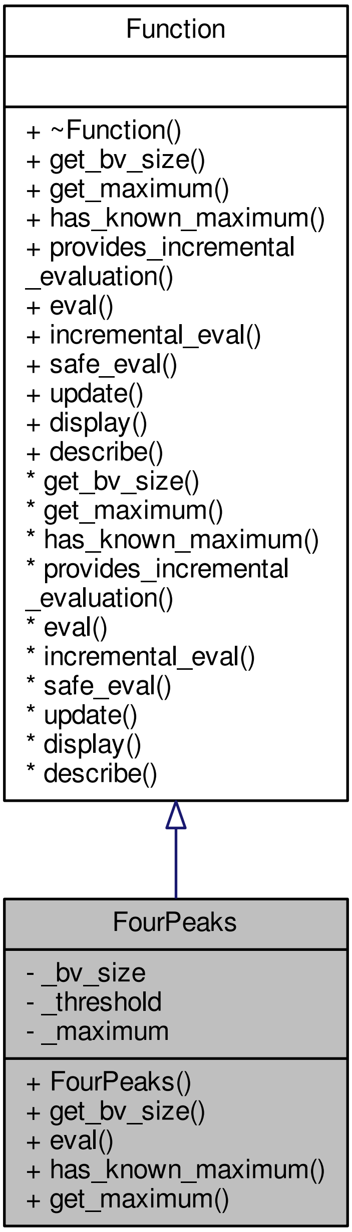
- lib/hnco/functions/factorization.hh
- lib/hnco/functions/factorization.cc

5.18 FourPeaks Class Reference

Four Peaks.

```
#include <hnco/functions/four-peaks.hh>
```

Inheritance diagram for FourPeaks:



Public Member Functions

- FourPeaks (int bv_size, int threshold)

    Constructor.
- size_t get_bv_size ()

    Get bit vector size.
- double eval (const bit_vector_t &)

Evaluate a bit vector.
- bool has_known_maximum ()

  Check for a known maximum.
- double get_maximum ()

  Get the global maximum.

Private Attributes

- size_t _bv_size

  Bit vector size.
- int _threshold

  Threshold.
- int _maximum

  Maximum.

5.18.1 Detailed Description

Four Peaks.

It is defined by

f(x) = max{head(x, 1) + tail(x, 0)} + R(x)

where:

- head(x, 1) is the length of the longest prefix of x made of ones;
- tail(x, 0) is the length of the longest suffix of x made of zeros;
- R(x) is the reward;
- R(x) = n if (head(x, 1) $>$ t and tail(x, 0) $>$ t);
- R(x) = 0 otherwise;
- the threshold t is a parameter of the function.

This function has four maxima, of which exactly two are global ones.

For example, if n = 6 and t = 1:

- f(111111) = 6 (local maximum)
- f(111110) = 5
- f(111100) = 10 (global maximum)

De Bonet, J. S., Isbell, C. L., & Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. Advances in neural information processing systems, 424-430.

Definition at line 57 of file four-peaks.hh.

5.18.2 Member Function Documentation

5.18.2.1 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true

Reimplemented from Function.

Definition at line 84 of file four-peaks.hh.

The documentation for this class was generated from the following files:

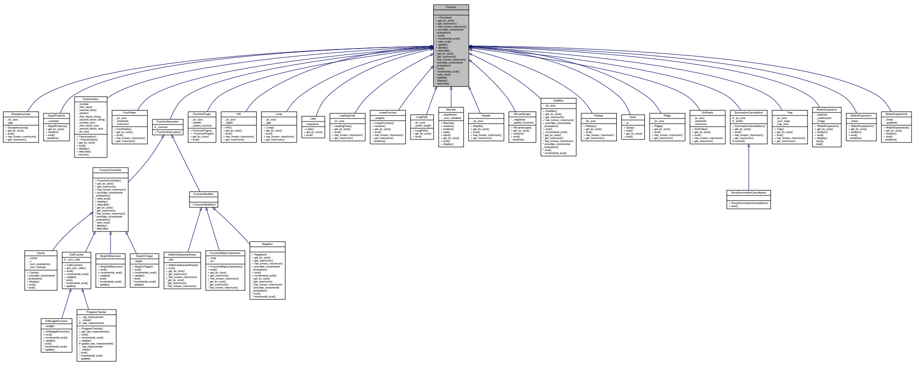
- lib/hnco/functions/four-peaks.hh
- lib/hnco/functions/four-peaks.cc

5.19 Function Class Reference

Function.

```
#include <hnco/functions/function.hh>
```

Inheritance diagram for Function:



Public Member Functions

- virtual ∼Function ()

  Destructor.

Information about the function

- virtual size_t get_bv_size ()=0

  Get bit vector size.
- virtual double get_maximum ()

  Get the global maximum.
- virtual bool has_known_maximum ()

  Check for a known maximum.
- virtual bool provides_incremental_evaluation ()

  Check whether the function provides incremental evaluation.

Evaluation

- virtual double eval (const bit_vector_t &)=0

  Evaluate a bit vector.
- virtual double incremental_eval (const bit_vector_t &x, double value, const hnco::sparse_bit_vector_t &flipped_bits)

  Incremental evaluation.
- virtual double safe_eval (const bit_vector_t &x)

  Safely evaluate a bit vector.
- virtual void update (const bit_vector_t &x, double value)

  Update after a safe evaluation.

Display

- virtual void display (std::ostream &stream)

  Display.
- virtual void describe (const bit_vector_t &x, std::ostream &stream)

  Describe a bit vector.

5.19.1 Detailed Description

Function.

Definition at line 35 of file function.hh.

5.19.2 Member Function Documentation

5.19.2.1 get_maximum()

```
virtual double get_maximum ( )  [inline], [virtual]
```

Get the global maximum.

Exceptions

Error	

Reimplemented in Plateau, Ridge, AdditiveGaussianNoise, Hiff, SixPeaks, FunctionMapComposition, Needle, LeadingOnes, DeceptiveJump, FourPeaks, SummationCancellation, LinearFunction, Trap, Negation, FunctionController, Jump, and OneMax.

Definition at line 51 of file function.hh.

5.19.2.2 incremental_eval()

```
virtual double incremental_eval (
             const bit_vector_t & x,
             double value,
             const hnco::sparse_bit_vector_t & flipped_bits )  [inline], [virtual]
```

Incremental evaluation.

Exceptions

Error	

Reimplemented in OnBudgetFunction, ProgressTracker, CallCounter, StopOnTarget, StopOnMaximum, Negation, and OneMax.

Definition at line 75 of file function.hh.

5.19.2.3 provides_incremental_evaluation()

```
virtual bool provides_incremental_evaluation ( )  [inline], [virtual]
```

Check whether the function provides incremental evaluation.

Returns

false

Reimplemented in Cache, Negation, FunctionController, and OneMax.

Definition at line 59 of file function.hh.

5.19.2.4 safe_eval()

```
virtual double safe_eval (
             const bit_vector_t & x )  [inline], [virtual]
```

Safely evaluate a bit vector.

Must be thread-safe, that is must avoid throwing exceptions and updating global states (e.g. maximum) in function decorators.

Reimplemented in FunctionController.

Definition at line 85 of file function.hh.

The documentation for this class was generated from the following file:

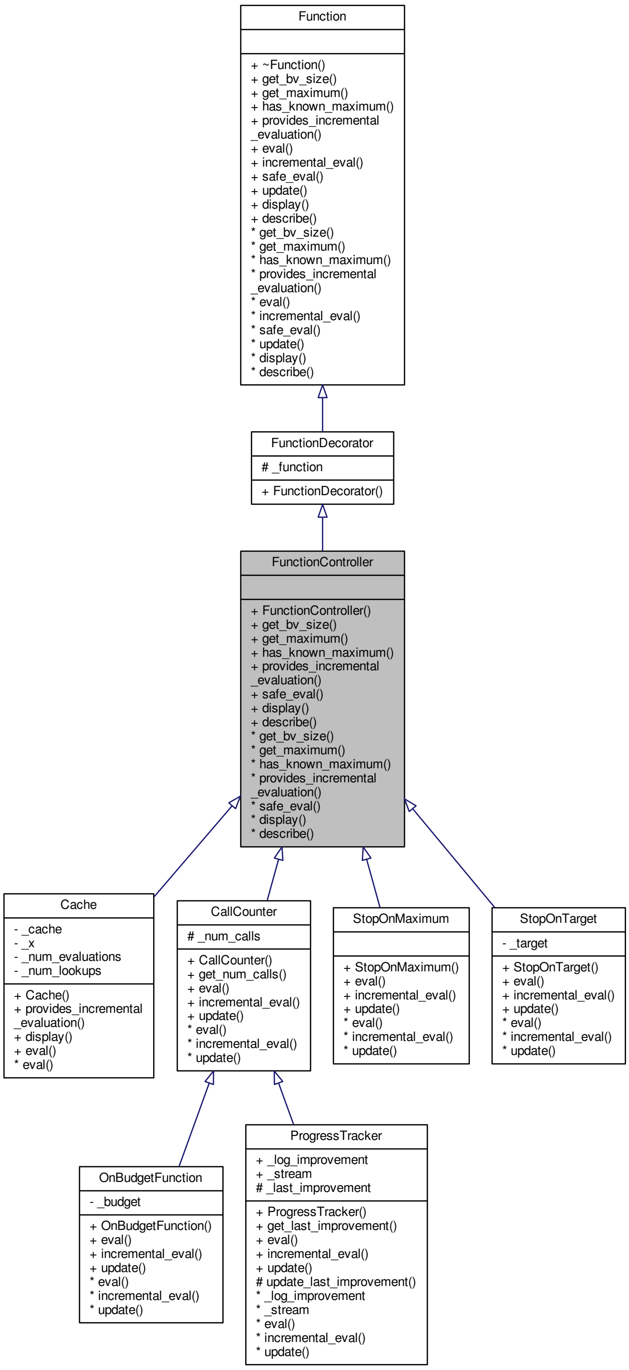
- lib/hnco/functions/function.hh

5.20 FunctionController Class Reference

Function controller.

```
#include <hnco/functions/decorators/function-controller.hh>
```

Inheritance diagram for FunctionController:



Public Member Functions

- FunctionController (Function ∗function)

  Constructor.

Information about the function

- size_t get_bv_size ()
  Get bit vector size.
- double get_maximum ()
  Get the global maximum.
- bool has_known_maximum ()
  Check for a known maximum.
- bool provides_incremental_evaluation ()
  Check whether the function provides incremental evaluation.

Evaluation

- double safe_eval (const bit_vector_t &x)
  Safely evaluate a bit vector.

Display

- void display (std::ostream &stream)
  Display.
- void describe (const bit_vector_t &x, std::ostream &stream)
  Describe a bit vector.

Additional Inherited Members

5.20.1 Detailed Description

Function controller.

Definition at line 38 of file function-controller.hh.

5.20.2 Member Function Documentation

5.20.2.1 provides_incremental_evaluation()

```
bool provides_incremental_evaluation ( )  [inline], [virtual]
```

Check whether the function provides incremental evaluation.

Returns

true if the decorated function does

Reimplemented from Function.

Reimplemented in Cache.

Definition at line 63 of file function-controller.hh.

The documentation for this class was generated from the following file:

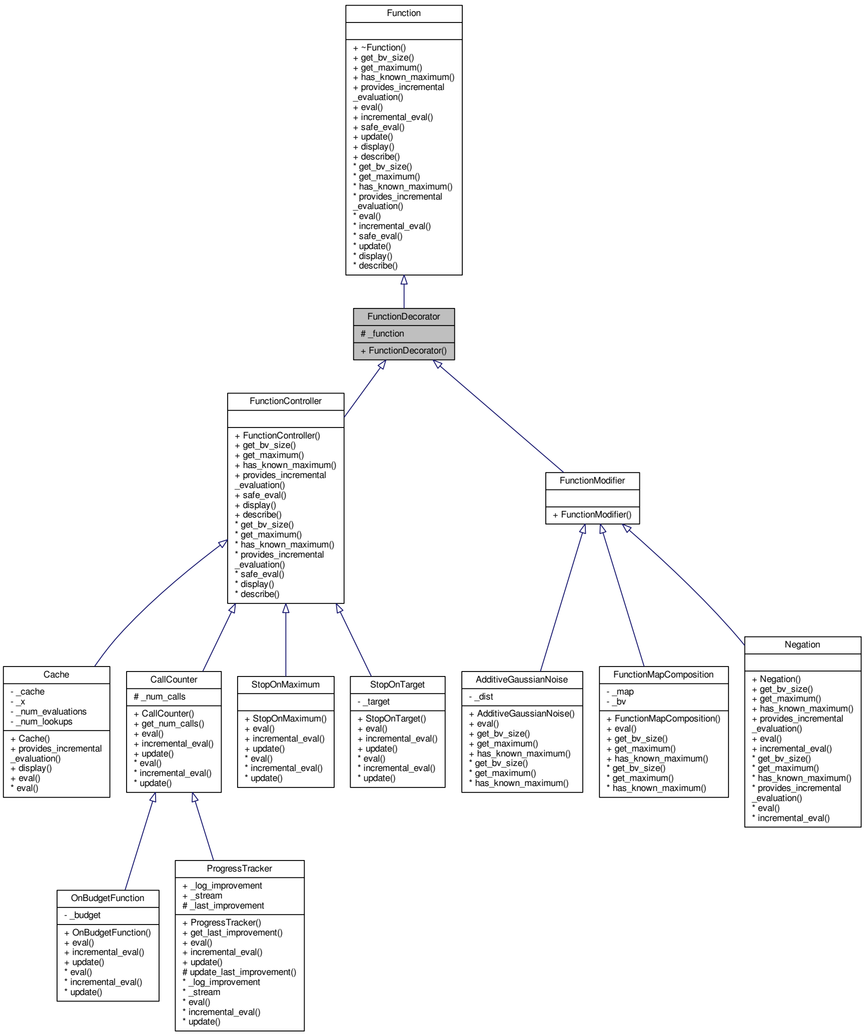
- lib/hnco/functions/decorators/function-controller.hh

5.21 FunctionDecorator Class Reference

Function decorator.

```
#include <hnco/functions/decorators/function-decorator.hh>
```

Inheritance diagram for FunctionDecorator:



Public Member Functions

- FunctionDecorator (Function ∗function)

  Constructor.

Protected Attributes

- Function ∗ _function

  Decorated function.

5.21.1 Detailed Description

Function decorator.

Definition at line 37 of file function-decorator.hh.

The documentation for this class was generated from the following file:

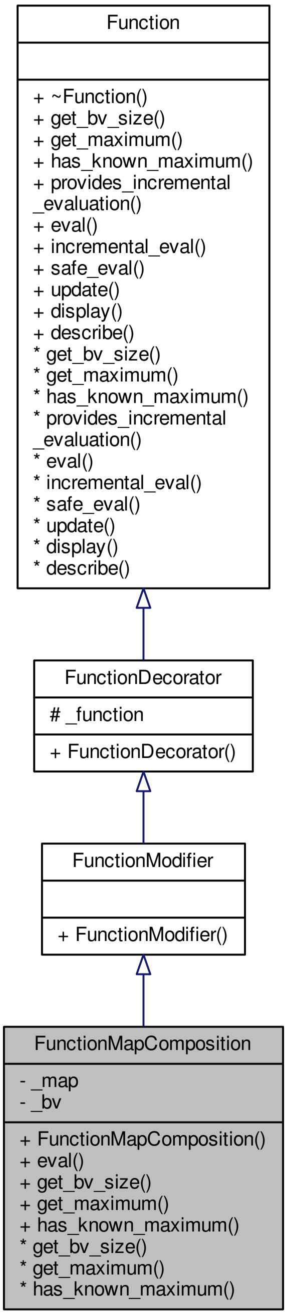
- lib/hnco/functions/decorators/function-decorator.hh

5.22 FunctionMapComposition Class Reference

Composition of a function and a map.

```
#include <hnco/functions/decorators/function-modifier.hh>
```

Inheritance diagram for FunctionMapComposition:



Public Member Functions

- FunctionMapComposition (Function ∗function, Map ∗map)

    Constructor.

- double eval (const bit_vector_t &)

    Evaluate a bit vector.

Information about the function

- size_t get_bv_size ()
  Get bit vector size.
- double get_maximum ()
  Get the global maximum.
- bool has_known_maximum ()
  Check for a known maximum.

Private Attributes

- Map ∗ _map
  Map.
- bit_vector_t _bv
  _bv

Additional Inherited Members

5.22.1 Detailed Description

Composition of a function and a map.

Definition at line 106 of file function-modifier.hh.

5.22.2 Constructor & Destructor Documentation

5.22.2.1 FunctionMapComposition()

```
FunctionMapComposition (
            Function ∗ function,
            Map ∗ map )  [inline]
```

Constructor.

Precondition

map->get_output_size() == function->get_bv_size()

Exceptions

Error	

Definition at line 121 of file function-modifier.hh.

5.22.3 Member Function Documentation

5.22.3.1 get_maximum()

```
double get_maximum ( )  [inline], [virtual]
```

Get the global maximum.

Exceptions

Error	

Reimplemented from Function.

Definition at line 141 of file function-modifier.hh.

5.22.3.2 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true if the function has a known maximum and the map is bijective.

Reimplemented from Function.

Definition at line 151 of file function-modifier.hh.

The documentation for this class was generated from the following files:

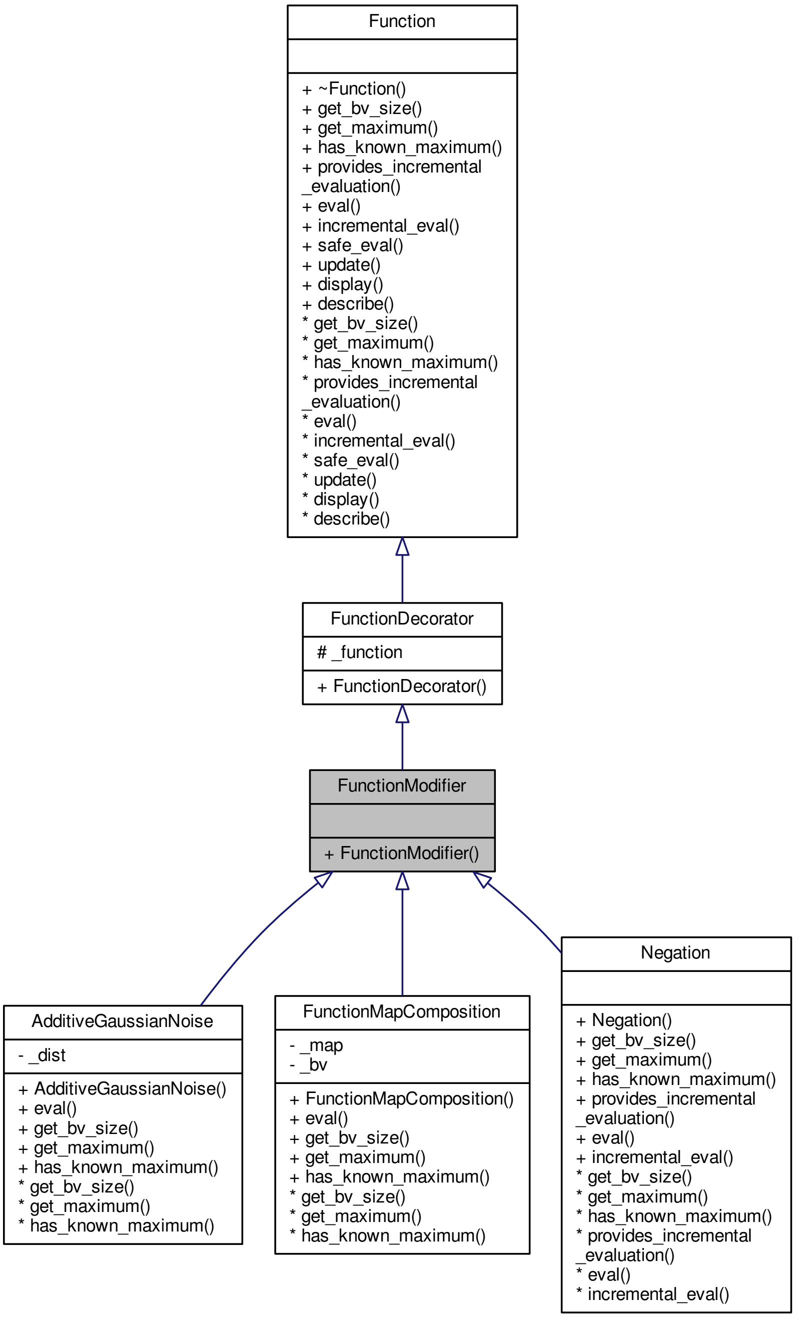
- lib/hnco/functions/decorators/function-modifier.hh
- lib/hnco/functions/decorators/function-modifier.cc

5.23 FunctionModifier Class Reference

Function modifier.

`#include <hnco/functions/decorators/function-modifier.hh>`

Inheritance diagram for FunctionModifier:



Public Member Functions

- FunctionModifier (Function ∗function)

  Constructor.

Additional Inherited Members

5.23.1 Detailed Description

Function modifier.

Definition at line 37 of file function-modifier.hh.

The documentation for this class was generated from the following file:

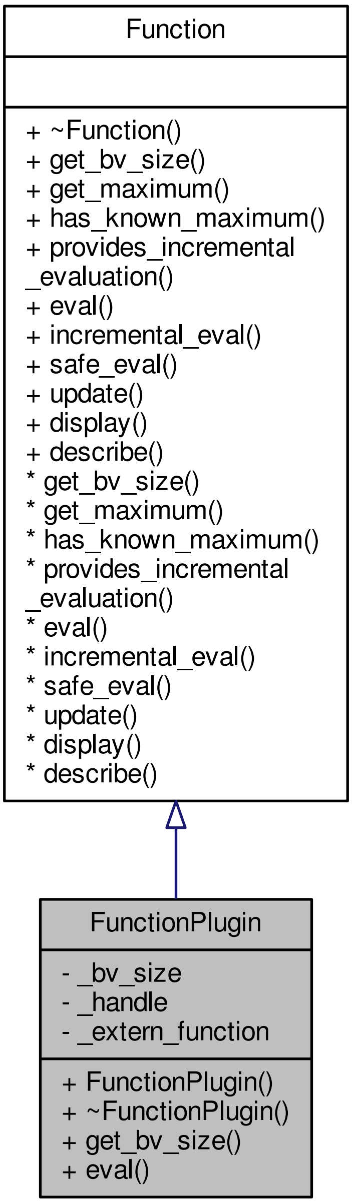
- lib/hnco/functions/decorators/function-modifier.hh

5.24 FunctionPlugin Class Reference

Function plugin.

```
#include <hnco/functions/plugin.hh>
```

Inheritance diagram for FunctionPlugin:



Public Member Functions

- FunctionPlugin (int bv_size, std::string path, std::string name)

  Constructor.
- ∼FunctionPlugin ()

  Destructor.
- size_t get_bv_size ()

  Get bit vector size.

- double eval (const bit_vector_t &)

  Evaluate a bit vector.

Private Types

- typedef double(∗ extern_function_t) (const char[ ], size_t)

  Type of an extern function.

Private Attributes

- size_t _bv_size

  Bit vector size.
- void ∗ _handle

  Handle returned by dlopen.
- extern_function_t _extern_function

  Extern function.

5.24.1 Detailed Description

Function plugin.

Definition at line 34 of file plugin.hh.

5.24.2 Constructor & Destructor Documentation

5.24.2.1 FunctionPlugin()

```
FunctionPlugin (
            int bv_size,
            std::string path,
            std::string name )
```

Constructor.

Parameters

bv_size	Size of bit vectors
path	Path to a shared library
name	Name of a function of the shared library

Definition at line 35 of file plugin.cc.

The documentation for this class was generated from the following files:

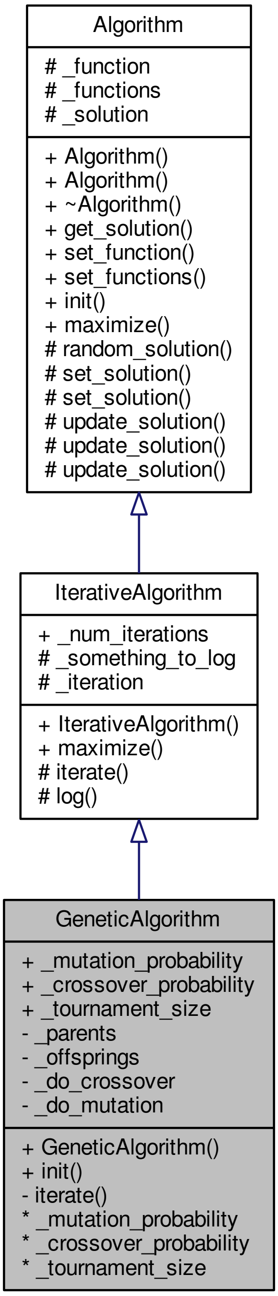
- lib/hnco/functions/plugin.hh
- lib/hnco/functions/plugin.cc

5.25 GeneticAlgorithm Class Reference

Genetic algorithm.

`#include <hnco/algorithms/ea/genetic-algorithm.hh>`

Inheritance diagram for GeneticAlgorithm:



Public Member Functions

- GeneticAlgorithm (int n, int mu)

  Constructor.
- void init ()

  Initialization.

Public Attributes

Parameters

- double _mutation_probability

  Mutation probability.
- double _crossover_probability = 0.5

  Crossover probability.
- int _tournament_size = 10

  Tournament size.

Private Member Functions

- void iterate ()

  Single iteration.

Private Attributes

- TournamentSelection _parents

  Parents.
- TournamentSelection _offsprings

  Offsprings.
- std::bernoulli_distribution _do_crossover

  Do crossover.
- std::bernoulli_distribution _do_mutation

  Do mutation.

Additional Inherited Members

5.25.1 Detailed Description

Genetic algorithm.

- Tournament selection for reproduction
- Uniform crossover
- Mutation
- (mu, mu) selection (offspring population replaces parent population)

Definition at line 42 of file genetic-algorithm.hh.

The documentation for this class was generated from the following files:

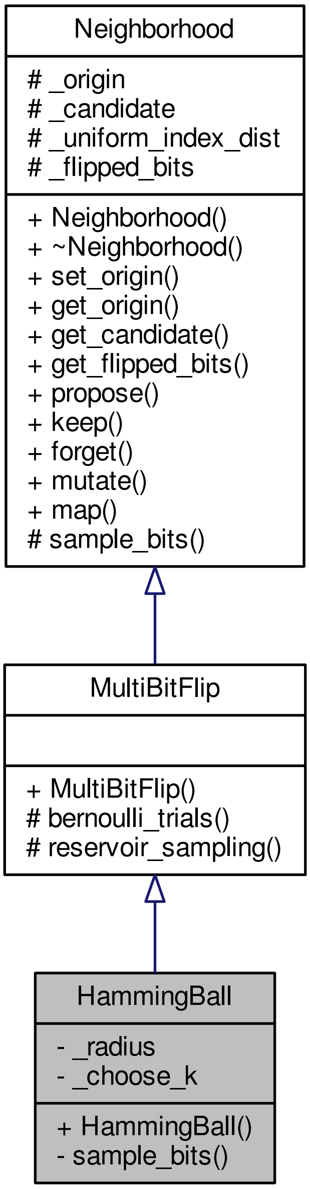
- lib/hnco/algorithms/ea/genetic-algorithm.hh
- lib/hnco/algorithms/ea/genetic-algorithm.cc

5.26 HammingBall Class Reference

Hamming ball.

`#include <hnco/neighborhoods/neighborhood.hh>`

Inheritance diagram for HammingBall:



Public Member Functions

- HammingBall (int n, int r)

  Constructor.

Private Member Functions

- void sample_bits ()

  Sample bits.

Private Attributes

- int _radius

  Radius of the ball.
- std::uniform_int_distribution< int > _choose_k

  Choose the distance to the center.

Additional Inherited Members

5.26.1 Detailed Description

Hamming ball.

Choose k uniformly on [1..r], where r is the radius of the ball, choose k bits uniformly among n and flip them.

Definition at line 295 of file neighborhood.hh.

5.26.2 Constructor & Destructor Documentation

5.26.2.1 HammingBall()

```
HammingBall (
              int n,
              int r )   [inline]
```

Constructor.

Parameters

n	Size of bit vectors
r	Radius of the ball

Definition at line 314 of file neighborhood.hh.

The documentation for this class was generated from the following files:

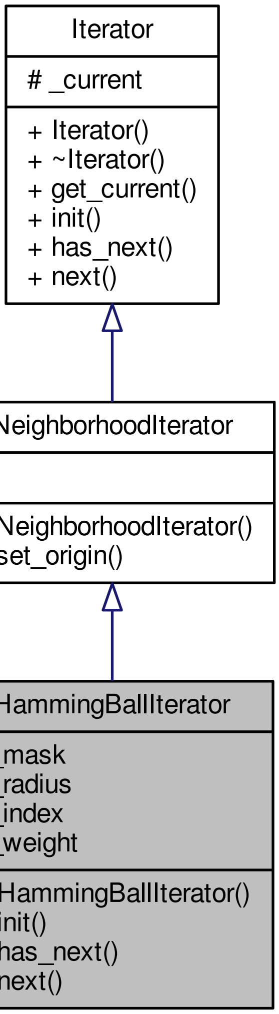
- lib/hnco/neighborhoods/neighborhood.hh
- lib/hnco/neighborhoods/neighborhood.cc

5.27 HammingBallIterator Class Reference

Hamming ball neighborhood iterator.

```
#include <hnco/neighborhoods/neighborhood-iterator.hh>
```

Inheritance diagram for HammingBallIterator:



Public Member Functions

- HammingBallIterator (int n, int r)

  Constructor.
- void init ()

  Initialization.
- bool has_next ()

  Has next bit vector.
- void next ()

  Next bit vector.

Private Attributes

- bit_vector_t _mask

  Mutation mask.
- int _radius

  Radius of the ball.
- int _index

  Index of the next bit to shift to the right.
- int _weight

  Partial Hamming weight.

Additional Inherited Members

5.27.1 Detailed Description

Hamming ball neighborhood iterator.

Definition at line 86 of file neighborhood-iterator.hh.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 HammingBallIterator()

```
HammingBallIterator (
              int n,
              int r )  [inline]
```

Constructor.

Parameters

n	Size of bit vectors
r	Radius of Hamming Ball

Definition at line 108 of file neighborhood-iterator.hh.

The documentation for this class was generated from the following files:

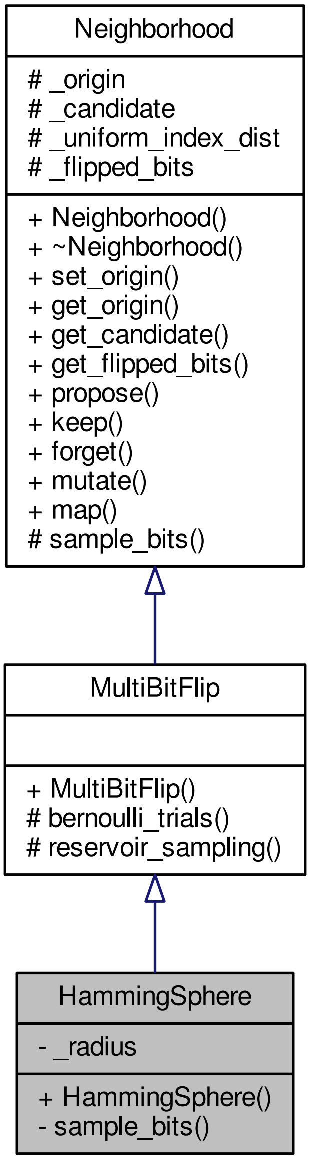
- lib/hnco/neighborhoods/neighborhood-iterator.hh
- lib/hnco/neighborhoods/neighborhood-iterator.cc

5.28 HammingSphere Class Reference

Hamming sphere.

```
#include <hnco/neighborhoods/neighborhood.hh>
```

Inheritance diagram for HammingSphere:



Public Member Functions

- HammingSphere (int n, int r)

  Constructor.

Private Member Functions

- void sample_bits ()

  Sample bits.

Private Attributes

- int _radius

  Radius of the sphere.

Additional Inherited Members

5.28.1 Detailed Description

Hamming sphere.

Uniformly choose r bits among n and flip them, where r is the radius of the sphere.

Definition at line 332 of file neighborhood.hh.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 HammingSphere()

```
HammingSphere (
             int n,
             int r )  [inline]
```

Constructor.

Parameters

n	Size of bit vectors
r	Radius of the sphere

Definition at line 348 of file neighborhood.hh.

The documentation for this class was generated from the following files:

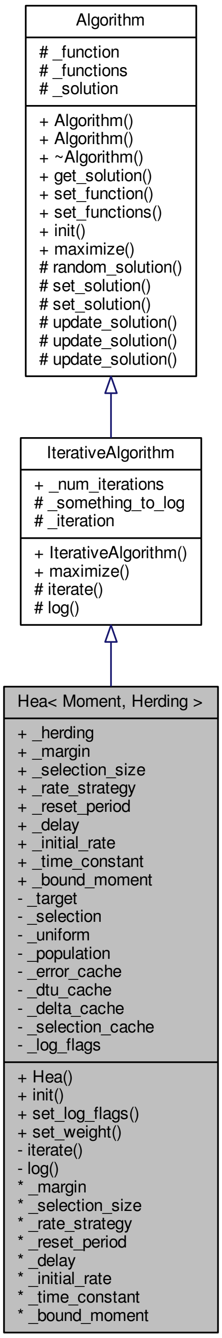
- lib/hnco/neighborhoods/neighborhood.hh
- lib/hnco/neighborhoods/neighborhood.cc

5.29 Hea< Moment, Herding > Class Template Reference

Herding evolutionary algorithm.

```
#include <hnco/algorithms/hea/hea.hh>
```

Inheritance diagram for Hea< Moment, Herding >:



Public Types

- enum {
  LOG_ERROR, LOG_DTU, LOG_DELTA, LOG_SELECTION,
  LAST_LOG }
- enum { RATE_CONSTANT, RATE_EXPONENTIAL, RATE_INVERSE }
- typedef std::bitset< LAST_LOG > log_flags_t

  Type for log flags.

Public Member Functions

- Hea (int n, int population_size)

  Constructor.
- void init ()

  Initialization.
- void set_log_flags (const log_flags_t &lf)

  Set log flags.
- void set_weight (double weight)

  Set weight.

Public Attributes

- Herding _herding

  Herding.

Parameters

- double _margin

  Moment margin.
- int _selection_size = 1

  Selection size (number of selected individuals in the population)
- int _rate_strategy = RATE_CONSTANT

  Rate strategy.
- int _reset_period = 0

  Reset period (<= 0 means no reset)
- int _delay = 10000

  Delay.
- double _initial_rate = 1e-4

  Initial value of the learning rate.
- double _time_constant = 1000

  Time constant.
- bool _bound_moment = false

  Bound moment after update.

Private Member Functions

- void iterate ()

  Single iteration.
- void log ()

  Log.

Private Attributes

- Moment _target

  Moment.
- Moment _selection

  Moment of selected individuals.
- Moment _uniform

  Uniform moment.
- algorithm::Population _population

  Population.
- double _error_cache

  Error cache.
- double _dtu_cache

  Distance to uniform cache.
- double _delta_cache

  Delta cache.
- double _selection_cache

  Selection distance cache.
- log_flags_t _log_flags

  Log flags.

Additional Inherited Members

5.29.1 Detailed Description

template<class Moment, class Herding>
class hnco::algorithm::hea::Hea< Moment, Herding >

Herding evolutionary algorithm.

Definition at line 39 of file hea.hh.

5.29.2 Member Enumeration Documentation

5.29.2.1 anonymous enum

```
anonymous enum
```

Enumerator

LOG_ERROR	Log error.
LOG_DTU	Log distance to uniform.
LOG_DELTA	Log delta (moment increment)
LOG_SELECTION	Log the distance between the target and the selection moment.

Definition at line 44 of file hea.hh.

5.29.2.2 anonymous enum

```
anonymous enum
```

Enumerator

RATE_CONSTANT	Constant rate.
RATE_EXPONENTIAL	Exponentiel decay.
RATE_INVERSE	Inverse decay.

Definition at line 192 of file hea.hh.

The documentation for this class was generated from the following file:

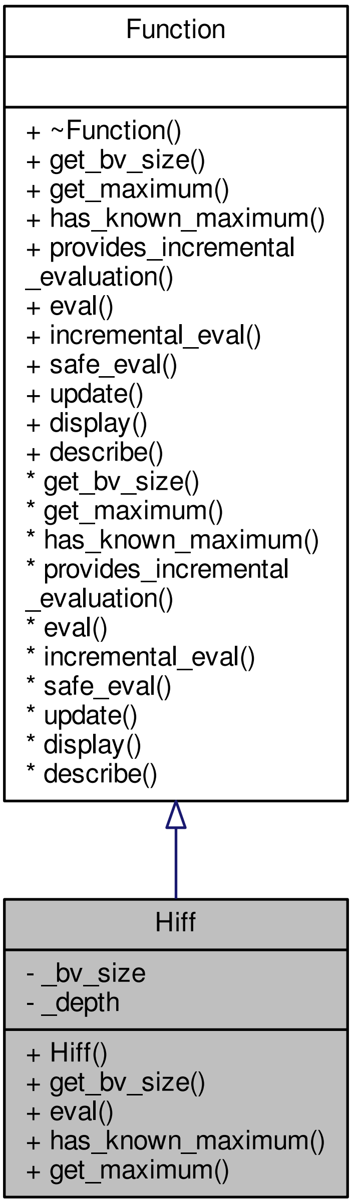
- lib/hnco/algorithms/hea/hea.hh

5.30 Hiff Class Reference

Hierarchical if and only if.

```
#include <hnco/functions/theory.hh>
```

Inheritance diagram for Hiff:



Public Member Functions

- Hiff (int bv_size)

  Constructor.
- size_t get_bv_size ()

  Get bit vector size.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.

- bool has_known_maximum ()

    Check for a known maximum.
- double get_maximum ()

    Get the global maximum.

Private Attributes

- size_t _bv_size

    Bit vector size.
- size_t _depth

    Tree depth.

5.30.1 Detailed Description

Hierarchical if and only if.

Definition at line 136 of file theory.hh.

5.30.2 Member Function Documentation

5.30.2.1 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true

Reimplemented from Function.

Definition at line 158 of file theory.hh.

The documentation for this class was generated from the following files:

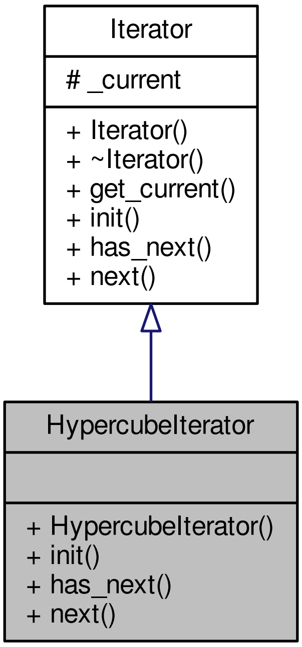
- lib/hnco/functions/theory.hh
- lib/hnco/functions/theory.cc

5.31 HypercubeIterator Class Reference

Hypercube iterator.

`#include <hnco/iterator.hh>`

Inheritance diagram for HypercubeIterator:



Public Member Functions

- HypercubeIterator (int n)

    Constructor.
- void init ()

    Initialization.
- bool has_next ()

    Has next bit vector.
- void next ()

    Next bit vector.

Additional Inherited Members

5.31.1 Detailed Description

Hypercube iterator.

Definition at line 71 of file iterator.hh.

5.31.2 Member Function Documentation

5.31.2.1 next()

```
void next ( )  [virtual]
```

Next bit vector.

Implemented as a simple binary adder.

Implements Iterator.

Definition at line 28 of file iterator.cc.

The documentation for this class was generated from the following files:

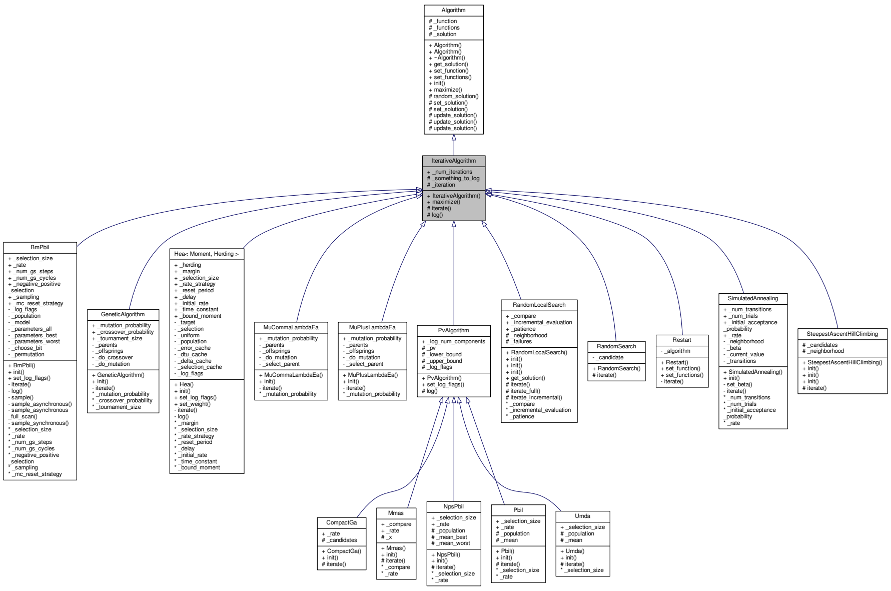
- lib/hnco/iterator.hh
- lib/hnco/iterator.cc

5.32 IterativeAlgorithm Class Reference

Iterative search.

```
#include <hnco/algorithms/algorithm.hh>
```

Inheritance diagram for IterativeAlgorithm:



Public Member Functions

- IterativeAlgorithm (int n)

    Constructor.
- void maximize ()

    Maximize.

Public Attributes

- int _num_iterations = 0

    Number of iterations.

Protected Member Functions

- virtual void iterate ()=0

    Single iteration.
- virtual void log ()

    Log.

Protected Attributes

- bool _something_to_log

    Something to log.
- int _iteration

    Current iteration.

5.32.1 Detailed Description

Iterative search.

Definition at line 110 of file algorithm.hh.

5.32.2 Constructor & Destructor Documentation

5.32.2.1 IterativeAlgorithm()

```
IterativeAlgorithm (
              int n )   [inline]
```

Constructor.

Parameters

n	Size of bit vectors

Definition at line 132 of file algorithm.hh.

5.32.3 Member Function Documentation

5.32.3.1 maximize()

```
void maximize ( )  [virtual]
```

Maximize.

Inside the loop:

- call iterate()
- call log()

Warning

> If an exception such as LocalMaximum is thrown by iterate(), log() will not be called. However, hnco reports the maximum at the end of the search.

Implements Algorithm.

Definition at line 77 of file algorithm.cc.

5.32.4 Member Data Documentation

5.32.4.1 _num_iterations

```
int _num_iterations = 0
```

Number of iterations.

_num_iterations <= 0 means indefinite

Definition at line 149 of file algorithm.hh.

The documentation for this class was generated from the following files:

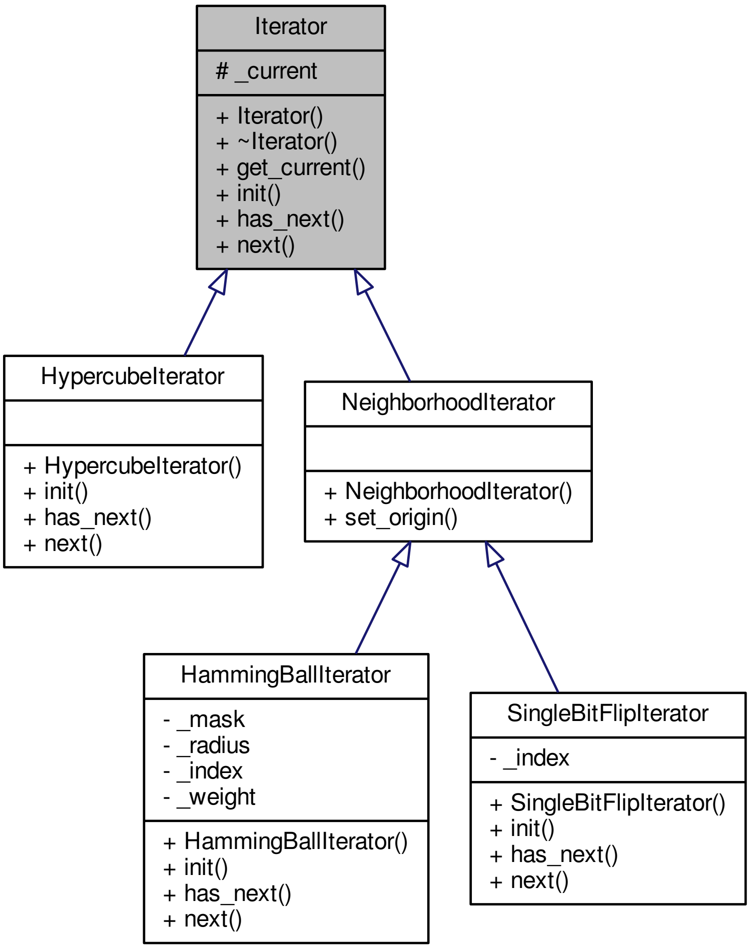
- lib/hnco/algorithms/algorithm.hh
- lib/hnco/algorithms/algorithm.cc

5.33 Iterator Class Reference

Iterator over bit vectors.

```
#include <hnco/iterator.hh>
```

Inheritance diagram for Iterator:



Public Member Functions

- Iterator (int n)

  Constructor.
- virtual ∼Iterator ()

  Destructor.
- virtual const bit_vector_t & get_current ()

  Current bit vector.
- virtual void init ()=0

  Initialization.

- virtual bool has_next ()=0

  Has next bit vector.
- virtual void next ()=0

  Next bit vector.

Protected Attributes

- bit_vector_t _current

  Current bit vector.

5.33.1 Detailed Description

Iterator over bit vectors.

Definition at line 40 of file iterator.hh.

The documentation for this class was generated from the following file:

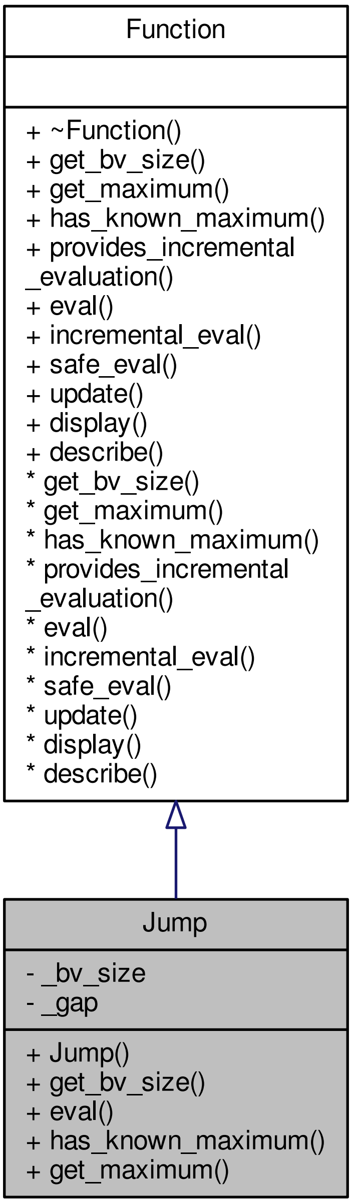
- lib/hnco/iterator.hh

5.34 Jump Class Reference

Jump.

```
#include <hnco/functions/jump.hh>
```

Inheritance diagram for Jump:



Public Member Functions

- Jump (int bv_size, int gap)

  Constructor.
- size_t get_bv_size ()

  Get bit vector size.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.

- bool has_known_maximum ()

  Check for a known maximum.
- double get_maximum ()

  Get the global maximum.

Private Attributes

- size_t _bv_size

  Bit vector size.
- int _gap

  Gap.

5.34.1 Detailed Description

Jump.

Definition at line 30 of file jump.hh.

5.34.2 Member Function Documentation

5.34.2.1 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true

Reimplemented from Function.

Definition at line 52 of file jump.hh.

The documentation for this class was generated from the following files:

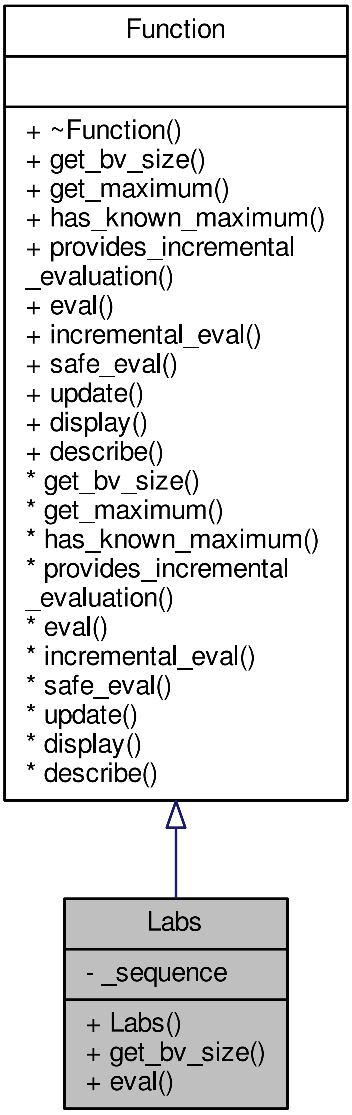
- lib/hnco/functions/jump.hh
- lib/hnco/functions/jump.cc

5.35 Labs Class Reference

Low autocorrelation binary sequences.

```
#include <hnco/functions/labs.hh>
```

Inheritance diagram for Labs:



Public Member Functions

- Labs (int n)

    Constructor.

- size_t get_bv_size ()

  Get bit vector size.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.

Private Attributes

- std::vector< int > _sequence

  Binary sequence written using 1 and -1.

5.35.1 Detailed Description

Low autocorrelation binary sequences.

Definition at line 32 of file labs.hh.

The documentation for this class was generated from the following files:

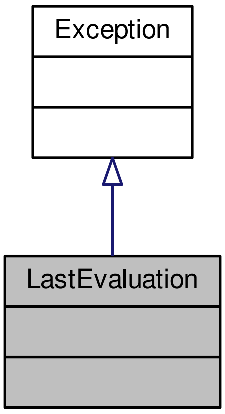
- lib/hnco/functions/labs.hh
- lib/hnco/functions/labs.cc

5.36 LastEvaluation Class Reference

Last evaluation.

```
#include <hnco/exception.hh>
```

Inheritance diagram for LastEvaluation:



5.36.1 Detailed Description

Last evaluation.

Definition at line 79 of file exception.hh.

The documentation for this class was generated from the following file:

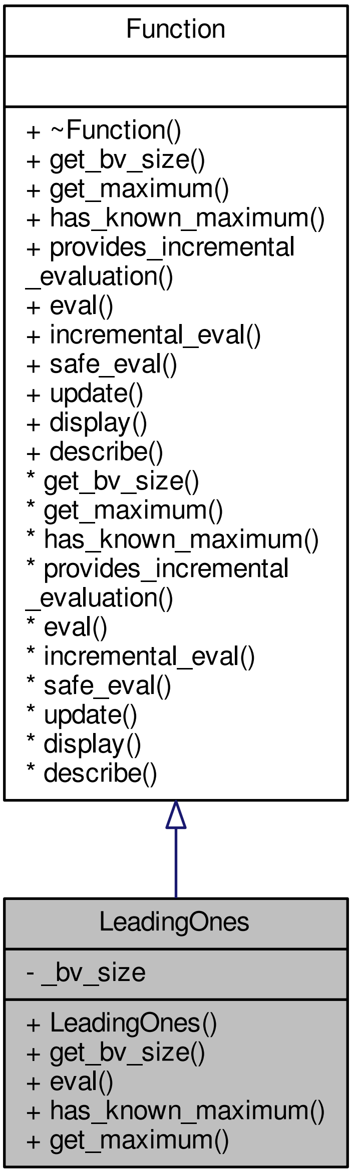
- lib/hnco/exception.hh

5.37 LeadingOnes Class Reference

Leading ones.

```
#include <hnco/functions/theory.hh>
```

Inheritance diagram for LeadingOnes:



Public Member Functions

- LeadingOnes (int bv_size)

  Constructor.
- size_t get_bv_size ()

  Get bit vector size.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.
- bool has_known_maximum ()

Check for a known maximum.

- double get_maximum ()

  Get the global maximum.

Private Attributes

- size_t _bv_size

  Bit vector size.

5.37.1 Detailed Description

Leading ones.

Definition at line 80 of file theory.hh.

5.37.2 Member Function Documentation

5.37.2.1 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true

Reimplemented from Function.

Definition at line 100 of file theory.hh.

The documentation for this class was generated from the following files:

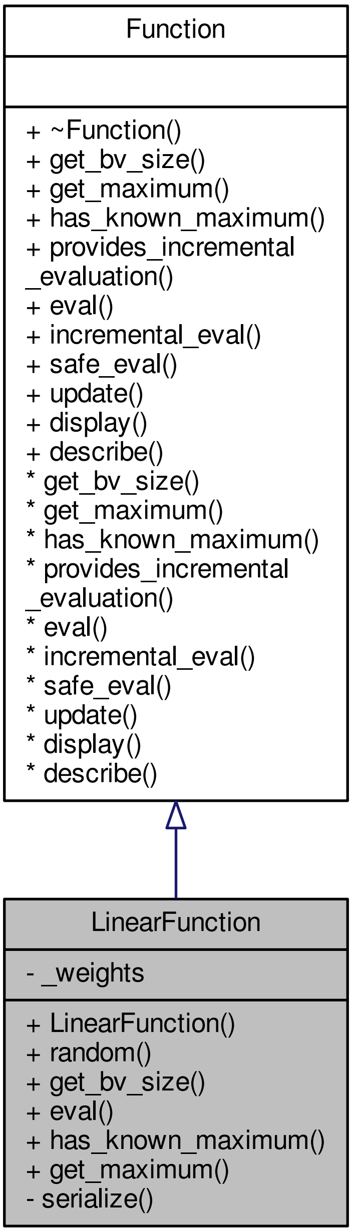
- lib/hnco/functions/theory.hh
- lib/hnco/functions/theory.cc

5.38 LinearFunction Class Reference

Linear function.

```
#include <hnco/functions/linear-function.hh>
```

Inheritance diagram for LinearFunction:



Public Member Functions

- LinearFunction ()

Constructor.
- void random (int n)

  Random instance.
- size_t get_bv_size ()

  Get bit vector size.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.
- bool has_known_maximum ()

  Check for a known maximum.
- double get_maximum ()

  Get the global maximum.

Private Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serialize.

Private Attributes

- std::vector< double > _weights

  Weights.

Friends

- class **boost::serialization::access**

5.38.1 Detailed Description

Linear function.

Definition at line 40 of file linear-function.hh.

5.38.2 Member Function Documentation

5.38.2.1 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true

Reimplemented from Function.

Definition at line 76 of file linear-function.hh.

5.38.2.2 random()

```
void random (
             int n )
```

Random instance.

Parameters

n	Size of bit vectors

Definition at line 34 of file linear-function.cc.

The documentation for this class was generated from the following files:

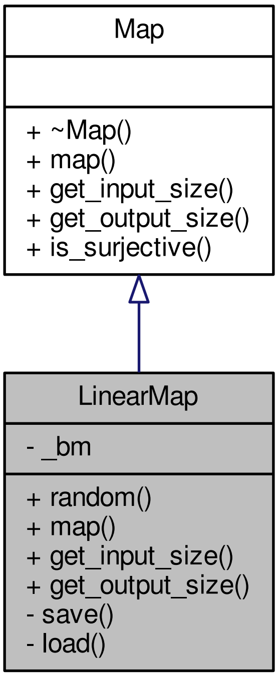
- lib/hnco/functions/linear-function.hh
- lib/hnco/functions/linear-function.cc

5.39 LinearMap Class Reference

Linear map.

```
#include <hnco/map.hh>
```

Inheritance diagram for LinearMap:



Public Member Functions

- void random (int n, int m)

  Random instance.
- void map (const bit_vector_t &input, bit_vector_t &output)

  Map.
- size_t get_input_size ()

  Get input size.
- size_t get_output_size ()

  Get output size.

Private Member Functions

- template<class Archive >
  void save (Archive &ar, const unsigned int version) const

  Save.
- template<class Archive >
  void load (Archive &ar, const unsigned int version)

  Load.

Private Attributes

- bit_matrix_t _bm

  Bit matrix.

Friends

- class **boost::serialization::access**

5.39.1 Detailed Description

Linear map.

A linear map f from Z_2^m to Z_2^n is defined by $f(x) = Ax$, where A is an m x n bit matrix.

Warning

> The class does not reimplement the member function is_surjective hence a linear map is always considered not surjective.

Definition at line 197 of file map.hh.

The documentation for this class was generated from the following files:

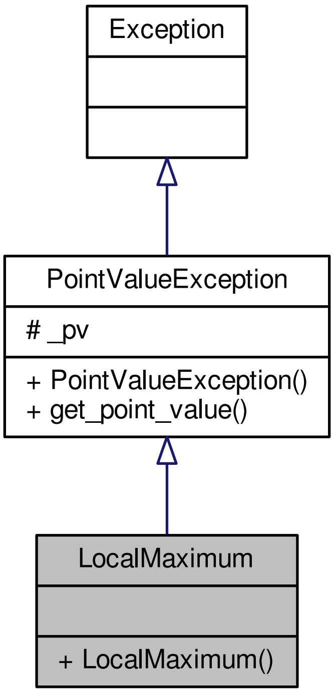
- lib/hnco/map.hh
- lib/hnco/map.cc

5.40 LocalMaximum Class Reference

Local maximum.

`#include <hnco/exception.hh>`

Inheritance diagram for LocalMaximum:



Public Member Functions

- LocalMaximum (const point_value_t &pv)

  Const.

Additional Inherited Members

5.40.1 Detailed Description

Local maximum.

Definition at line 70 of file exception.hh.

The documentation for this class was generated from the following file:

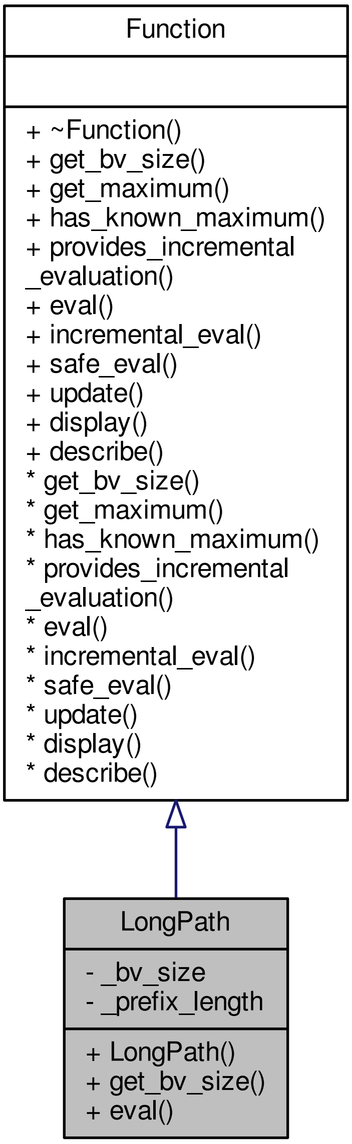
- lib/hnco/exception.hh

5.41 LongPath Class Reference

Long path.

```
#include <hnco/functions/long-path.hh>
```

Inheritance diagram for LongPath:



Public Member Functions

- LongPath (int bv_size, int prefix_length)

    Constructor.

- size_t get_bv_size ()

  Get bit vector size.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.

Private Attributes

- size_t _bv_size

  Bit vector size.
- int _prefix_length

  Prefix length.

5.41.1 Detailed Description

Long path.

Long paths have been introduced in:

Jeffrey Horn, David E. Goldberg, and Kalyanmoy Deb, "Long Path Problems", PPSN III, 1994.

Here we follow the definition given in "Analyzing evolutionary algorithms" by Thomas Jansen.

As an example, here is the 2-long path of dimension 4:

- 0000
- 0001
- 0011
- 0111
- 1111
- 1101
- 1100

The fitness is increasing along the path. The fitness on the complementary of the path is defined as a linear function pointing to the beginning of the path.

Definition at line 54 of file long-path.hh.

The documentation for this class was generated from the following files:

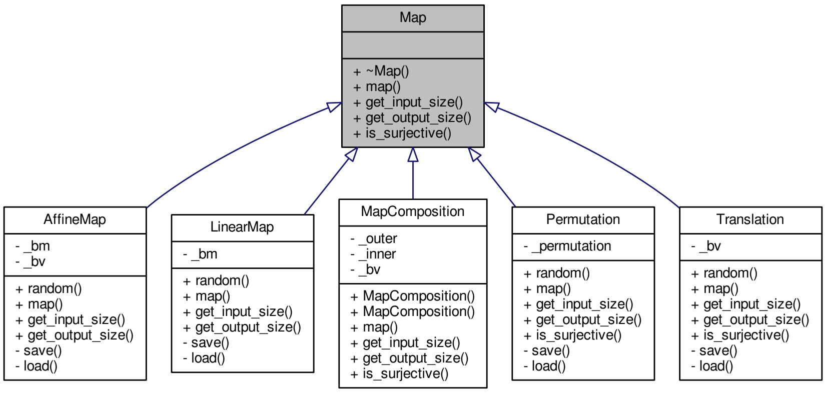
- lib/hnco/functions/long-path.hh
- lib/hnco/functions/long-path.cc

5.42 Map Class Reference

Map.

`#include <hnco/map.hh>`

Inheritance diagram for Map:



Public Member Functions

- virtual ∼Map ()

  Destructor.
- virtual void map (const bit_vector_t &input, bit_vector_t &output)=0

  Map.
- virtual size_t get_input_size ()=0

  Get input size.
- virtual size_t get_output_size ()=0

  Get output size.
- virtual bool is_surjective ()

  Check for surjective map.

5.42.1 Detailed Description

Map.

Definition at line 39 of file map.hh.

5.42.2 Member Function Documentation

5.42.2.1 is_surjective()

```
virtual bool is_surjective ( )  [inline], [virtual]
```

Check for surjective map.

Returns

false

Reimplemented in MapComposition, Permutation, and Translation.

Definition at line 59 of file map.hh.

The documentation for this class was generated from the following file:

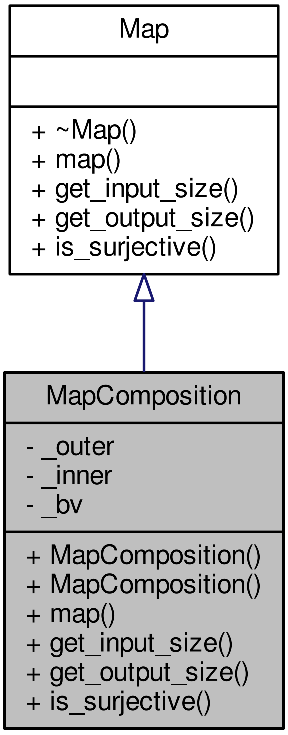
- lib/hnco/map.hh

5.43 MapComposition Class Reference

Map composition.

```
#include <hnco/map.hh>
```

Inheritance diagram for MapComposition:



Public Member Functions

- MapComposition ()

  Default constructor.
- MapComposition (Map ∗outer, Map ∗inner)

  Constructor.
- void map (const bit_vector_t &input, bit_vector_t &output)

  Map.
- size_t get_input_size ()

  Get input size.
- size_t get_output_size ()

  Get output size.
- bool is_surjective ()

  Check for surjective map.

Private Attributes

- Map ∗ _outer

  Outer map.
- Map ∗ _inner

  Inner map.
- bit_vector_t _bv

  Temporary bit vector.

5.43.1 Detailed Description

Map composition.

The resulting composition f is defined for all bit vector x by f(x) = outer(inner(x)).

Definition at line 326 of file map.hh.

5.43.2 Constructor & Destructor Documentation

5.43.2.1 MapComposition()

```
MapComposition (
            Map ∗ outer,
            Map ∗ inner )  [inline]
```

Constructor.

Parameters

outer	outer map
inner	inner map

Precondition

outer->get_input_size() == inner->get_output_size()

Definition at line 350 of file map.hh.

5.43.3 Member Function Documentation

5.43.3.1 is_surjective()

```
bool is_surjective ( )  [inline], [virtual]
```

Check for surjective map.

Returns

true if both maps are surjective

Reimplemented from Map.

Definition at line 374 of file map.hh.

The documentation for this class was generated from the following file:

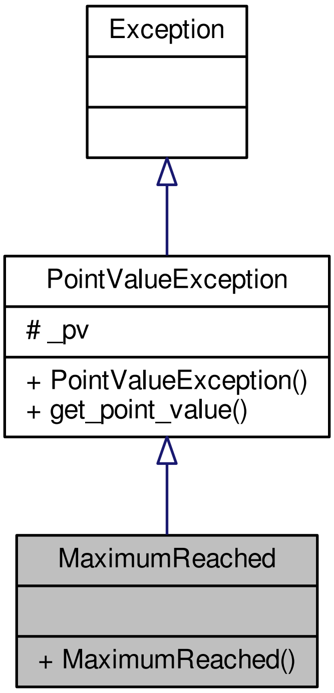
- lib/hnco/map.hh

5.44 MaximumReached Class Reference

Maximum reached.

```
#include <hnco/exception.hh>
```

Inheritance diagram for MaximumReached:



Public Member Functions

- MaximumReached (const point_value_t &pv)

  Constructor.

Additional Inherited Members

5.44.1 Detailed Description

Maximum reached.

Definition at line 52 of file exception.hh.

The documentation for this class was generated from the following file:

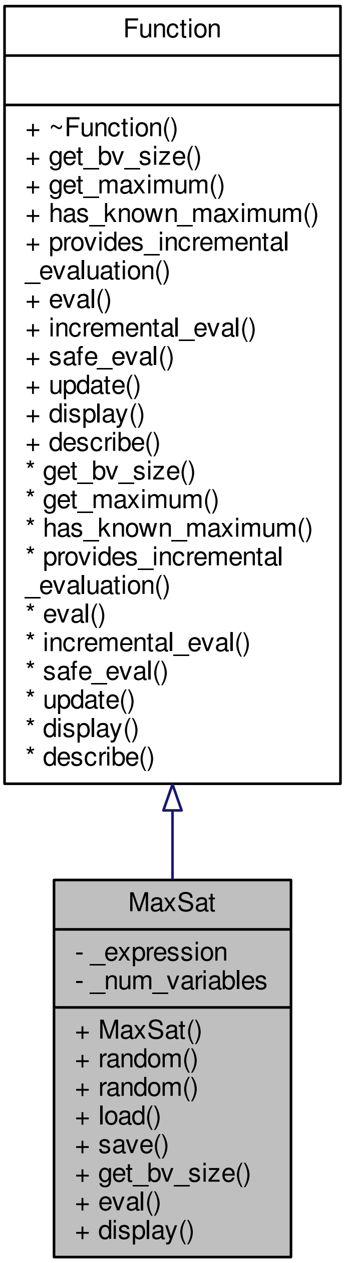
- lib/hnco/exception.hh

5.45 MaxSat Class Reference

MAX-SAT.

```
#include <hnco/functions/max-sat.hh>
```

Inheritance diagram for MaxSat:



Public Member Functions

- MaxSat ()

  Default constructor.
- void random (int n, int k, int c)

  Random instance.
- void random (const bit_vector_t &solution, int k, int c)

  Random instance with satisfiable expression.
- void load (std::istream &stream)

  Load an instance.
- void save (std::ostream &stream)

  Save an instance.
- size_t get_bv_size ()

  Get bit vector size.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.
- void display (std::ostream &stream)

  Display the expression.

Private Attributes

- std::vector< std::vector< int > > _expression

  Expression.
- size_t _num_variables

  Number of variables.

5.45.1 Detailed Description

MAX-SAT.

Definition at line 35 of file max-sat.hh.

5.45.2 Member Function Documentation

5.45.2.1 load()

```
void load (
            std::istream & stream )
```

Load an instance.

Exceptions

Error	

Definition at line 134 of file max-sat.cc.

5.45.2.2 random() [1/2]

```
void random (
            int n,
            int k,
            int c )
```

Random instance.

Parameters

n	Size of bit vectors
k	Number of literals per clause
c	Number of clauses

Definition at line 39 of file max-sat.cc.

5.45.2.3 random() [2/2]

```
void random (
            const bit_vector_t & solution,
            int k,
            int c )
```

Random instance with satisfiable expression.

Warning

> Since the expression is satisfiable, the maximum of the function is equal to the number of clauses in the expression. However, this information is lost in the save and load cycle as the archive format only manages the expression itself.

Parameters

solution	Solution
k	Number of literals per clause
c	Number of clauses

Definition at line 67 of file max-sat.cc.

5.45.3 Member Data Documentation

5.45.3.1 _expression

```
std::vector<std::vector<int> > _expression  [private]
```

Expression.

An expression is represented by a vector of clauses. A clause is represented by a vector of literals. A literal is represented by a non null integer; if the integer is positive then the literal is a variable; if it is negative then it is the logical negation of a variable.

Definition at line 45 of file max-sat.hh.

The documentation for this class was generated from the following files:

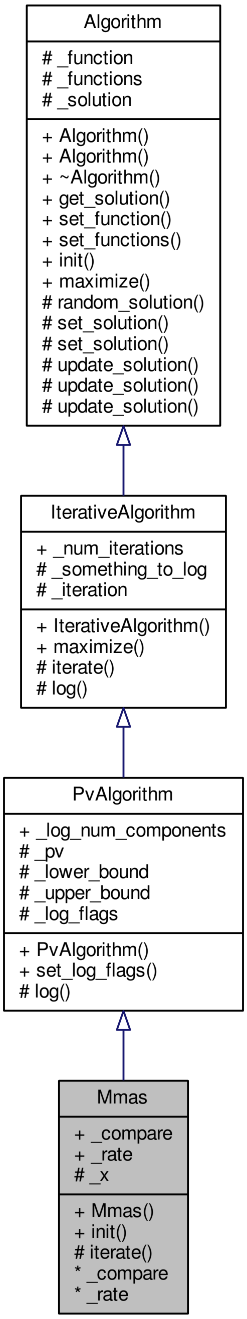
- lib/hnco/functions/max-sat.hh
- lib/hnco/functions/max-sat.cc

5.46 Mmas Class Reference

Max-min ant system.

```
#include <hnco/algorithms/pv/mmas.hh>
```

Inheritance diagram for Mmas:



Public Member Functions

- Mmas (int n)

    Constructor.

- void init ()

    Initialization.

Public Attributes

Parameters

- std::function< bool(double, double)> _compare = std::greater_equal<double>()

  Binary operator for comparing evaluations.
- double _rate

  Learning rate.

Protected Member Functions

- void iterate ()

  Single iteration.

Protected Attributes

- bit_vector_t _x

  Candidate solution.

Additional Inherited Members

5.46.1 Detailed Description

Max-min ant system.

Definition at line 33 of file mmas.hh.

The documentation for this class was generated from the following files:

- lib/hnco/algorithms/pv/mmas.hh
- lib/hnco/algorithms/pv/mmas.cc

5.47 Model Class Reference

Model of a Boltzmann machine.

```
#include <hnco/algorithms/bm-pbil/model.hh>
```

Public Member Functions

- Model (int n)

  Constructor.
- void init ()

  Initialize.
- void reset_mc ()

  Reset Markov chain.
- void gibbs_sampler (size_t i)

  A Gibbs sampler cycle.
- void gibbs_sampler_synchronous ()

  A synchronous Gibbs sampler.
- const bit_vector_t & get_state ()

  Get the state of the Gibbs sampler.
- void update (const ModelParameters &p, const ModelParameters &q, double rate)

  Update parameters in the direction of p and away from q.
- double norm_infinite ()

  Infinite norm of the parameters.
- double norm_l1 ()

  l1 norm of the parameters

Private Attributes

- ModelParameters _model_parameters

  Model parameters.
- bit_vector_t _state

  State of the Gibbs sampler.
- pv_t _pv

  Probability vector for synchronous Gibbs sampling.

5.47.1 Detailed Description

Model of a Boltzmann machine.

Definition at line 75 of file model.hh.

The documentation for this class was generated from the following files:

- lib/hnco/algorithms/bm-pbil/model.hh
- lib/hnco/algorithms/bm-pbil/model.cc

5.48 ModelParameters Class Reference

Parameters of a Boltzmann machine.

```
#include <hnco/algorithms/bm-pbil/model.hh>
```

Public Member Functions

- ModelParameters (int n)

  Constructor.
- void init ()

  Initialize.
- void add (const bit_vector_t &x)

  Add a bit_vector_t.
- void average (int count)

  Compute averages.
- void update (const ModelParameters &p, const ModelParameters &q, double rate)

  Update parameters in the direction of p and away from q.
- double norm_infinite ()

  Infinite norm of the parameters.
- double norm_l1 ()

  l1 norm of the parameters

Private Attributes

- std::vector< std::vector< double > > _weight

  Weights.
- std::vector< double > _bias

  Bias.

Friends

- class **Model**

5.48.1 Detailed Description

Parameters of a Boltzmann machine.

Definition at line 36 of file model.hh.

The documentation for this class was generated from the following files:

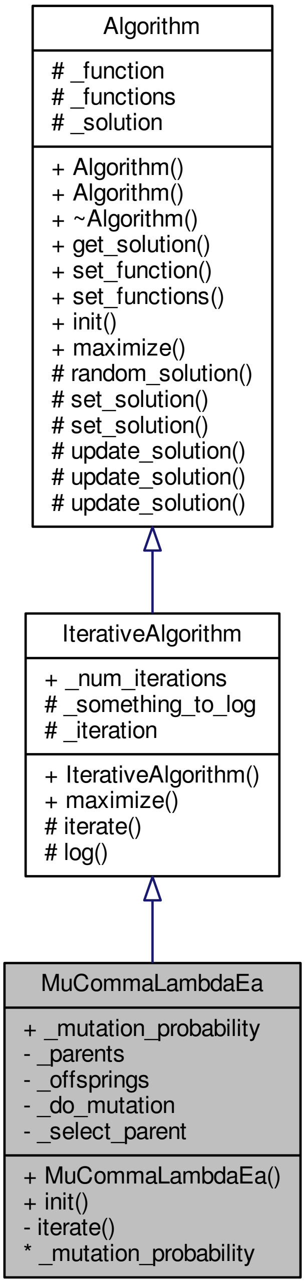
- lib/hnco/algorithms/bm-pbil/model.hh
- lib/hnco/algorithms/bm-pbil/model.cc

5.49 MuCommaLambdaEa Class Reference

(mu,lambda) EA

```
#include <hnco/algorithms/ea/mu-comma-lambda-ea.hh>
```

Inheritance diagram for MuCommaLambdaEa:



Public Member Functions

- MuCommaLambdaEa (int n, int mu, int lambda)

    Constructor.
- void init ()

    Initialization.

Public Attributes

Parameters

- double _mutation_probability

    Mutation probability.

Private Member Functions

- void iterate ()

    Single iteration.

Private Attributes

- Population _parents

    Parents.
- Population _offsprings

    Offsprings.
- std::bernoulli_distribution _do_mutation

    Do mutation.
- std::uniform_int_distribution< int > _select_parent

    Select parent.

Additional Inherited Members

5.49.1 Detailed Description

(mu,lambda) EA

Definition at line 35 of file mu-comma-lambda-ea.hh.

The documentation for this class was generated from the following files:

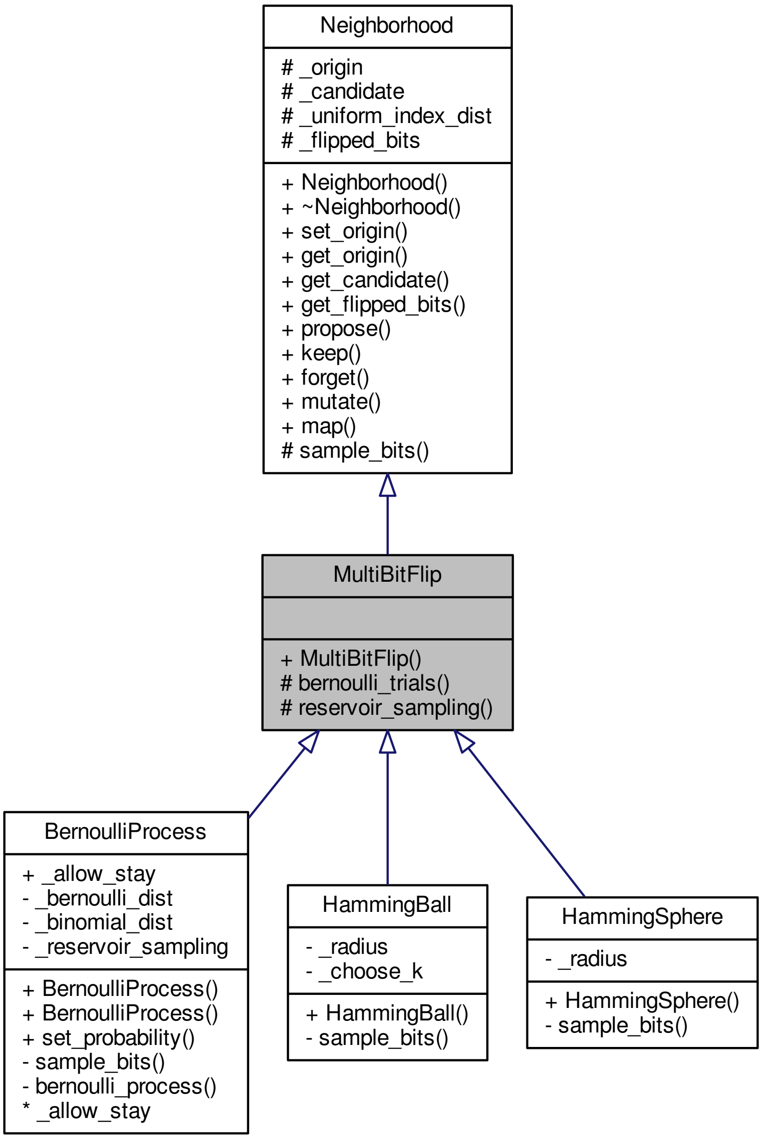
- lib/hnco/algorithms/ea/mu-comma-lambda-ea.hh
- lib/hnco/algorithms/ea/mu-comma-lambda-ea.cc

5.50 MultiBitFlip Class Reference

Multi bit flip.

```
#include <hnco/neighborhoods/neighborhood.hh>
```

Inheritance diagram for MultiBitFlip:



Public Member Functions

- MultiBitFlip (int n)

  Constructor.

Protected Member Functions

- void bernoulli_trials (int k)

  Sample a given number of bits using Bernoulli trials.
- void reservoir_sampling (int k)

  Sample a given number of bits using resevoir sampling.

Additional Inherited Members

5.50.1 Detailed Description

Multi bit flip.

Definition at line 183 of file neighborhood.hh.

5.50.2 Constructor & Destructor Documentation

5.50.2.1 MultiBitFlip()

```
MultiBitFlip (
             int n )  [inline]
```

Constructor.

Parameters

n	Size of bit vectors

Definition at line 206 of file neighborhood.hh.

5.50.3 Member Function Documentation

5.50.3.1 bernoulli_trials()

```
void bernoulli_trials (
             int k )  [protected]
```

Sample a given number of bits using Bernoulli trials.

Parameters

k	Number of bits to sample

Definition at line 34 of file neighborhood.cc.

5.50.3.2 reservoir_sampling()

```
void reservoir_sampling (
             int k )  [protected]
```

Sample a given number of bits using resevoir sampling.

Parameters

k	Number of bits to sample

Definition at line 52 of file neighborhood.cc.

The documentation for this class was generated from the following files:

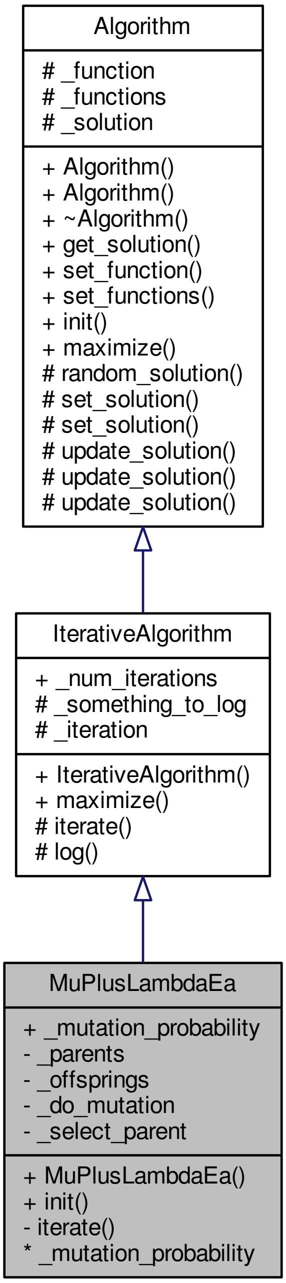
- lib/hnco/neighborhoods/neighborhood.hh
- lib/hnco/neighborhoods/neighborhood.cc

5.51 MuPlusLambdaEa Class Reference

(mu+lambda) EA

`#include <hnco/algorithms/ea/mu-plus-lambda-ea.hh>`

Inheritance diagram for MuPlusLambdaEa:



Public Member Functions

- MuPlusLambdaEa (int n, int mu, int lambda)

    Constructor.
- void init ()

    Initialization.

Public Attributes

Parameters

- double _mutation_probability

  Mutation probability.

Private Member Functions

- void iterate ()

  Single iteration.

Private Attributes

- Population _parents

  Parents.
- Population _offsprings

  Offsprings.
- std::bernoulli_distribution _do_mutation

  Do mutation.
- std::uniform_int_distribution< int > _select_parent

  Select parent.

Additional Inherited Members

5.51.1 Detailed Description

(mu+lambda) EA

Definition at line 34 of file mu-plus-lambda-ea.hh.

The documentation for this class was generated from the following files:

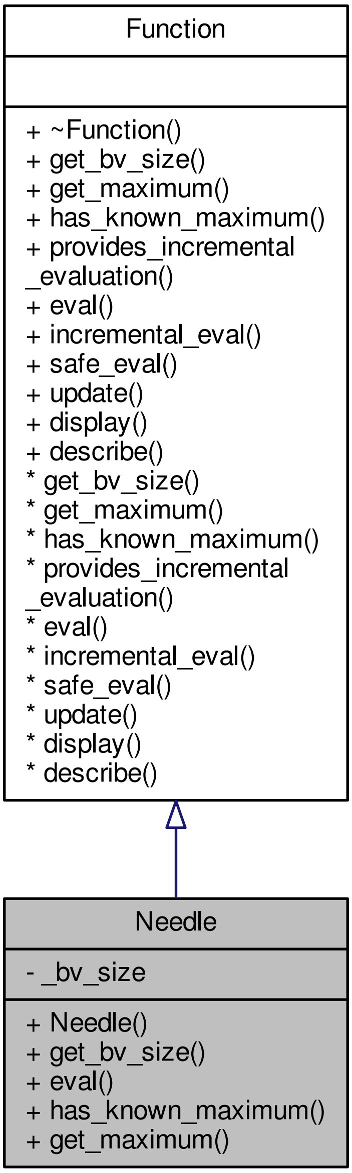
- lib/hnco/algorithms/ea/mu-plus-lambda-ea.hh
- lib/hnco/algorithms/ea/mu-plus-lambda-ea.cc

5.52 Needle Class Reference

Needle in a haystack.

```
#include <hnco/functions/theory.hh>
```

Inheritance diagram for Needle:



Public Member Functions

- Needle (int bv_size)

Constructor.

- size_t get_bv_size ()

  Get bit vector size.

- double eval (const bit_vector_t &)

  Evaluate a bit vector.

- bool has_known_maximum ()

  Check for a known maximum.

- double get_maximum ()

  Get the global maximum.

Private Attributes

- size_t _bv_size

  Bit vector size.

5.52.1 Detailed Description

Needle in a haystack.

Definition at line 108 of file theory.hh.

5.52.2 Member Function Documentation

5.52.2.1 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true

Reimplemented from Function.

Definition at line 128 of file theory.hh.

The documentation for this class was generated from the following files:

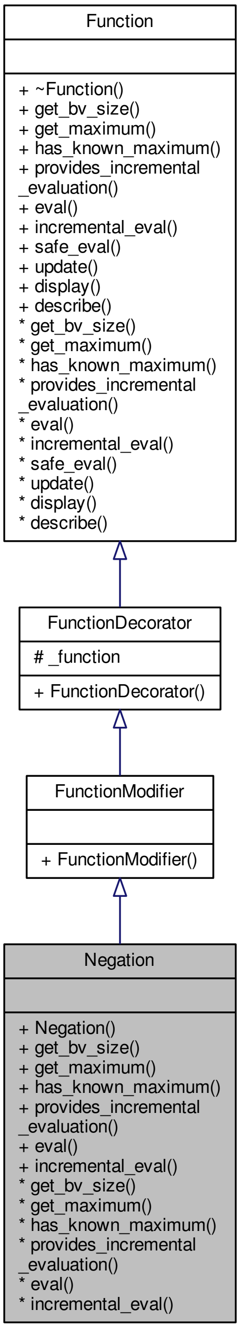
- lib/hnco/functions/theory.hh
- lib/hnco/functions/theory.cc

5.53 Negation Class Reference

Negation.

```
#include <hnco/functions/decorators/function-modifier.hh>
```

Inheritance diagram for Negation:



Public Member Functions

- Negation (Function ∗function)

  Constructor.

Information about the function

- size_t get_bv_size ()

  Get bit vector size.
- double get_maximum ()

  Get the global maximum.
- bool has_known_maximum ()

  Check for a known maximum.
- bool provides_incremental_evaluation ()

  Check whether the function provides incremental evaluation.

Evaluation

- double eval (const bit_vector_t &)

  Evaluate a bit vector.
- double incremental_eval (const bit_vector_t &x, double value, const hnco::sparse_bit_vector_t &flipped←_bits)

  Incremental evaluation.

Additional Inherited Members

5.53.1 Detailed Description

Negation.

Use cases:

- for algorithms which minimize rather than maximize a function
- for functions one wishes to minimize
- when minimization is needed inside an algorithm

Definition at line 58 of file function-modifier.hh.

5.53.2 Member Function Documentation

5.53.2.1 get_maximum()

```
double get_maximum ( )  [inline], [virtual]
```

Get the global maximum.

Exceptions

Error	

Reimplemented from Function.

Definition at line 76 of file function-modifier.hh.

5.53.2.2 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

false

Reimplemented from Function.

Definition at line 80 of file function-modifier.hh.

5.53.2.3 provides_incremental_evaluation()

```
bool provides_incremental_evaluation ( )  [inline], [virtual]
```

Check whether the function provides incremental evaluation.

Returns

true

Reimplemented from Function.

Definition at line 85 of file function-modifier.hh.

The documentation for this class was generated from the following files:

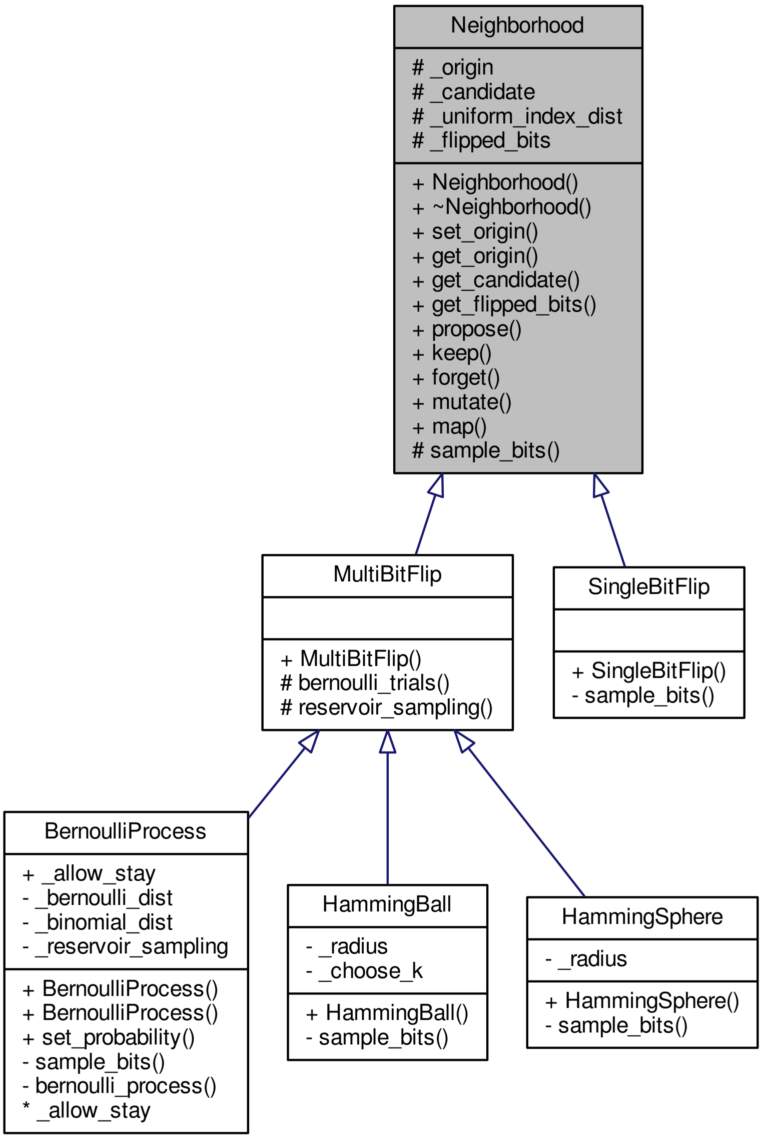
- lib/hnco/functions/decorators/function-modifier.hh
- lib/hnco/functions/decorators/function-modifier.cc

5.54 Neighborhood Class Reference

Neighborhood.

```
#include <hnco/neighborhoods/neighborhood.hh>
```

Inheritance diagram for Neighborhood:



Public Member Functions

- Neighborhood (int n)

  Constructor.
- virtual ∼Neighborhood ()

Destructor.
- virtual void set_origin (const bit_vector_t &x)

  Set the origin.
- virtual const bit_vector_t & get_origin ()

  Get the origin.
- virtual const bit_vector_t & get_candidate ()

  Get the candidate bit vector.
- virtual const sparse_bit_vector_t & get_flipped_bits ()

  Get flipped bits.
- virtual void propose ()

  Propose a candidate bit vector.
- virtual void keep ()

  Keep the candidate bit vector.
- virtual void forget ()

  Forget the candidate bit vector.
- virtual void mutate (bit_vector_t &bv)

  Mutate.
- virtual void map (const bit_vector_t &input, bit_vector_t &output)

  Map.

Protected Member Functions

- virtual void sample_bits ()=0

  Sample bits.

Protected Attributes

- bit_vector_t _origin

  Origin of the neighborhood.
- bit_vector_t _candidate

  candidate bit vector
- std::uniform_int_distribution< int > _uniform_index_dist

  Uniform index distribution.
- sparse_bit_vector_t _flipped_bits

  Flipped bits.

5.54.1 Detailed Description

Neighborhood.

A neighborhood maintains two points, _origin and _candidate. They are initialized in the same state by set_origin. A Neighborhood class must implement the member function sample_bits which samples the bits to flip in _origin to get a _candidate. The following member functions take care of the modifications:

- propose: flip _candidate
- keep: flip _origin
- forget flip _candidate

After keep or forget, _origin and _candidate are in the same state again.

A Neighborhood class can also behave as a mutation operator through the member functions mutate and map.

Definition at line 61 of file neighborhood.hh.

5.54.2 Constructor & Destructor Documentation

5.54.2.1 Neighborhood()

```
Neighborhood (
            int n )  [inline]
```

Constructor.

Parameters

n	Size of bit vectors

Definition at line 86 of file neighborhood.hh.

5.54.3 Member Function Documentation

5.54.3.1 map()

```
virtual void map (
            const bit_vector_t & input,
            bit_vector_t & output )  [inline], [virtual]
```

Map.

The output bit vector is a mutated version of the input bit vector.

Parameters

input	Input bit vector
output	Output bit vector

Definition at line 148 of file neighborhood.hh.

5.54.3.2 mutate()

```
virtual void mutate (
            bit_vector_t & bv )  [inline], [virtual]
```

Mutate.

In-place mutation of the bit vector.

Parameters

bv	Bit vector to mutate

Definition at line 134 of file neighborhood.hh.

The documentation for this class was generated from the following file:

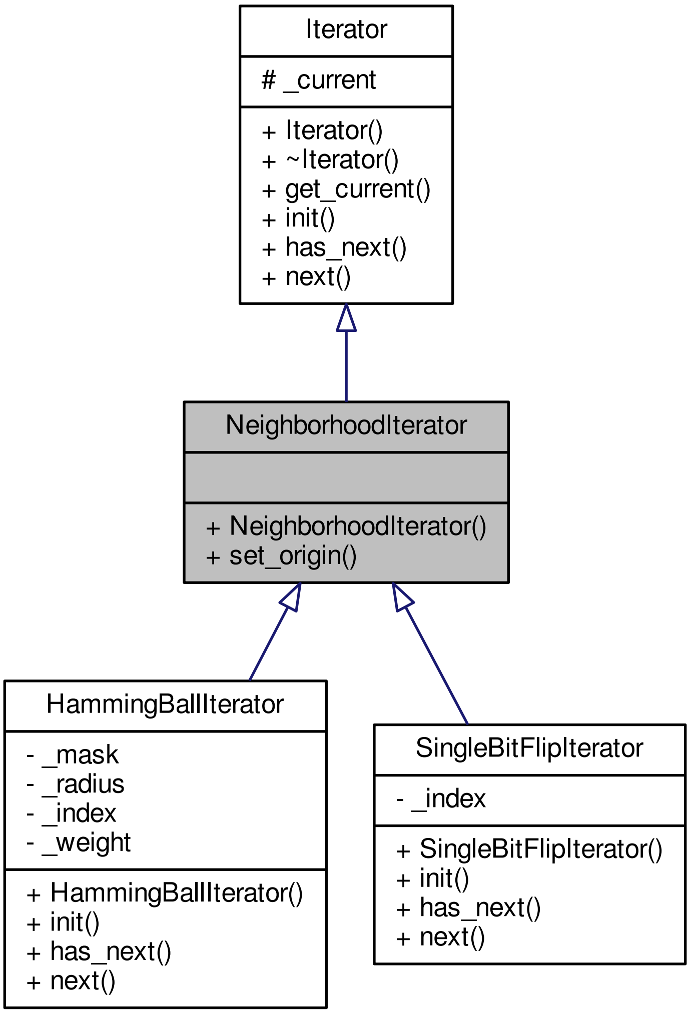
- lib/hnco/neighborhoods/neighborhood.hh

5.55 NeighborhoodIterator Class Reference

Neighborhood iterator.

```
#include <hnco/neighborhoods/neighborhood-iterator.hh>
```

Inheritance diagram for NeighborhoodIterator:



Public Member Functions

- NeighborhoodIterator (int n)

  Constructor.
- virtual void set_origin (const bit_vector_t &x)

  Set origin.

Additional Inherited Members

5.55.1 Detailed Description

Neighborhood iterator.

Definition at line 38 of file neighborhood-iterator.hh.

5.55.2 Constructor & Destructor Documentation

5.55.2.1 NeighborhoodIterator()

```
NeighborhoodIterator (
                int n )   [inline]
```

Constructor.

Parameters

n	Size of bit vectors

Definition at line 49 of file neighborhood-iterator.hh.

The documentation for this class was generated from the following files:

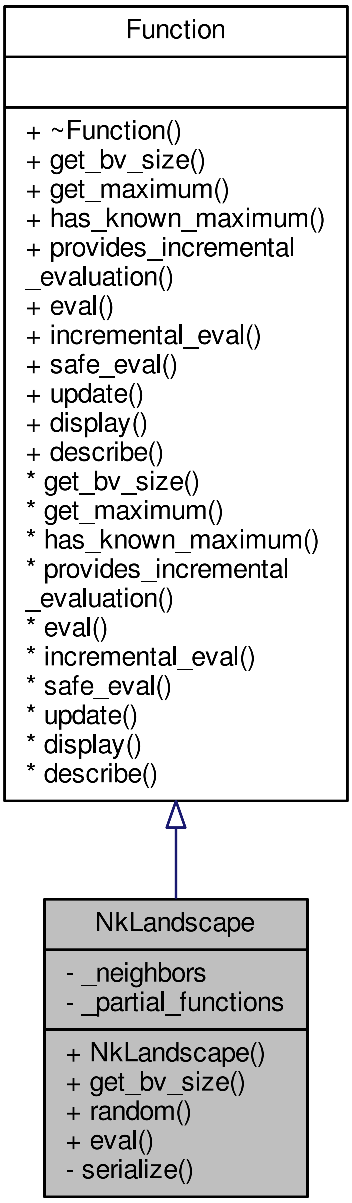
- lib/hnco/neighborhoods/neighborhood-iterator.hh
- lib/hnco/neighborhoods/neighborhood-iterator.cc

5.56 NkLandscape Class Reference

NK landscape.

```
#include <hnco/functions/nk-landscape.hh>
```

Inheritance diagram for NkLandscape:



Public Member Functions

- NkLandscape ()

  Default constructor.
- size_t get_bv_size ()

  Get bit vector size.
- void random (int n, int k, double stddev)

  Random instance.

- double eval (const bit_vector_t &)

  Evaluate a bit vector.

Private Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serialize.

Private Attributes

- std::vector< std::vector< int > > _neighbors

  Bit neighbors.
- std::vector< std::vector< double > > _partial_functions

  Partial functions.

Friends

- class **boost::serialization::access**

5.56.1 Detailed Description

NK landscape.

Source: Kauffman

Definition at line 43 of file nk-landscape.hh.

5.56.2 Member Function Documentation

5.56.2.1 random()

```
void random (
             int n,
             int k,
             double stddev )
```

Random instance.

Parameters

n	Size of bit vector
k	Number of neighbors of each bit
stddev	Standard deviation of the values of the partial functions

Definition at line 33 of file nk-landscape.cc.

The documentation for this class was generated from the following files:

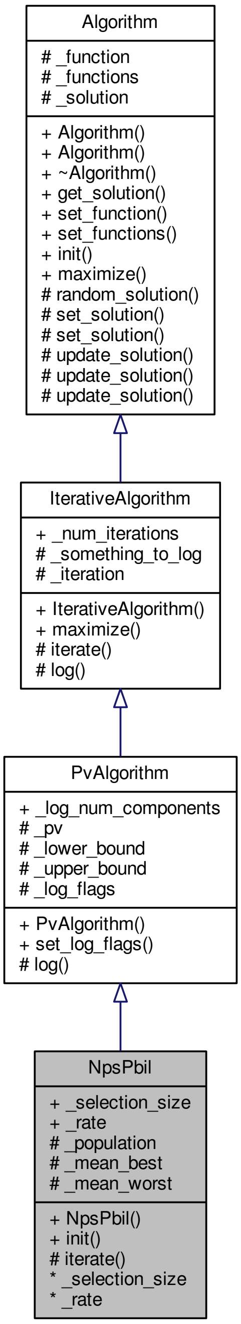
- lib/hnco/functions/nk-landscape.hh
- lib/hnco/functions/nk-landscape.cc

5.57 NpsPbil Class Reference

Population-based incremental learning with negative and positive selection.

```
#include <hnco/algorithms/pv/nps-pbil.hh>
```

Inheritance diagram for NpsPbil:



Public Member Functions

- NpsPbil (int n, int population_size)

  Constructor.

- void init ()

  Initialization.

Public Attributes

Parameters

- int _selection_size = 1

  Selection size.
- double _rate = 1e-3

  Learning rate.

Protected Member Functions

- void iterate ()

  Single iteration.

Protected Attributes

- Population _population

  Population.
- pv_t _mean_best

  Mean of best individuals.
- pv_t _mean_worst

  Mean of worst individuals.

Additional Inherited Members

5.57.1 Detailed Description

Population-based incremental learning with negative and positive selection.

Definition at line 32 of file nps-pbil.hh.

The documentation for this class was generated from the following files:

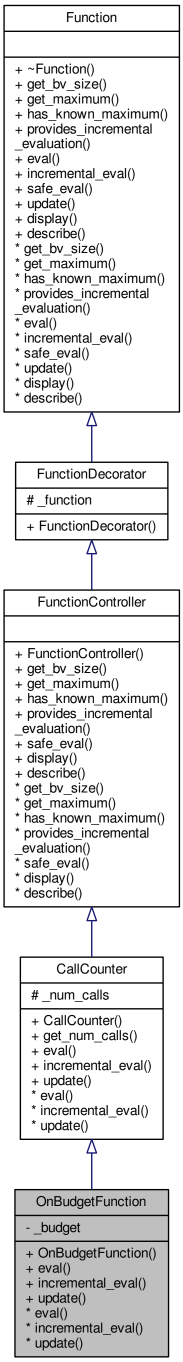
- lib/hnco/algorithms/pv/nps-pbil.hh
- lib/hnco/algorithms/pv/nps-pbil.cc

5.58 OnBudgetFunction Class Reference

CallCounter with a limited number of evaluations.

```
#include <hnco/functions/decorators/function-controller.hh>
```

Inheritance diagram for OnBudgetFunction:



Public Member Functions

- OnBudgetFunction (Function ∗function, int budget)

  Constructor.

Evaluation

- double eval (const bit_vector_t &)

  Evaluate a bit vector.
- double incremental_eval (const bit_vector_t &x, double value, const hnco::sparse_bit_vector_t &flipped←_bits)

  Incremental evaluation.
- void update (const bit_vector_t &x, double value)

  Update after a safe evaluation.

Private Attributes

- int _budget

  Budget.

Additional Inherited Members

5.58.1 Detailed Description

CallCounter with a limited number of evaluations.

Definition at line 299 of file function-controller.hh.

5.58.2 Member Function Documentation

5.58.2.1 eval()

```
double eval (
            const bit_vector_t & x )  [virtual]
```

Evaluate a bit vector.

Exceptions

LastEvaluation	

Reimplemented from CallCounter.

Definition at line 121 of file function-controller.cc.

5.58.2.2 incremental_eval()

```
double incremental_eval (
            const bit_vector_t & x,
            double value,
            const hnco::sparse_bit_vector_t & flipped_bits )  [virtual]
```

Incremental evaluation.

Exceptions

LastEvaluation	

Reimplemented from CallCounter.

Definition at line 132 of file function-controller.cc.

5.58.2.3 update()

```
void update (
            const bit_vector_t & x,
            double value )  [virtual]
```

Update after a safe evaluation.

Exceptions

LastEvaluation	

Reimplemented from CallCounter.

Definition at line 143 of file function-controller.cc.

The documentation for this class was generated from the following files:

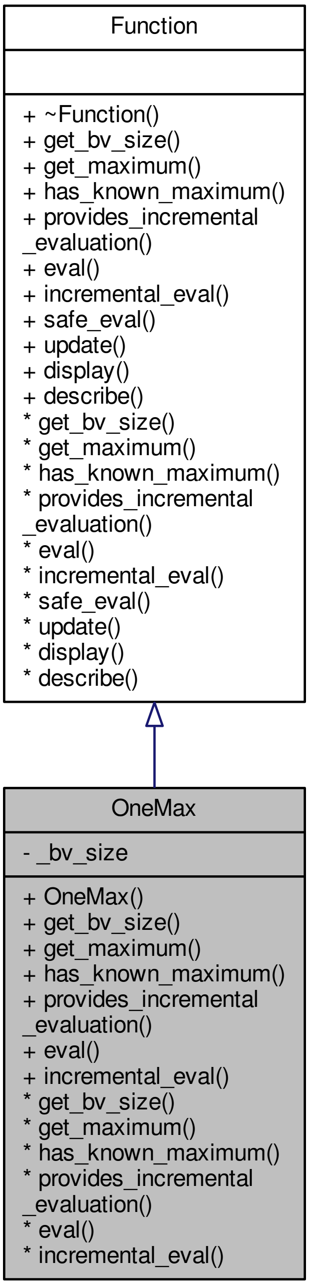
- lib/hnco/functions/decorators/function-controller.hh
- lib/hnco/functions/decorators/function-controller.cc

5.59 OneMax Class Reference

OneMax.

```
#include <hnco/functions/theory.hh>
```

Inheritance diagram for OneMax:



Public Member Functions

- OneMax (int bv_size)

  Constructor.

Information about the function

- size_t get_bv_size ()

    Get bit vector size.
- double get_maximum ()

    Get the global maximum.
- bool has_known_maximum ()

    Check for a known maximum.
- bool provides_incremental_evaluation ()

    Check whether the function provides incremental evaluation.

Evaluation

- double eval (const bit_vector_t &)

    Evaluate a bit vector.
- double incremental_eval (const bit_vector_t &x, double v, const hnco::sparse_bit_vector_t &flipped_bits)

    Incremental evaluation.

Private Attributes

- size_t _bv_size

    Bit vector size.

5.59.1 Detailed Description

OneMax.

Definition at line 30 of file theory.hh.

5.59.2 Member Function Documentation

5.59.2.1 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true

Reimplemented from Function.

Definition at line 54 of file theory.hh.

5.59.2.2 provides_incremental_evaluation()

```
bool provides_incremental_evaluation ( ) [inline], [virtual]
```

Check whether the function provides incremental evaluation.

Returns

true

Reimplemented from Function.

Definition at line 59 of file theory.hh.

The documentation for this class was generated from the following files:

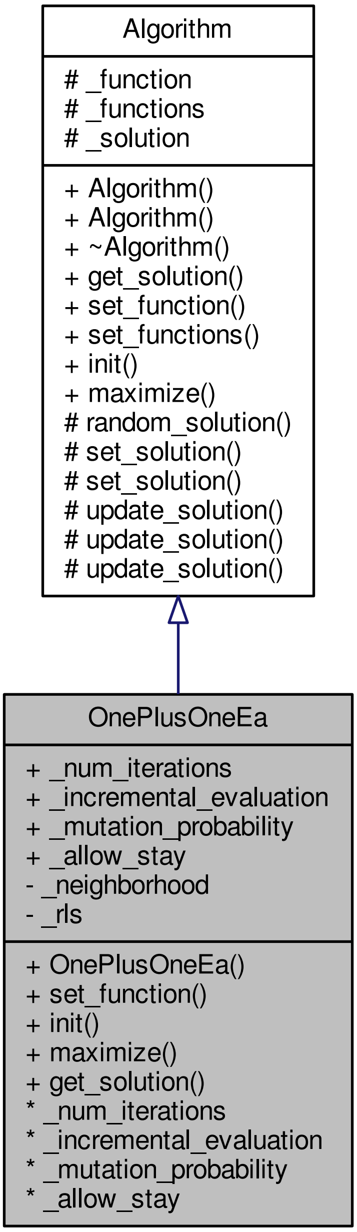
- lib/hnco/functions/theory.hh
- lib/hnco/functions/theory.cc

5.60 OnePlusOneEa Class Reference

(1+1) EA.

```
#include <hnco/algorithms/ea/one-plus-one-ea.hh>
```

Inheritance diagram for OnePlusOneEa:



Public Member Functions

- OnePlusOneEa (int n)

  Constructor.
- void set_function (function::Function ∗function)

  Set function.
- void init ()

Initialization.

- void maximize ()

  Maximize.
- const point_value_t & get_solution ()

  Solution.

Public Attributes

Parameters

- int _num_iterations = 0

  Number of iterations.
- bool _incremental_evaluation = false

  Incremental evaluation.
- double _mutation_probability

  Mutation probability.
- bool _allow_stay = false

  Allow stay.

Private Attributes

- neighborhood::BernoulliProcess _neighborhood

  Neighborhood.
- RandomLocalSearch _rls

  Random local search.

Additional Inherited Members

5.60.1 Detailed Description

(1+1) EA.

(1+1) EA is implemented as a RandomLocalSearch with a BernoulliProcess neighborhood and infinite patience. Thus it does derive from IterativeAlgorithm.

Definition at line 39 of file one-plus-one-ea.hh.

5.60.2 Constructor & Destructor Documentation

5.60.2.1 OnePlusOneEa()

```
OnePlusOneEa (
              int n )   [inline]
```

Constructor.

Parameters

n	Size of bit vectors

_mutation_probability is initialized to 1 / n.

Definition at line 56 of file one-plus-one-ea.hh.

5.60.3 Member Data Documentation

5.60.3.1 _allow_stay

```
bool _allow_stay = false
```

Allow stay.

In case no mutation occurs allow the current bit vector to stay unchanged.

Definition at line 102 of file one-plus-one-ea.hh.

5.60.3.2 _num_iterations

```
int _num_iterations = 0
```

Number of iterations.

_num_iterations <= 0 means indefinite

Definition at line 89 of file one-plus-one-ea.hh.

The documentation for this class was generated from the following file:

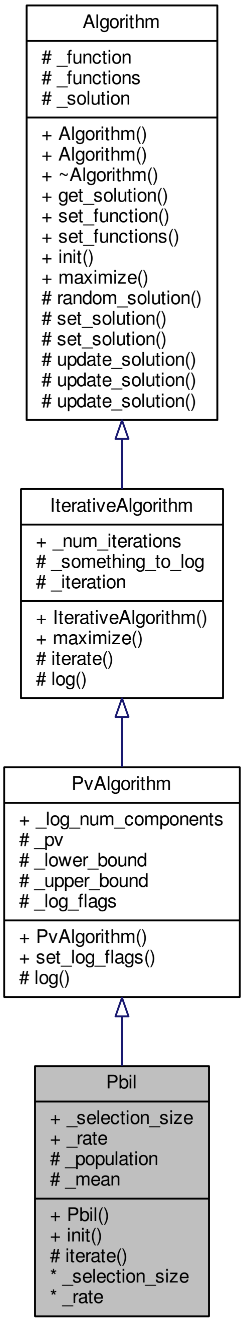
- lib/hnco/algorithms/ea/one-plus-one-ea.hh

5.61 Pbil Class Reference

Population-based incremental learning.

```
#include <hnco/algorithms/pv/pbil.hh>
```

Inheritance diagram for Pbil:



Public Member Functions

- Pbil (int n, int population_size)

  Constructor.
- void init ()

  Initialization.

Public Attributes

Parameters

- int _selection_size = 1

  Selection size.
- double _rate = 1e-3

  Learning rate.

Protected Member Functions

- void iterate ()

  Single iteration.

Protected Attributes

- Population _population

  Population.
- pv_t _mean

  Mean of selected bit vectors.

Additional Inherited Members

5.61.1 Detailed Description

Population-based incremental learning.

Definition at line 32 of file pbil.hh.

The documentation for this class was generated from the following files:

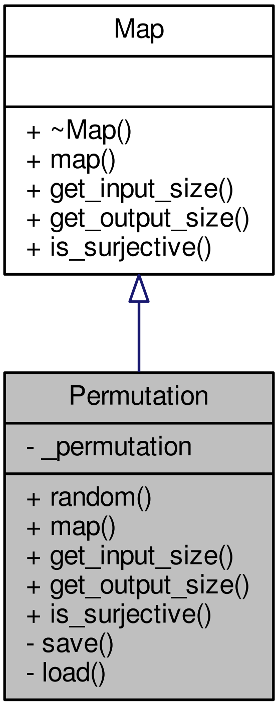
- lib/hnco/algorithms/pv/pbil.hh
- lib/hnco/algorithms/pv/pbil.cc

5.62 Permutation Class Reference

Permutation.

```
#include <hnco/map.hh>
```

Inheritance diagram for Permutation:



Public Member Functions

- void random (int n)

  Random instance.
- void map (const bit_vector_t &input, bit_vector_t &output)

  Map.
- size_t get_input_size ()

  Get input size.
- size_t get_output_size ()

  Get output size.
- bool is_surjective ()

  Check for surjective map.

Private Member Functions

- template<class Archive >
  void save (Archive &ar, const unsigned int version) const
  Save.
- template<class Archive >
  void load (Archive &ar, const unsigned int version)
  Load.

Private Attributes

- permutation_t _permutation
  Permutation.

Friends

- class **boost::serialization::access**

5.62.1 Detailed Description

Permutation.

A permutation is a linear map f from Z_2^n to itself defined by $f(x) = y$, where $y_i = x_{\sigma_i}$ and σ is a permutation of 0, 1, ..., n - 1.

Definition at line 132 of file map.hh.

5.62.2 Member Function Documentation

5.62.2.1 is_surjective()

```
bool is_surjective ( ) [inline], [virtual]
```

Check for surjective map.

Returns

true

Reimplemented from Map.

Definition at line 183 of file map.hh.

The documentation for this class was generated from the following files:

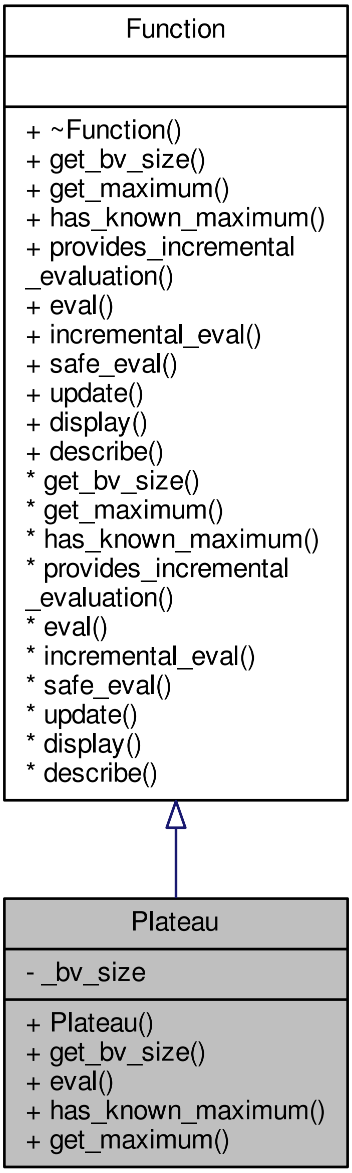
- lib/hnco/map.hh
- lib/hnco/map.cc

5.63 Plateau Class Reference

Plateau.

```
#include <hnco/functions/theory.hh>
```

Inheritance diagram for Plateau:



Public Member Functions

- Plateau (int bv_size)

Constructor.

- size_t get_bv_size ()

  Get bit vector size.

- double eval (const bit_vector_t &)

  Evaluate a bit vector.

- bool has_known_maximum ()

  Check for a known maximum.

- double get_maximum ()

  Get the global maximum.

Private Attributes

- size_t _bv_size

  Bit vector size.

5.63.1 Detailed Description

Plateau.

Definition at line 196 of file theory.hh.

5.63.2 Member Function Documentation

5.63.2.1 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true

Reimplemented from Function.

Definition at line 216 of file theory.hh.

The documentation for this class was generated from the following files:

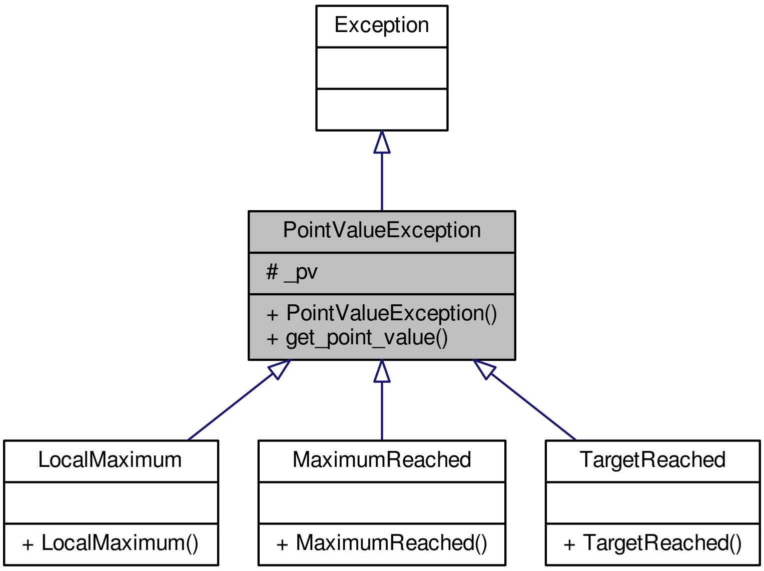
- lib/hnco/functions/theory.hh
- lib/hnco/functions/theory.cc

5.64 PointValueException Class Reference

Point-value exception.

```
#include <hnco/exception.hh>
```

Inheritance diagram for PointValueException:



Public Member Functions

- PointValueException (const point_value_t &pv)

  Constructor.
- const point_value_t & get_point_value () const

  Get point-value.

Protected Attributes

- point_value_t _pv

  Point-value.

5.64.1 Detailed Description

Point-value exception.

Definition at line 38 of file exception.hh.

The documentation for this class was generated from the following file:

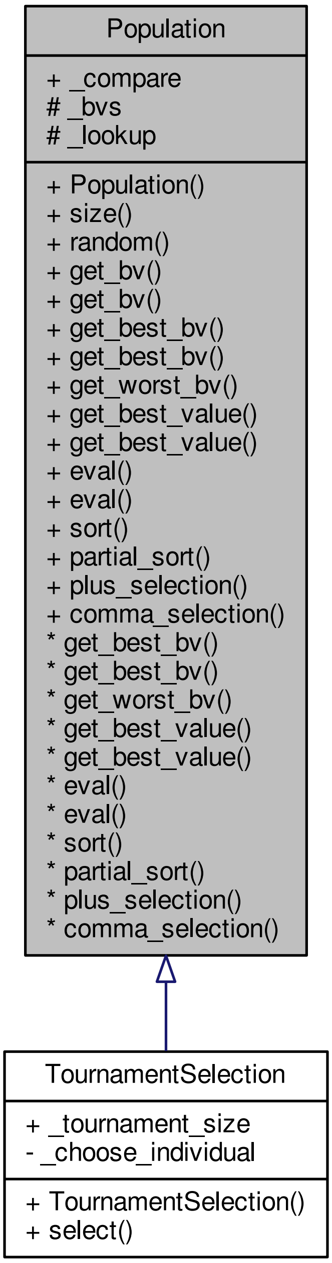
- lib/hnco/exception.hh

5.65 Population Class Reference

Population.

```
#include <hnco/algorithms/population.hh>
```

Inheritance diagram for Population:



Public Types

- typedef std::pair< size_t, double > index_value_t

  Index-value type.

Public Member Functions

- Population (int population_size, int n)

  Constructor.
- std::size_t size () const

  Size.
- void random ()

  Initialize the population with random bit vectors.
- bit_vector_t & get_bv (int i)

  Get a bit vector.
- const bit_vector_t & get_bv (int i) const

  Get a bit vector.

Get sorted bit vectors

- const bit_vector_t & get_best_bv (int i) const

  Get best bit vector.
- const bit_vector_t & get_best_bv () const

  Get best bit vector.
- const bit_vector_t & get_worst_bv (int i) const

  Get worst bit vector.

Get sorted values

- double get_best_value (int i) const

  Get best value.
- double get_best_value () const

  Get best value.

Evaluation and sorting

- void eval (function::Function ∗function)

  Evaluate the population.
- void eval (const std::vector< function::Function ∗> &functions)

  Parallel evaluation of the population.
- void sort ()

  Sort the lookup table.
- void partial_sort (int selection_size)

  Partially sort the lookup table.

Selection

- void plus_selection (const Population &offsprings)

  Plus selection.
- void comma_selection (const Population &offsprings)

  Comma selection.

Public Attributes

- std::function< bool(const index_value_t &, const index_value_t &)> _compare

    Binary operator for comparing index-value pairs.

Protected Attributes

- std::vector< bit_vector_t > _bvs

    Bit vectors.
- std::vector< index_value_t > _lookup

    Lookup table.

5.65.1 Detailed Description

Population.

Definition at line 35 of file population.hh.

5.65.2 Member Function Documentation

5.65.2.1 comma_selection()

```
void comma_selection (
             const Population & offsprings )
```

Comma selection.

Precondition

Offspring population must be sorted.

Warning

The function does not break ties randomly as it should.

Definition at line 108 of file population.cc.

5.65.2.2 get_best_bv() [1/2]

```
const bit_vector_t& get_best_bv (
             int i ) const  [inline]
```

Get best bit vector.

Parameters

i	Index in the sorted population

Precondition

The population must be sorted.

Definition at line 89 of file population.hh.

5.65.2.3 get_best_bv() [2/2]

```
const bit_vector_t& get_best_bv ( ) const  [inline]
```

Get best bit vector.

Precondition

The population must be sorted.

Definition at line 95 of file population.hh.

5.65.2.4 get_best_value() [1/2]

```
double get_best_value (
             int i ) const  [inline]
```

Get best value.

Parameters

i	Index in the sorted population

Precondition

The population must be sorted.

Definition at line 118 of file population.hh.

5.65.2.5 get_best_value() [2/2]

```
double get_best_value ( ) const  [inline]
```

Get best value.

Precondition

The population must be sorted.

Definition at line 124 of file population.hh.

5.65.2.6 get_worst_bv()

```
const bit_vector_t& get_worst_bv (
             int i ) const  [inline]
```

Get worst bit vector.

Parameters

i	Index in the sorted population

Precondition

The population must be sorted.

Definition at line 103 of file population.hh.

5.65.2.7 plus_selection()

```
void plus_selection (
             const Population & offsprings )
```

Plus selection.

Precondition

Both populations must be sorted.

Warning

The function does not break ties randomly as it should.

Definition at line 89 of file population.cc.

5.65.3 Member Data Documentation

5.65.3.1 _compare

```
std::function<bool(const index_value_t&, const index_value_t&)> _compare
```

Initial value:

```
=
      [](const index_value_t& a, const index_value_t& b) { return a.second > b.
      second; }
```

Binary operator for comparing index-value pairs.

Definition at line 43 of file population.hh.

5.65.3.2 _lookup

```
std::vector<index_value_t> _lookup  [protected]
```

Lookup table.

Let p be of type std::pair<size_t, double>. Then p.first is the bv index in the unsorted population whereas p.second is the bv value.

Definition at line 57 of file population.hh.

The documentation for this class was generated from the following files:

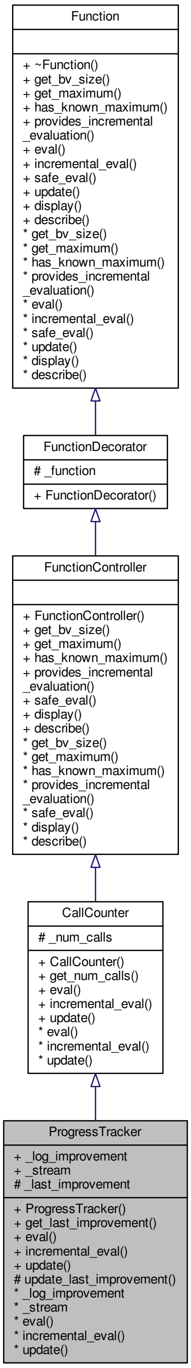
- lib/hnco/algorithms/population.hh
- lib/hnco/algorithms/population.cc

5.66 ProgressTracker Class Reference

ProgressTracker.

```
#include <hnco/functions/decorators/function-controller.hh>
```

Inheritance diagram for ProgressTracker:



Classes

- struct Event

    Event.

Public Member Functions

- ProgressTracker (Function ∗function)

  Constructor.
- const Event & get_last_improvement ()

  Get the last improvement.

Evaluation

- double eval (const bit_vector_t &)

  Evaluate a bit vector.
- double incremental_eval (const bit_vector_t &x, double value, const hnco::sparse_bit_vector_t &flipped↩ _bits)

  Incremental evaluation.
- void update (const bit_vector_t &x, double value)

  Update after a safe evaluation.

Public Attributes

Parameters

- bool _log_improvement = false

  Log improvement.
- std::ostream & _stream = std::cout

  Output stream.

Protected Member Functions

- void update_last_improvement (double value)

  Update last improvement.

Protected Attributes

- Event _last_improvement

  Last improvement.

5.66.1 Detailed Description

ProgressTracker.

A ProgressTracker is a CallCounter which records the last event, that is the time and value of the last improvement.

Definition at line 212 of file function-controller.hh.

5.66.2 Member Function Documentation

5.66.2.1 eval()

```
double eval (
            const bit_vector_t & x )  [virtual]
```

Evaluate a bit vector.

Exceptions

MaximumReached	
TargetReached	

Reimplemented from CallCounter.

Definition at line 153 of file function-controller.cc.

5.66.2.2 get_last_improvement()

```
const Event& get_last_improvement ( )  [inline]
```

Get the last improvement.

Warning

> If _last_improvement.time is zero then _function has never been called. The Event returned by get_last_improvement has therefore no meaning.

Definition at line 276 of file function-controller.hh.

5.66.2.3 incremental_eval()

```
double incremental_eval (
             const bit_vector_t & x,
             double value,
             const hnco::sparse_bit_vector_t & flipped_bits )  [virtual]
```

Incremental evaluation.

Exceptions

MaximumReached	
TargetReached	

Reimplemented from CallCounter.

Definition at line 172 of file function-controller.cc.

5.66.2.4 update()

```
void update (
             const bit_vector_t & x,
             double value )  [virtual]
```

Update after a safe evaluation.

Exceptions

MaximumReached	
TargetReached	

Reimplemented from CallCounter.

Definition at line 191 of file function-controller.cc.

The documentation for this class was generated from the following files:

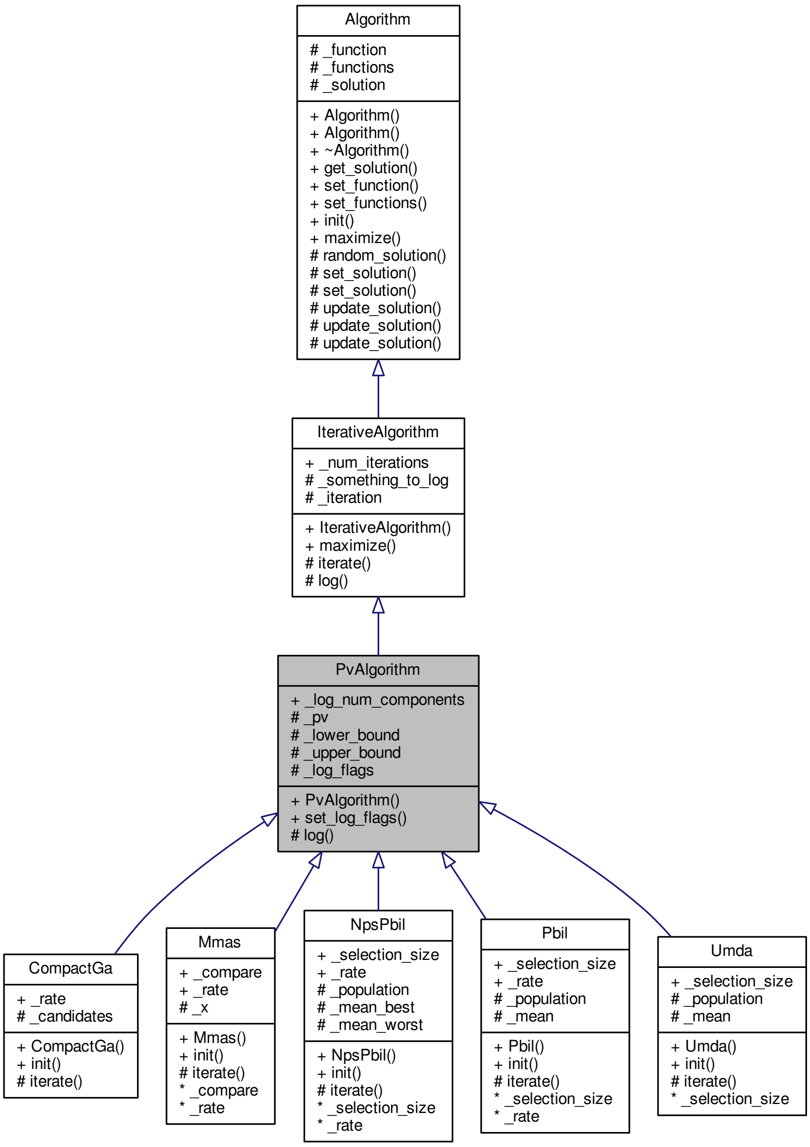
- lib/hnco/functions/decorators/function-controller.hh
- lib/hnco/functions/decorators/function-controller.cc

5.67 PvAlgorithm Class Reference

Probability vector algorithm.

```
#include <hnco/algorithms/pv/pv-algorithm.hh>
```

Inheritance diagram for PvAlgorithm:



Public Types

- enum { LOG_PV, LOG_ENTROPY, **LAST_LOG** }
- typedef std::bitset< LAST_LOG > **log_flags_t**

Public Member Functions

- PvAlgorithm (int n)

  Constructor.
- void set_log_flags (const log_flags_t &lf)

  Set log flags.

Public Attributes

- int _log_num_components = 5

  Number of probability vector components to log.

Protected Member Functions

- void log ()

  Log.

Protected Attributes

- pv_t _pv

  Probability vector.
- double _lower_bound

  Lower bound of probability.
- double _upper_bound

  Upper bound of probability.
- log_flags_t _log_flags

  Log flags.

5.67.1 Detailed Description

Probability vector algorithm.

Definition at line 34 of file pv-algorithm.hh.

5.67.2 Member Enumeration Documentation

5.67.2.1 anonymous enum

```
anonymous enum
```

Enumerator

LOG_PV	Log probability vector.
LOG_ENTROPY	Log entropy.

Definition at line 39 of file pv-algorithm.hh.

The documentation for this class was generated from the following files:

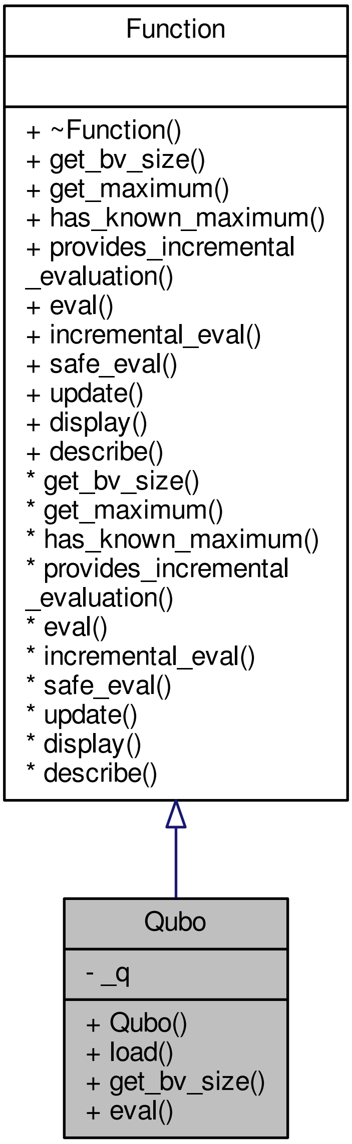
- lib/hnco/algorithms/pv/pv-algorithm.hh
- lib/hnco/algorithms/pv/pv-algorithm.cc

5.68 Qubo Class Reference

Quadratic unconstrained binary optimization.

```
#include <hnco/functions/qubo.hh>
```

Inheritance diagram for Qubo:



Public Member Functions

- Qubo ()

  Constructor.

- void load (std::istream &stream)

  Load an instance.
- size_t get_bv_size ()

  Get bit vector size.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.

Private Attributes

- std::vector$<$ std::vector$<$ double $>$ $>$ _q

  Matrix.

5.68.1 Detailed Description

Quadratic unconstrained binary optimization.

Its expression is of the form $f(x) = \sum_i Q_{ii}x_i + \sum_{i<j} Q_{ij}x_ix_{ij} = x^TQx$, where Q is an n x n upper-triangular matrix.

Qubo is the problem addressed by qbsolv. Here is its description as given on github:

Qbsolv, a decomposing solver, finds a minimum value of a large quadratic unconstrained binary optimization (Q↩ UBO) problem by splitting it into pieces solved either via a D-Wave system or a classical tabu solver.

There are some differences between WalshExpansion2 and Qubo:

- WalshExpansion2 maps 0/1 variables into -1/1 variables whereas Qubo directly deals with binary variables.
- Hence, there is a separate linear part in WalshExpansion2 whereas the linear part in Qubo stems from the diagonal elements of the given matrix.

qbsolv aims at minimizing quadratic functions whereas hnco algorithms aim at maximizing them. Hence Qubo::load negates all elements so that maximizing the resulting function is equivalent to minimizing the original Qubo.

References:

- `https://github.com/dwavesystems/qbsolv`
- `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/bqpinfo.html`

Definition at line 67 of file qubo.hh.

5.68.2 Member Function Documentation

5.68.2.1 load()

```
void load (
            std::istream & stream )
```

Load an instance.

Exceptions

Error	

Definition at line 36 of file qubo.cc.

5.68.3 Member Data Documentation

5.68.3.1 _q

```
std::vector<std::vector<double> > _q  [private]
```

Matrix.

n x n upper triangular matrix.

Definition at line 76 of file qubo.hh.

The documentation for this class was generated from the following files:

- lib/hnco/functions/qubo.hh
- lib/hnco/functions/qubo.cc

5.69 Random Struct Reference

Random numbers.

```
#include <hnco/random.hh>
```

Static Public Member Functions

- static double uniform ()

  Next uniformly distributed sample.
- static double normal ()

  Next normally distributed sample.
- static bool random_bit ()

  Next random bit.

Static Public Attributes

- static std::mt19937 engine

  Engine.

5.69.1 Detailed Description

Random numbers.

Definition at line 33 of file random.hh.

The documentation for this struct was generated from the following files:

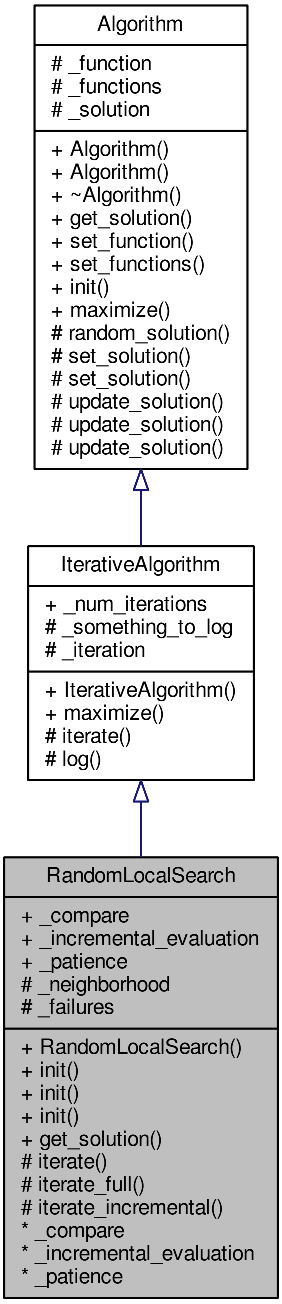
- lib/hnco/random.hh
- lib/hnco/random.cc

5.70 RandomLocalSearch Class Reference

Random local search.

```
#include <hnco/algorithms/ls/random-local-search.hh>
```

Inheritance diagram for RandomLocalSearch:



Public Member Functions

- RandomLocalSearch (int n, neighborhood::Neighborhood ∗neighborhood)

  Constructor.
- void init ()

  Random initialization.
- void init (const bit_vector_t &x)

Explicit initialization.

- void init (const bit_vector_t &x, double value)

  Explicit initialization.
- const point_value_t & get_solution ()

  Solution.

Public Attributes

Parameters

- std::function< bool(double, double)> _compare = std::greater_equal<double>()

  Binary operator for comparing evaluations.
- bool _incremental_evaluation = false

  Incremental evaluation.
- int _patience = 50

  Patience.

Protected Member Functions

- void iterate ()

  Single iteration.
- void iterate_full ()

  Single iteration with full evaluation.
- void iterate_incremental ()

  Single iteration with incremental evaluation.

Protected Attributes

- neighborhood::Neighborhood ∗ _neighborhood

  Neighborhood.
- int _failures

  Number of failure.

5.70.1 Detailed Description

Random local search.

Definition at line 39 of file random-local-search.hh.

5.70.2 Member Data Documentation

5.70.2.1 _patience

```
int _patience = 50
```

Patience.

Number of consecutive rejected moves before throwing a LocalMaximum exception

If patience <= 0 then patience is considered infinite, meaning that the algorithm will never throw any LocalMaximum exception.

Definition at line 97 of file random-local-search.hh.

The documentation for this class was generated from the following files:

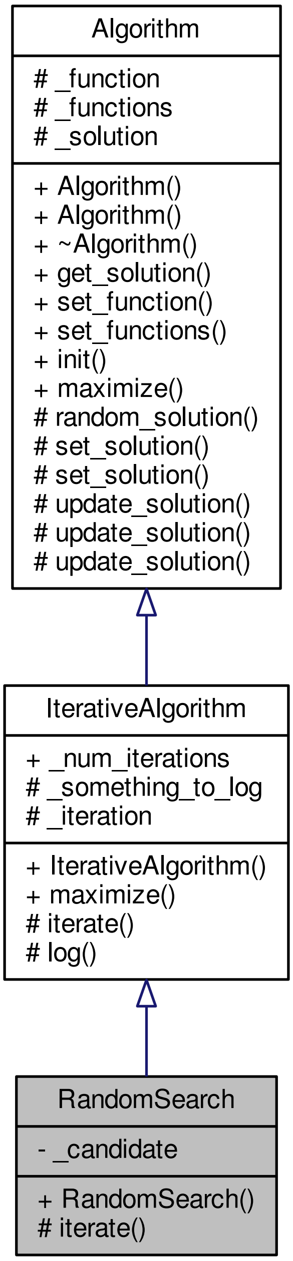
- lib/hnco/algorithms/ls/random-local-search.hh
- lib/hnco/algorithms/ls/random-local-search.cc

5.71 RandomSearch Class Reference

Random search.

```
#include <hnco/algorithms/random-search.hh>
```

Inheritance diagram for RandomSearch:



Public Member Functions

- RandomSearch (int n)

  Constructor.

Protected Member Functions

- void iterate ()

  Single iteration.

Private Attributes

- bit_vector_t _candidate

  Candidate.

Additional Inherited Members

5.71.1 Detailed Description

Random search.

Definition at line 30 of file random-search.hh.

The documentation for this class was generated from the following files:

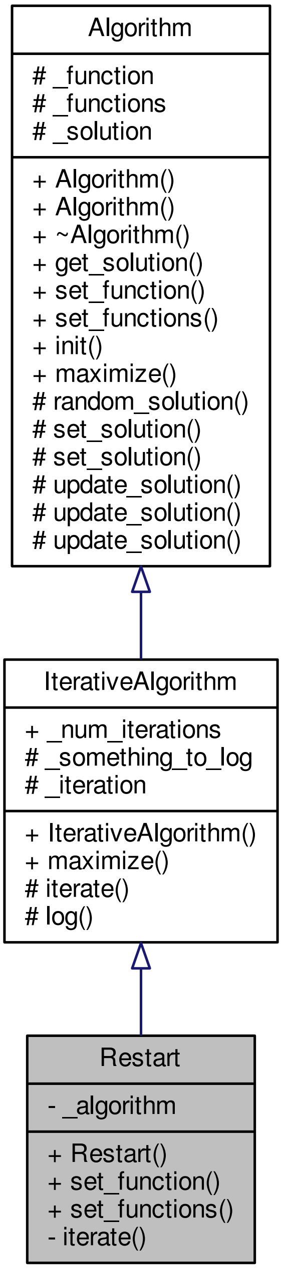
- lib/hnco/algorithms/random-search.hh
- lib/hnco/algorithms/random-search.cc

5.72 Restart Class Reference

Restart.

```
#include <hnco/algorithms/restart.hh>
```

Inheritance diagram for Restart:



Public Member Functions

- Restart (int n, Algorithm $*$algorithm)

  Constructor.
- void set_function (function::Function $*$function)

  Set function.
- void set_functions (const std::vector$<$ function::Function $*>$ functions)

  Set functions.

Private Member Functions

- void iterate ()

  Optimize.

Private Attributes

- Algorithm ∗ _algorithm

  Algorithm.

Additional Inherited Members

5.72.1 Detailed Description

Restart.

Restart an Algorithm an indefinite number of times. Should be used in conjonction with OnBudgetFunction or StopOnMaximum.

Definition at line 38 of file restart.hh.

The documentation for this class was generated from the following files:

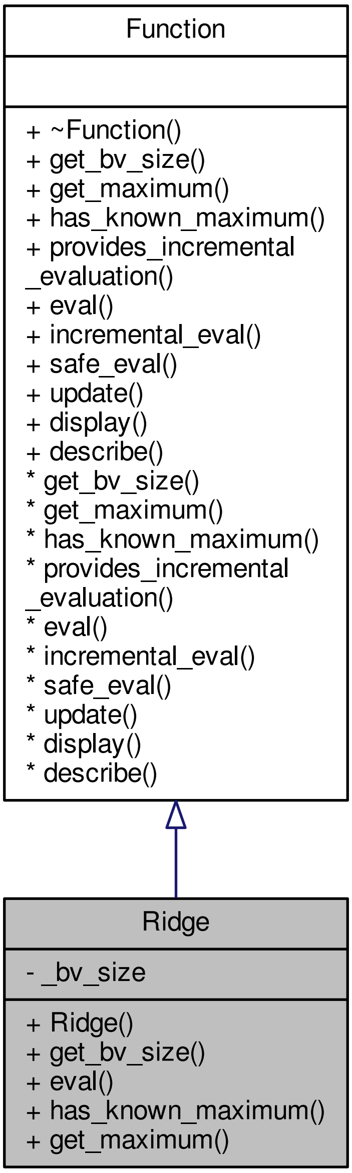
- lib/hnco/algorithms/restart.hh
- lib/hnco/algorithms/restart.cc

5.73 Ridge Class Reference

Ridge.

```
#include <hnco/functions/theory.hh>
```

Inheritance diagram for Ridge:



Public Member Functions

- Ridge (int bv_size)

  Constructor.
- size_t get_bv_size ()

  Get bit vector size.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.
- bool has_known_maximum ()

Check for a known maximum.

- double get_maximum ()

  Get the global maximum.

Private Attributes

- size_t _bv_size

  Bit vector size.

5.73.1 Detailed Description

Ridge.

Definition at line 167 of file theory.hh.

5.73.2 Member Function Documentation

5.73.2.1 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true

Reimplemented from Function.

Definition at line 187 of file theory.hh.

The documentation for this class was generated from the following files:

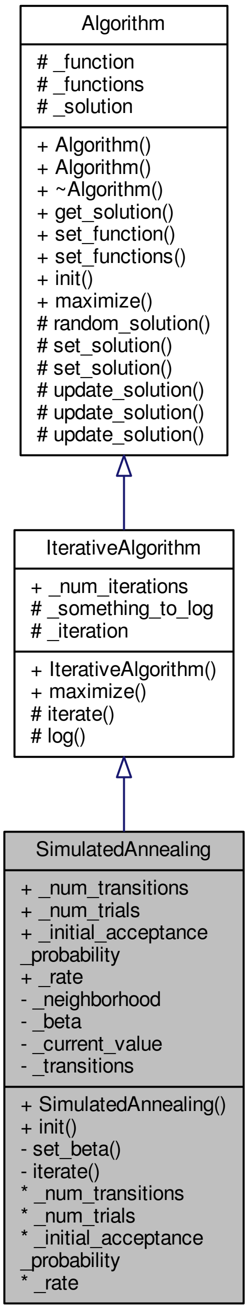
- lib/hnco/functions/theory.hh
- lib/hnco/functions/theory.cc

5.74 SimulatedAnnealing Class Reference

Simulated annealing.

```
#include <hnco/algorithms/ls/simulated-annealing.hh>
```

Inheritance diagram for SimulatedAnnealing:



Public Member Functions

- SimulatedAnnealing (int n, neighborhood::Neighborhood ∗neighborhood)

  Constructor.
- void init ()

  Initialization.

Public Attributes

Parameters

- int _num_transitions = 50

  Number of accepted transitions before annealing.
- int _num_trials = 100

  Number of trials.
- double _initial_acceptance_probability = 0.6

  Initial acceptance probability.
- double _rate = 1.2

  Increase rate for inverse temperature.

Private Member Functions

- void set_beta ()

  Set beta.
- void iterate ()

  Single iteration.

Private Attributes

- neighborhood::Neighborhood ∗ _neighborhood

  Neighborhood.
- double _beta

  Inverse temperature.
- double _current_value

  Current value.
- int _transitions

  Number of accepted transitions.

Additional Inherited Members

5.74.1 Detailed Description

Simulated annealing.

Definition at line 36 of file simulated-annealing.hh.

5.74.2 Member Function Documentation

5.74.2.1 set_beta()

```
void set_beta ( )  [private]
```

Set beta.

Requires (2 ∗ num_trials) evaluations. This should be taken into account when using OnBudgetFunction.

Definition at line 34 of file simulated-annealing.cc.

The documentation for this class was generated from the following files:

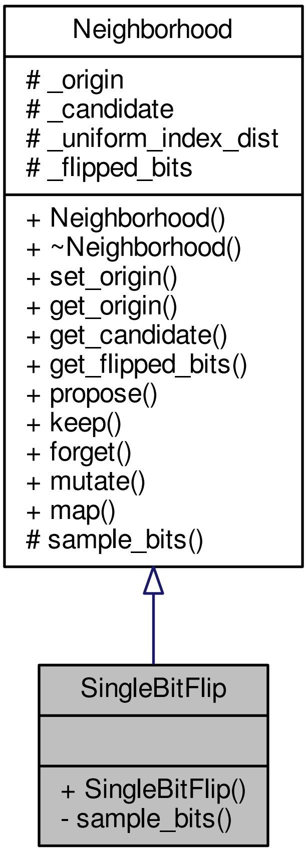
- lib/hnco/algorithms/ls/simulated-annealing.hh
- lib/hnco/algorithms/ls/simulated-annealing.cc

5.75 SingleBitFlip Class Reference

One bit neighborhood.

```
#include <hnco/neighborhoods/neighborhood.hh>
```

Inheritance diagram for SingleBitFlip:



Public Member Functions

- SingleBitFlip (int n)

  Constructor.

Private Member Functions

- void sample_bits ()

  Sample bits.

Additional Inherited Members

5.75.1 Detailed Description

One bit neighborhood.

Definition at line 160 of file neighborhood.hh.

The documentation for this class was generated from the following file:

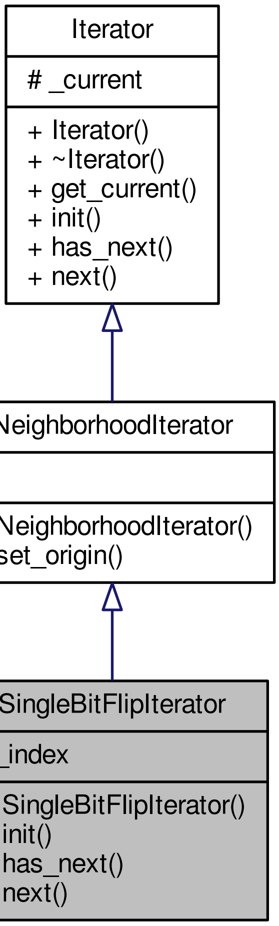
- lib/hnco/neighborhoods/neighborhood.hh

5.76 SingleBitFlipIterator Class Reference

Single bit flip neighborhood iterator.

```
#include <hnco/neighborhoods/neighborhood-iterator.hh>
```

Inheritance diagram for SingleBitFlipIterator:



Public Member Functions

- SingleBitFlipIterator (int n)

  Constructor.
- void init ()

  Initialization.
- bool has_next ()

  Has next bit vector.
- void next ()

  Next bit vector.

Private Attributes

- size_t _index

  Index of the last flipped bit.

Additional Inherited Members

5.76.1 Detailed Description

Single bit flip neighborhood iterator.

Definition at line 58 of file neighborhood-iterator.hh.

5.76.2 Constructor & Destructor Documentation

5.76.2.1 SingleBitFlipIterator()

```
SingleBitFlipIterator (
             int n )  [inline]
```

Constructor.

Parameters

n	Size of bit vectors

Definition at line 70 of file neighborhood-iterator.hh.

The documentation for this class was generated from the following files:

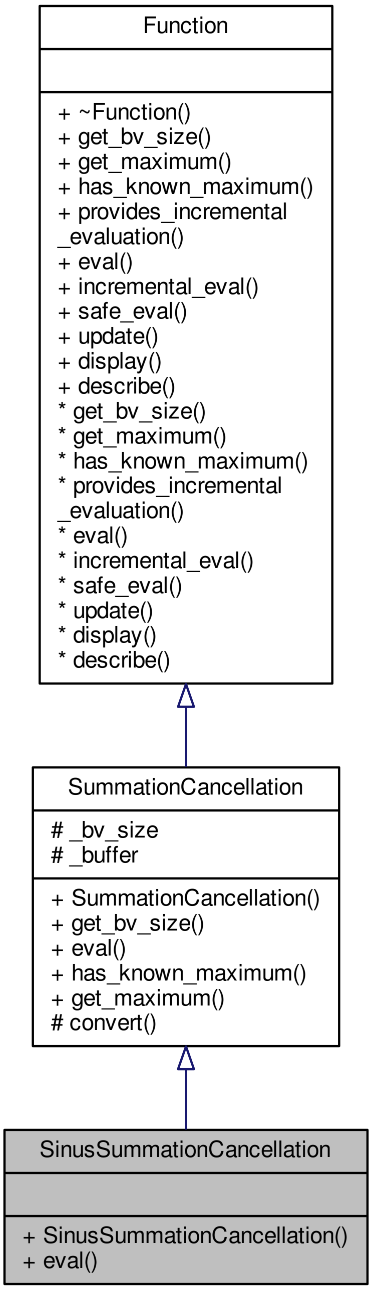
- lib/hnco/neighborhoods/neighborhood-iterator.hh
- lib/hnco/neighborhoods/neighborhood-iterator.cc

5.77 SinusSummationCancellation Class Reference

Summation cancellation with sinus.

```
#include <hnco/functions/cancellation.hh>
```

Inheritance diagram for SinusSummationCancellation:



Public Member Functions

- SinusSummationCancellation (int n)

  Constructor.
- double eval (const bit_vector_t &x)

  Evaluate a bit vector.

Additional Inherited Members

5.77.1 Detailed Description

Summation cancellation with sinus.

Definition at line 87 of file cancellation.hh.

The documentation for this class was generated from the following files:

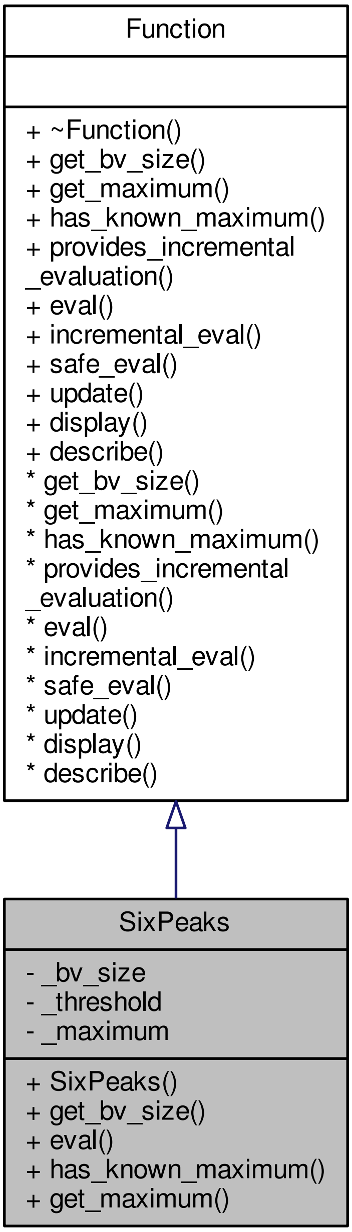
- lib/hnco/functions/cancellation.hh
- lib/hnco/functions/cancellation.cc

5.78 SixPeaks Class Reference

Six Peaks.

```
#include <hnco/functions/four-peaks.hh>
```

Inheritance diagram for SixPeaks:



Public Member Functions

- SixPeaks (int bv_size, int threshold)

    Constructor.
- size_t get_bv_size ()

    Get bit vector size.
- double eval (const bit_vector_t &)

Evaluate a bit vector.

- bool has_known_maximum ()

  Check for a known maximum.
- double get_maximum ()

  Get the global maximum.

Private Attributes

- size_t _bv_size

  Bit vector size.
- int _threshold

  Threshold.
- int _maximum

  Maximum.

5.78.1 Detailed Description

Six Peaks.

It is defined by

f(x) = max{head(x, 0) + tail(x, 1) + head(x, 1) + tail(x, 0)} + R(x)

where:

- head(x, 0) is the length of the longest prefix of x made of zeros;
- head(x, 1) is the length of the longest prefix of x made of ones;
- tail(x, 0) is the length of the longest suffix of x made of zeros;
- tail(x, 1) is the length of the longest suffix of x made of ones;
- R(x) is the reward;
- R(x) = n if (head(x, 0) $>$ t and tail(x, 1) $>$ t) or (head(x, 1) $>$ t and tail(x, 0) $>$ t);
- R(x) = 0 otherwise;
- the threshold t is a parameter of the function.

This function has six maxima, of which exactly four are global ones.

For example, if n = 6 and t = 1:

- f(111111) = 6 (local maximum)
- f(111110) = 5
- f(111100) = 10 (global maximum)

De Bonet, J. S., Isbell, C. L., & Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. Advances in neural information processing systems, 424-430.

Definition at line 120 of file four-peaks.hh.

5.78.2 Member Function Documentation

5.78.2.1 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true

Reimplemented from Function.

Definition at line 147 of file four-peaks.hh.

The documentation for this class was generated from the following files:

- lib/hnco/functions/four-peaks.hh
- lib/hnco/functions/four-peaks.cc

5.79 SpinHerding Class Reference

Herding with spin variables.

```
#include <hnco/algorithms/hea/herding-spin.hh>
```

Public Types

- enum {
  SAMPLE_GREEDY, SAMPLE_RLS, SAMPLE_DLS, SAMPLE_NN,
  LAST_SAMPLE }

Public Member Functions

- SpinHerding (int n)

  Constructor.
- void init ()

  Initialization.
- double error (const SpinMoment &target)

  Compute the error.
- double delta (const SpinMoment &target)

  Compute the norm of the moment increment.
- void sample (const SpinMoment &target, bit_vector_t &x)

  Sample a bit vector.

Public Attributes

Parameters

- bool _randomize_bit_order = false

  Randomize bit order.
- int _sampling_method = SAMPLE_GREEDY

  Sampling method.
- int _num_seq_updates

  Number of sequential updates per sample.
- int _num_par_updates = 1

  Number of parallel updates per sample.
- double _weight = 1

  Weight of second order moments.

Protected Member Functions

- void compute_delta (const SpinMoment &target)

  Compute delta.
- void sample_greedy (bit_vector_t &x)

  Sample by means of a greedy algorithm.
- double q_derivative (const bit_vector_t &x, size_t i)

  Derivative of q.
- double q_variation (const bit_vector_t &x, size_t i)

  Variation of q.
- void sample_rls (bit_vector_t &x)

  Sample by means of random local search.
- void sample_dls (bit_vector_t &x)

  Sample by means of deterministic local search.
- void sample_nn (bit_vector_t &x)

  Sample by means of a neural network.
- void update_counters (const bit_vector_t &x)

  Update counters.

Protected Attributes

- SpinMoment _delta

  Delta moment.
- SpinMoment _count

  Counter moment.
- bit_vector_t _state

  State.
- permutation_t _permutation

  Permutation.
- std::uniform_int_distribution< int > _choose_bit

  Choose bit.
- int _time

  Time.

5.79.1 Detailed Description

Herding with spin variables.

By spin variables, we mean variables taking values 1 or -1, instead of 0 or 1 in the case of binary variables.

Definition at line 37 of file herding-spin.hh.

5.79.2 Member Enumeration Documentation

5.79.2.1 anonymous enum

```
anonymous enum
```

Enumerator

SAMPLE_GREEDY	Greedy algorithm.
SAMPLE_RLS	Random local search.
SAMPLE_DLS	Deterministic local search.
SAMPLE_NN	Neural network.

Definition at line 90 of file herding-spin.hh.

5.79.3 Member Function Documentation

5.79.3.1 q_variation()

```
double q_variation (
             const bit_vector_t & x,
             size_t i )  [protected]
```

Variation of q.

Up to a positive multiplicative constant. Only the sign of the variation matters to local search.

Definition at line 155 of file herding-spin.cc.

The documentation for this class was generated from the following files:

- lib/hnco/algorithms/hea/herding-spin.hh
- lib/hnco/algorithms/hea/herding-spin.cc

5.80 SpinMoment Struct Reference

Moment for spin variables.

```
#include <hnco/algorithms/hea/moment-spin.hh>
```

Public Member Functions

- SpinMoment (int n)

  Constructor.
- void uniform ()

  Set the moment to that of the uniform distribution.
- void init ()

  Initialize accumulators.
- void add (const bit_vector_t &x)

  Update accumulators.
- void average (int count)

  Compute average.
- void update (const SpinMoment &p, double rate)

  Update moment.
- void bound (double margin)

  Bound moment.
- double distance (const SpinMoment &p) const

  Distance.
- double norm_2 () const

  Compute the norm 2.
- double diameter () const

  Compute the diameter.
- size_t size () const

  Size.

Public Attributes

- std::vector< double > _first

  First moment.
- std::vector< std::vector< double > > _second

  Second moment.
- double _weight = 1

  Weight of second order moments.

5.80.1 Detailed Description

Moment for spin variables.

Definition at line 35 of file moment-spin.hh.

The documentation for this struct was generated from the following files:

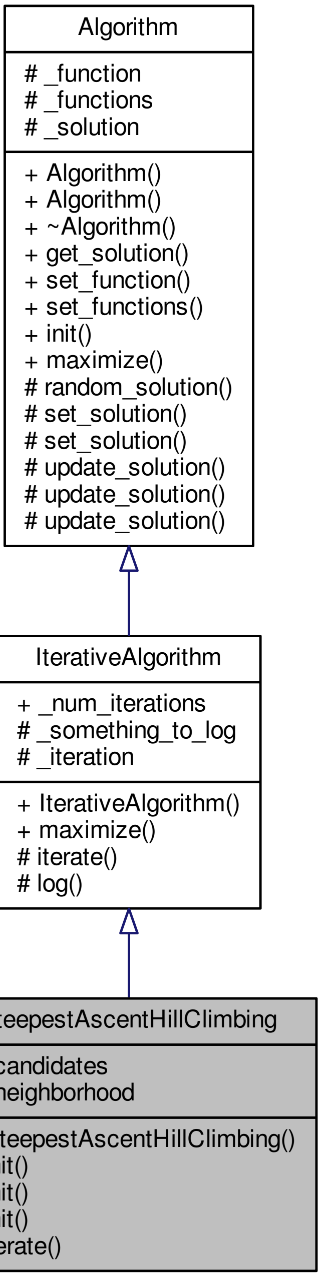
- lib/hnco/algorithms/hea/moment-spin.hh
- lib/hnco/algorithms/hea/moment-spin.cc

5.81 SteepestAscentHillClimbing Class Reference

Steepest ascent hill climbing.

```
#include <hnco/algorithms/ls/steepest-ascent-hill-climbing.hh>
```

Inheritance diagram for SteepestAscentHillClimbing:



Public Member Functions

- SteepestAscentHillClimbing (int n, neighborhood::NeighborhoodIterator ∗neighborhood)

  Constructor.
- void init ()

  Random initialization.
- void init (const bit_vector_t &x)

  Explicit initialization.
- void init (const bit_vector_t &x, double value)

  Explicit initialization.

Protected Member Functions

- void iterate ()

  Single iteration.

Protected Attributes

- std::vector< bit_vector_t > _candidates

  Potential candidate.
- neighborhood::NeighborhoodIterator ∗ _neighborhood

  Neighborhood.

Additional Inherited Members

5.81.1 Detailed Description

Steepest ascent hill climbing.

Definition at line 39 of file steepest-ascent-hill-climbing.hh.

The documentation for this class was generated from the following files:

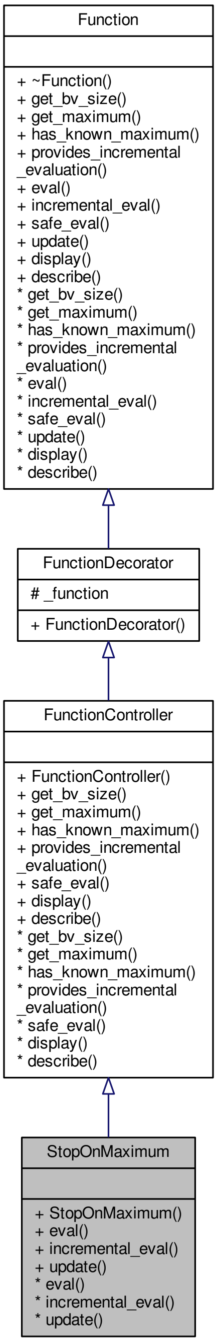
- lib/hnco/algorithms/ls/steepest-ascent-hill-climbing.hh
- lib/hnco/algorithms/ls/steepest-ascent-hill-climbing.cc

5.82 StopOnMaximum Class Reference

Stop on maximum.

```
#include <hnco/functions/decorators/function-controller.hh>
```

Inheritance diagram for StopOnMaximum:



Public Member Functions

- StopOnMaximum (Function ∗function)

  Constructor.

Evaluation

- double eval (const bit_vector_t &)

  Evaluate a bit vector.
- double incremental_eval (const bit_vector_t &x, double value, const hnco::sparse_bit_vector_t &flipped↩ _bits)

  Incremental evaluation.
- void update (const bit_vector_t &x, double value)

  Update after a safe evaluation.

Additional Inherited Members

5.82.1 Detailed Description

Stop on maximum.

The eval() member function throws a MaximumReached exception when its argument maximizes the decorated function.

Definition at line 98 of file function-controller.hh.

5.82.2 Constructor & Destructor Documentation

5.82.2.1 StopOnMaximum()

```
StopOnMaximum (
             Function ∗ function )   [inline]
```

Constructor.

Parameters

function	Decorated function

Precondition

function->has_known_maximum()

Definition at line 106 of file function-controller.hh.

5.82.3 Member Function Documentation

5.82.3.1 eval()

```
double eval (
             const bit_vector_t & x )  [virtual]
```

Evaluate a bit vector.

Exceptions

MaximumReached	

Implements Function.

Definition at line 31 of file function-controller.cc.

5.82.3.2 incremental_eval()

```
double incremental_eval (
             const bit_vector_t & x,
             double value,
             const hnco::sparse_bit_vector_t & flipped_bits )  [virtual]
```

Incremental evaluation.

Exceptions

MaximumReached	

Reimplemented from Function.

Definition at line 43 of file function-controller.cc.

5.82.3.3 update()

```
void update (
             const bit_vector_t & x,
             double value )  [virtual]
```

Update after a safe evaluation.

Exceptions

MaximumReached	

Reimplemented from Function.

Definition at line 55 of file function-controller.cc.

The documentation for this class was generated from the following files:

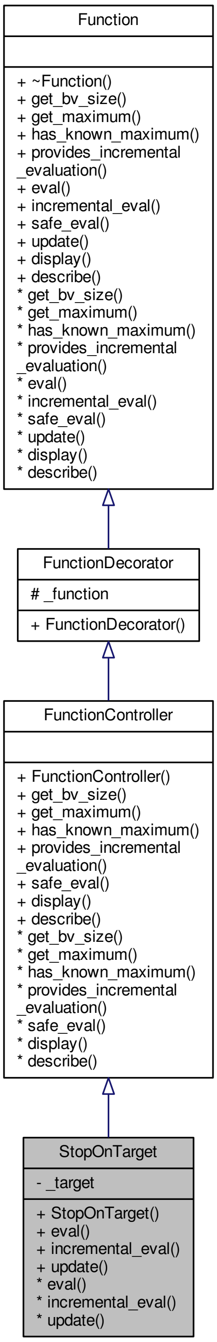
- lib/hnco/functions/decorators/function-controller.hh
- lib/hnco/functions/decorators/function-controller.cc

5.83 StopOnTarget Class Reference

Stop on target.

```
#include <hnco/functions/decorators/function-controller.hh>
```

Inheritance diagram for StopOnTarget:



Public Member Functions

- StopOnTarget (Function ∗function, double target)

  Constructor.

Evaluation

- double eval (const bit_vector_t &)

  Evaluate a bit vector.
- double incremental_eval (const bit_vector_t &x, double value, const hnco::sparse_bit_vector_t &flipped↩_bits)

  Incremental evaluation.
- void update (const bit_vector_t &x, double value)

  Update after a safe evaluation.

Private Attributes

- double _target

  Target.

Additional Inherited Members

5.83.1 Detailed Description

Stop on target.

Definition at line 134 of file function-controller.hh.

5.83.2 Constructor & Destructor Documentation

5.83.2.1 StopOnTarget()

```
StopOnTarget (
            Function * function,
            double target )  [inline]
```

Constructor.

Parameters

function	Decorated function

Definition at line 144 of file function-controller.hh.

5.83.3 Member Function Documentation

5.83.3.1 eval()

```
double eval (
            const bit_vector_t & x )  [virtual]
```

Evaluate a bit vector.

Exceptions

TargetReached	

Implements Function.

Definition at line 66 of file function-controller.cc.

5.83.3.2 incremental_eval()

```
double incremental_eval (
            const bit_vector_t & x,
            double value,
            const hnco::sparse_bit_vector_t & flipped_bits )  [virtual]
```

Incremental evaluation.

Exceptions

TargetReached	

Reimplemented from Function.

Definition at line 76 of file function-controller.cc.

5.83.3.3 update()

```
void update (
            const bit_vector_t & x,
            double value )  [virtual]
```

Update after a safe evaluation.

Exceptions

TargetReached	

Reimplemented from Function.

Definition at line 86 of file function-controller.cc.

The documentation for this class was generated from the following files:

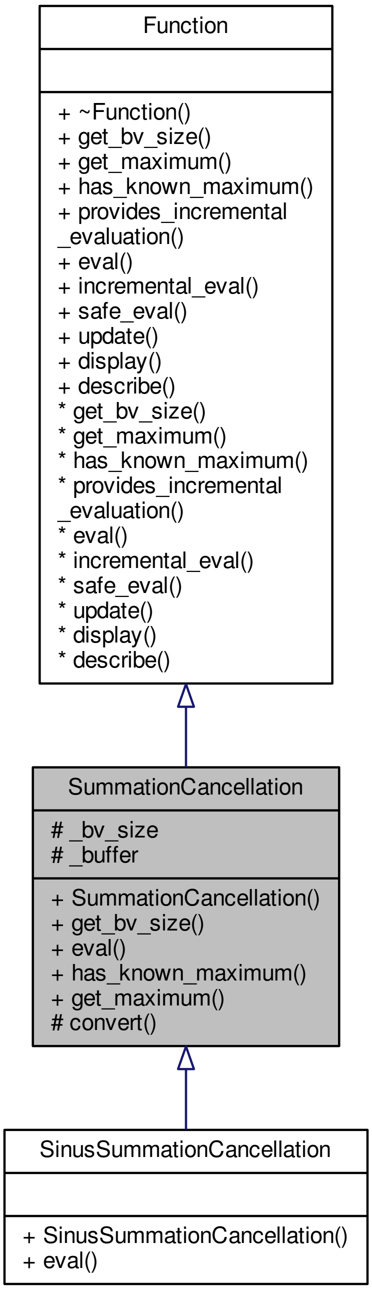
- lib/hnco/functions/decorators/function-controller.hh
- lib/hnco/functions/decorators/function-controller.cc

5.84 SummationCancellation Class Reference

Summation cancellation.

```
#include <hnco/functions/cancellation.hh>
```

Inheritance diagram for SummationCancellation:



Public Member Functions

- SummationCancellation (int n)

  Constructor.
- size_t get_bv_size ()

  Get bit vector size.
- double eval (const bit_vector_t &x)

Evaluate a bit vector.

- bool has_known_maximum ()

  Check for a known maximum.
- double get_maximum ()

  Get the global maximum.

Protected Member Functions

- void convert (const bit_vector_t &x)

  Convert a bit vector into a real vector.

Protected Attributes

- size_t _bv_size

  Bit vector size.
- std::vector< double > _buffer

  Buffer.

5.84.1 Detailed Description

Summation cancellation.

Encoding of a signed integer:

- bit 0: sign
- bits 1 to 8: two's complement representation

Definition at line 39 of file cancellation.hh.

5.84.2 Constructor & Destructor Documentation

5.84.2.1 SummationCancellation()

```
SummationCancellation (
              int n )  [inline]
```

Constructor.

The bit vector size n must be a multiple of 9. The size of _buffer is then n / 9.

Parameters

n	Size of the bit vector

Definition at line 62 of file cancellation.hh.

5.84.3 Member Function Documentation

5.84.3.1 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true

Reimplemented from Function.

Definition at line 78 of file cancellation.hh.

The documentation for this class was generated from the following files:

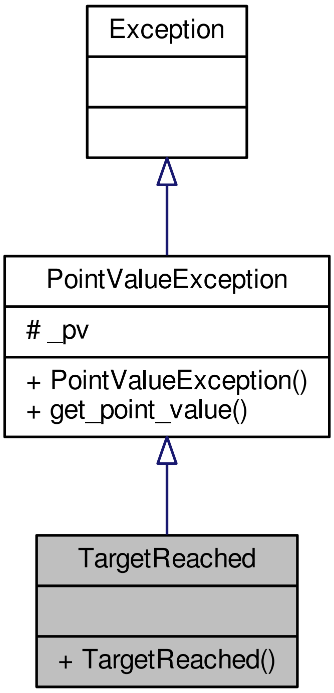
- lib/hnco/functions/cancellation.hh
- lib/hnco/functions/cancellation.cc

5.85 TargetReached Class Reference

target reached

```
#include <hnco/exception.hh>
```

Inheritance diagram for TargetReached:



Public Member Functions

- TargetReached (const point_value_t &pv)

  Constructor.

Additional Inherited Members

5.85.1 Detailed Description

target reached

Definition at line 61 of file exception.hh.

The documentation for this class was generated from the following file:

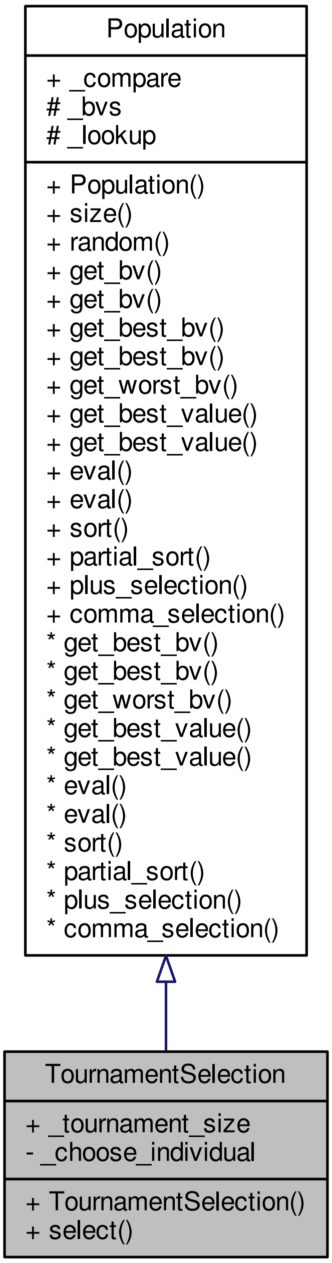
- lib/hnco/exception.hh

5.86 TournamentSelection Class Reference

Population with tournament selection.

```
#include <hnco/algorithms/ea/tournament-selection.hh>
```

Inheritance diagram for TournamentSelection:



Public Member Functions

- TournamentSelection (int population_size, int n)

  Constructor.
- const bit_vector_t & select ()

  Selection.

Public Attributes

- int _tournament_size = 10

  Tournament size.

Private Attributes

- std::uniform_int_distribution< int > _choose_individual

  Random index.

Additional Inherited Members

5.86.1 Detailed Description

Population with tournament selection.

Definition at line 34 of file tournament-selection.hh.

5.86.2 Member Function Documentation

5.86.2.1 select()

```
const bit_vector_t & select ( )
```

Selection.

The selection only requires that the population be evaluated, not necessarily sorted.

Precondition

The population must be evaluated.

Definition at line 33 of file tournament-selection.cc.

The documentation for this class was generated from the following files:

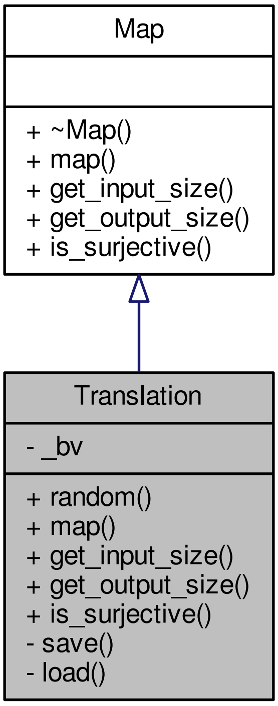
- lib/hnco/algorithms/ea/tournament-selection.hh
- lib/hnco/algorithms/ea/tournament-selection.cc

5.87 Translation Class Reference

Translation.

```
#include <hnco/map.hh>
```

Inheritance diagram for Translation:



Public Member Functions

- void random (int n)

  Random instance.
- void map (const bit_vector_t &input, bit_vector_t &output)

  Map.
- size_t get_input_size ()

  Get input size.
- size_t get_output_size ()

  Get output size.
- bool is_surjective ()

  Check for surjective map.

Private Member Functions

- template<class Archive >
  void save (Archive &ar, const unsigned int version) const

  Save.
- template<class Archive >
  void load (Archive &ar, const unsigned int version)

  Load.

Private Attributes

- bit_vector_t _bv

  Translation vector.

Friends

- class **boost::serialization::access**

5.87.1 Detailed Description

Translation.

A translation is an affine map f from Z_2^n to itself defined by $f(x) = x + b$, where b is an n-dimensional bit vector.

Definition at line 70 of file map.hh.

5.87.2 Member Function Documentation

5.87.2.1 is_surjective()

```
bool is_surjective ( )  [inline], [virtual]
```

Check for surjective map.

Returns

true

Reimplemented from Map.

Definition at line 121 of file map.hh.

The documentation for this class was generated from the following files:

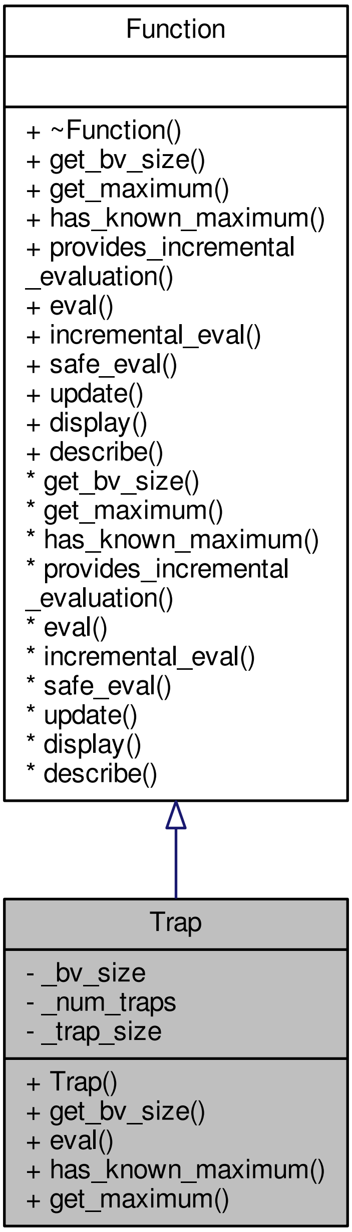
- lib/hnco/map.hh
- lib/hnco/map.cc

5.88 Trap Class Reference

Trap.

`#include <hnco/functions/trap.hh>`

Inheritance diagram for Trap:



Public Member Functions

- Trap (int bv_size, int num_traps)

Constructor.

- size_t get_bv_size ()

  Get bit vector size.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.
- bool has_known_maximum ()

  Check for a known maximum.
- double get_maximum ()

  Get the global maximum.

Private Attributes

- size_t _bv_size

  Bit vector size.
- int _num_traps

  Number of traps.
- int _trap_size

  Trap size.

5.88.1 Detailed Description

Trap.

Definition at line 34 of file trap.hh.

5.88.2 Constructor & Destructor Documentation

5.88.2.1 Trap()

```
Trap (
            int bv_size,
            int num_traps )  [inline]
```

Constructor.

Parameters

bv_size	Bit vector size
num_traps	Number of traps

Warning

> bv_size must be a multiple of num_traps

Definition at line 55 of file trap.hh.

5.88.3 Member Function Documentation

5.88.3.1 has_known_maximum()

```
bool has_known_maximum ( )  [inline], [virtual]
```

Check for a known maximum.

Returns

true

Reimplemented from Function.

Definition at line 75 of file trap.hh.

The documentation for this class was generated from the following files:

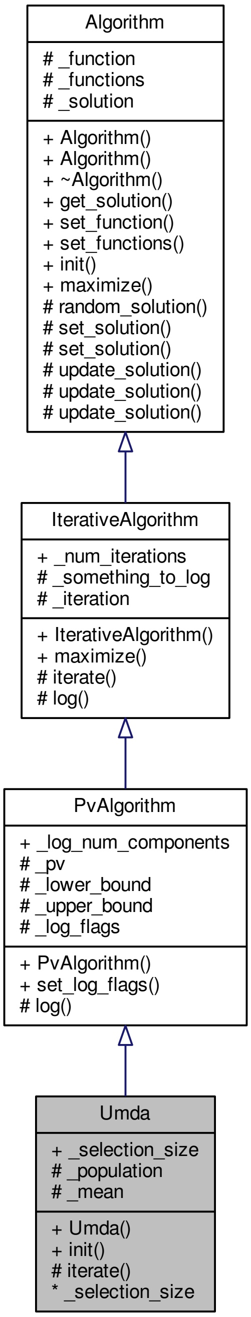
- lib/hnco/functions/trap.hh
- lib/hnco/functions/trap.cc

5.89 Umda Class Reference

Univariate marginal distribution algorithm.

```
#include <hnco/algorithms/pv/umda.hh>
```

Inheritance diagram for Umda:



Public Member Functions

- Umda (int n, int population_size)

  Constructor.
- void init ()

  Initialization.

Public Attributes

Parameters

- int _selection_size = 1

  Selection size.

Protected Member Functions

- void iterate ()

  Single iteration.

Protected Attributes

- Population _population

  Population.
- pv_t _mean

  Mean of selected bit vectors.

Additional Inherited Members

5.89.1 Detailed Description

Univariate marginal distribution algorithm.

Definition at line 32 of file umda.hh.

The documentation for this class was generated from the following files:

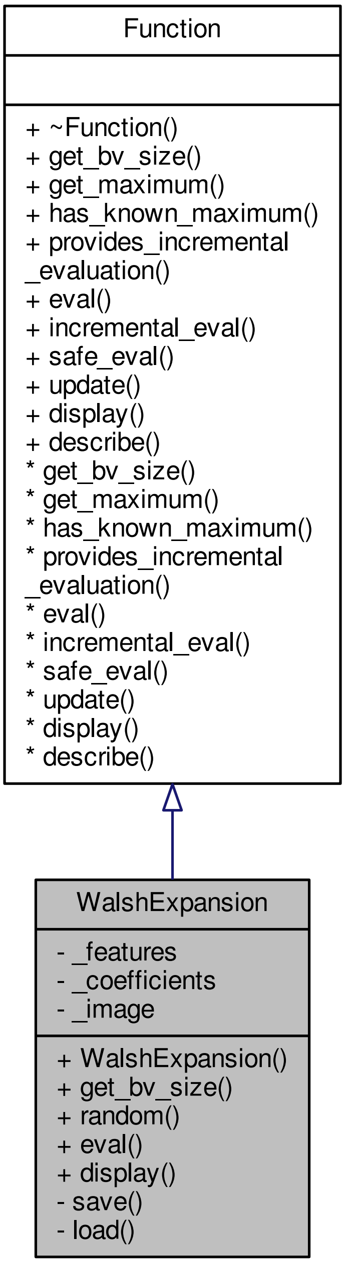
- lib/hnco/algorithms/pv/umda.hh
- lib/hnco/algorithms/pv/umda.cc

5.90 WalshExpansion Class Reference

Walsh expansion.

```
#include <hnco/functions/walsh/walsh-expansion.hh>
```

Inheritance diagram for WalshExpansion:



Public Member Functions

- WalshExpansion ()

  Constructor.
- size_t get_bv_size ()

  Get bit vector size.
- void random (int n, int num_features, double stddev)

Random instance.

- double eval (const bit_vector_t &)

  Evaluate a bit vector.
- void display (std::ostream &stream)

  Display.

Private Member Functions

- template<class Archive >
  void save (Archive &ar, const unsigned int version) const

  Save.
- template<class Archive >
  void load (Archive &ar, const unsigned int version)

  Load.

Private Attributes

- hnco::bit_matrix_t _features

  Features.
- std::vector< double > _coefficients

  Coefficients.
- bit_vector_t _image

  Image of bit vectors under the feature matrix.

Friends

- class **boost::serialization::access**

5.90.1 Detailed Description

Walsh expansion.

Its expression is of the form

$f(x) = \sum_u a_u (-1)^{x \cdot u}$

where the sum is over a subset of $\{0,1\}^n$ and $x \cdot u = \sum_i x_i u_i$ is mod 2. The real numbers a_u are the coefficients of the expansion and the bit vectors u are its feature vectors.

Definition at line 53 of file walsh-expansion.hh.

5.90.2 Member Function Documentation

5.90.2.1 random()

```
void random (
             int n,
             int num_features,
             double stddev )
```

Random instance.

Parameters

n	Size of bit vector
num_features	Number of feature vectors
stddev	Standard deviation of the coefficients

Definition at line 34 of file walsh-expansion.cc.

The documentation for this class was generated from the following files:

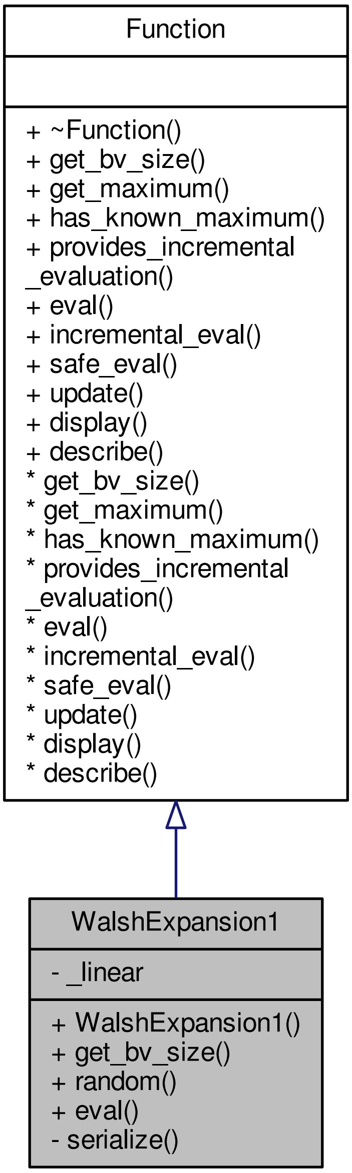
- lib/hnco/functions/walsh/walsh-expansion.hh
- lib/hnco/functions/walsh/walsh-expansion.cc

5.91 WalshExpansion1 Class Reference

Walsh expansion of degree 1.

```
#include <hnco/functions/walsh/walsh-expansion-1.hh>
```

Inheritance diagram for WalshExpansion1:



Public Member Functions

- WalshExpansion1 ()

  Constructor.
- size_t get_bv_size ()

  Get bit vector size.
- void random (int n, double stddev)

  Random instance.
- double eval (const bit_vector_t &)

  Evaluate a bit vector.

Private Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  Serialize.

Private Attributes

- std::vector< double > _linear

  Linear part.

Friends

- class **boost::serialization::access**

5.91.1 Detailed Description

Walsh expansion of degree 1.

Its expression is of the form

$f(x) = \sum_i a_i(1 - 2x_i)$

or equivalently

$f(x) = \sum_i a_i(-1)^{x_i}$

Definition at line 50 of file walsh-expansion-1.hh.

5.91.2 Member Function Documentation

5.91.2.1 random()

```
void random (
            int n,
            double stddev )
```

Random instance.

Parameters

n	Size of bit vector
stddev	Standard deviation of the coefficients

Definition at line 33 of file walsh-expansion-1.cc.

The documentation for this class was generated from the following files:

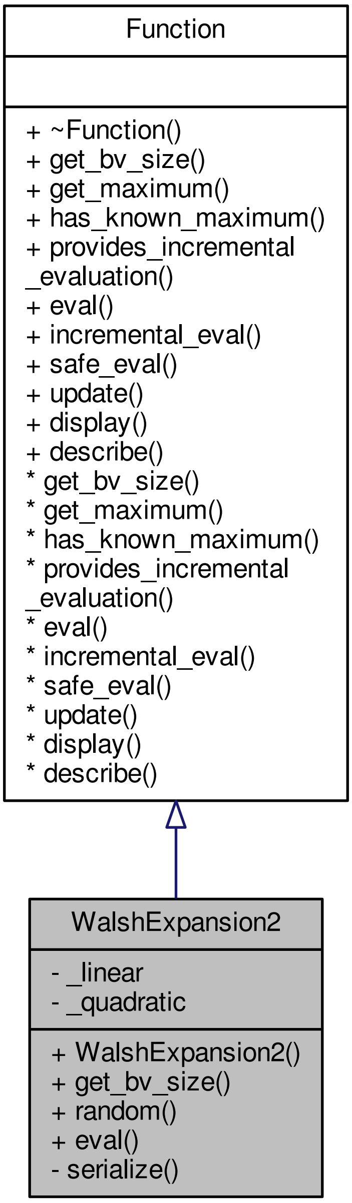
- lib/hnco/functions/walsh/walsh-expansion-1.hh
- lib/hnco/functions/walsh/walsh-expansion-1.cc

5.92 WalshExpansion2 Class Reference

Walsh expansion of degree 2.

```
#include <hnco/functions/walsh/walsh-expansion-2.hh>
```

Inheritance diagram for WalshExpansion2:



Public Member Functions

- WalshExpansion2 ()

    Constructor.
- size_t get_bv_size ()

    Get bit vector size.
- void random (int n, double stddev_lin, double stddev_quad)

    Random instance.
- double eval (const bit_vector_t &)

    Evaluate a bit vector.

Private Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

    Serialize.

Private Attributes

- std::vector< double > _linear

    Linear part.
- std::vector< std::vector< double > > _quadratic

    Quadratic part.

Friends

- class **boost::serialization::access**

5.92.1 Detailed Description

Walsh expansion of degree 2.

Its expression is of the form

$f(x) = \sum_i a_i(1-2x_i) + \sum_{i<j} a_{ij}(1-2x_i)(1-2x_j)$

or equivalently

$f(x) = \sum_i a_i(-1)^{x_i} + \sum_{i<j} a_{ij}(-1)^{x_i+x_j}$

where the sum $x_i + x_j$ is mod 2 (xor).

Definition at line 52 of file walsh-expansion-2.hh.

5.92.2 Member Function Documentation

5.92.2.1 random()

```
void random (
            int n,
            double stddev_lin,
            double stddev_quad )
```

Random instance.

Parameters

n	Size of bit vector
stddev_lin	Standard deviation of the coefficients of the linear part
stddev_quad	Standard deviation of the coefficients of the quadratic part

Definition at line 33 of file walsh-expansion-2.cc.

5.92.3 Member Data Documentation

5.92.3.1 _quadratic

```
std::vector<std::vector<double> > _quadratic  [private]
```

Quadratic part.

Represented as a lower triangular matrix (without its diagonal).

Definition at line 75 of file walsh-expansion-2.hh.

The documentation for this class was generated from the following files:

- lib/hnco/functions/walsh/walsh-expansion-2.hh
- lib/hnco/functions/walsh/walsh-expansion-2.cc

Index