

Optimal Partition

Generated by Doxygen 1.8.6

Mon Nov 9 2015 19:14:46

Contents

1	Main Page	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Class Documentation	9
5.1	AbstractSet Class Reference	9
5.1.1	Detailed Description	10
5.1.2	Constructor & Destructor Documentation	11
5.1.2.1	~AbstractSet	11
5.1.3	Member Function Documentation	11
5.1.3.1	computeOptimalPartition	11
5.1.3.2	getOptimalPartition	11
5.1.3.3	getOptimalPartitionList	11
5.1.3.4	printOptimalPartition	12
5.1.3.5	printOptimalPartitionList	13
5.1.3.6	printOptimalPartitionListInCSV	13
5.1.3.7	setObjectiveFunction	13
5.2	BiPart Class Reference	13
5.2.1	Detailed Description	14
5.3	BiSet Class Reference	14
5.3.1	Member Function Documentation	15
5.3.1.1	computeOptimalPartition	15
5.3.1.2	getOptimalPartition	15
5.3.1.3	printOptimalPartition	15
5.3.1.4	setObjectiveFunction	16
5.4	BiSubset Class Reference	16

5.5	BottleneckObjectiveValue Class Reference	17
5.6	CompleteGraph Class Reference	17
5.7	DataPointStruct Struct Reference	18
5.8	Dataset Class Reference	18
5.9	Datatree Class Reference	19
5.10	FiliformGraph Class Reference	20
5.11	Graph Class Reference	20
5.11.1	Member Function Documentation	22
5.11.1.1	computeOptimalPartition	22
5.11.1.2	getOptimalPartition	22
5.11.1.3	printOptimalPartition	22
5.11.1.4	setObjectiveFunction	22
5.12	GraphComponent Class Reference	22
5.13	HHNode Class Reference	23
5.14	HierarchicalHierarchicalSet Class Reference	24
5.14.1	Member Function Documentation	25
5.14.1.1	computeOptimalPartition	25
5.14.1.2	getOptimalPartition	25
5.14.1.3	printOptimalPartition	25
5.14.1.4	setObjectiveFunction	25
5.15	HierarchicalOrderedSet Class Reference	26
5.15.1	Member Function Documentation	26
5.15.1.1	computeOptimalPartition	26
5.15.1.2	getOptimalPartition	27
5.15.1.3	printOptimalPartition	27
5.15.1.4	setObjectiveFunction	27
5.16	HierarchicalSet Class Reference	27
5.16.1	Member Function Documentation	28
5.16.1.1	computeOptimalPartition	28
5.16.1.2	getOptimalPartition	28
5.16.1.3	printOptimalPartition	29
5.16.1.4	setObjectiveFunction	29
5.17	HierarchicalUniSet Class Reference	29
5.17.1	Detailed Description	30
5.17.2	Constructor & Destructor Documentation	30
5.17.2.1	HierarchicalUniSet	30
5.18	HNode Class Reference	30
5.19	HONode Class Reference	31
5.20	InformationBottleneck Class Reference	32
5.21	LogarithmicScore Class Reference	32

5.21.1 Detailed Description	33
5.21.2 Constructor & Destructor Documentation	33
5.21.2.1 LogarithmicScore	33
5.22 LogarithmicScoreValue Class Reference	33
5.23 MultiPart Class Reference	34
5.23.1 Detailed Description	34
5.24 MultiSet Class Reference	35
5.24.1 Detailed Description	36
5.24.2 Constructor & Destructor Documentation	36
5.24.2.1 MultiSet	36
5.24.2.2 MultiSet	36
5.24.2.3 ~MultiSet	36
5.24.3 Member Function Documentation	37
5.24.3.1 computeOptimalPartition	37
5.24.3.2 getAtomicMultiSubset	38
5.24.3.3 getOptimalPartition	38
5.24.3.4 getRandomAtomicMultiSubset	38
5.24.3.5 printOptimalPartition	38
5.24.3.6 setObjectiveFunction	38
5.25 MultiSubset Class Reference	39
5.26 NHONode Class Reference	39
5.27 NHOSet Class Reference	40
5.27.1 Member Function Documentation	41
5.27.1.1 computeOptimalPartition	41
5.27.1.2 getOptimalPartition	41
5.27.1.3 printOptimalPartition	41
5.27.1.4 setObjectiveFunction	42
5.28 NonconstrainedOrderedSet Class Reference	43
5.28.1 Member Function Documentation	44
5.28.1.1 computeOptimalPartition	44
5.28.1.2 getOptimalPartition	44
5.28.1.3 printOptimalPartition	44
5.28.1.4 setObjectiveFunction	44
5.29 NonconstrainedSet Class Reference	45
5.29.1 Member Function Documentation	46
5.29.1.1 computeOptimalPartition	46
5.29.1.2 getOptimalPartition	46
5.29.1.3 printOptimalPartition	46
5.29.1.4 setObjectiveFunction	46
5.30 ObjectiveFunction Class Reference	47

5.30.1 Detailed Description	47
5.30.2 Constructor & Destructor Documentation	47
5.30.2.1 ObjectiveFunction	47
5.31 ObjectiveValue Class Reference	48
5.32 OrderedDatatree Class Reference	48
5.33 OrderedSet Class Reference	49
5.33.1 Member Function Documentation	50
5.33.1.1 computeOptimalPartition	50
5.33.1.2 getOptimalPartition	50
5.33.1.3 printOptimalPartition	51
5.33.1.4 setObjectiveFunction	51
5.34 OrderedUniSet Class Reference	51
5.34.1 Detailed Description	51
5.34.2 Constructor & Destructor Documentation	52
5.34.2.1 OrderedUniSet	52
5.35 Part Class Reference	52
5.35.1 Detailed Description	52
5.36 Partition Class Reference	53
5.36.1 Detailed Description	53
5.37 PredictionDataset Class Reference	53
5.37.1 Detailed Description	54
5.37.2 Constructor & Destructor Documentation	54
5.37.2.1 PredictionDataset	54
5.37.3 Member Function Documentation	54
5.37.3.1 addTestValue	54
5.37.3.2 addTestValue	55
5.37.3.3 addTrainValue	55
5.37.3.4 addTrainValue	55
5.38 RandomGraph Class Reference	55
5.39 RelativeEntropy Class Reference	56
5.40 RelativeObjectiveValue Class Reference	57
5.41 Ring Class Reference	57
5.41.1 Member Function Documentation	58
5.41.1.1 computeOptimalPartition	58
5.41.1.2 getOptimalPartition	59
5.41.1.3 printOptimalPartition	59
5.41.1.4 setObjectiveFunction	59
5.42 RingGraph Class Reference	59
5.43 Timer Class Reference	60
5.44 TreeToAdd Struct Reference	60

5.45	UniSet Class Reference	61
5.45.1	Detailed Description	62
5.45.2	Constructor & Destructor Documentation	62
5.45.2.1	UniSet	62
5.45.3	Member Function Documentation	62
5.45.3.1	initReached	62
5.46	UniSubset Class Reference	62
5.46.1	Detailed Description	63
5.46.2	Constructor & Destructor Documentation	63
5.46.2.1	UniSubset	63
6	File Documentation	65
6.1	/home/lamarche/programming/optimal_partition/src/abstract_set.hpp File Reference	65
6.1.1	Detailed Description	65
6.2	/home/lamarche/programming/optimal_partition/src/logarithmic_score.hpp File Reference	65
6.2.1	Detailed Description	66
6.3	/home/lamarche/programming/optimal_partition/src/multi_set.hpp File Reference	66
6.3.1	Detailed Description	66
6.4	/home/lamarche/programming/optimal_partition/src/objective_function.hpp File Reference	67
6.4.1	Detailed Description	67
6.5	/home/lamarche/programming/optimal_partition/src/prediction_dataset.hpp File Reference	67
6.5.1	Detailed Description	68
6.6	/home/lamarche/programming/optimal_partition/src/uni_set.hpp File Reference	68
6.6.1	Detailed Description	68
	Index	69

Chapter 1

Main Page

This program is a toolbox to solve special versions of the Set Partitioning Problem, that is the combinatorial optimisation of a decomposable objective over a set of feasible partitions (defined according to specific algebraic structures: e.g., hierarchies, sets of intervals, graphs). The objectives are mainly based on information theory, in the perspective of multilevel analysis of large-scale datasets, and the algorithms are based on dynamic programming. For details regarding the formal grounds of this work, please refer to:

Robin Lamarche-Perrin, Yves Demazeau and Jean-Marc Vincent. A Generic Set Partitioning Algorithm with Applications to Hierarchical and Ordered Sets. Technical Report 105/2014, Max-Planck-Institute for Mathematics in the Sciences, Leipzig, Germany, May 2014.

<http://www.mis.mpg.de/publications/preprints/2014/prepr2014-105.html>

Copyright © 2015 Robin Lamarche-Perrin (Robin.Lamarche-Perrin@lip6.fr)

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

AbstractSet	9
BiSet	14
Graph	20
CompleteGraph	17
FiliformGraph	20
RandomGraph	55
RingGraph	59
HierarchicalHierarchicalSet	24
HierarchicalOrderedSet	26
HierarchicalSet	27
MultiSet	35
NHOSet	40
NonconstrainedOrderedSet	43
NonconstrainedSet	45
OrderedSet	49
Ring	57
BiSubset	16
DataPointStruct	18
Dataset	18
Datatree	19
GraphComponent	22
HHNode	23
HNode	30
HONode	31
MultiSubset	39
NHONode	39
ObjectiveFunction	47
InformationBottleneck	32
LogarithmicScore	32
RelativeEntropy	56
ObjectiveValue	48
BottleneckObjectiveValue	17
LogarithmicScoreValue	33
RelativeObjectiveValue	57
OrderedDatatree	48
Part	52
BiPart	13

MultiPart	34
Partition	53
PredictionDataset	53
Timer	60
TreeToAdd	60
UniSet	61
HierarchicalUniSet	29
OrderedUniSet	51
UniSubset	62

Chapter 3

Class Index

3.1 Class List

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

AbstractSet	Abstract class defining a set of elements that one wants to partition while optimising some (decomposable) objective and preserving some algebraic constraints (set of feasible parts)	9
BiPart	A bi-part is a subset of a bi-dimensional set of elements (individuals)	13
BiSet		14
BiSubset		16
BottleneckObjectiveValue		17
CompleteGraph		17
DataPointStruct		18
Dataset		18
Datatree		19
FiliformGraph		20
Graph		20
GraphComponent		22
HHNode		23
HierarchicalHierarchicalSet		24
HierarchicalOrderedSet		26
HierarchicalSet		27
HierarchicalUniSet	A uni-dimensional set of elements structured according to a complete binary hierarchy, and such that the feasible subsets are all the nodes of the hierarchy	29
HNode		30
HONode		31
InformationBottleneck		32
LogarithmicScore	Class to define and compute the logarithmic score function in the case of point prediction . . .	32
LogarithmicScoreValue		33
MultiPart	A multi-part is a subset of a multi-dimensional set of elements (individuals)	34
MultiSet	A multi-dimensional set of elements based on the Cartesian product of several uni-dimensional sets (UniSet) and their algebraic structures (feasible subsets and feasible refinements)	35
MultiSubset		39
NHONode		39
NHOSet		40
NonconstrainedOrderedSet		43
NonconstrainedSet		45

ObjectiveFunction	
Abstract class defining an objective function to be associated to a constrained set in order to define the optimisation problem that one wants to solve	47
ObjectiveValue	48
OrderedDatatree	48
OrderedSet	49
OrderedUniSet	
A uni-dimensional set of elements with a total order, and such that the feasible subsets are all the intervals induced by this order	51
Part	
A part is a subset of a set of elements (individuals) represented by integers	52
Partition	
A partition is a collection of pairwise-disjoint and covering subsets (parts) of a set of elements .	53
PredictionDataset	
Class to represent a data set for the prediction of a post-measurement from the knowledge of a pre-measurement (composed of a train set and a test set)	53
RandomGraph	55
RelativeEntropy	56
RelativeObjectiveValue	57
Ring	57
RingGraph	59
Timer	60
TreeToAdd	60
UniSet	
A uni-dimensional set of elements and its algebraic structure (feasible subsets and feasible refinements)	61
UniSubset	
A feasible subset associated to a uni-dimensional set of elements (UniSet)	62

Chapter 4

File Index

4.1 File List

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

/home/lamarche/programming/optimal_partition/src/ abstract_set.hpp	
Abstract class defining a set of elements that one wants to partition while optimising some (decomposable) objective and preserving some algebraic constraints (set of feasible parts)	65
/home/lamarche/programming/optimal_partition/src/ bi_set.hpp	??
/home/lamarche/programming/optimal_partition/src/ bidimensional_relative_entropy.hpp	??
/home/lamarche/programming/optimal_partition/src/ check_graph_datatree.hpp	??
/home/lamarche/programming/optimal_partition/src/ csv_tools.hpp	??
/home/lamarche/programming/optimal_partition/src/ dataset.hpp	??
/home/lamarche/programming/optimal_partition/src/ datatree.hpp	??
/home/lamarche/programming/optimal_partition/src/ graph.hpp	??
/home/lamarche/programming/optimal_partition/src/ hierarchical_hierarchical_set.hpp	??
/home/lamarche/programming/optimal_partition/src/ hierarchical_ordered_set.hpp	??
/home/lamarche/programming/optimal_partition/src/ hierarchical_set.hpp	??
/home/lamarche/programming/optimal_partition/src/ information_bottleneck.hpp	??
/home/lamarche/programming/optimal_partition/src/ logarithmic_score.hpp	
Classes to define and compute the logarithmic score function in the case of point prediction	65
/home/lamarche/programming/optimal_partition/src/ main.hpp	??
/home/lamarche/programming/optimal_partition/src/ multi_set.hpp	
Classes to represent multi-dimensional sets of elements and their algebraic structure (feasible subsets and feasible refinements)	66
/home/lamarche/programming/optimal_partition/src/ NHO_set.hpp	??
/home/lamarche/programming/optimal_partition/src/ nonconstrained_ordered_set.hpp	??
/home/lamarche/programming/optimal_partition/src/ nonconstrained_set.hpp	??
/home/lamarche/programming/optimal_partition/src/ objective_function.hpp	
Abstract class defining an objective function to be associated to a constrained set in order to define the optimisation problem that one wants to solve	67
/home/lamarche/programming/optimal_partition/src/ orderedset.hpp	??
/home/lamarche/programming/optimal_partition/src/ partition.hpp	??
/home/lamarche/programming/optimal_partition/src/ prediction_dataset.hpp	
Class to represent a data set for the prediction of a post-measurement from the knowledge of a pre-measurement (composed of a train set and a test set)	67
/home/lamarche/programming/optimal_partition/src/ prediction_programs.hpp	??
/home/lamarche/programming/optimal_partition/src/ programs.hpp	??
/home/lamarche/programming/optimal_partition/src/ relative_entropy.hpp	??
/home/lamarche/programming/optimal_partition/src/ ring.hpp	??
/home/lamarche/programming/optimal_partition/src/ timer.hpp	??

[/home/lamarche/programming/optimal_partition/src/uni_set.hpp](#)

Some classes to represent uni-dimensional sets of elements and their algebraic structure (feasible subsets and feasible refinements) 68

Chapter 5

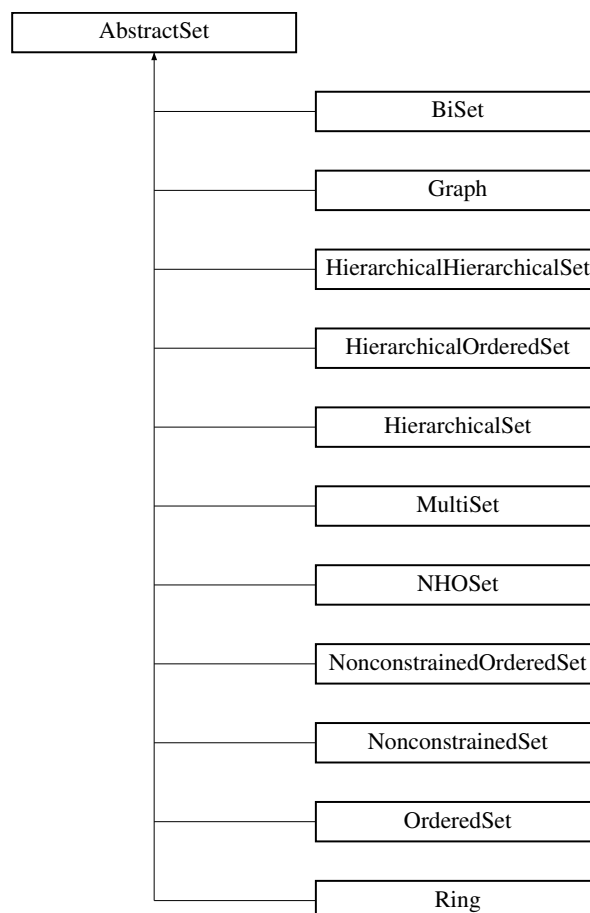
Class Documentation

5.1 AbstractSet Class Reference

Abstract class defining a set of elements that one wants to partition while optimising some (decomposable) objective and preserving some algebraic constraints (set of feasible parts)

```
#include <abstract_set.hpp>
```

Inheritance diagram for AbstractSet:



Public Member Functions

- virtual `~AbstractSet ()`
The objective that one wants to optimise (assumed to be decomposable: the objective of a partition is function of the objectives of its parts)
- virtual void `setRandom ()=0`
Randomly set the algebraic constraints for quick experiments (warning: this method is not always implemented)
- virtual void `buildDataStructure ()=0`
Build a proper data structure to represent the set and its algebraic constraints (warning: this method should always be called after instantiating and parameterising a set, and before calling any other method, such as `print()`, `computeObjectiveValues()`, `computeOptimalPartition` (double parameter), etc.)
- virtual void `setObjectiveFunction (ObjectiveFunction *objective)=0`
Set the objective that one wants to optimise.
- virtual void `print ()=0`
Print the set and its algebraic constraints.
- virtual void `computeObjectiveValues ()=0`
*Compute the value of the objective function for each feasible part (warning: `setObjectiveFunction (ObjectiveFunction *objective)` should have been called first)*
- virtual void `normalizeObjectiveValues ()=0`
Finish computing the value of the objective function for each feasible part when normalisation is required (warning: only after `computeObjectiveValues()` has been called)
- virtual void `printObjectiveValues ()=0`
Print the value of the objective function for each feasible part.
- virtual void `computeOptimalPartition` (double parameter)=0
Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.
- virtual `Partition *` `getOptimalPartition` (double parameter)=0
Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition` (double parameter) and by calling `print()` on the result)
- virtual void `printOptimalPartition` (double parameter)=0
Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition` (double parameter) and by calling `print()` on the result)
- `PartitionList *` `getOptimalPartitionList` (double threshold)
Compute and return a list of partitions that fit with the algebraic constraints and that optimises the objective function that has been specified, while the parameter of the objective function varies on a proper ranged (defined by the objective itself)
- void `printOptimalPartitionList` (double threshold)
Compute and print a list of partitions that fit with the algebraic constraints and that optimises the objective function that has been specified, while the parameter of the objective function varies on a proper ranged (defined by the objective itself)
- void `printOptimalPartitionListInCSV` (double threshold, `Dataset *`data, int dim, std::string fileName)
Compute and print in a CSV file a list of partitions that fit with the algebraic constraints and that optimises the objective function that has been specified, while the parameter of the objective function varies on a proper ranged (defined by the objective itself)

Public Attributes

- `ObjectiveFunction *` **objective**

5.1.1 Detailed Description

Abstract class defining a set of elements that one wants to partition while optimising some (decomposable) objective and preserving some algebraic constraints (set of feasible parts)

5.1.2 Constructor & Destructor Documentation

5.1.2.1 AbstractSet::~AbstractSet () [virtual]

The objective that one wants to optimise (assumed to be decomposable: the objective of a partition is function of the objectives of its parts)

Destructor

5.1.3 Member Function Documentation

5.1.3.1 virtual void AbstractSet::computeOptimalPartition (double *parameter*) [pure virtual]

Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implemented in [Graph](#), [MultiSet](#), [BiSet](#), [NonconstrainedSet](#), [Ring](#), [NHOSet](#), [HierarchicalHierarchicalSet](#), [OrderedSet](#), [NonconstrainedOrderedSet](#), [HierarchicalOrderedSet](#), and [HierarchicalSet](#).

5.1.3.2 virtual Partition* AbstractSet::getOptimalPartition (double *parameter*) [pure virtual]

Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition (double parameter)` and by calling `print()` on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Returns

: The resulting optimal partition

Implemented in [Graph](#), [MultiSet](#), [BiSet](#), [NonconstrainedSet](#), [Ring](#), [NHOSet](#), [HierarchicalHierarchicalSet](#), [OrderedSet](#), [NonconstrainedOrderedSet](#), [HierarchicalOrderedSet](#), and [HierarchicalSet](#).

5.1.3.3 PartitionList * AbstractSet::getOptimalPartitionList (double *threshold*)

Compute and return a list of partitions that fit with the algebraic constraints and that optimises the objective function that has been specified, while the parameter of the objective function varies on a proper ranged (defined by the objective itself)

Parameters

<i>threshold</i>	: The minimal distance between two successive parameters giving birth to two different partitions
------------------	---

Returns

: The resulting list of optimal partitions

5.1.3.4 `virtual void AbstractSet::printOptimalPartition (double parameter)` `[pure virtual]`

Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition (double parameter)` and by calling `print()` on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implemented in [Graph](#), [MultiSet](#), [BiSet](#), [NonconstrainedSet](#), [Ring](#), [NHOSet](#), [HierarchicalHierarchicalSet](#), [OrderedSet](#), [NonconstrainedOrderedSet](#), [HierarchicalOrderedSet](#), and [HierarchicalSet](#).

5.1.3.5 void AbstractSet::printOptimalPartitionList (double *threshold*)

Compute and print a list of partitions that fit with the algebraic constraints and that optimises the objective function that has been specified, while the parameter of the objective function varies on a proper ranged (defined by the objective itself)

Parameters

<i>threshold</i>	: The minimal distance between two successive parameters giving birth to two different partitions
------------------	---

5.1.3.6 void AbstractSet::printOptimalPartitionListInCSV (double *threshold*, Dataset * *data*, int *dim*, std::string *fileName*)

Compute and print in a CSV file a list of partitions that fit with the algebraic constraints and that optimises the objective function that has been specified, while the parameter of the objective function varies on a proper ranged (defined by the objective itself)

Parameters

<i>threshold</i>	: The minimal distance between two successive parameters giving birth to two different partitions
------------------	---

5.1.3.7 virtual void AbstractSet::setObjectiveFunction (ObjectiveFunction * *objective*) [pure virtual]

Set the objective that one wants to optimise.

Parameters

<i>objective</i>	: The objective function itself
------------------	---------------------------------

Implemented in [MultiSet](#), [Graph](#), [BiSet](#), [Ring](#), [NonconstrainedSet](#), [NHOSet](#), [HierarchicalHierarchicalSet](#), [OrderedSet](#), [NonconstrainedOrderedSet](#), [HierarchicalOrderedSet](#), and [HierarchicalSet](#).

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

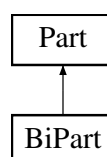
- /home/lamarche/programming/optimal_partition/src/[abstract_set.hpp](#)
- /home/lamarche/programming/optimal_partition/src/[abstract_set.cpp](#)

5.2 BiPart Class Reference

A bi-part is a subset of a bi-dimensional set of elements (individuals)

```
#include <partition.hpp>
```

Inheritance diagram for BiPart:



Public Member Functions

- **BiPart** ([Part](#) *part1, [Part](#) *part2, [ObjectiveValue](#) *value=0)
- **BiPart** ([BiPart](#) *biPart)
- bool **equal** ([Part](#) *p)
- void **print** (bool endl=false)
- int **printSize** ()

Public Attributes

- [Part](#) * **firstPart**
- [Part](#) * **secondPart**

5.2.1 Detailed Description

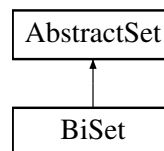
A bi-part is a subset of a bi-dimensional set of elements (individuals)

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

- /home/lamarche/programming/optimal_partition/src/partition.hpp
- /home/lamarche/programming/optimal_partition/src/partition.cpp

5.3 BiSet Class Reference

Inheritance diagram for BiSet:



Public Member Functions

- **BiSet** ([UniSet](#) *uniSet1, [UniSet](#) *uniSet2)
- void **initReached** ()
- void **setRandom** ()
Randomly set the algebraic constraints for quick experiments (warning: this method is not always implemented)
- void **setObjectiveFunction** ([ObjectiveFunction](#) *m)
Set the objective that one wants to optimise.
- void **print** ()
Print the set and its algebraic constraints.
- void **buildDataStructure** ()
Build a proper data structure to represent the set and its algebraic constraints (warning: this method should always be called after instantiating and parameterising a set, and before calling any other method, such as [print\(\)](#), [computeObjectiveValues\(\)](#), [computeOptimalPartition](#) (double parameter), etc.)
- void **computeObjectiveValues** ()
*Compute the value of the objective function for each feasible part (warning: [setObjectiveFunction](#) ([ObjectiveFunction](#) *objective) should have been called first)*
- void **normalizeObjectiveValues** ()
Finish computing the value of the objective function for each feasible part when normalisation is required (warning: only after [computeObjectiveValues\(\)](#) has been called)

- void [printObjectiveValues](#) ()
Print the value of the objective function for each feasible part.
- void [computeOptimalPartition](#) (double parameter)
Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.
- void [printOptimalPartition](#) (double parameter)
Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)
- [Partition](#) * [getOptimalPartition](#) (double parameter)
Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)

Public Attributes

- [UniSet](#) * **uniSet1**
- [UniSet](#) * **uniSet2**
- int **biSubsetNumber**
- int **atomicBiSubsetNumber**
- [BiSubset](#) * **firstBiSubset**
- [BiSubset](#) ** **biSubsetArray**

5.3.1 Member Function Documentation

5.3.1.1 void BiSet::computeOptimalPartition (double parameter) [virtual]

Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.3.1.2 Partition * BiSet::getOptimalPartition (double parameter) [virtual]

Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Returns

: The resulting optimal partition

Implements [AbstractSet](#).

5.3.1.3 void BiSet::printOptimalPartition (double parameter) [virtual]

Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.3.1.4 void BiSet::setObjectiveFunction (**ObjectiveFunction** * *objective*) [virtual]

Set the objective that one wants to optimise.

Parameters

<i>objective</i>	: The objective function itself
------------------	---------------------------------

Implements [AbstractSet](#).

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

- /home/lamarche/programming/optimal_partition/src/bi_set.hpp
- /home/lamarche/programming/optimal_partition/src/bi_set.cpp

5.4 BiSubset Class Reference

Public Member Functions

- **BiSubset** ([UniSubset](#) *uniSubset1, [UniSubset](#) *uniSubset2)
- void **print** ()
- void **printIndexSet** (bool endl=false)
- void **addBiSubsetSet** (BiSubsetSet *biSubsetSet)
- void **setObjectiveFunction** ([ObjectiveFunction](#) *m)
- void **buildDataStructure** ()
- void **computeObjectiveValues** ()
- void **normalizeObjectiveValues** ([ObjectiveValue](#) *maxQual=0)
- void **printObjectiveValues** ()
- void **computeOptimalPartition** (double parameter)
- void **printOptimalPartition** (double parameter)
- void **buildOptimalPartition** ([Partition](#) *partition)

Public Attributes

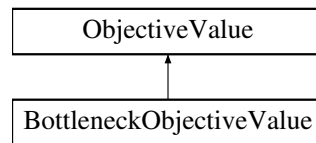
- [UniSubset](#) * **uniSubset1**
- [UniSubset](#) * **uniSubset2**
- int **num**
- bool **isAtomic**
- bool **reached**
- BiSubsetSetSet * **biSubsetSetSet**
- [BiSet](#) * **biSet**
- [ObjectiveFunction](#) * **objective**
- [ObjectiveValue](#) * **value**
- double **optimalValue**
- BiSubsetSet * **optimalCut**

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

- /home/lamarche/programming/optimal_partition/src/bi_set.hpp
- /home/lamarche/programming/optimal_partition/src/bi_set.cpp

5.5 BottleneckObjectiveValue Class Reference

Inheritance diagram for BottleneckObjectiveValue:



Public Member Functions

- **BottleneckObjectiveValue** ([InformationBottleneck](#) *objective, int index=-1)
- void **add** ([ObjectiveValue](#) *value)
- void **compute** ()
- void **compute** ([ObjectiveValue](#) *value1, [ObjectiveValue](#) *value2)
- void **compute** ([ObjectiveValueSet](#) *valueSet)
- void **normalize** ([ObjectiveValue](#) *q)
- void **print** (bool verbose=true)
- double **getValue** (double param)

Public Attributes

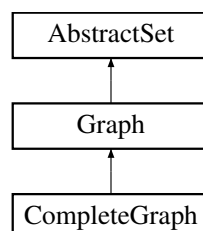
- int **index**
- double **pk**
- double * **pkj**
- double * **pj**
- double **lki**
- double **lkj**

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

- /home/lamarche/programming/optimal_partition/src/information_bottleneck.hpp
- /home/lamarche/programming/optimal_partition/src/information_bottleneck.cpp

5.6 CompleteGraph Class Reference

Inheritance diagram for CompleteGraph:



Public Member Functions

- **CompleteGraph** (int vNum)

Additional Inherited Members

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

- /home/lamarche/programming/optimal_partition/src/graph.hpp

5.7 DataPointStruct Struct Reference

Public Attributes

- std::vector< int > **parameters**
- float **time**
- int **memory**

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

- /home/lamarche/programming/optimal_partition/src/timer.hpp

5.8 Dataset Class Reference

Public Member Functions

- void **print** ()
- void **buildDataset** ()
- void **initValues** (double v=0)
- void **initRefValues** (double v=0)
- void **addLabel1** (std::string str)
- void **addLabel2** (std::string str)
- std::string **getLabel1** (int i)
- std::string **getLabel2** (int j)
- int **getIndex1** (std::string s1)
- int **getIndex2** (std::string s2)
- double **getValue** (int i, int j)
- double **getRefValue** (int i, int j)
- double **getValue** (std::string s1, std::string s2)
- double **getRefValue** (std::string s1, std::string s2)
- void **setValue** (int i, int j, double v)
- void **setRefValue** (int i, int j, double v)
- void **setValue** (std::string s1, std::string s2, double v)
- void **setRefValue** (std::string s1, std::string s2, double v)
- void **incrementValue** (int i, int j)
- void **incrementRefValue** (int i, int j)
- void **incrementValue** (std::string s1, std::string s2)
- void **incrementRefValue** (std::string s1, std::string s2)
- double * **getValues1** (std::string s2)
- double * **getValues2** (std::string s1)
- double * **getRefValues1** (std::string s2)
- double * **getRefValues2** (std::string s1)
- double * **getValues** (bool order=true)
- double * **getRefValues** (bool order=true)

Public Attributes

- int **size1**
- int **size2**
- std::map< std::string, int > * **indices1**
- std::map< std::string, int > * **indices2**
- std::map< int, std::string > * **labels1**
- std::map< int, std::string > * **labels2**
- double * **values**
- double * **refValues**

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

- /home/lamarche/programming/optimal_partition/src/dataset.hpp
- /home/lamarche/programming/optimal_partition/src/dataset.cpp

5.9 Datatree Class Reference

Public Member Functions

- **Datatree** ([Datatree](#) &tree)
- **Datatree** (int vertex=-1)
- void **setObjectiveFunction** ([ObjectiveFunction](#) *objective)
- std::string **toString** ()
- Vertices * **getAllVertices** ()
- [Datatree](#) * **addChild** (int v, bool print=true)
- [Datatree](#) * **findChild** (int v)
- [Datatree](#) * **findOrAddChild** (int v, bool print=true)
- void **addBipartition** ([Datatree](#) *n1, [Datatree](#) *n2)
- void **computeObjectiveValues** ()
- void **normalizeObjectiveValues** ([ObjectiveValue](#) *maxObjectiveValue=0)
- void **printObjectiveValues** ()
- void **computeOptimalPartition** (double parameter)
- void **printOptimalPartition** (double parameter)
- [Partition](#) * **getOptimalPartition** (double parameter)
- void **print** (bool verbose=false)
- void **printVertices** (bool endl=true)
- PartSet * **getParts** ()
- void **printParts** ()
- PartitionList * **getAllPartitions** ()
- int **printPartitions** (bool print=true)

Public Attributes

- int **size**
- int **vertex**
- bool **wholeSet**
- [ObjectiveFunction](#) * **objective**
- [Datatree](#) * **parent**
- [Datatree](#) * **complement**
- TreesList * **complementList**
- TreesSet * **children**
- [ObjectiveValue](#) * **value**

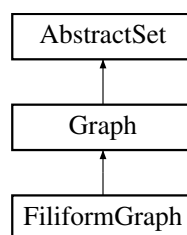
- BipartitionsSet * **bipartitions**
- double **optimalValue**
- Bipartition * **optimalBipartition**
- bool **optimized**

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

- /home/lamarche/programming/optimal_partition/src/datatree.hpp
- /home/lamarche/programming/optimal_partition/src/datatree.cpp

5.10 FiliformGraph Class Reference

Inheritance diagram for FiliformGraph:



Public Member Functions

- **FiliformGraph** (int vNum)

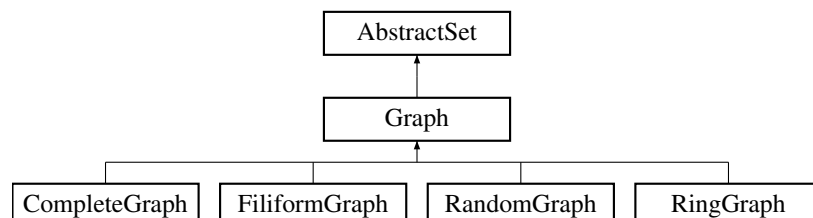
Additional Inherited Members

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

- /home/lamarche/programming/optimal_partition/src/graph.hpp
- /home/lamarche/programming/optimal_partition/src/graph.cpp

5.11 Graph Class Reference

Inheritance diagram for Graph:



Public Member Functions

- **Graph** (int size)
- void **setRandom** ()

- Randomly set the algebraic constraints for quick experiments (warning: this method is not always implemented)*
- void **setObjectiveFunction** ([ObjectiveFunction](#) *m)
 - Set the objective that one wants to optimise.*
- void **print** ()
 - Print the set and its algebraic constraints.*
- void **addEdge** (int v1, int v2)
- void **buildFromBinary** (int index)
- bool **areAdjacent** (int v1, int v2)
- bool **areAdjacent** (int v1, Vertices *v2)
- bool **areAdjacent** (Vertices *v1, int v2)
- bool **areAdjacent** (Vertices *v1, Vertices *v2)
- bool **areAdjacent** (int v1, VVertices *v2)
- bool **areAdjacent** (VVertices *v1, int v2)
- bool **areAdjacent** (VVertices *v1, VVertices *v2)
- Vertices * **getAdjacentVertices** (int v, int vMax=-1)
- void **printVertices** (Vertices *V)
- bool **isConnected** ()
- bool **isConnected** (Vertices *V)
- void **printDataStructure** (bool verbose=true)
- PartSet * **getParts** ()
- void **printParts** ()
- int **printPartitions** (bool [print](#)=true)
- void **buildDataStructure** ()
 - Build a proper data structure to represent the set and its algebraic constraints (warning: this method should always be called after instantiating and parameterising a set, and before calling any other method, such as [print\(\)](#), [computeObjectiveValues\(\)](#), [computeOptimalPartition](#) (double parameter), etc.)*
- void **computeObjectiveValues** ()
 - Compute the value of the objective function for each feasible part (warning: setObjectiveFunction ([ObjectiveFunction](#) *objective) should have been called first)*
- void **normalizeObjectiveValues** ()
 - Finish computing the value of the objective function for each feasible part when normalisation is required (warning: only after [computeObjectiveValues\(\)](#) has been called)*
- void **printObjectiveValues** ()
 - Print the value of the objective function for each feasible part.*
- void **buildDataStructureWithSlyce** ()
- void **slyce** (VVertices *R, VVertices *F, int m)
- void **enumerateSubsets** (VVertices *R, VVertices *F, int m, int n, VVertices *T, int q, int r)
- void **computeOptimalPartition** (double parameter)
 - Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.*
- void **printOptimalPartition** (double parameter)
 - Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)*
- [Partition](#) * **getOptimalPartition** (double parameter)
 - Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)*

Public Attributes

- int **size**
- Vertices ** **adjacencySets**
- [GraphComponent](#) ** **graphComponents**
- [GraphComponentSet](#) * **graphComponentSet**
- bool * **reachedVertices**
- [ObjectiveValue](#) * **value**

5.11.1 Member Function Documentation

5.11.1.1 void Graph::computeOptimalPartition (double *parameter*) [virtual]

Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.11.1.2 Partition * Graph::getOptimalPartition (double *parameter*) [virtual]

Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using getOptimalPartition (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Returns

: The resulting optimal partition

Implements [AbstractSet](#).

5.11.1.3 void Graph::printOptimalPartition (double *parameter*) [virtual]

Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using getOptimalPartition (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.11.1.4 void Graph::setObjectiveFunction (ObjectiveFunction * *objective*) [virtual]

Set the objective that one wants to optimise.

Parameters

<i>objective</i>	: The objective function itself
------------------	---------------------------------

Implements [AbstractSet](#).

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

- /home/lamarche/programming/optimal_partition/src/graph.hpp
- /home/lamarche/programming/optimal_partition/src/graph.cpp

5.12 GraphComponent Class Reference

Public Member Functions

- **GraphComponent** ([Graph](#) *graph)
- void **setVertices** (std::list< int > *vertexList)
- void **printDataStructure** (bool verbose=true)
- void **buildDataStructure** ()
- void **computeObjectiveValues** ()
- void **normalizeObjectiveValues** ()
- void **printObjectiveValues** ()
- void **computeOptimalPartition** (double parameter)
- void **printOptimalPartition** (double parameter)
- [Partition](#) * **getOptimalPartition** (double parameter)

Public Attributes

- int **size**
- int * **vertices**
- std::map< int, int > * **order**
- [Graph](#) * **graph**
- [Datatree](#) * **datatree**

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

- /home/lamarche/programming/optimal_partition/src/graph.hpp
- /home/lamarche/programming/optimal_partition/src/graph.cpp

5.13 HHNode Class Reference

Public Member Functions

- **HHNode** ([HNode](#) *node1, [HNode](#) *node2)
- void **addChild1** ([HHNode](#) *node)
- void **addChild2** ([HHNode](#) *node)
- void **setObjectiveFunction** ([ObjectiveFunction](#) *m)
- void **print** ()
- void **printIndices** (bool endl=false)
- void **buildDataStructure** ()
- void **computeObjectiveValues** ()
- void **normalizeObjectiveValues** ([ObjectiveValue](#) *maxQual=0)
- void **printObjectiveValues** ()
- void **computeOptimalPartition** (double parameter)
- void **printOptimalPartition** (double parameter)
- void **buildOptimalPartition** ([Partition](#) *partition)

Public Attributes

- [HNode](#) * **node1**
- [HNode](#) * **node2**
- HHNodeSet * **children1**
- HHNodeSet * **children2**
- [ObjectiveFunction](#) * **objective**
- [ObjectiveValue](#) * **value**

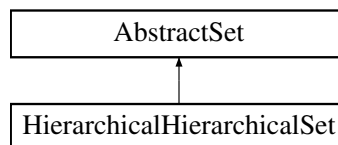
- double **optimalValue**
- int **optimalCut**

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

- /home/lamarche/programming/optimal_partition/src/hierarchical_hierarchical_set.hpp
- /home/lamarche/programming/optimal_partition/src/hierarchical_hierarchical_set.cpp

5.14 HierarchicalHierarchicalSet Class Reference

Inheritance diagram for HierarchicalHierarchicalSet:



Public Member Functions

- **HierarchicalHierarchicalSet** ([HNode](#) *hierarchy1, [HNode](#) *hierarchy2)
- void [setRandom](#) ()
Randomly set the algebraic constraints for quick experiments (warning: this method is not always implemented)
- void [setObjectiveFunction](#) ([ObjectiveFunction](#) *m)
Set the objective that one wants to optimise.
- void [print](#) ()
Print the set and its algebraic constraints.
- void [buildDataStructure](#) ()
Build a proper data structure to represent the set and its algebraic constraints (warning: this method should always be called after instantiating and parameterising a set, and before calling any other method, such as [print\(\)](#), [computeObjectiveValues\(\)](#), [computeOptimalPartition](#) (double parameter), etc.)
- void [computeObjectiveValues](#) ()
*Compute the value of the objective function for each feasible part (warning: [setObjectiveFunction](#) ([ObjectiveFunction](#) *objective) should have been called first)*
- void [normalizeObjectiveValues](#) ()
Finish computing the value of the objective function for each feasible part when normalisation is required (warning: only after [computeObjectiveValues\(\)](#) has been called)
- void [printObjectiveValues](#) ()
Print the value of the objective function for each feasible part.
- void [computeOptimalPartition](#) (double parameter)
Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.
- void [printOptimalPartition](#) (double parameter)
Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)
- [Partition](#) * [getOptimalPartition](#) (double parameter)
Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)

Public Attributes

- [HNode](#) * **hierarchy1**
- [HNode](#) * **hierarchy2**
- [HHNode](#) * **hyperarchy**

5.14.1 Member Function Documentation

5.14.1.1 void HierarchicalHierarchicalSet::computeOptimalPartition (double *parameter*) [virtual]

Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.14.1.2 Partition * HierarchicalHierarchicalSet::getOptimalPartition (double *parameter*) [virtual]

Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using getOptimalPartition (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Returns

: The resulting optimal partition

Implements [AbstractSet](#).

5.14.1.3 void HierarchicalHierarchicalSet::printOptimalPartition (double *parameter*) [virtual]

Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using getOptimalPartition (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.14.1.4 void HierarchicalHierarchicalSet::setObjectiveFunction (ObjectiveFunction * *objective*) [virtual]

Set the objective that one wants to optimise.

Parameters

<i>objective</i>	: The objective function itself
------------------	---------------------------------

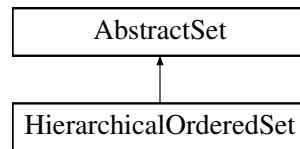
Implements [AbstractSet](#).

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

- /home/lamarche/programming/optimal_partition/src/hierarchical_hierarchical_set.hpp
- /home/lamarche/programming/optimal_partition/src/hierarchical_hierarchical_set.cpp

5.15 HierarchicalOrderedSet Class Reference

Inheritance diagram for HierarchicalOrderedSet:



Public Member Functions

- **HierarchicalOrderedSet** ([HONode](#) *hierarchy, int size)
- void [setRandom](#) ()
Randomly set the algebraic constraints for quick experiments (warning: this method is not always implemented)
- void [setObjectiveFunction](#) ([ObjectiveFunction](#) *m)
Set the objective that one wants to optimise.
- void [print](#) ()
Print the set and its algebraic constraints.
- void [buildDataStructure](#) ()
Build a proper data structure to represent the set and its algebraic constraints (warning: this method should always be called after instantiating and parameterising a set, and before calling any other method, such as [print\(\)](#), [computeObjectiveValues\(\)](#), [computeOptimalPartition](#) (double parameter), etc.)
- void [computeObjectiveValues](#) ()
*Compute the value of the objective function for each feasible part (warning: setObjectiveFunction ([ObjectiveFunction](#) *objective) should have been called first)*
- void [normalizeObjectiveValues](#) ()
Finish computing the value of the objective function for each feasible part when normalisation is required (warning: only after [computeObjectiveValues\(\)](#) has been called)
- void [printObjectiveValues](#) ()
Print the value of the objective function for each feasible part.
- void [computeOptimalPartition](#) (double parameter)
Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.
- void [printOptimalPartition](#) (double parameter)
Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)
- [Partition](#) * [getOptimalPartition](#) (double parameter)
Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)

Public Attributes

- int **size**
- [HONode](#) * **hierarchy**

5.15.1 Member Function Documentation

5.15.1.1 void HierarchicalOrderedSet::computeOptimalPartition (double parameter) [virtual]

Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.15.1.2 Partition * HierarchicalOrderedSet::getOptimalPartition (double *parameter*) [virtual]

Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using getOptimalPartition (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Returns

: The resulting optimal partition

Implements [AbstractSet](#).

5.15.1.3 void HierarchicalOrderedSet::printOptimalPartition (double *parameter*) [virtual]

Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using getOptimalPartition (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.15.1.4 void HierarchicalOrderedSet::setObjectiveFunction (ObjectiveFunction * *objective*) [virtual]

Set the objective that one wants to optimise.

Parameters

<i>objective</i>	: The objective function itself
------------------	---------------------------------

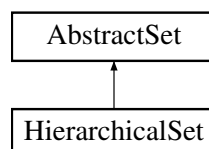
Implements [AbstractSet](#).

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

- /home/lamarche/programming/optimal_partition/src/hierarchical_ordered_set.hpp
- /home/lamarche/programming/optimal_partition/src/hierarchical_ordered_set.cpp

5.16 HierarchicalSet Class Reference

Inheritance diagram for HierarchicalSet:



Public Member Functions

- **HierarchicalSet** ([HNode](#) *hierarchy)
- void [setRandom](#) ()
Randomly set the algebraic constraints for quick experiments (warning: this method is not always implemented)
- void [setObjectiveFunction](#) ([ObjectiveFunction](#) *m)
Set the objective that one wants to optimise.
- void [print](#) ()
Print the set and its algebraic constraints.
- void [buildDataStructure](#) ()
Build a proper data structure to represent the set and its algebraic constraints (warning: this method should always be called after instantiating and parameterising a set, and before calling any other method, such as [print\(\)](#), [computeObjectiveValues\(\)](#), [computeOptimalPartition](#) (double parameter), etc.)
- void [computeObjectiveValues](#) ()
*Compute the value of the objective function for each feasible part (warning: [setObjectiveFunction](#) ([ObjectiveFunction](#) *objective) should have been called first)*
- void [normalizeObjectiveValues](#) ()
Finish computing the value of the objective function for each feasible part when normalisation is required (warning: only after [computeObjectiveValues\(\)](#) has been called)
- void [printObjectiveValues](#) ()
Print the value of the objective function for each feasible part.
- void [computeOptimalPartition](#) (double parameter)
Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.
- void [printOptimalPartition](#) (double parameter)
Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)
- [Partition](#) * [getOptimalPartition](#) (double parameter)
Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)

Public Attributes

- [HNode](#) * **hierarchy**

5.16.1 Member Function Documentation

5.16.1.1 void HierarchicalSet::computeOptimalPartition (double *parameter*) [virtual]

Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.16.1.2 [Partition](#) * HierarchicalSet::getOptimalPartition (double *parameter*) [virtual]

Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Returns

: The resulting optimal partition

Implements [AbstractSet](#).

5.16.1.3 void HierarchicalSet::printOptimalPartition (double *parameter*) [virtual]

Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition (double parameter)` and by calling `print()` on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.16.1.4 void HierarchicalSet::setObjectiveFunction (ObjectiveFunction * *objective*) [virtual]

Set the objective that one wants to optimise.

Parameters

<i>objective</i>	: The objective function itself
------------------	---------------------------------

Implements [AbstractSet](#).

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

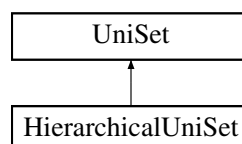
- /home/lamarche/programming/optimal_partition/src/hierarchical_set.hpp
- /home/lamarche/programming/optimal_partition/src/hierarchical_set.cpp

5.17 HierarchicalUniSet Class Reference

A uni-dimensional set of elements structured according to a complete binary hierarchy, and such that the feasible subsets are all the nodes of the hierarchy.

```
#include <uni_set.hpp>
```

Inheritance diagram for HierarchicalUniSet:



Public Member Functions

- [HierarchicalUniSet](#) (int depth)
Depth of the complete binary hierarchy.

Public Attributes

- int **depth**

Additional Inherited Members

5.17.1 Detailed Description

A uni-dimensional set of elements structured according to a complete binary hierarchy, and such that the feasible subsets are all the nodes of the hierarchy.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 HierarchicalUniSet::HierarchicalUniSet (int *depth*)

Depth of the complete binary hierarchy.

Constructor

Parameters

<i>size</i>	: Depth of the complete binary hierarchy
-------------	--

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

- /home/lamarche/programming/optimal_partition/src/[uni_set.hpp](#)
- /home/lamarche/programming/optimal_partition/src/[uni_set.cpp](#)

5.18 HNode Class Reference

Public Member Functions

- **HNode** (int index=-1)
- void **addChild** ([HNode](#) *node)
- void **setObjectiveFunction** ([ObjectiveFunction](#) *m)
- void **print** ()
- void **printIndices** (bool endl=false)
- void **buildDataStructure** ([HNode](#) *root=0, int level=0, int num=0)
- void **computeObjectiveValues** ()
- void **normalizeObjectiveValues** ([ObjectiveValue](#) *maxQual=0)
- void **printObjectiveValues** ()
- void **computeOptimalPartition** (double parameter)
- void **printOptimalPartition** (double parameter)
- [Partition](#) * **getOptimalPartition** (double parameter)
- void **buildOptimalPartition** ([Partition](#) *partition)

Public Attributes

- int **index**
- std::set< int > * **indices**
- int **level**
- int **size**
- int **width**
- int **num**

- [HNode](#) * **root**
- HNodeSet * **children**
- [ObjectiveFunction](#) * **objective**
- [ObjectiveValue](#) * **value**
- double **optimalValue**
- bool **optimalCut**

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

- /home/lamarche/programming/optimal_partition/src/hierarchical_set.hpp
- /home/lamarche/programming/optimal_partition/src/hierarchical_set.cpp

5.19 HONode Class Reference

Public Member Functions

- **HONode** (int size, int index=-1)
- int **getIndex** (int i, int j)
- void **addChild** ([HONode](#) *node)
- void **setObjectiveFunction** ([ObjectiveFunction](#) *m)
- void **print** ()
- void **buildDataStructure** (int level=0)
- void **computeObjectiveValues** ()
- void **normalizeObjectiveValues** ([ObjectiveValue](#) *maxQual=0)
- void **printObjectiveValues** ()
- void **computeOptimalPartition** (double parameter)
- void **printOptimalPartition** (double parameter)
- [Partition](#) * **getOptimalPartition** (double parameter)
- void **buildOptimalPartition** ([Partition](#) *partition, int pi=0, int pj=-1)

Public Attributes

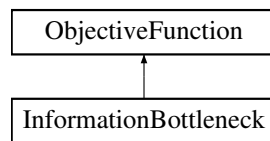
- int **index**
- int **level**
- std::set< int > * **indices**
- HONodeSet * **children**
- [ObjectiveFunction](#) * **objective**
- int **size**
- [ObjectiveValue](#) ** **qualities**
- double * **optimalValues**
- int * **optimalCuts**

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

- /home/lamarche/programming/optimal_partition/src/hierarchical_ordered_set.hpp
- /home/lamarche/programming/optimal_partition/src/hierarchical_ordered_set.cpp

5.20 InformationBottleneck Class Reference

Inheritance diagram for InformationBottleneck:



Public Member Functions

- **InformationBottleneck** (MarkovProcess *process)
- void **setRandom** ()
Randomly set the initial data from which the objective function is computed.
- **ObjectiveValue** * **newObjectiveValue** (int index=-1)
*This method is called by child classes of **AbstractSet** (do not use directly)*
- double **getParameter** (double unit)
- double **getUnitDistance** (double uMin, double uMax)
- double **getIntermediaryUnit** (double uMin, double uMax)

Public Attributes

- MarkovProcess * **process**

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

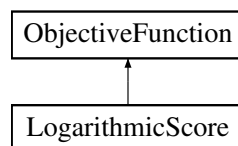
- /home/lamarche/programming/optimal_partition/src/information_bottleneck.hpp
- /home/lamarche/programming/optimal_partition/src/information_bottleneck.cpp

5.21 LogarithmicScore Class Reference

Class to define and compute the logarithmic score function in the case of point prediction.

```
#include <logarithmic_score.hpp>
```

Inheritance diagram for LogarithmicScore:



Public Member Functions

- **LogarithmicScore** (**PredictionDataset** *dataset, int prior=0)
Constructor.
- **~LogarithmicScore** ()
Destructor.
- void **setRandom** ()

- Randomly set the initial data from which the objective function is computed.
- **ObjectiveValue** * **newObjectiveValue** (int index=-1)
This method is called by child classes of **AbstractSet** (do not use directly)
- void **computeObjectiveValues** ()
This method is called by child classes of **AbstractSet** (do not use directly)
- void **printObjectiveValues** (bool verbose=true)
This method is called by child classes of **AbstractSet** (do not use directly)
- double **getParameter** (double unit)
- double **getUnitDistance** (double uMin, double uMax)
- double **getIntermediaryUnit** (double uMin, double uMax)

Additional Inherited Members

5.21.1 Detailed Description

Class to define and compute the logarithmic score function in the case of point prediction.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 LogarithmicScore::LogarithmicScore (**PredictionDataset** * *dataset*, int *prior* = 0)

Constructor.

Parameters

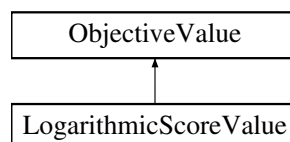
<i>dataset</i>	: The prediction data set that is use to evaluate the score function (from a train set and a test set both containing pre-observations and post-observations)
<i>prior</i>	: A prior giving the number of times each couple of (pre and post) observations has been observed in addition to the train set

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

- /home/lamarche/programming/optimal_partition/src/[logarithmic_score.hpp](#)
- /home/lamarche/programming/optimal_partition/src/logarithmic_score.cpp

5.22 LogarithmicScoreValue Class Reference

Inheritance diagram for LogarithmicScoreValue:



Public Member Functions

- **LogarithmicScoreValue** (**LogarithmicScore** *objective)
- void **add** (**ObjectiveValue** *value)
- void **compute** ()
- void **compute** (**ObjectiveValue** *value1, **ObjectiveValue** *value2)
- void **compute** (**ObjectiveValueSet** *valueset)

- void **normalize** ([ObjectiveValue](#) *q)
- void **print** (bool verbose=true)
- double **getValue** (double param)

Public Attributes

- int **preSize**
- int **postSize**
- int * **trainCountArray**
- int * **testCountArray**
- int **trainCountTotal**
- int **testCountTotal**
- double **score**

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

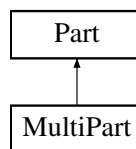
- /home/lamarche/programming/optimal_partition/src/[logarithmic_score.hpp](#)
- /home/lamarche/programming/optimal_partition/src/logarithmic_score.cpp

5.23 MultiPart Class Reference

A multi-part is a subset of a multi-dimensional set of elements (individuals)

```
#include <partition.hpp>
```

Inheritance diagram for MultiPart:



Public Member Functions

- **MultiPart** ([Part](#) **partArray, int dimension, [ObjectiveValue](#) *value=0)
- **MultiPart** ([MultiPart](#) *multiPart)
- bool **equal** ([Part](#) *p)
- void **print** (bool endl=false)
- int **printSize** ()

Public Attributes

- int **dimension**
- [Part](#) ** **partArray**

5.23.1 Detailed Description

A multi-part is a subset of a multi-dimensional set of elements (individuals)

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

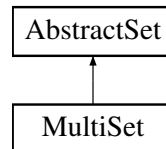
- /home/lamarche/programming/optimal_partition/src/partition.hpp
- /home/lamarche/programming/optimal_partition/src/partition.cpp

5.24 MultiSet Class Reference

A multi-dimensional set of elements based on the Cartesian product of several uni-dimensional sets ([UniSet](#)) and their algebraic structures (feasible subsets and feasible refinements)

```
#include <multi_set.hpp>
```

Inheritance diagram for MultiSet:



Public Member Functions

- [MultiSet](#) ([UniSet](#) *uniSet)
Number of feasible subsets.
- [MultiSet](#) ([UniSet](#) **uniSetArray, int dimension)
Constructor.
- virtual [~MultiSet](#) ()
- [MultiSubset](#) * [getAtomicMultiSubset](#) (int index)
- [MultiSubset](#) * [getRandomAtomicMultiSubset](#) ()
- void [setRandom](#) ()
Randomly set the algebraic constraints for quick experiments (warning: this method is not always implemented)
- void [setObjectiveFunction](#) ([ObjectiveFunction](#) *objective)
Set the objective that one wants to optimise.
- void [print](#) ()
Print the set and its algebraic constraints.
- void [buildDataStructure](#) ()
Build a proper data structure to represent the set and its algebraic constraints (warning: this method should always be called after instantiating and parameterising a set, and before calling any other method, such as [print\(\)](#), [computeObjectiveValues\(\)](#), [computeOptimalPartition](#) (double parameter), etc.)
- void [computeObjectiveValues](#) ()
*Compute the value of the objective function for each feasible part (warning: setObjectiveFunction ([ObjectiveFunction](#) *objective) should have been called first)*
- void [normalizeObjectiveValues](#) ()
Finish computing the value of the objective function for each feasible part when normalisation is required (warning: only after [computeObjectiveValues\(\)](#) has been called)
- void [printObjectiveValues](#) ()
Print the value of the objective function for each feasible part.
- void [computeOptimalPartition](#) (double parameter)
Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.
- void [printOptimalPartition](#) (double parameter)
Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)
- [Partition](#) * [getOptimalPartition](#) (double parameter)
Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)

Public Attributes

- int **dimension**
- int [atomicMultiSubsetNumber](#)
Number of dimensions.
- int [multiSubsetNumber](#)
Number of elements (e.g., atomic feasible subsets)

Protected Member Functions

- int **getNum** (int *multiNum)
- int * **getMultiNum** (int num)
- void **initReached** ()

Protected Attributes

- [UniSet](#) ** **uniSetArray**
- [MultiSubset](#) * **firstMultiSubset**
- [MultiSubset](#) ** **multiSubsetArray**
- [MultiSubset](#) ** **atomicMultiSubsetArray**

5.24.1 Detailed Description

A multi-dimensional set of elements based on the Cartesian product of several uni-dimensional sets ([UniSet](#)) and their algebraic structures (feasible subsets and feasible refinements)

5.24.2 Constructor & Destructor Documentation

5.24.2.1 MultiSet::MultiSet ([UniSet](#) * *uniSet*)

Number of feasible subsets.

Constructor for a one-dimensional set

Parameters

<i>uniSet</i>	: Pointer to one uni-dimensional set (UniSet)
---------------	---

5.24.2.2 MultiSet::MultiSet ([UniSet](#) ** *uniSetArray*, int *dimension*)

Constructor.

Parameters

<i>uniSetArray</i>	: Array of pointers to uni-dimensional sets from which the Cartesian product is computed
<i>dimension</i>	: Number of uni-dimensional sets

5.24.2.3 MultiSet::~MultiSet () [virtual]

Destructor

5.24.3 Member Function Documentation

5.24.3.1 void MultiSet::computeOptimalPartition (double *parameter*) [virtual]

Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.24.3.2 MultiSubset * MultiSet::getAtomicMultiSubset (int *index*)

Access to an element (e.g., atomic feasible subset) from its index /param The index of the element to access /return A pointer to the unique atomic feasible subset that contains the element

5.24.3.3 Partition * MultiSet::getOptimalPartition (double *parameter*) [virtual]

Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using getOptimalPartition (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Returns

: The resulting optimal partition

Implements [AbstractSet](#).

5.24.3.4 MultiSubset * MultiSet::getRandomAtomicMultiSubset ()

Access to a random element (e.g., atomic feasible subset) /return A pointer to the unique atomic feasible subset that contains the element

5.24.3.5 void MultiSet::printOptimalPartition (double *parameter*) [virtual]

Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using getOptimalPartition (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.24.3.6 void MultiSet::setObjectiveFunction (ObjectiveFunction * *objective*) [virtual]

Set the objective that one wants to optimise.

Parameters

<i>objective</i>	: The objective function itself
------------------	---------------------------------

Implements [AbstractSet](#).

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

- /home/lamarche/programming/optimal_partition/src/[multi_set.hpp](#)
- /home/lamarche/programming/optimal_partition/src/[multi_set.cpp](#)

5.25 MultiSubset Class Reference

Public Member Functions

- **MultiSubset** ([UniSubset](#) **uniSubsetArray, int dimension)
- void **print** ()
- void **printIndexSet** (bool endl=false)
- void **addMultiSubsetSet** (MultiSubsetSet *multiSubsetSet)
- void **setObjectiveFunction** ([ObjectiveFunction](#) *m)
- void **buildDataStructure** ()
- void **computeObjectiveValues** ()
- void **normalizeObjectiveValues** ([ObjectiveValue](#) *maxQual=0)
- void **printObjectiveValues** ()
- void **computeOptimalPartition** (double parameter)
- void **printOptimalPartition** (double parameter)
- void **buildOptimalPartition** ([Partition](#) *partition)

Public Attributes

- int **dimension**
- [UniSubset](#) ** **uniSubsetArray**
- int **num**
- int **atomicNum**
- bool **isAtomic**
- bool **reached**
- MultiSubsetSetSet * **multiSubsetSetSet**
- [MultiSet](#) * **multiSet**
- [ObjectiveFunction](#) * **objective**
- [ObjectiveValue](#) * **value**
- double **optimalValue**
- MultiSubsetSet * **optimalCut**

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

- /home/lamarche/programming/optimal_partition/src/[multi_set.hpp](#)
- /home/lamarche/programming/optimal_partition/src/[multi_set.cpp](#)

5.26 NHONode Class Reference

Public Member Functions

- **HONode** (int size, int index=-1)
- int **getIndex** (int i, int j)
- void **addChild** ([HONode](#) *node)
- void **setObjectiveFunction** ([ObjectiveFunction](#) *m)
- void **print** ()
- void **buildDataStructure** (int level=0)
- void **computeObjectiveValues** ()
- void **normalizeObjectiveValues** ([ObjectiveValue](#) *maxQual=0)
- void **printObjectiveValues** ()
- void **computeOptimalPartition** (double parameter)
- void **printOptimalPartition** (double parameter)
- [Partition](#) * **getOptimalPartition** (double parameter)
- void **buildOptimalPartition** ([Partition](#) *partition, int pi=0, int pj=-1)

Public Attributes

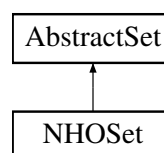
- int **index**
- int **level**
- std::set< int > * **indices**
- HONodeSet * **children**
- [ObjectiveFunction](#) * **objective**
- int **size**
- [ObjectiveValue](#) ** **qualities**
- double * **optimalValues**
- int * **optimalCuts**

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

- /home/lamarche/programming/optimal_partition/src/NHO_set.hpp

5.27 NHOSet Class Reference

Inheritance diagram for NHOSet:



Public Member Functions

- **NHOSet** (int NSize, [HNode](#) *HHierarchy, int OSize)
- void [setRandom](#) ()
Randomly set the algebraic constraints for quick experiments (warning: this method is not always implemented)
- void [setObjectiveFunction](#) ([ObjectiveFunction](#) *m)
Set the objective that one wants to optimise.
- void [print](#) ()
Print the set and its algebraic constraints.
- void [buildDataStructure](#) ()
Build a proper data structure to represent the set and its algebraic constraints (warning: this method should always be called after instantiating and parameterising a set, and before calling any other method, such as [print\(\)](#), [computeObjectiveValues\(\)](#), [computeOptimalPartition](#) (double parameter), etc.)
- void [computeObjectiveValues](#) ()
*Compute the value of the objective function for each feasible part (warning: setObjectiveFunction ([ObjectiveFunction](#) *objective) should have been called first)*
- void [normalizeObjectiveValues](#) ()
Finish computing the value of the objective function for each feasible part when normalisation is required (warning: only after [computeObjectiveValues\(\)](#) has been called)
- void [printObjectiveValues](#) ()
Print the value of the objective function for each feasible part.
- void [computeOptimalPartition](#) (double parameter)
Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.
- void [printOptimalPartition](#) (double parameter)

Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition` (double parameter) and by calling `print()` on the result)

- `Partition * getOptimalPartition` (double parameter)

Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition` (double parameter) and by calling `print()` on the result)

Public Attributes

- `int NSize`
- `HNode * HHierarchy`
- `int OSize`
- `NHODatatree * NHODatatree`

5.27.1 Member Function Documentation

5.27.1.1 `void NHOSet::computeOptimalPartition (double parameter) [virtual]`

Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.

Parameters

<code>parameter</code>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------------	--

Implements [AbstractSet](#).

5.27.1.2 `Partition* NHOSet::getOptimalPartition (double parameter) [virtual]`

Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition` (double parameter) and by calling `print()` on the result)

Parameters

<code>parameter</code>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------------	--

Returns

: The resulting optimal partition

Implements [AbstractSet](#).

5.27.1.3 `void NHOSet::printOptimalPartition (double parameter) [virtual]`

Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition` (double parameter) and by calling `print()` on the result)

Parameters

<code>parameter</code>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------------	--

Implements [AbstractSet](#).

5.27.1.4 void NHOSet::setObjectiveFunction (ObjectiveFunction * *objective*) [virtual]

Set the objective that one wants to optimise.

Parameters

<i>objective</i>	: The objective function itself
------------------	---------------------------------

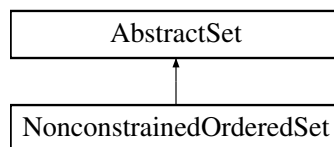
Implements [AbstractSet](#).

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

- /home/lamarche/programming/optimal_partition/src/NHO_set.hpp

5.28 NonconstrainedOrderedSet Class Reference

Inheritance diagram for NonconstrainedOrderedSet:



Public Member Functions

- **NonconstrainedOrderedSet** (int size1, int size2)
- void **setRandom** ()
Randomly set the algebraic constraints for quick experiments (warning: this method is not always implemented)
- void **setObjectiveFunction** ([ObjectiveFunction](#) *m)
Set the objective that one wants to optimise.
- void **print** ()
Print the set and its algebraic constraints.
- void **buildDataStructure** ()
Build a proper data structure to represent the set and its algebraic constraints (warning: this method should always be called after instantiating and parameterising a set, and before calling any other method, such as [print\(\)](#), [computeObjectiveValues\(\)](#), [computeOptimalPartition](#) (double parameter), etc.)
- void **computeObjectiveValues** ()
*Compute the value of the objective function for each feasible part (warning: [setObjectiveFunction](#) ([ObjectiveFunction](#) *objective) should have been called first)*
- void **normalizeObjectiveValues** ()
Finish computing the value of the objective function for each feasible part when normalisation is required (warning: only after [computeObjectiveValues\(\)](#) has been called)
- void **printObjectiveValues** ()
Print the value of the objective function for each feasible part.
- void **computeOptimalPartition** (double parameter)
Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.
- void **printOptimalPartition** (double parameter)
Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)
- [Partition](#) * **getOptimalPartition** (double parameter)
Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)

Public Attributes

- int **size1**
- int **size2**
- [OrderedDatatree](#) * **dataTree**
- [ObjectiveValue](#) ** **qualities**

5.28.1 Member Function Documentation

5.28.1.1 void NonconstrainedOrderedSet::computeOptimalPartition (double *parameter*) [virtual]

Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.28.1.2 Partition * NonconstrainedOrderedSet::getOptimalPartition (double *parameter*) [virtual]

Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using getOptimalPartition (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Returns

: The resulting optimal partition

Implements [AbstractSet](#).

5.28.1.3 void NonconstrainedOrderedSet::printOptimalPartition (double *parameter*) [virtual]

Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using getOptimalPartition (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.28.1.4 void NonconstrainedOrderedSet::setObjectiveFunction (ObjectiveFunction * *objective*) [virtual]

Set the objective that one wants to optimise.

Parameters

<i>objective</i>	: The objective function itself
------------------	---------------------------------

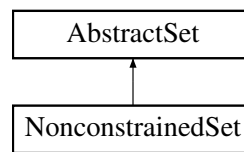
Implements [AbstractSet](#).

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

- /home/lamarche/programming/optimal_partition/src/nonconstrained_ordered_set.hpp
- /home/lamarche/programming/optimal_partition/src/nonconstrained_ordered_set.cpp

5.29 NonconstrainedSet Class Reference

Inheritance diagram for NonconstrainedSet:



Public Member Functions

- **NonconstrainedSet** (int size)
- void **setRandom** ()
Randomly set the algebraic constraints for quick experiments (warning: this method is not always implemented)
- void **setObjectiveFunction** ([ObjectiveFunction](#) *m)
Set the objective that one wants to optimise.
- void **print** ()
Print the set and its algebraic constraints.
- void **printDataTree** (bool verbose=true)
- void **printParts** ()
- int **printPartitions** (bool [print](#)=true)
- void **buildDataStructure** ()
Build a proper data structure to represent the set and its algebraic constraints (warning: this method should always be called after instantiating and parameterising a set, and before calling any other method, such as [print\(\)](#), [computeObjectiveValues\(\)](#), [computeOptimalPartition](#) (double parameter), etc.)
- void **computeObjectiveValues** ()
*Compute the value of the objective function for each feasible part (warning: setObjectiveFunction ([ObjectiveFunction](#) *objective) should have been called first)*
- void **normalizeObjectiveValues** ()
Finish computing the value of the objective function for each feasible part when normalisation is required (warning: only after [computeObjectiveValues\(\)](#) has been called)
- void **printObjectiveValues** ()
Print the value of the objective function for each feasible part.
- void **computeOptimalPartition** (double parameter)
Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.
- void **printOptimalPartition** (double parameter)
Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)
- [Partition](#) * **getOptimalPartition** (double parameter)
Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)

Public Attributes

- int **size**
- [Datatree](#) * **dataTree**
- [ObjectiveValue](#) ** **qualities**

5.29.1 Member Function Documentation

5.29.1.1 void NonconstrainedSet::computeOptimalPartition (double *parameter*) [virtual]

Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.29.1.2 Partition * NonconstrainedSet::getOptimalPartition (double *parameter*) [virtual]

Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using getOptimalPartition (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Returns

: The resulting optimal partition

Implements [AbstractSet](#).

5.29.1.3 void NonconstrainedSet::printOptimalPartition (double *parameter*) [virtual]

Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using getOptimalPartition (double parameter) and by calling [print\(\)](#) on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.29.1.4 void NonconstrainedSet::setObjectiveFunction (ObjectiveFunction * *objective*) [virtual]

Set the objective that one wants to optimise.

Parameters

<i>objective</i>	: The objective function itself
------------------	---------------------------------

Implements [AbstractSet](#).

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

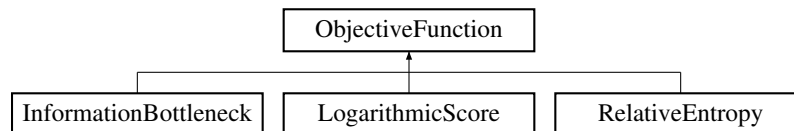
- /home/lamarche/programming/optimal_partition/src/nonconstrained_set.hpp
- /home/lamarche/programming/optimal_partition/src/nonconstrained_set.cpp

5.30 ObjectiveFunction Class Reference

Abstract class defining an objective function to be associated to a constrained set in order to define the optimisation problem that one wants to solve.

```
#include <objective_function.hpp>
```

Inheritance diagram for ObjectiveFunction:



Public Member Functions

- [ObjectiveFunction](#) ()
True if one deals with a maximisation problem, and false if one deals with a minimisation problem.
- virtual [~ObjectiveFunction](#) ()
Destructor.
- virtual void [setRandom](#) ()=0
Randomly set the initial data from which the objective function is computed.
- virtual void [computeObjectiveValues](#) ()=0
This method is called by child classes of [AbstractSet](#) (do not use directly)
- virtual void [printObjectiveValues](#) (bool verbose=true)=0
This method is called by child classes of [AbstractSet](#) (do not use directly)
- virtual [ObjectiveValue](#) * [newObjectiveValue](#) (int index=-1)=0
This method is called by child classes of [AbstractSet](#) (do not use directly)

Public Attributes

- bool **maximize**

Friends

- class **AbstractSet**

5.30.1 Detailed Description

Abstract class defining an objective function to be associated to a constrained set in order to define the optimisation problem that one wants to solve.

5.30.2 Constructor & Destructor Documentation

5.30.2.1 ObjectiveFunction::ObjectiveFunction ()

True if one deals with a maximisation problem, and false if one deals with a minimisation problem.

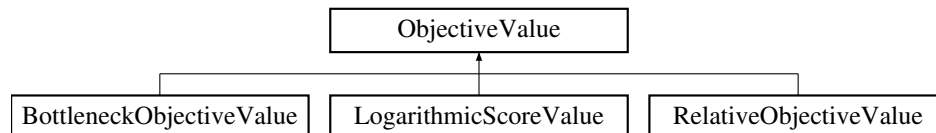
Constructor

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

- /home/lamarche/programming/optimal_partition/src/objective_function.hpp
- /home/lamarche/programming/optimal_partition/src/objective_function.cpp

5.31 ObjectiveValue Class Reference

Inheritance diagram for ObjectiveValue:



Public Member Functions

- virtual void **add** ([ObjectiveValue](#) *value)=0
- virtual void **compute** ()=0
- virtual void **compute** ([ObjectiveValue](#) *value1, [ObjectiveValue](#) *value2)=0
- virtual void **compute** ([ObjectiveValueSet](#) *valueSet)=0
- virtual void **normalize** ([ObjectiveValue](#) *normalizingValue)=0
- virtual double **getValue** (double param)=0
- virtual void **print** (bool verbose=true)=0

Public Attributes

- [ObjectiveFunction](#) * **objective**

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

- /home/lamarche/programming/optimal_partition/src/objective_function.hpp
- /home/lamarche/programming/optimal_partition/src/objective_function.cpp

5.32 OrderedDatatree Class Reference

Public Member Functions

- **OrderedDatatree** ([OrderedDatatree](#) &tree)
- **OrderedDatatree** (int size2, int vertex=-1)
- void **setObjectiveFunction** ([ObjectiveFunction](#) *objective)
- Vertices * **getAllVertices** ()
- int **getIndex** (int i, int j)
- [OrderedDatatree](#) * **addChild** (int v, bool print=true)
- [OrderedDatatree](#) * **findChild** (int v)
- [OrderedDatatree](#) * **findOrAddChild** (int v, bool print=true)
- void **addBipartition** ([OrderedDatatree](#) *n1, [OrderedDatatree](#) *n2)
- void **computeObjectiveValues** ()
- void **normalizeObjectiveValues** ([ObjectiveValue](#) *maxObjectiveValue=0)
- void **printObjectiveValues** ()
- void **buildOptimalPartition** ([Partition](#) *partition, int pi=0, int pj=-1)
- void **computeOptimalPartition** (double parameter)

- void **printOptimalPartition** (double parameter)
- [Partition](#) * **getOptimalPartition** (double parameter)
- void **print** (bool verbose=false)
- void **printVertices** (bool endl=true)

Public Attributes

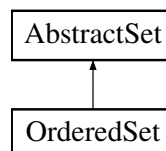
- int **size1**
- int **size2**
- int **vertex**
- bool **wholeSet**
- [ObjectiveFunction](#) * **objective**
- [OrderedDatatree](#) * **parent**
- [OrderedDatatree](#) * **complement**
- [OrderedTreesList](#) * **complementList**
- [OrderedTreesSet](#) * **children**
- [OrderedBipartitionsSet](#) * **bipartitions**
- [ObjectiveValue](#) ** **qualities**
- double * **optimalValues**
- int * **optimalCuts**
- [OrderedBipartition](#) ** **optimalBipartitions**
- bool **optimized**

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

- /home/lamarche/programming/optimal_partition/src/datatree.hpp
- /home/lamarche/programming/optimal_partition/src/datatree.cpp

5.33 OrderedSet Class Reference

Inheritance diagram for OrderedSet:



Public Member Functions

- **OrderedSet** (int s)
- int **getIndex** (int i, int j)
- void **setRandom** ()
Randomly set the algebraic constraints for quick experiments (warning: this method is not always implemented)
- void **setObjectiveFunction** ([ObjectiveFunction](#) *m)
Set the objective that one wants to optimise.
- void **print** ()
Print the set and its algebraic constraints.
- void **buildDataStructure** ()
Build a proper data structure to represent the set and its algebraic constraints (warning: this method should always be called after instantiating and parameterising a set, and before calling any other method, such as [print\(\)](#), [computeObjectiveValues\(\)](#), [computeOptimalPartition](#) (double parameter), etc.)

- void `computeObjectiveValues` ()
*Compute the value of the objective function for each feasible part (warning: `setObjectiveFunction` ([ObjectiveFunction](#) *objective) should have been called first)*
- void `normalizeObjectiveValues` ()
Finish computing the value of the objective function for each feasible part when normalisation is required (warning: only after `computeObjectiveValues()` has been called)
- void `printObjectiveValues` ()
Print the value of the objective function for each feasible part.
- void `computeOptimalPartition` (double parameter)
Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.
- void `printOptimalPartition` (double parameter)
Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition` (double parameter) and by calling `print()` on the result)
- [Partition](#) * `getOptimalPartition` (double parameter)
Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition` (double parameter) and by calling `print()` on the result)

Public Attributes

- int **size**
- [ObjectiveValue](#) ** **qualities**
- double * **optimalValues**
- int * **optimalCuts**

5.33.1 Member Function Documentation

5.33.1.1 void `OrderedSet::computeOptimalPartition` (double *parameter*) [virtual]

Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.33.1.2 [Partition](#) * `OrderedSet::getOptimalPartition` (double *parameter*) [virtual]

Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition` (double parameter) and by calling `print()` on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Returns

: The resulting optimal partition

Implements [AbstractSet](#).

5.33.1.3 void OrderedSet::printOptimalPartition (double *parameter*) [virtual]

Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition (double parameter)` and by calling `print()` on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.33.1.4 void OrderedSet::setObjectiveFunction (ObjectiveFunction * *objective*) [virtual]

Set the objective that one wants to optimise.

Parameters

<i>objective</i>	: The objective function itself
------------------	---------------------------------

Implements [AbstractSet](#).

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

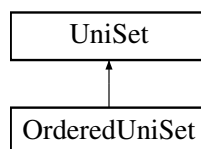
- /home/lamarche/programming/optimal_partition/src/orderedset.hpp
- /home/lamarche/programming/optimal_partition/src/orderedset.cpp

5.34 OrderedUniSet Class Reference

A uni-dimensional set of elements with a total order, and such that the feasible subsets are all the intervals induced by this order.

```
#include <uni_set.hpp>
```

Inheritance diagram for OrderedUniSet:



Public Member Functions

- [OrderedUniSet](#) (int size)
Number of ordered elements.

Public Attributes

- int **size**

Additional Inherited Members

5.34.1 Detailed Description

A uni-dimensional set of elements with a total order, and such that the feasible subsets are all the intervals induced by this order.

5.34.2 Constructor & Destructor Documentation

5.34.2.1 OrderedUniSet::OrderedUniSet (int *size*)

Number of ordered elements.

Constructor

Parameters

<i>size</i>	: Number of ordered elements
-------------	------------------------------

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

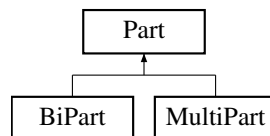
- [/home/lamarche/programming/optimal_partition/src/uni_set.hpp](#)
- [/home/lamarche/programming/optimal_partition/src/uni_set.cpp](#)

5.35 Part Class Reference

A part is a subset of a set of elements (individuals) represented by integers.

```
#include <partition.hpp>
```

Inheritance diagram for Part:



Public Member Functions

- **Part** ([ObjectiveValue](#) *value=0)
- **Part** ([Part](#) *part)
- **Part** ([Datatree](#) *node, [ObjectiveValue](#) *value=0)
- void **addIndividual** (int i, bool front=false)
- Vertices * **getVertices** ()
- virtual bool **equal** ([Part](#) *p)
- virtual void **print** (bool endl=false)
- virtual int **printSize** ()

Public Attributes

- std::list< int > * **individuals**
- [ObjectiveValue](#) * **value**

5.35.1 Detailed Description

A part is a subset of a set of elements (individuals) represented by integers.

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

- [/home/lamarche/programming/optimal_partition/src/partition.hpp](#)
- [/home/lamarche/programming/optimal_partition/src/partition.cpp](#)

5.36 Partition Class Reference

A partition is a collection of pairwise-disjoint and covering subsets (parts) of a set of elements.

```
#include <partition.hpp>
```

Public Member Functions

- **Partition** ([ObjectiveFunction](#) *objective=0, double parameter=0)
- **Partition** ([Partition](#) *partition)
- void **addPart** ([Part](#) *p, bool front=false)
- bool **equal** ([Partition](#) *p)
- void **print** (bool endl=false)

Public Attributes

- double **parameter**
- std::list< [Part](#) * > * **parts**
- [ObjectiveValue](#) * **value**

5.36.1 Detailed Description

A partition is a collection of pairwise-disjoint and covering subsets (parts) of a set of elements.

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

- /home/lamarche/programming/optimal_partition/src/partition.hpp
- /home/lamarche/programming/optimal_partition/src/partition.cpp

5.37 PredictionDataset Class Reference

Class to represent a data set for the prediction of a post-measurement from the knowledge of a pre-measurement (composed of a train set and a test set)

```
#include <prediction_dataset.hpp>
```

Public Member Functions

- [PredictionDataset](#) ([MultiSet](#) *preMeasurement, [MultiSet](#) *postMeasurement)
Constructor.
- [~PredictionDataset](#) ()
Destructor.
- void **addTrainValue** ([MultiSubset](#) *preValue, [MultiSubset](#) *postValue, int count=1)
Add a couple of (pre and post) observations to the train set.
- void **addTestValue** ([MultiSubset](#) *preValue, [MultiSubset](#) *postValue, int count=1)
Add a couple of (pre and post) observations to the test set.
- void **addTrainValue** (int preIndex, int postIndex, int count=1)
*Add a couple of (pre and post) observations to the train set **in the case of one-dimensional measurements***
- void **addTestValue** (int preIndex, int postIndex, int count=1)
*Add a couple of (pre and post) observations to the test set **in the case of one-dimensional measurements***

Public Attributes

- [MultiSet](#) * **preMultiSet**
- [MultiSet](#) * **postMultiSet**
The pre-measurement modelled as a structured multi-dimensional set of elements (the possible observation values)
- `std::vector< MultiSubset * > * trainPreValues`
The post-measurement modelled as a structured multi-dimensional set of elements (the possible observation values)
- `std::vector< MultiSubset * > * trainPostValues`
A vector of pre-observations that are used to train the predictor.
- `std::vector< int > * trainCountValues`
A vector of post-observations that are used to train the predictor.
- `std::vector< MultiSubset * > * testPreValues`
- `std::vector< MultiSubset * > * testPostValues`
A vector of pre-observations that are used to test the predictor.
- `std::vector< int > * testCountValues`
A vector of post-observations that are used to test the predictor.

5.37.1 Detailed Description

Class to represent a data set for the prediction of a post-measurement from the knowledge of a pre-measurement (composed of a train set and a test set)

5.37.2 Constructor & Destructor Documentation

5.37.2.1 PredictionDataset::PredictionDataset ([MultiSet](#) * *preMeasurement*, [MultiSet](#) * *postMeasurement*)

Constructor.

Parameters

<i>preMeasurement</i>	: The pre-measurement modelled as a structured multi-dimensional set of elements (the possible observation values)
<i>post-Measurement</i>	: The post-measurement modelled as a structured multi-dimensional set of elements (the possible observation values)

5.37.3 Member Function Documentation

5.37.3.1 void PredictionDataset::addTestValue ([MultiSubset](#) * *preValue*, [MultiSubset](#) * *postValue*, int *count* = 1)

Add a couple of (pre and post) observations to the test set.

Parameters

<i>preValue</i>	: A pointer to the feasible subset that has been pre-observed in the multi-dimensional set representing the pre-measurement. It should always be an element of the set, that is an atomic feasible subset
<i>preValue</i>	: A pointer to the feasible subset that has been post-observed in the multi-dimensional set representing the pre-measurement. It should always be an element of the set, that is an atomic feasible subset

<i>count</i>	: The number of times the couple has been observed
--------------	--

5.37.3.2 void PredictionDataset::addTestValue (int *preIndex*, int *postIndex*, int *count* = 1)

Add a couple of (pre and post) observations to the test set **in the case of one-dimensional measurements**

Parameters

<i>preIndex</i>	: The index of the element that has been pre-observed in the one-dimensional set representing the pre-measurement
<i>postIndex</i>	: The index of the element that has been post-observed in the one-dimensional set representing the pre-measurement
<i>count</i>	: The number of times the couple has been observed

5.37.3.3 void PredictionDataset::addTrainValue (MultiSubset * *preValue*, MultiSubset * *postValue*, int *count* = 1)

Add a couple of (pre and post) observations to the train set.

Parameters

<i>preValue</i>	: A pointer to the feasible subset that have been pre-observed in the multi-dimensional set representing the pre-measurement. It should always be an element of the set, that is an atomic feasible subset
<i>preValue</i>	: A pointer to the feasible subset that have been post-observed in the multi-dimensional set representing the pre-measurement. It should always be an element of the set, that is an atomic feasible subset
<i>count</i>	: The number of times the couple has been observed

5.37.3.4 void PredictionDataset::addTrainValue (int *preIndex*, int *postIndex*, int *count* = 1)

Add a couple of (pre and post) observations to the train set **in the case of one-dimensional measurements**

Parameters

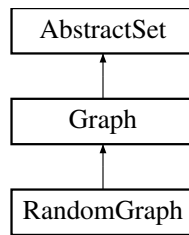
<i>preIndex</i>	: The index of the element that has been pre-observed in the one-dimensional set representing the pre-measurement
<i>postIndex</i>	: The index of the element that has been post-observed in the one-dimensional set representing the pre-measurement
<i>count</i>	: The number of times the couple has been observed

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

- /home/lamarche/programming/optimal_partition/src/prediction_dataset.hpp
- /home/lamarche/programming/optimal_partition/src/prediction_dataset.cpp

5.38 RandomGraph Class Reference

Inheritance diagram for RandomGraph:



Public Member Functions

- **RandomGraph** (int vNum, int eNum)

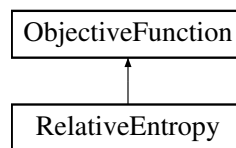
Additional Inherited Members

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

- /home/lamarche/programming/optimal_partition/src/graph.hpp
- /home/lamarche/programming/optimal_partition/src/graph.cpp

5.39 RelativeEntropy Class Reference

Inheritance diagram for RelativeEntropy:



Public Member Functions

- **RelativeEntropy** (int size, double *values=0, double *refValues=0)
- void **setRandom** ()
Randomly set the initial data from which the objective function is computed.
- **ObjectiveValue** * **newObjectiveValue** (int index=-1)
This method is called by child classes of [AbstractSet](#) (do not use directly)
- void **computeObjectiveValues** ()
This method is called by child classes of [AbstractSet](#) (do not use directly)
- void **printObjectiveValues** (bool verbose=true)
This method is called by child classes of [AbstractSet](#) (do not use directly)
- double **getParameter** (double unit)
- double **getUnitDistance** (double uMin, double uMax)
- double **getIntermediaryUnit** (double uMin, double uMax)

Public Attributes

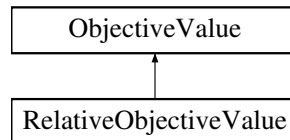
- int **size**
- double * **values**
- double * **refValues**

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

- /home/lamarche/programming/optimal_partition/src/relative_entropy.hpp
- /home/lamarche/programming/optimal_partition/src/relative_entropy.cpp

5.40 RelativeObjectiveValue Class Reference

Inheritance diagram for RelativeObjectiveValue:



Public Member Functions

- **RelativeObjectiveValue** ([RelativeEntropy](#) *objective, int index=-1)
- void **add** ([ObjectiveValue](#) *value)
- void **compute** ()
- void **compute** ([ObjectiveValue](#) *value1, [ObjectiveValue](#) *value2)
- void **compute** ([ObjectiveValueSet](#) *valueset)
- void **normalize** ([ObjectiveValue](#) *q)
- void **print** (bool verbose=true)
- double **getValue** (double param)

Public Attributes

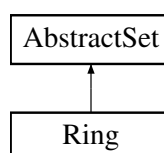
- int **index**
- double **sumValue**
- double **sumRefValue**
- double **microInfo**
- double **divergence**
- double **sizeReduction**

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

- /home/lamarche/programming/optimal_partition/src/relative_entropy.hpp
- /home/lamarche/programming/optimal_partition/src/relative_entropy.cpp

5.41 Ring Class Reference

Inheritance diagram for Ring:



Public Member Functions

- **Ring** (int s)
- int **getIndex** (int i, int j)
- void **setObjectiveFunction** ([ObjectiveFunction](#) *m)
Set the objective that one wants to optimise.
- void **setRandom** ()
Randomly set the algebraic constraints for quick experiments (warning: this method is not always implemented)
- void **print** ()
Print the set and its algebraic constraints.
- void **buildDataStructure** ()
Build a proper data structure to represent the set and its algebraic constraints (warning: this method should always be called after instantiating and parameterising a set, and before calling any other method, such as [print\(\)](#), [computeObjectiveValues\(\)](#), [computeOptimalPartition](#) (double parameter), etc.)
- void **computeObjectiveValues** ()
*Compute the value of the objective function for each feasible part (warning: [setObjectiveFunction](#) ([ObjectiveFunction](#) *objective) should have been called first)*
- void **normalizeObjectiveValues** ()
Finish computing the value of the objective function for each feasible part when normalisation is required (warning: only after [computeObjectiveValues\(\)](#) has been called)
- void **printObjectiveValues** ()
Print the value of the objective function for each feasible part.
- void **computeOptimalPartition** (double parameter)
Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.
- void **printOptimalPartition** (double parameter)
Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)
- [Partition](#) * **getOptimalPartition** (double parameter)
Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using [getOptimalPartition](#) (double parameter) and by calling [print\(\)](#) on the result)
- [AbstractSet](#) * **getRandomSet** (int size)

Public Attributes

- int **size**
- double * **values**
- double * **sumValues**
- double * **microInfos**
- double * **sizeReductions**
- double * **divergences**
- double * **optimalQualities**
- int * **optimalCuts**
- int **firstOptimalCut**
- int **lastOptimalCut**

5.41.1 Member Function Documentation

5.41.1.1 void [Ring::computeOptimalPartition](#) (double *parameter*) [virtual]

Compute a partition that fits with the algebraic constraints and that optimises the objective function that has been specified.

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.41.1.2 Partition * Ring::getOptimalPartition (double *parameter*) [virtual]

Compute and return a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition (double parameter)` and by calling `print()` on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Returns

: The resulting optimal partition

Implements [AbstractSet](#).

5.41.1.3 void Ring::printOptimalPartition (double *parameter*) [virtual]

Compute and print a partition that fits with the algebraic constraints and that optimises the objective function that has been specified (warning: this method is not always implemented, but one can obtain a similar result by using `getOptimalPartition (double parameter)` and by calling `print()` on the result)

Parameters

<i>parameter</i>	: The parameter of the objective function to be optimised (if the objective is parametrised)
------------------	--

Implements [AbstractSet](#).

5.41.1.4 void Ring::setObjectiveFunction (ObjectiveFunction * *objective*) [virtual]

Set the objective that one wants to optimise.

Parameters

<i>objective</i>	: The objective function itself
------------------	---------------------------------

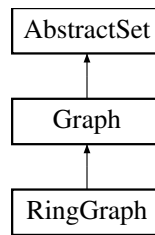
Implements [AbstractSet](#).

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

- /home/lamarche/programming/optimal_partition/src/ring.hpp
- /home/lamarche/programming/optimal_partition/src/ring.cpp

5.42 RingGraph Class Reference

Inheritance diagram for RingGraph:



Public Member Functions

- **RingGraph** (int vNum)

Additional Inherited Members

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

- /home/lamarche/programming/optimal_partition/src/graph.hpp
- /home/lamarche/programming/optimal_partition/src/graph.cpp

5.43 Timer Class Reference

Public Member Functions

- **Timer** (char *file=0, int dimension=1, bool append=false)
- void **start** (int size, std::string text="")
- void **start** (std::vector< int > parameters, std::string text="")
- void **startTime** ()
- void **startMemory** ()
- void **stop** (std::string text="")
- void **stopTime** ()
- void **stopMemory** ()
- void **step** (std::string text="")
- void **print** (char *fName)

Public Attributes

- float **time**
- int **memory**

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

- /home/lamarche/programming/optimal_partition/src/timer.hpp
- /home/lamarche/programming/optimal_partition/src/timer.cpp

5.44 TreeToAdd Struct Reference

Public Attributes

- [Datatree](#) * **node**
- [Datatree](#) * **nodeToAdd**

- Vertices * **vertices**
- Vertices * **adjVertices**

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

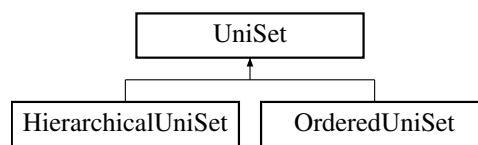
- /home/lamarche/programming/optimal_partition/src/graph.cpp

5.45 UniSet Class Reference

A uni-dimensional set of elements and its algebraic structure (feasible subsets and feasible refinements)

```
#include <uni_set.hpp>
```

Inheritance diagram for UniSet:



Public Member Functions

- [UniSet \(UniSubset *firstUniSubset\)](#)
Constructor.
- [~UniSet \(\)](#)
Destructor.
- void [buildDataStructure \(\)](#)
Build a proper data structure to represent the uni-dimensional set of elements and its algebraic structure (warning: this method should be called after construction, and before actually using the set)
- void [print \(\)](#)
Print the current state of the set and its algebraic structure.

Protected Member Functions

- void [initReached \(\)](#)
Top subset in the lattice of feasible subsets (assumed to be unique and to include all feasible subsets)

Protected Attributes

- int **atomicUniSubsetNumber**
- int [uniSubsetNumber](#)
Number of elements (i.e., atomic feasible subsets)
- [UniSubset ** atomicUniSubsetArray](#)
Number of feasible subsets.
- [UniSubset ** uniSubsetArray](#)
Array of pointers to all elements (i.e., atomic feasible subsets)
- [UniSubset * firstUniSubset](#)
Array of pointers to all feasible subsets.

5.45.1 Detailed Description

A uni-dimensional set of elements and its algebraic structure (feasible subsets and feasible refinements)

5.45.2 Constructor & Destructor Documentation

5.45.2.1 UniSet::UniSet (UniSubset * firstUniSubset)

Constructor.

Parameters

<i>firstUniSubset</i>	: Top subset in the lattice of feasible subsets
-----------------------	---

5.45.3 Member Function Documentation

5.45.3.1 void UniSet::initReached () [protected]

Top subset in the lattice of feasible subsets (assumed to be unique and to include all feasible subsets)

Initialise the `reached` field of all feasible subsets to `false` (used by other methods to run through the algebraic structure in a recursive fashion without considering twice the same subset)

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

- [/home/lamarche/programming/optimal_partition/src/uni_set.hpp](#)
- [/home/lamarche/programming/optimal_partition/src/uni_set.cpp](#)

5.46 UniSubset Class Reference

A feasible subset associated to a uni-dimensional set of elements ([UniSet](#))

```
#include <uni_set.hpp>
```

Public Member Functions

- [UniSubset](#) (int index=-1)
Set of the refinements of this subset, that is the set of all partitions of this subset that are made of other feasible subsets; this hence properly defines the algebraic structure.
- [~UniSubset](#) ()
Destructor.
- void [print](#) ()
Print the actual state of this subset.
- void [printIndexSet](#) (bool endl=false)
Print indexes of the elements in this subset.
- void [addUniSubsetSet](#) (UniSubsetSet *uniSubsetSet)
Add a refinement to this subset, that is a partition of this subset that is made of other feasible subsets.

Public Attributes

- [UniSet](#) * [uniSet](#)
- bool [isAtomic](#)
Pointer to the set of elements to which this subset is associated.

- int [atomicNum](#)
true if and only if this subset is actually an element of the associated set (i.e., an atomic feasible subset)
- int [num](#)
If this subset is an element (i.e., an atomic feasible subset), identifier of this subset among all the elements of the associated set; if not, always equal to -1
- IndexSet * [indexSet](#)
Identifier of this subset among all the feasible subsets of the associated set.
- UniSubsetSetSet * [uniSubsetSetSet](#)
Indexes of all the elements in this subset (only one index / one element in the case of an atomic subset)

5.46.1 Detailed Description

A feasible subset associated to a uni-dimensional set of elements ([UniSet](#))

5.46.2 Constructor & Destructor Documentation

5.46.2.1 UniSubset::UniSubset (int *index* = -1)

Set of the refinements of this subset, that is the set of all partitions of this subset that are made of other feasible subsets; this hence properly defines the algebraic structure.

Constructor

Parameters

<i>index</i>	: The index of the unique element in the case of an atomic subset
--------------	---

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

- /home/lamarche/programming/optimal_partition/src/[uni_set.hpp](#)
- /home/lamarche/programming/optimal_partition/src/[uni_set.cpp](#)

Chapter 6

File Documentation

6.1 /home/lamarche/programming/optimal_partition/src/abstract_set.hpp File Reference

Abstract class defining a set of elements that one wants to partition while optimising some (decomposable) objective and preserving some algebraic constraints (set of feasible parts)

```
#include "timer.hpp"
#include "objective_function.hpp"
#include "partition.hpp"
#include "dataset.hpp"
```

Classes

- class [AbstractSet](#)

Abstract class defining a set of elements that one wants to partition while optimising some (decomposable) objective and preserving some algebraic constraints (set of feasible parts)

6.1.1 Detailed Description

Abstract class defining a set of elements that one wants to partition while optimising some (decomposable) objective and preserving some algebraic constraints (set of feasible parts)

Author

Robin Lamarche-Perrin

Date

06/11/2015

6.2 /home/lamarche/programming/optimal_partition/src/logarithmic_score.hpp File Reference

Classes to define and compute the logarithmic score function in the case of point prediction.

```
#include "objective_function.hpp"
#include "prediction_dataset.hpp"
```

Classes

- class [LogarithmicScore](#)
Class to define and compute the logarithmic score function in the case of point prediction.
- class [LogarithmicScoreValue](#)

6.2.1 Detailed Description

Classes to define and compute the logarithmic score function in the case of point prediction.

Author

Robin Lamarche-Perrin

Date

06/11/2015

6.3 /home/lamarche/programming/optimal_partition/src/multi_set.hpp File Reference

Classes to represent multi-dimensional sets of elements and their algebraic structure (feasible subsets and feasible refinements)

```
#include <list>
#include "uni_set.hpp"
#include "abstract_set.hpp"
```

Classes

- class [MultiSet](#)
A multi-dimensional set of elements based on the Cartesian product of several uni-dimensional sets ([UniSet](#)) and their algebraic structures (feasible subsets and feasible refinements)
- class [MultiSubset](#)

Typedefs

- typedef std::list< [MultiSubset](#) * > **MultiSubsetSet**
- typedef std::list
 < MultiSubsetSet * > **MultiSubsetSetSet**

6.3.1 Detailed Description

Classes to represent multi-dimensional sets of elements and their algebraic structure (feasible subsets and feasible refinements)

Author

Robin Lamarche-Perrin

Date

06/11/2015

6.4 /home/lamarche/programming/optimal_partition/src/objective_function.hpp File Reference

Abstract class defining an objective function to be associated to a constrained set in order to define the optimisation problem that one wants to solve.

```
#include <set>
```

Classes

- class [ObjectiveFunction](#)

Abstract class defining an objective function to be associated to a constrained set in order to define the optimisation problem that one wants to solve.

- class [ObjectiveValue](#)

Typedefs

- typedef std::set
< [ObjectiveValue](#) * > **ObjectiveValueSet**

6.4.1 Detailed Description

Abstract class defining an objective function to be associated to a constrained set in order to define the optimisation problem that one wants to solve.

Author

Robin Lamarche-Perrin

Date

06/11/2015

6.5 /home/lamarche/programming/optimal_partition/src/prediction_dataset.hpp File Reference

Class to represent a data set for the prediction of a post-measurement from the knowledge of a pre-measurement (composed of a train set and a test set)

```
#include "multi_set.hpp"
```

Classes

- class [PredictionDataset](#)

Class to represent a data set for the prediction of a post-measurement from the knowledge of a pre-measurement (composed of a train set and a test set)

6.5.1 Detailed Description

Class to represent a data set for the prediction of a post-measurement from the knowledge of a pre-measurement (composed of a train set and a test set)

Author

Robin Lamarche-Perrin

Date

06/11/2015

6.6 /home/lamarche/programming/optimal_partition/src/uni_set.hpp File Reference

Some classes to represent uni-dimensional sets of elements and their algebraic structure (feasible subsets and feasible refinements)

```
#include <list>
#include "bi_set.hpp"
#include "multi_set.hpp"
```

Classes

- class [UniSet](#)
A uni-dimensional set of elements and its algebraic structure (feasible subsets and feasible refinements)
- class [OrderedUniSet](#)
A uni-dimensional set of elements with a total order, and such that the feasible subsets are all the intervals induced by this order.
- class [HierarchicalUniSet](#)
A uni-dimensional set of elements structured according to a complete binary hierarchy, and such that the feasible subsets are all the nodes of the hierarchy.
- class [UniSubset](#)
A feasible subset associated to a uni-dimensional set of elements ([UniSet](#))

Typedefs

- typedef std::list< [UniSubset](#) * > **UniSubsetSet**
- typedef std::list< UniSubsetSet * > **UniSubsetSetSet**
- typedef std::list< int > **IndexSet**

6.6.1 Detailed Description

Some classes to represent uni-dimensional sets of elements and their algebraic structure (feasible subsets and feasible refinements)

Author

Robin Lamarche-Perrin

Date

06/11/2015

Index

- ~AbstractSet
 - AbstractSet, 11
- ~MultiSet
 - MultiSet, 36
- /home/lamarche/programming/optimal_partition/src/abstract-
_set.hpp, 65
- /home/lamarche/programming/optimal_partition/src/logarithmic-
_score.hpp, 65
- /home/lamarche/programming/optimal_partition/src/multi-
_set.hpp, 66
- /home/lamarche/programming/optimal_partition/src/objective-
_function.hpp, 67
- /home/lamarche/programming/optimal_partition/src/prediction-
_dataset.hpp, 67
- /home/lamarche/programming/optimal_partition/src/uni-
_set.hpp, 68
- AbstractSet, 9
 - ~AbstractSet, 11
 - computeOptimalPartition, 11
 - getOptimalPartition, 11
 - getOptimalPartitionList, 11
 - printOptimalPartition, 11
 - printOptimalPartitionList, 13
 - printOptimalPartitionListInCSV, 13
 - setObjectiveFunction, 13
- addTestValue
 - PredictionDataset, 54, 55
- addTrainValue
 - PredictionDataset, 55
- BiPart, 13
- BiSet, 14
 - computeOptimalPartition, 15
 - getOptimalPartition, 15
 - printOptimalPartition, 15
 - setObjectiveFunction, 16
- BiSubset, 16
- BottleneckObjectiveValue, 17
- CompleteGraph, 17
- computeOptimalPartition
 - AbstractSet, 11
 - BiSet, 15
 - Graph, 22
 - HierarchicalHierarchicalSet, 25
 - HierarchicalOrderedSet, 26
 - HierarchicalSet, 28
 - MultiSet, 37
 - NHOSet, 41
 - NonconstrainedOrderedSet, 44
 - NonconstrainedSet, 46
 - OrderedSet, 50
 - Ring, 58
- DataPointStruct, 18
- Dataset, 18
- Datree, 19
- FiliformGraph, 20
- getAtomicMultiSubset
 - MultiSet, 38
- getOptimalPartition
 - AbstractSet, 11
 - BiSet, 15
 - Graph, 22
 - HierarchicalHierarchicalSet, 25
 - HierarchicalOrderedSet, 27
 - HierarchicalSet, 28
 - MultiSet, 38
 - NHOSet, 41
 - NonconstrainedOrderedSet, 44
 - NonconstrainedSet, 46
 - OrderedSet, 50
 - Ring, 59
- getOptimalPartitionList
 - AbstractSet, 11
- getRandomAtomicMultiSubset
 - MultiSet, 38
- Graph, 20
 - computeOptimalPartition, 22
 - getOptimalPartition, 22
 - printOptimalPartition, 22
 - setObjectiveFunction, 22
- GraphComponent, 22
- HHNode, 23
- HNode, 30
- HONode, 31
- HierarchicalHierarchicalSet, 24
 - computeOptimalPartition, 25
 - getOptimalPartition, 25
 - printOptimalPartition, 25
 - setObjectiveFunction, 25
- HierarchicalOrderedSet, 26
 - computeOptimalPartition, 26
 - getOptimalPartition, 27
 - printOptimalPartition, 27
 - setObjectiveFunction, 27

- HierarchicalSet, 27
 - computeOptimalPartition, 28
 - getOptimalPartition, 28
 - printOptimalPartition, 29
 - setObjectiveFunction, 29
- HierarchicalUniSet, 29
 - HierarchicalUniSet, 30
 - HierarchicalUniSet, 30
- InformationBottleneck, 32
- initReached
 - UniSet, 62
- LogarithmicScore, 32
 - LogarithmicScore, 33
 - LogarithmicScore, 33
- LogarithmicScoreValue, 33
- MultiPart, 34
- MultiSet, 35
 - ~MultiSet, 36
 - computeOptimalPartition, 37
 - getAtomicMultiSubset, 38
 - getOptimalPartition, 38
 - getRandomAtomicMultiSubset, 38
 - MultiSet, 36
 - MultiSet, 36
 - printOptimalPartition, 38
 - setObjectiveFunction, 38
- MultiSubset, 39
- NHONode, 39
- NHOSet, 40
 - computeOptimalPartition, 41
 - getOptimalPartition, 41
 - printOptimalPartition, 41
 - setObjectiveFunction, 41
- NonconstrainedOrderedSet, 43
 - computeOptimalPartition, 44
 - getOptimalPartition, 44
 - printOptimalPartition, 44
 - setObjectiveFunction, 44
- NonconstrainedSet, 45
 - computeOptimalPartition, 46
 - getOptimalPartition, 46
 - printOptimalPartition, 46
 - setObjectiveFunction, 46
- ObjectiveFunction, 47
 - ObjectiveFunction, 47
 - ObjectiveFunction, 47
- ObjectiveValue, 48
- OrderedDatatree, 48
- OrderedSet, 49
 - computeOptimalPartition, 50
 - getOptimalPartition, 50
 - printOptimalPartition, 50
 - setObjectiveFunction, 51
- OrderedUniSet, 51
- OrderedUniSet, 52
- OrderedUniSet, 52
- Part, 52
- Partition, 53
- PredictionDataset, 53
 - addTestValue, 54, 55
 - addTrainValue, 55
 - PredictionDataset, 54
 - PredictionDataset, 54
- printOptimalPartition
 - AbstractSet, 11
 - BiSet, 15
 - Graph, 22
 - HierarchicalHierarchicalSet, 25
 - HierarchicalOrderedSet, 27
 - HierarchicalSet, 29
 - MultiSet, 38
 - NHOSet, 41
 - NonconstrainedOrderedSet, 44
 - NonconstrainedSet, 46
 - OrderedSet, 50
 - Ring, 59
- printOptimalPartitionList
 - AbstractSet, 13
- printOptimalPartitionListInCSV
 - AbstractSet, 13
- RandomGraph, 55
- RelativeEntropy, 56
- RelativeObjectiveValue, 57
- Ring, 57
 - computeOptimalPartition, 58
 - getOptimalPartition, 59
 - printOptimalPartition, 59
 - setObjectiveFunction, 59
- RingGraph, 59
- setObjectiveFunction
 - AbstractSet, 13
 - BiSet, 16
 - Graph, 22
 - HierarchicalHierarchicalSet, 25
 - HierarchicalOrderedSet, 27
 - HierarchicalSet, 29
 - MultiSet, 38
 - NHOSet, 41
 - NonconstrainedOrderedSet, 44
 - NonconstrainedSet, 46
 - OrderedSet, 51
 - Ring, 59
- Timer, 60
- TreeToAdd, 60
- UniSet, 61
 - initReached, 62
 - UniSet, 62
 - UniSet, 62

UniSubset, [62](#)
UniSubset, [63](#)
UniSubset, [63](#)