

HNCO

0.21

Generated by Doxygen 1.9.1

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	7
3.1 Class List	7
4 Namespace Documentation	15
4.1 hnco Namespace Reference	15
4.1.1 Detailed Description	20
4.1.2 Typedef Documentation	20
4.1.2.1 sparse_bit_vector_t	20
4.1.3 Function Documentation	20
4.1.3.1 bm_add_columns()	20
4.1.3.2 bm_add_rows()	21
4.1.3.3 bm_identity() [1/2]	21
4.1.3.4 bm_identity() [2/2]	21
4.1.3.5 bm_invert()	22
4.1.3.6 bm_multiply()	22
4.1.3.7 bm_rank()	23
4.1.3.8 bm_row_echelon_form()	23
4.1.3.9 bm_set_column()	23
4.1.3.10 bm_solve()	24
4.1.3.11 bm_solve_upper_triangular()	25
4.1.3.12 bm_transpose() [1/2]	26
4.1.3.13 bm_transpose() [2/2]	26
4.1.3.14 bv_add() [1/2]	27
4.1.3.15 bv_add() [2/2]	27
4.1.3.16 bv_from_size_type()	28
4.1.3.17 bv_from_stream()	28
4.1.3.18 bv_from_string()	29
4.1.3.19 bv_from_vector_bool()	29
4.1.3.20 bv_to_size_type() [1/2]	29
4.1.3.21 bv_to_size_type() [2/2]	30
4.1.3.22 bv_to_vector_bool()	30
4.1.3.23 ensure()	31
4.1.3.24 is_in_range() [1/2]	31
4.1.3.25 is_in_range() [2/2]	32
4.1.3.26 load_from_archive()	32
4.1.3.27 perm_identity()	32
4.1.3.28 perm_random()	33

4.1.3.29 <code>save_to_archive()</code>	33
4.1.3.30 <code>sbv_flip()</code>	33
4.1.3.31 <code>sbv_is_valid()</code> [1/2]	34
4.1.3.32 <code>sbv_is_valid()</code> [2/2]	34
4.2 <code>hnco::algorithm</code> Namespace Reference	35
4.2.1 Detailed Description	37
4.2.2 Function Documentation	37
4.2.2.1 <code>pv_add()</code>	37
4.2.2.2 <code>pv_average()</code>	38
4.2.2.3 <code>pv_bound()</code>	38
4.2.2.4 <code>pv_init()</code>	38
4.2.2.5 <code>pv_sample()</code>	39
4.2.2.6 <code>pv_uniform()</code>	39
4.2.2.7 <code>pv_update()</code> [1/2]	40
4.2.2.8 <code>pv_update()</code> [2/2]	40
4.3 <code>hnco::algorithm::fast_efficient_p3</code> Namespace Reference	40
4.3.1 Detailed Description	41
4.4 <code>hnco::algorithm::walsh_moment</code> Namespace Reference	41
4.4.1 Detailed Description	41
4.5 <code>hnco::app</code> Namespace Reference	41
4.5.1 Detailed Description	42
4.6 <code>hnco::exception</code> Namespace Reference	42
4.6.1 Detailed Description	43
4.7 <code>hnco::function</code> Namespace Reference	43
4.7.1 Detailed Description	45
4.7.2 Function Documentation	45
4.7.2.1 <code>compute_fast_walsh_transform()</code>	45
4.7.2.2 <code>compute_walsh_transform()</code>	46
4.8 <code>hnco::function::controller</code> Namespace Reference	46
4.8.1 Detailed Description	47
4.9 <code>hnco::function::modifier</code> Namespace Reference	47
4.9.1 Detailed Description	47
4.10 <code>hnco::logging</code> Namespace Reference	48
4.10.1 Detailed Description	48
4.11 <code>hnco::map</code> Namespace Reference	48
4.11.1 Detailed Description	49
4.11.2 Typedef Documentation	49
4.11.2.1 <code>transvection_sequence_t</code>	50
4.11.3 Function Documentation	50
4.11.3.1 <code>ts_is_valid()</code> [1/2]	50
4.11.3.2 <code>ts_is_valid()</code> [2/2]	50
4.11.3.3 <code>ts_multiply()</code> [1/2]	51

4.11.3.4 ts_multiply() [2/2]	51
4.11.3.5 ts_random()	52
4.11.3.6 ts_random_commuting()	52
4.11.3.7 ts_random_disjoint()	53
4.11.3.8 ts_random_non_commuting()	53
4.11.3.9 ts_random_unique_destination()	54
4.11.3.10 ts_random_unique_source()	54
4.12 hnco::multiobjective Namespace Reference	55
4.12.1 Detailed Description	55
4.13 hnco::multiobjective::algorithm Namespace Reference	55
4.13.1 Detailed Description	56
4.13.2 Function Documentation	56
4.13.2.1 operator<()	56
4.14 hnco::multiobjective::app Namespace Reference	56
4.14.1 Detailed Description	57
4.15 hnco::multiobjective::function Namespace Reference	57
4.15.1 Detailed Description	57
4.15.2 Typedef Documentation	58
4.15.2.1 value_t	58
4.15.3 Function Documentation	58
4.15.3.1 dominates()	58
4.16 hnco::neighborhood Namespace Reference	58
4.16.1 Detailed Description	59
4.17 hnco::random Namespace Reference	59
4.17.1 Detailed Description	59
4.18 hnco::representation Namespace Reference	59
4.18.1 Detailed Description	60
4.18.2 Function Documentation	60
4.18.2.1 difference_is_safe()	60
5 Class Documentation	63
5.1 AbstractMaxSat Class Reference	63
5.1.1 Detailed Description	64
5.1.2 Member Function Documentation	64
5.1.2.1 load()	64
5.1.2.2 load_()	64
5.1.2.3 save()	65
5.1.2.4 save_()	65
5.1.3 Member Data Documentation	66
5.1.3.1 _expression	66
5.2 AdditiveGaussianNoise Class Reference	66
5.2.1 Detailed Description	67

5.3 AffineMap Class Reference	67
5.3.1 Detailed Description	68
5.3.2 Member Function Documentation	69
5.3.2.1 is_surjective()	69
5.3.2.2 load()	69
5.3.2.3 random()	69
5.3.2.4 save()	70
5.4 Algorithm Class Reference	70
5.4.1 Detailed Description	72
5.4.2 Member Function Documentation	72
5.4.2.1 finalize()	72
5.4.2.2 set_solution()	73
5.4.2.3 update_solution()	73
5.4.3 Member Data Documentation	73
5.4.3.1 _functions	73
5.5 Algorithm Class Reference	74
5.5.1 Detailed Description	75
5.5.2 Constructor & Destructor Documentation	75
5.5.2.1 Algorithm()	75
5.5.3 Member Data Documentation	75
5.5.3.1 _functions	75
5.6 AlgorithmFactory Class Reference	76
5.6.1 Detailed Description	76
5.6.2 Member Function Documentation	76
5.6.2.1 make()	76
5.7 AlgorithmFactory Class Reference	77
5.7.1 Detailed Description	77
5.7.2 Member Function Documentation	77
5.7.2.1 make()	77
5.8 BiasedCrossover Class Reference	78
5.8.1 Detailed Description	79
5.8.2 Member Function Documentation	79
5.8.2.1 recombine()	79
5.9 BmPbil< GibbsSampler > Class Template Reference	79
5.9.1 Detailed Description	82
5.9.2 Member Enumeration Documentation	82
5.9.2.1 anonymous enum	82
5.9.2.2 anonymous enum	83
5.9.3 Member Function Documentation	83
5.9.3.1 set_selection_size()	83
5.10 Cache Class Reference	83
5.10.1 Detailed Description	85

5.10.2 Constructor & Destructor Documentation	85
5.10.2.1 Cache()	85
5.10.3 Member Function Documentation	85
5.10.3.1 provides_incremental_evaluation()	85
5.11 CallCounter Class Reference	86
5.11.1 Detailed Description	87
5.12 CommandLineAlgorithmFactory Class Reference	87
5.12.1 Detailed Description	88
5.12.2 Member Function Documentation	88
5.12.2.1 make()	88
5.13 CommandLineAlgorithmFactory Class Reference	88
5.13.1 Detailed Description	89
5.13.2 Member Function Documentation	89
5.13.2.1 make()	89
5.14 CommandLineApplication Class Reference	89
5.14.1 Detailed Description	90
5.14.2 Constructor & Destructor Documentation	91
5.14.2.1 CommandLineApplication()	91
5.15 CommandLineApplication Class Reference	91
5.15.1 Detailed Description	92
5.15.2 Constructor & Destructor Documentation	92
5.15.2.1 CommandLineApplication()	92
5.16 CommandLineFunctionFactory Class Reference	93
5.16.1 Detailed Description	94
5.17 CommandLineFunctionFactory Class Reference	94
5.17.1 Detailed Description	95
5.18 CommaSelection Class Reference	95
5.18.1 Detailed Description	95
5.18.2 Constructor & Destructor Documentation	95
5.18.2.1 CommaSelection()	95
5.19 CompactGa Class Reference	96
5.19.1 Detailed Description	97
5.20 CompleteSearch Class Reference	97
5.20.1 Detailed Description	98
5.21 ComplexToDouble< T > Struct Template Reference	98
5.21.1 Detailed Description	98
5.22 Controller Class Reference	99
5.22.1 Detailed Description	100
5.22.2 Member Function Documentation	100
5.22.2.1 provides_incremental_evaluation()	100
5.23 Crossover Class Reference	100
5.23.1 Detailed Description	101

5.23.2 Member Function Documentation	101
5.23.2.1 recombine()	101
5.24 DeceptiveJump Class Reference	101
5.24.1 Detailed Description	102
5.24.2 Member Function Documentation	103
5.24.2.1 get_maximum()	103
5.24.2.2 has_known_maximum()	103
5.25 DecoratedFunctionFactory Class Reference	103
5.25.1 Detailed Description	104
5.25.2 Member Function Documentation	104
5.25.2.1 make_function_controller()	104
5.26 Decorator Class Reference	105
5.26.1 Detailed Description	106
5.27 DyadicComplexRepresentation< T > Class Template Reference	106
5.27.1 Detailed Description	107
5.27.2 Constructor & Destructor Documentation	107
5.27.2.1 DyadicComplexRepresentation() [1/2]	107
5.27.2.2 DyadicComplexRepresentation() [2/2]	108
5.28 DyadicFloatRepresentation< T > Class Template Reference	108
5.28.1 Detailed Description	109
5.28.2 Constructor & Destructor Documentation	109
5.28.2.1 DyadicFloatRepresentation() [1/2]	109
5.28.2.2 DyadicFloatRepresentation() [2/2]	110
5.28.3 Member Function Documentation	110
5.28.3.1 compute_lengths()	110
5.29 DyadicIntegerRepresentation< T > Class Template Reference	111
5.29.1 Detailed Description	112
5.29.2 Constructor & Destructor Documentation	112
5.29.2.1 DyadicIntegerRepresentation() [1/2]	112
5.29.2.2 DyadicIntegerRepresentation() [2/2]	112
5.30 PermutationRepresentation::Element Struct Reference	113
5.30.1 Detailed Description	113
5.31 EqualProducts Class Reference	114
5.31.1 Detailed Description	115
5.31.2 Member Function Documentation	115
5.31.2.1 generate()	115
5.31.2.2 load()	116
5.31.2.3 random()	116
5.31.2.4 save()	116
5.32 ProgressTracker::Event Struct Reference	117
5.32.1 Detailed Description	117
5.33 ExtendedHypercubeliterator Class Reference	117

5.33.1 Detailed Description	118
5.34 Factorization Class Reference	118
5.34.1 Detailed Description	120
5.34.2 Constructor & Destructor Documentation	120
5.34.2.1 Factorization()	120
5.34.3 Member Function Documentation	120
5.34.3.1 load()	120
5.35 FfgOptions Class Reference	121
5.35.1 Detailed Description	126
5.36 FirstAscentHillClimbing Class Reference	126
5.36.1 Detailed Description	127
5.37 FourPeaks Class Reference	128
5.37.1 Detailed Description	129
5.37.2 Member Function Documentation	129
5.37.2.1 get_maximum()	129
5.37.2.2 has_known_maximum()	130
5.38 FrontDistancePair Struct Reference	130
5.38.1 Detailed Description	130
5.39 Function Class Reference	131
5.39.1 Detailed Description	132
5.39.2 Member Function Documentation	132
5.39.2.1 describe()	133
5.39.2.2 evaluate()	133
5.39.2.3 evaluate_incrementally()	133
5.39.2.4 evaluate_safely()	134
5.39.2.5 get_maximum()	134
5.39.2.6 provides_incremental_evaluation()	134
5.39.2.7 update()	135
5.40 Function Class Reference	135
5.40.1 Detailed Description	136
5.40.2 Member Function Documentation	136
5.40.2.1 describe()	136
5.40.2.2 evaluate()	136
5.41 FunctionFactory Class Reference	137
5.41.1 Detailed Description	137
5.42 FunctionFactory Class Reference	138
5.42.1 Detailed Description	138
5.43 FunctionMapComposition Class Reference	138
5.43.1 Detailed Description	140
5.43.2 Constructor & Destructor Documentation	140
5.43.2.1 FunctionMapComposition()	140
5.43.3 Member Function Documentation	140

5.43.3.1	get_maximum()	140
5.43.3.2	has_known_maximum()	141
5.44	FunctionPlugin Class Reference	141
5.44.1	Detailed Description	142
5.44.2	Constructor & Destructor Documentation	142
5.44.2.1	FunctionPlugin()	142
5.45	Generator Struct Reference	143
5.45.1	Detailed Description	143
5.45.2	Member Function Documentation	143
5.45.2.1	reset()	143
5.45.2.2	set_seed()	144
5.46	GeneticAlgorithm Class Reference	144
5.46.1	Detailed Description	145
5.46.2	Constructor & Destructor Documentation	146
5.46.2.1	GeneticAlgorithm()	146
5.47	HammingBall Class Reference	146
5.47.1	Detailed Description	147
5.47.2	Constructor & Destructor Documentation	147
5.47.2.1	HammingBall()	147
5.48	HammingSphere Class Reference	148
5.48.1	Detailed Description	149
5.48.2	Constructor & Destructor Documentation	149
5.48.2.1	HammingSphere()	149
5.49	HammingSphereIterator Class Reference	149
5.49.1	Detailed Description	150
5.49.2	Constructor & Destructor Documentation	151
5.49.2.1	HammingSphereIterator()	151
5.50	Hboa Class Reference	151
5.50.1	Detailed Description	152
5.50.2	Member Data Documentation	153
5.50.2.1	_pimpl	153
5.51	Hea< Herding > Class Template Reference	153
5.51.1	Detailed Description	155
5.51.2	Constructor & Destructor Documentation	155
5.51.2.1	Hea()	155
5.51.3	Member Function Documentation	156
5.51.3.1	set_reset_period()	156
5.51.3.2	set_selection_size()	156
5.52	Hiff Class Reference	157
5.52.1	Detailed Description	157
5.52.2	Member Function Documentation	158
5.52.2.1	get_maximum()	158

5.52.2.2 has_known_maximum()	158
5.53 HncoEvaluator Class Reference	158
5.53.1 Detailed Description	159
5.54 HncoOptions Class Reference	159
5.54.1 Detailed Description	176
5.55 HncoOptions Class Reference	177
5.55.1 Detailed Description	182
5.56 Human Class Reference	183
5.56.1 Detailed Description	184
5.57 Hypercubeliterator Class Reference	184
5.57.1 Detailed Description	184
5.58 Implementation Struct Reference	185
5.58.1 Detailed Description	185
5.59 Injection Class Reference	185
5.59.1 Detailed Description	186
5.59.2 Constructor & Destructor Documentation	186
5.59.2.1 Injection()	186
5.60 IntegerCategoricalRepresentation Class Reference	187
5.60.1 Detailed Description	188
5.60.2 Constructor & Destructor Documentation	188
5.60.2.1 IntegerCategoricalRepresentation()	188
5.61 IterativeAlgorithm Class Reference	188
5.61.1 Detailed Description	190
5.61.2 Constructor & Destructor Documentation	190
5.61.2.1 IterativeAlgorithm()	190
5.61.3 Member Function Documentation	190
5.61.3.1 loop()	190
5.61.3.2 maximize()	191
5.61.3.3 set_num_iterations()	191
5.62 IterativeAlgorithm Class Reference	191
5.62.1 Detailed Description	193
5.62.2 Constructor & Destructor Documentation	193
5.62.2.1 IterativeAlgorithm()	193
5.62.3 Member Function Documentation	193
5.62.3.1 loop()	194
5.62.3.2 minimize()	194
5.62.3.3 set_num_iterations()	194
5.63 Iterator Class Reference	195
5.63.1 Detailed Description	195
5.64 Jump Class Reference	196
5.64.1 Detailed Description	196
5.64.2 Member Function Documentation	197

5.64.2.1 <code>get_maximum()</code>	197
5.64.2.2 <code>has_known_maximum()</code>	197
5.65 Labs Class Reference	197
5.65.1 Detailed Description	198
5.66 LastEvaluation Class Reference	199
5.66.1 Detailed Description	199
5.67 LeadingOnes Class Reference	199
5.67.1 Detailed Description	200
5.67.2 Member Function Documentation	200
5.67.2.1 <code>get_maximum()</code>	200
5.67.2.2 <code>has_known_maximum()</code>	201
5.68 LinearCategoricalRepresentation Class Reference	201
5.68.1 Detailed Description	202
5.68.2 Constructor & Destructor Documentation	202
5.68.2.1 <code>LinearCategoricalRepresentation()</code>	202
5.69 LinearFunction Class Reference	202
5.69.1 Detailed Description	204
5.69.2 Member Function Documentation	204
5.69.2.1 <code>generate()</code>	204
5.69.2.2 <code>has_known_maximum()</code>	204
5.69.2.3 <code>load()</code>	205
5.69.2.4 <code>provides_incremental_evaluation()</code>	205
5.69.2.5 <code>random()</code>	205
5.69.2.6 <code>save()</code>	206
5.70 LinearMap Class Reference	206
5.70.1 Detailed Description	207
5.70.2 Member Function Documentation	207
5.70.2.1 <code>is_surjective()</code>	208
5.70.2.2 <code>load()</code>	208
5.70.2.3 <code>random()</code>	208
5.70.2.4 <code>save()</code>	209
5.71 LocalSearchAlgorithm< Neighborhood > Class Template Reference	209
5.71.1 Detailed Description	210
5.72 LogContext Class Reference	211
5.72.1 Detailed Description	211
5.73 Logger Class Reference	211
5.73.1 Detailed Description	212
5.73.2 Constructor & Destructor Documentation	212
5.73.2.1 <code>Logger()</code>	212
5.73.2.2 <code>~Logger()</code>	213
5.74 LongPath Class Reference	213
5.74.1 Detailed Description	214

5.74.2 Member Function Documentation	215
5.74.2.1 get_maximum()	215
5.74.2.2 has_known_maximum()	215
5.75 LowerTriangularWalshMoment2 Struct Reference	215
5.75.1 Detailed Description	216
5.75.2 Constructor & Destructor Documentation	216
5.75.2.1 LowerTriangularWalshMoment2()	216
5.75.3 Member Function Documentation	217
5.75.3.1 bound()	217
5.75.3.2 display()	217
5.75.3.3 scaled_difference()	217
5.75.3.4 update() [1/2]	218
5.75.3.5 update() [2/2]	218
5.76 LowerTriangularWalshMoment2GibbsSampler Class Reference	219
5.76.1 Detailed Description	220
5.77 LowerTriangularWalshMoment2Herding Class Reference	220
5.77.1 Detailed Description	221
5.77.2 Constructor & Destructor Documentation	221
5.77.2.1 LowerTriangularWalshMoment2Herding()	221
5.78 Ltga Class Reference	222
5.78.1 Detailed Description	222
5.78.2 Member Data Documentation	223
5.78.2.1 _pimpl	223
5.79 Map Class Reference	223
5.79.1 Detailed Description	224
5.79.2 Member Function Documentation	224
5.79.2.1 is_surjective()	224
5.80 MapComposition Class Reference	225
5.80.1 Detailed Description	225
5.80.2 Constructor & Destructor Documentation	226
5.80.2.1 MapComposition()	226
5.80.3 Member Function Documentation	226
5.80.3.1 is_surjective()	226
5.81 MapgenOptions Class Reference	227
5.81.1 Detailed Description	229
5.82 MaxNae3Sat Class Reference	229
5.82.1 Detailed Description	230
5.82.2 Member Function Documentation	230
5.82.2.1 load()	230
5.83 MaxSat Class Reference	230
5.83.1 Detailed Description	231
5.83.2 Member Function Documentation	231

5.83.2.1 random() [1/2]	232
5.83.2.2 random() [2/2]	232
5.84 Mimic Class Reference	232
5.84.1 Detailed Description	234
5.85 Mmas Class Reference	235
5.85.1 Detailed Description	236
5.86 Modifier Class Reference	236
5.86.1 Detailed Description	237
5.87 MuCommaLambdaEa Class Reference	237
5.87.1 Detailed Description	238
5.87.2 Constructor & Destructor Documentation	238
5.87.2.1 MuCommaLambdaEa()	238
5.88 MultiBitFlip Class Reference	239
5.88.1 Detailed Description	240
5.88.2 Constructor & Destructor Documentation	240
5.88.2.1 MultiBitFlip()	240
5.88.3 Member Function Documentation	240
5.88.3.1 bernoulli_trials()	240
5.88.3.2 rejection_sampling()	241
5.89 MultivariateFunctionAdapter< Fn, Rep, Conv > Class Template Reference	241
5.89.1 Detailed Description	242
5.89.2 Constructor & Destructor Documentation	242
5.89.2.1 MultivariateFunctionAdapter()	242
5.90 MultivariateFunctionAdapter< Fn, Rep, Conv > Class Template Reference	243
5.90.1 Detailed Description	244
5.90.2 Constructor & Destructor Documentation	244
5.90.2.1 MultivariateFunctionAdapter()	244
5.91 MuPlusLambdaEa Class Reference	245
5.91.1 Detailed Description	246
5.91.2 Constructor & Destructor Documentation	246
5.91.2.1 MuPlusLambdaEa()	246
5.92 NearestNeighborIsingModel1 Class Reference	247
5.92.1 Detailed Description	249
5.92.2 Member Function Documentation	249
5.92.2.1 evaluate()	249
5.92.2.2 generate()	249
5.92.2.3 load()	250
5.92.2.4 provides_incremental_evaluation()	250
5.92.2.5 random()	250
5.92.2.6 save()	251
5.93 NearestNeighborIsingModel2 Class Reference	251
5.93.1 Detailed Description	253

5.93.2 Member Function Documentation	253
5.93.2.1 evaluate()	254
5.93.2.2 generate()	254
5.93.2.3 load()	254
5.93.2.4 provides_incremental_evaluation()	255
5.93.2.5 random()	255
5.93.2.6 save()	255
5.94 Needle Class Reference	256
5.94.1 Detailed Description	257
5.94.2 Member Function Documentation	257
5.94.2.1 get_maximum()	257
5.94.2.2 has_known_maximum()	257
5.95 Negation Class Reference	258
5.95.1 Detailed Description	259
5.95.2 Member Function Documentation	259
5.95.2.1 provides_incremental_evaluation()	259
5.96 Neighborhood Class Reference	260
5.96.1 Detailed Description	261
5.96.2 Constructor & Destructor Documentation	261
5.96.2.1 Neighborhood()	261
5.96.3 Member Function Documentation	262
5.96.3.1 map()	262
5.96.3.2 mutate()	262
5.97 NeighborhoodIterator Class Reference	263
5.97.1 Detailed Description	263
5.97.2 Constructor & Destructor Documentation	263
5.97.2.1 NeighborhoodIterator()	263
5.98 NkLandscape Class Reference	264
5.98.1 Detailed Description	265
5.98.2 Member Function Documentation	265
5.98.2.1 generate()	265
5.98.2.2 load()	266
5.98.2.3 random()	266
5.98.2.4 random_structure()	266
5.98.2.5 save()	267
5.99 NpsPbil Class Reference	267
5.99.1 Detailed Description	269
5.100 Nsga2 Class Reference	269
5.100.1 Detailed Description	271
5.100.2 Constructor & Destructor Documentation	271
5.100.2.1 Nsga2()	271
5.101 Nsga2ParetoFrontComputation Class Reference	272

5.101.1 Detailed Description	272
5.101.2 Member Function Documentation	272
5.101.2.1 compute()	272
5.101.2.2 is_non_dominated()	273
5.101.3 Member Data Documentation	273
5.101.3.1 _dominated	273
5.102 OnBudgetFunction Class Reference	274
5.102.1 Detailed Description	275
5.102.2 Member Function Documentation	275
5.102.2.1 evaluate()	275
5.102.2.2 evaluate_incrementally()	275
5.102.2.3 update()	276
5.103 OneMax Class Reference	276
5.103.1 Detailed Description	277
5.103.2 Member Function Documentation	277
5.103.2.1 get_maximum()	278
5.103.2.2 has_known_maximum()	278
5.103.2.3 provides_incremental_evaluation()	278
5.104 OnePlusLambdaCommaLambdaGa Class Reference	279
5.104.1 Detailed Description	280
5.104.2 Constructor & Destructor Documentation	280
5.104.2.1 OnePlusLambdaCommaLambdaGa()	280
5.105 OnePlusOneEa Class Reference	281
5.105.1 Detailed Description	282
5.105.2 Constructor & Destructor Documentation	282
5.105.2.1 OnePlusOneEa()	282
5.105.3 Member Function Documentation	283
5.105.3.1 set_num_iterations()	283
5.106 ParameterLessPopulationPyramid Class Reference	283
5.106.1 Detailed Description	284
5.106.2 Member Data Documentation	284
5.106.2.1 _pimpl	284
5.107 ParsedModifier Class Reference	285
5.107.1 Detailed Description	286
5.107.2 Constructor & Destructor Documentation	286
5.107.2.1 ParsedModifier()	286
5.108 ParsedMultivariateFunction< Parser > Class Template Reference	286
5.108.1 Detailed Description	287
5.108.2 Constructor & Destructor Documentation	287
5.108.2.1 ParsedMultivariateFunction()	287
5.109 ParsedMultivariateFunction< Parser > Class Template Reference	288
5.109.1 Detailed Description	289

5.109.2 Constructor & Destructor Documentation	289
5.109.2.1 ParsedMultivariateFunction()	289
5.109.3 Member Data Documentation	289
5.109.3.1 _index_of	289
5.109.3.2 _lookup_tables	290
5.109.3.3 _names	290
5.109.3.4 _variables	290
5.110 Partition Class Reference	290
5.110.1 Detailed Description	292
5.110.2 Member Function Documentation	292
5.110.2.1 generate()	292
5.110.2.2 load()	292
5.110.2.3 random()	293
5.110.2.4 save()	293
5.111 Pbil Class Reference	293
5.111.1 Detailed Description	295
5.112 Permutation Class Reference	295
5.112.1 Detailed Description	296
5.112.2 Member Function Documentation	297
5.112.2.1 is_surjective()	297
5.112.2.2 load()	297
5.112.2.3 save()	297
5.113 PermutationFunctionAdapter< Fn > Class Template Reference	298
5.113.1 Detailed Description	299
5.113.2 Constructor & Destructor Documentation	299
5.113.2.1 PermutationFunctionAdapter()	299
5.114 PermutationRepresentation Class Reference	300
5.114.1 Detailed Description	300
5.114.2 Constructor & Destructor Documentation	300
5.114.2.1 PermutationRepresentation()	300
5.115 Plateau Class Reference	301
5.115.1 Detailed Description	302
5.115.2 Member Function Documentation	302
5.115.2.1 get_maximum()	302
5.115.2.2 has_known_maximum()	302
5.116 PlusSelection Class Reference	303
5.116.1 Detailed Description	303
5.116.2 Constructor & Destructor Documentation	303
5.116.2.1 PlusSelection()	303
5.117 Population Struct Reference	304
5.117.1 Detailed Description	305
5.117.2 Constructor & Destructor Documentation	305

5.117.2.1 Population()	305
5.117.3 Member Function Documentation	305
5.117.3.1 get_best_bv() [1/2]	305
5.117.3.2 get_best_bv() [2/2]	306
5.117.3.3 get_best_value() [1/2]	306
5.117.3.4 get_best_value() [2/2]	306
5.117.3.5 get_equivalent_bvs()	307
5.117.3.6 get_worst_bv()	307
5.117.3.7 partial_sort()	308
5.117.3.8 sort()	308
5.118 Population Struct Reference	308
5.118.1 Detailed Description	309
5.118.2 Constructor & Destructor Documentation	309
5.118.2.1 Population()	309
5.118.3 Member Function Documentation	310
5.118.3.1 resize()	310
5.118.3.2 shrink()	310
5.119 PriorNoise Class Reference	311
5.119.1 Detailed Description	312
5.119.2 Member Function Documentation	312
5.119.2.1 get_maximum()	312
5.119.2.2 has_known_maximum()	312
5.119.2.3 provides_incremental_evaluation()	313
5.120 ProgressTracker Class Reference	313
5.120.1 Detailed Description	315
5.120.2 Member Function Documentation	315
5.120.2.1 get_last_improvement()	315
5.120.3 Member Data Documentation	315
5.120.3.1 _record_evaluation_time	315
5.121 ProgressTrackerContext Class Reference	316
5.121.1 Detailed Description	316
5.122 Projection Class Reference	317
5.122.1 Detailed Description	317
5.122.2 Constructor & Destructor Documentation	318
5.122.2.1 Projection()	318
5.122.3 Member Function Documentation	318
5.122.3.1 is_surjective()	318
5.123 PvAlgorithm Class Reference	319
5.123.1 Detailed Description	320
5.124 PythonFunction Class Reference	320
5.124.1 Detailed Description	321
5.124.2 Constructor & Destructor Documentation	321

5.124.2.1 PythonFunction()	321
5.125 PythonFunction Class Reference	322
5.125.1 Detailed Description	323
5.125.2 Constructor & Destructor Documentation	323
5.125.2.1 PythonFunction()	323
5.126 Qubo Class Reference	323
5.126.1 Detailed Description	324
5.126.2 Member Function Documentation	325
5.126.2.1 load() [1/2]	325
5.126.2.2 load() [2/2]	325
5.126.3 Member Data Documentation	326
5.126.3.1 _q	326
5.127 RandomLocalSearch Class Reference	326
5.127.1 Detailed Description	328
5.127.2 Member Function Documentation	328
5.127.2.1 set_patience()	328
5.127.3 Member Data Documentation	329
5.127.3.1 _patience	329
5.128 RandomSearch Class Reference	329
5.128.1 Detailed Description	330
5.129 RandomSelection Class Reference	330
5.129.1 Detailed Description	331
5.129.2 Constructor & Destructor Documentation	331
5.129.2.1 RandomSelection()	331
5.130 RandomWalk Class Reference	331
5.130.1 Detailed Description	333
5.131 Restart Class Reference	333
5.131.1 Detailed Description	334
5.132 Ridge Class Reference	334
5.132.1 Detailed Description	335
5.132.2 Member Function Documentation	335
5.132.2.1 get_maximum()	335
5.132.2.2 has_known_maximum()	336
5.133 ScalarToDouble< T > Struct Template Reference	336
5.133.1 Detailed Description	336
5.134 SelfAdjustingOnePlusOneEa Class Reference	337
5.134.1 Detailed Description	339
5.135 SimulatedAnnealing Class Reference	339
5.135.1 Detailed Description	341
5.135.2 Member Function Documentation	341
5.135.2.1 init_beta()	341
5.136 SingleBitFlip Class Reference	341

5.136.1 Detailed Description	342
5.137 SingleBitFlipIterator Class Reference	342
5.137.1 Detailed Description	343
5.137.2 Constructor & Destructor Documentation	343
5.137.2.1 SingleBitFlipIterator()	343
5.138 SinusSummationCancellation Class Reference	344
5.138.1 Detailed Description	344
5.139 SixPeaks Class Reference	345
5.139.1 Detailed Description	346
5.139.2 Member Function Documentation	346
5.139.2.1 get_maximum()	346
5.139.2.2 has_known_maximum()	347
5.140 StandardBitMutation Class Reference	347
5.140.1 Detailed Description	348
5.140.2 Constructor & Destructor Documentation	348
5.140.2.1 StandardBitMutation() [1/2]	348
5.140.2.2 StandardBitMutation() [2/2]	349
5.140.3 Member Function Documentation	349
5.140.3.1 set_mutation_rate()	349
5.141 SteepestAscentHillClimbing Class Reference	350
5.141.1 Detailed Description	351
5.142 StopOnMaximum Class Reference	351
5.142.1 Detailed Description	352
5.142.2 Constructor & Destructor Documentation	352
5.142.2.1 StopOnMaximum()	352
5.143 StopOnTarget Class Reference	353
5.143.1 Detailed Description	354
5.143.2 Constructor & Destructor Documentation	354
5.143.2.1 StopOnTarget()	354
5.143.3 Member Function Documentation	354
5.143.3.1 evaluate()	355
5.143.3.2 evaluate_incrementally()	355
5.143.3.3 update()	355
5.144 StopWatch Class Reference	356
5.144.1 Detailed Description	356
5.145 Sudoku Class Reference	356
5.145.1 Detailed Description	358
5.145.2 Member Function Documentation	358
5.145.2.1 load()	358
5.145.2.2 load_()	358
5.145.2.3 random()	358
5.145.2.4 save()	359

5.146 SummationCancellation Class Reference	359
5.146.1 Detailed Description	361
5.146.2 Constructor & Destructor Documentation	361
5.146.2.1 SummationCancellation()	361
5.146.3 Member Function Documentation	361
5.146.3.1 has_known_maximum()	361
5.147 SymmetricWalshMoment2 Struct Reference	362
5.147.1 Detailed Description	363
5.147.2 Constructor & Destructor Documentation	363
5.147.2.1 SymmetricWalshMoment2()	363
5.147.3 Member Function Documentation	363
5.147.3.1 average()	363
5.147.3.2 bound()	364
5.147.3.3 display()	364
5.147.3.4 scaled_difference()	364
5.147.3.5 update() [1/2]	365
5.147.3.6 update() [2/2]	365
5.148 SymmetricWalshMoment2GibbsSampler Class Reference	366
5.148.1 Detailed Description	367
5.149 SymmetricWalshMoment2Herding Class Reference	367
5.149.1 Detailed Description	368
5.149.2 Constructor & Destructor Documentation	368
5.149.2.1 SymmetricWalshMoment2Herding()	368
5.150 TargetReached Class Reference	369
5.150.1 Detailed Description	369
5.151 TournamentSelection Class Reference	369
5.151.1 Detailed Description	370
5.151.2 Constructor & Destructor Documentation	370
5.151.2.1 TournamentSelection()	370
5.151.3 Member Function Documentation	371
5.151.3.1 select()	371
5.152 TournamentSelection< T, Compare > Class Template Reference	371
5.152.1 Detailed Description	372
5.153 Translation Class Reference	373
5.153.1 Detailed Description	374
5.153.2 Member Function Documentation	374
5.153.2.1 is_surjective()	374
5.153.2.2 load()	374
5.153.2.3 save()	375
5.154 Transvection Struct Reference	375
5.154.1 Detailed Description	376
5.154.2 Member Function Documentation	376

5.154.2.1 is_valid()	376
5.154.2.2 multiply() [1/2]	377
5.154.2.3 multiply() [2/2]	377
5.154.2.4 random()	378
5.154.2.5 random_non_commuting()	378
5.155 Trap Class Reference	378
5.155.1 Detailed Description	379
5.155.2 Constructor & Destructor Documentation	380
5.155.2.1 Trap()	380
5.155.3 Member Function Documentation	380
5.155.3.1 get_maximum()	380
5.155.3.2 has_known_maximum()	381
5.156 TsAffineMap Class Reference	381
5.156.1 Detailed Description	382
5.156.2 Member Enumeration Documentation	383
5.156.2.1 SamplingMode	383
5.156.3 Member Function Documentation	383
5.156.3.1 is_surjective()	383
5.156.3.2 load()	383
5.156.3.3 random()	384
5.156.3.4 save()	384
5.157 Tsp Class Reference	385
5.157.1 Detailed Description	386
5.157.2 Member Function Documentation	386
5.157.2.1 generate()	386
5.157.2.2 load()	387
5.157.2.3 load_()	387
5.157.2.4 random()	387
5.157.2.5 save()	388
5.158 Umda Class Reference	388
5.158.1 Detailed Description	390
5.159 UniformCrossover Class Reference	390
5.159.1 Detailed Description	390
5.159.2 Member Function Documentation	391
5.159.2.1 recombine()	391
5.160 UniformSelection Class Reference	391
5.160.1 Detailed Description	392
5.160.2 Constructor & Destructor Documentation	392
5.160.2.1 UniformSelection()	392
5.161 UniversalFunction Class Reference	392
5.161.1 Detailed Description	393
5.162 UniversalFunction Class Reference	393

5.162.1 Detailed Description	394
5.163 UniversalFunctionAdapter Class Reference	394
5.163.1 Detailed Description	395
5.163.2 Constructor & Destructor Documentation	395
5.163.2.1 UniversalFunctionAdapter()	396
5.164 UniversalFunctionAdapter Class Reference	396
5.164.1 Detailed Description	398
5.164.2 Constructor & Destructor Documentation	398
5.164.2.1 UniversalFunctionAdapter()	398
5.165 WalshExpansion Class Reference	399
5.165.1 Detailed Description	400
5.165.2 Member Function Documentation	400
5.165.2.1 generate()	400
5.165.2.2 load()	401
5.165.2.3 random()	401
5.165.2.4 save()	401
5.166 WalshExpansion1 Class Reference	402
5.166.1 Detailed Description	403
5.166.2 Member Function Documentation	404
5.166.2.1 generate()	404
5.166.2.2 has_known_maximum()	404
5.166.2.3 load()	404
5.166.2.4 provides_incremental_evaluation()	405
5.166.2.5 random()	405
5.166.2.6 save()	405
5.167 WalshExpansion2 Class Reference	406
5.167.1 Detailed Description	407
5.167.2 Member Function Documentation	408
5.167.2.1 generate()	408
5.167.2.2 generate_ising1_long_range()	408
5.167.2.3 generate_ising1_long_range_periodic()	409
5.167.2.4 load()	409
5.167.2.5 random()	410
5.167.2.6 save()	410
5.167.3 Member Data Documentation	410
5.167.3.1 _quadratic	410
5.168 WalshTerm Struct Reference	411
5.168.1 Detailed Description	411
5.168.2 Member Data Documentation	411
5.168.2.1 feature	411

Chapter 1

Namespace Index

1.1 Namespace List

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

hnco	Top-level HNCO namespace	15
hnco::algorithm	Algorithms	35
hnco::algorithm::fast_efficient_p3	Algorithms from the FastEfficientP3 library	40
hnco::algorithm::walsh_moment	Algorithms using Walsh moments	41
hnco::app	Classes for applications	41
hnco::exception	Exceptions	42
hnco::function	Functions defined on bit vectors	43
hnco::function::controller	Controllers	46
hnco::function::modifier	Modifiers	47
hnco::logging	Logging	48
hnco::map	Maps	48
hnco::multiobjective	Multiobjective optimization	55
hnco::multiobjective::algorithm	Multiobjective Algorithms	55
hnco::multiobjective::app	Classes for applications	56
hnco::multiobjective::function	Functions defined on bit vectors	57
hnco::neighborhood	Neighborhoods for local search	58
hnco::random	Random numbers	59
hnco::representation	Representations	59

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Algorithm	70
CompleteSearch	97
IterativeAlgorithm	188
LocalSearchAlgorithm< neighborhood::NeighborhoodIterator >	209
FirstAscentHillClimbing	126
SteepestAscentHillClimbing	350
LocalSearchAlgorithm< neighborhood::Neighborhood >	209
RandomLocalSearch	326
RandomWalk	331
SimulatedAnnealing	339
GeneticAlgorithm	144
Human	183
LocalSearchAlgorithm< Neighborhood >	209
Mimic	232
MuCommaLambdaEa	237
MuPlusLambdaEa	245
OnePlusLambdaCommaLambdaGa	279
PvAlgorithm	319
CompactGa	96
Mmas	235
NpsPbil	267
Pbil	293
Umda	388
RandomSearch	329
Restart	333
SelfAdjustingOnePlusOneEa	337
BmPbil< GibbsSampler >	79
Hea< Herding >	153
OnePlusOneEa	281
Hboa	151
Ltga	222
ParameterLessPopulationPyramid	283
Algorithm	74
IterativeAlgorithm	191

Nsga2	269
AlgorithmFactory	76
CommandLineAlgorithmFactory	87
AlgorithmFactory	77
CommandLineAlgorithmFactory	88
CommandLineApplication	89
CommandLineApplication	91
CommaSelection	95
ComplexToDouble< T >	98
Crossover	100
BiasedCrossover	78
UniformCrossover	390
DecoratedFunctionFactory	103
DyadicComplexRepresentation< T >	106
DyadicComplexRepresentation< double >	106
DyadicFloatRepresentation< T >	108
DyadicFloatRepresentation< double >	108
DyadicIntegerRepresentation< T >	111
DyadicIntegerRepresentation< int >	111
PermutationRepresentation::Element	113
Evaluator	
HncoEvaluator	158
ProgressTracker::Event	117
exception	
std::runtime_error	
LastEvaluation	199
TargetReached	369
FfgenOptions	121
FrontDistancePair	130
Function	131
AbstractMaxSat	63
MaxNae3Sat	229
MaxSat	230
DeceptiveJump	101
Decorator	105
Controller	99
Cache	83
CallCounter	86
OnBudgetFunction	274
ProgressTracker	313
StopOnTarget	353
StopOnMaximum	351
Modifier	236
AdditiveGaussianNoise	66
FunctionMapComposition	138
Negation	258
ParsedModifier	285
PriorNoise	311
EqualProducts	114
Factorization	118
FourPeaks	128
FunctionPlugin	141
Hiff	157
Jump	196
Labs	197
LeadingOnes	199
LinearFunction	202

LongPath	213
MultivariateFunctionAdapter< Fn, Rep, Conv >	241
NearestNeighborIsingModel1	247
NearestNeighborIsingModel2	251
Needle	256
NkLandscape	264
OneMax	276
Partition	290
PermutationFunctionAdapter< Fn >	298
Plateau	301
PythonFunction	320
Qubo	323
Ridge	334
SixPeaks	345
SummationCancellation	359
SinusSummationCancellation	344
Trap	378
UniversalFunctionAdapter	394
WalshExpansion	399
WalshExpansion1	402
WalshExpansion2	406
Function	135
MultivariateFunctionAdapter< Fn, Rep, Conv >	243
PythonFunction	322
UniversalFunctionAdapter	396
FunctionFactory	137
CommandLineFunctionFactory	93
FunctionFactory	138
CommandLineFunctionFactory	94
Generator	143
HncoOptions	159
HncoOptions	177
Implementation	185
IntegerCategoricalRepresentation	187
Iterator	195
ExtendedHypercubeIterator	117
HypercubeIterator	184
NeighborhoodIterator	263
HammingSphereIterator	149
SingleBitFlipIterator	342
LinearCategoricalRepresentation	201
LogContext	211
ProgressTrackerContext	316
Logger	211
LowerTriangularWalshMoment2	215
LowerTriangularWalshMoment2GibbsSampler	219
LowerTriangularWalshMoment2Herdning	220
Map	223
AffineMap	67
Injection	185
LinearMap	206
MapComposition	225
Permutation	295
Projection	317
Translation	373
TsAffineMap	381

MapgenOptions	227
Neighborhood	260
MultiBitFlip	239
HammingBall	146
HammingSphere	148
StandardBitMutation	347
SingleBitFlip	341
Nsga2ParetoFrontComputation	272
ParsedMultivariateFunction< Parser >	286
ParsedMultivariateFunction< Parser >	288
PermutationRepresentation	300
PlusSelection	303
Population	304
Population	308
RandomSelection	330
TournamentSelection	369
UniformSelection	391
ScalarToDouble< T >	336
StopWatch	356
Sudoku	356
SymmetricWalshMoment2	362
SymmetricWalshMoment2GibbsSampler	366
SymmetricWalshMoment2Herdin	367
TournamentSelection< T, Compare >	371
TournamentSelection< double, std::greater< double > >	371
TournamentSelection< hnco::multiobjective::algorithm::FrontDistancePair, std::less< hnco::multiobjective← ::algorithm::FrontDistancePair > >	371
Transvection	375
Tsp	385
UniversalFunction	392
UniversalFunction	393
WalshTerm	411

Chapter 3

Class Index

3.1 Class List

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

AbstractMaxSat	Abstract class for MaxSat-like functions	63
AdditiveGaussianNoise	Additive Gaussian Noise	66
AffineMap	Affine map	67
Algorithm	Abstract search algorithm	70
Algorithm	Abstract multiobjective search algorithm	74
AlgorithmFactory	Algorithm factory	76
AlgorithmFactory	Algorithm factory	77
BiasedCrossover	Biased crossover	78
BmPbil < GibbsSampler >	Boltzmann machine PBIL	79
Cache	Cache	83
CallCounter	Call counter	86
CommandLineAlgorithmFactory	Command line algorithm factory	87
CommandLineAlgorithmFactory	Command line algorithm factory	88
CommandLineApplication	Command line application	89
CommandLineApplication	Command line application	91
CommandLineFunctionFactory	Command line function factory	93
CommandLineFunctionFactory	Command line function factory	94
CommaSelection	Comma selection	95

CompactGa	
Compact genetic algorithm	96
CompleteSearch	
Complete search	97
ComplexToDouble< T >	
Convert a complex to a double	98
Controller	
Function controller	99
Crossover	
Crossover	100
DeceptiveJump	
Deceptive jump	101
DecoratedFunctionFactory	
Decorated function factory	103
Decorator	
Function decorator	105
DyadicComplexRepresentation< T >	
Dyadic complex representation	106
DyadicFloatRepresentation< T >	
Dyadic float representation	108
DyadicIntegerRepresentation< T >	
Dyadic integer representation	111
PermutationRepresentation::Element	
Element	113
EqualProducts	
Equal products	114
ProgressTracker::Event	
Event	117
ExtendedHypercubeIterator	
Extended Hypercube iterator	117
Factorization	
Factorization	118
FfgenOptions	
Command line options for ffgen	121
FirstAscentHillClimbing	
First ascent hill climbing	126
FourPeaks	
Four Peaks	128
FrontDistancePair	
Front-distance pair	130
Function	
Function	131
Function	
Function	135
FunctionFactory	
Function factory	137
FunctionFactory	
Function factory	138
FunctionMapComposition	
Composition of a function and a map	138
FunctionPlugin	
Function plugin	141
Generator	
Random number generator	143
GeneticAlgorithm	
Genetic algorithm	144
HammingBall	
Hamming ball	146

HammingSphere	
Hamming sphere	148
HammingSphereIterator	
Hamming sphere neighborhood iterator	149
Hboa	
Hierarchical Bayesian Optimization Algorithm	151
Hea< Herding >	
Herding evolutionary algorithm	153
Hiff	
Hierarchical if and only if	157
HncoEvaluator	
Evaluator for HNCO functions	158
HncoOptions	
Command line options for hnco	159
HncoOptions	
Command line options for hnco-mo	177
Human	
Human	183
HypercubeIterator	
Hypercube iterator	184
Implementation	
Implementation	185
Injection	
Injection	185
IntegerCategoricalRepresentation	
Integer categorical representation	187
IterativeAlgorithm	
Iterative search	188
IterativeAlgorithm	
Iterative algorithm	191
Iterator	
Iterator over bit vectors	195
Jump	
Jump	196
Labs	
Low autocorrelation binary sequences	197
LastEvaluation	
Last evaluation	199
LeadingOnes	
Leading ones	199
LinearCategoricalRepresentation	
Linear categorical representation	201
LinearFunction	
Linear function	202
LinearMap	
Linear map	206
LocalSearchAlgorithm< Neighborhood >	
Local search algorithm	209
LogContext	
Log context	211
Logger	
Logger	211
LongPath	
Long path	213
LowerTriangularWalshMoment2	
Lower triangular Walsh moment	215
LowerTriangularWalshMoment2GibbsSampler	
Gibbs sampler with lower triangular Walsh moments	219

LowerTriangularWalshMoment2Herdng	
Herdng with lower triangular Walsh moment	220
Ltga	
Linkage Tree Genetic Algorithm	222
Map	
Map	223
MapComposition	
Map composition	225
MapgenOptions	
Command line options for mapgen	227
MaxNae3Sat	
Max not-all-equal 3SAT	229
MaxSat	
MAX-SAT	230
Mimic	
Mutual information maximizing input clustering	232
Mmas	
Max-min ant system	235
Modifier	
Function modifier	236
MuCommaLambdaEa	
(mu, lambda) EA	237
MultiBitFlip	
Multi bit flip	239
MultivariateFunctionAdapter< Fn, Rep, Conv >	
Multivariate function adapter	241
MultivariateFunctionAdapter< Fn, Rep, Conv >	
Multivariate function adapter	243
MuPlusLambdaEa	
(mu+lambda) EA	245
NearestNeighborsIsingModel1	
Nearest neighbor Ising model in one dimension	247
NearestNeighborsIsingModel2	
Nearest neighbor Ising model in two dimensions	251
Needle	
Needle in a haystack	256
Negation	
Negation	258
Neighborhood	
Neighborhood	260
NeighborhoodIterator	
Neighborhood iterator	263
NkLandscape	
NK landscape	264
NpsPbil	
Population-based incremental learning with negative and positive selection	267
Nsga2	
NSGA-II	269
Nsga2ParetoFrontComputation	
Pareto front computation from the NSGA-II paper	272
OnBudgetFunction	
Function with a limited number of evaluations	274
OneMax	
OneMax	276
OnePlusLambdaCommaLambdaGa	
(1+(lambda, lambda)) genetic algorithm	279
OnePlusOneEa	
(1+1) EA	281

ParameterLessPopulationPyramid	
Parameter-less Population Pyramid	283
ParsedModifier	
Parsed modifier	285
ParsedMultivariateFunction< Parser >	
Parsed multivariate function	286
ParsedMultivariateFunction< Parser >	
Parsed multivariate function	288
Partition	
Partition	290
Pbil	
Population-based incremental learning	293
Permutation	
Permutation	295
PermutationFunctionAdapter< Fn >	
Permutation function adapter	298
PermutationRepresentation	
Permutation representation	300
Plateau	
Plateau	301
PlusSelection	
Plus selection	303
Population	
Population	304
Population	
Population	308
PriorNoise	
Prior noise	311
ProgressTracker	
ProgressTracker	313
ProgressTrackerContext	
Log context for ProgressTracker	316
Projection	
Projection	317
PvAlgorithm	
Probability vector algorithm	319
PythonFunction	
Python function	320
PythonFunction	
Python function	322
Qubo	
Quadratic unconstrained binary optimization	323
RandomLocalSearch	
Random local search	326
RandomSearch	
Random search	329
RandomSelection	
Random selection	330
RandomWalk	
Random walk	331
Restart	
Restart	333
Ridge	
Ridge	334
ScalarToDouble< T >	
Convert a scalar to a double	336
SelfAdjustingOnePlusOneEa	
Self-adjusting (1+1) evolutionary algorithm	337

SimulatedAnnealing	
Simulated annealing	339
SingleBitFlip	
One bit neighborhood	341
SingleBitFlipIterator	
Single bit flip neighborhood iterator	342
SinusSummationCancellation	
Summation cancellation with sinus	344
SixPeaks	
Six Peaks	345
StandardBitMutation	
Standard bit mutation	347
SteepestAscentHillClimbing	
Steepest ascent hill climbing	350
StopOnMaximum	
Stop on maximum	351
StopOnTarget	
Stop on target	353
StopWatch	
Stop watch	356
Sudoku	
Sudoku	356
SummationCancellation	
Summation cancellation	359
SymmetricWalshMoment2	
Symmetric Walsh moment	362
SymmetricWalshMoment2GibbsSampler	
Gibbs sampler with symmetric Walsh moments	366
SymmetricWalshMoment2Herdin	
Herdin with symmetric Walsh moment	367
TargetReached	
Target reached	369
TournamentSelection	
Tournament selection	369
TournamentSelection< T, Compare >	
Tournament selection	371
Translation	
Translation	373
Transvection	
Transvection	375
Trap	
Trap	378
TsAffineMap	
Transvection sequence affine map	381
Tsp	
Traveling salesman problem	385
Umda	
Univariate marginal distribution algorithm	388
UniformCrossover	
Uniform crossover	390
UniformSelection	
Uniform selection	391
UniversalFunction	
Universal function	392
UniversalFunction	
Universal function	393
UniversalFunctionAdapter	
Universal function adapter	394

UniversalFunctionAdapter	
Universal function adapter	396
WalshExpansion	
Walsh expansion	399
WalshExpansion1	
Walsh expansion of degree 1	402
WalshExpansion2	
Walsh expansion of degree 2	406
WalshTerm	
Walsh transform term	411

Chapter 4

Namespace Documentation

4.1 hnco Namespace Reference

top-level HNCO namespace

Namespaces

- [algorithm](#)
Algorithms.
- [app](#)
Classes for applications.
- [exception](#)
Exceptions.
- [function](#)
Functions defined on bit vectors.
- [logging](#)
Logging.
- [map](#)
Maps.
- [multiobjective](#)
Multiobjective optimization.
- [neighborhood](#)
Neighborhoods for local search.
- [random](#)
Random numbers.
- [representation](#)
Representations.

Classes

- class [Iterator](#)
Iterator over bit vectors
- class [HypercubeIterator](#)
Hypercube iterator.
- class [ExtendedHypercubeIterator](#)
Extended Hypercube iterator.
- class [StopWatch](#)
Stop watch.

Functions

- void [ensure](#) (bool b, const std::string message)
Ensure that a condition is satisfied or throw a runtime exception.
- template<class A , class B >
bool [have_same_size](#) (const A &a, const B &b)
Check whether two containers have the same size.
- template<class T >
T [square](#) (T x)
Generic square function.
- double [logistic](#) (double x)
Logistic function (sigmoid)
- template<typename Iter >
std::string [join](#) (Iter begin, Iter end, std::string const &separator)
Convert to string and join elements of a container (from SO)

Load from and save to boost archives

- template<typename T >
void [load_from_archive](#) (T &object, std::string path, std::string name)
Load from a boost archive.
- template<typename T >
void [save_to_archive](#) (const T &object, std::string path, std::string name)
Save to a boost archive.

Range checking

- bool [is_in_range](#) (int i, int a, int b)
Check whether an index is in a given range.
- bool [is_in_range](#) (int i, int n)
Check whether an index is in a given range.

Intervals

- bool [is_in_interval](#) (double x, double a, double b)
Check whether a double value belongs to a given interval.
- template<typename T >
T [clip_value](#) (T x, T low, T high)
Clip value between two bounds.

Types and functions related to bit matrices

Output and input-output function parameters appear at the beginning of the parameter list.

Output and input-output `bit_matrix_t` parameters are passed by reference and must have the right size for the considered function.

Input object parameters are passed by const reference.

- using [bit_matrix_t](#) = std::vector< [bit_vector_t](#) >
Bit matrix.
- [bit_matrix_t](#) [bm_rectangular](#) (int nrows, int ncols)

- Make a rectangular bit matrix.*

 - [bit_matrix_t bm_square](#) (int n)
- Make a square bit matrix.*

 - void [bm_identity](#) ([bit_matrix_t](#) &M)
- Set a matrix to the identity matrix.*

 - [bit_matrix_t bm_identity](#) (int n)
- Make an identity bit matrix.*

 - void [bm_transpose](#) ([bit_matrix_t](#) &N, const [bit_matrix_t](#) &M)
- Transpose a bit matrix.*

 - [bit_matrix_t bm_transpose](#) (const [bit_matrix_t](#) &M)
- Transpose a bit matrix.*

 - void [bm_display](#) (const [bit_matrix_t](#) &M, std::ostream &stream)
- Display bit matrix.*

 - bool [bm_is_valid](#) (const [bit_matrix_t](#) &M)
- Check whether a bit matrix is valid.*

 - int [bm_num_rows](#) (const [bit_matrix_t](#) &M)
- Number of rows.*

 - int [bm_num_columns](#) (const [bit_matrix_t](#) &M)
- Number of columns.*

 - bool [bm_is_square](#) (const [bit_matrix_t](#) &M)
- Check whether the matrix is a square matrix.*

 - bool [bm_is_identity](#) (const [bit_matrix_t](#) &M)
- Check whether the matrix is the identity matrix.*

 - bool [bm_is_upper_triangular](#) (const [bit_matrix_t](#) &M)
- Check whether the matrix is upper triangular.*

 - void [bm_resize](#) ([bit_matrix_t](#) &M, int nrows, int ncols)
- Resize a bit matrix.*

 - void [bm_resize](#) ([bit_matrix_t](#) &M, int nrows)
- Resize a bit matrix and make it a square matrix.*

 - void [bm_clear](#) ([bit_matrix_t](#) &M)
- Clear bit matrix.*

 - void [bm_random](#) ([bit_matrix_t](#) &M)
- Sample a random bit matrix.*

 - void [bm_swap_rows](#) ([bit_matrix_t](#) &M, int i, int j)
- Swap two rows.*

 - void [bm_add_rows](#) ([bit_matrix_t](#) &M, int dest, int src)
- Add two rows.*

 - void [bm_add_columns](#) ([bit_matrix_t](#) &M, int dest, int src)
- Add two columns.*

 - void [bm_set_column](#) ([bit_matrix_t](#) &M, int j, const [bit_vector_t](#) &bv)
- Set column.*

 - void [bm_row_echelon_form](#) ([bit_matrix_t](#) &A)
- Compute a row echelon form of a matrix.*

 - int [bm_rank](#) (const [bit_matrix_t](#) &A)
- Compute the rank of a matrix.*

 - bool [bm_solve](#) ([bit_matrix_t](#) &A, [bit_vector_t](#) &b)
- Solve a linear system.*

 - bool [bm_solve_upper_triangular](#) ([bit_matrix_t](#) &A, [bit_vector_t](#) &b)
- Solve a linear system in upper triangular form.*

 - bool [bm_invert](#) ([bit_matrix_t](#) &M, [bit_matrix_t](#) &N)
- Invert a bit matrix.*

 - void [bm_multiply](#) ([bit_vector_t](#) &y, const [bit_matrix_t](#) &M, const [bit_vector_t](#) &x)
- Multiply a bit matrix and a bit vector.*

Types and functions related to bit

- using `bit_t` = `std::uint8_t`
Bit.
- `bit_t bit_flip (bit_t b)`
Flip bit.
- `bit_t bit_random (double p)`
Sample a random bit.

Types and functions related to bit vectors

Output and input-output function parameters appear at the beginning of the parameter list.

Output and input-output `bit_vector_t` parameters are passed by reference and must have the right size for the considered function.

Input `bit_vector_t` parameters are passed by const reference.

- using `bit_vector_t` = `std::vector< bit_t >`
Bit vector.
- `std::string bv_domain (const bit_vector_t &x)`
Display bit vector.
- `void bv_display (const bit_vector_t &v, std::ostream &stream)`
Display bit vector.
- `bool bv_is_valid (const bit_vector_t &x)`
Check whether the bit vector is valid.
- `bool bv_is_zero (const bit_vector_t &x)`
Check whether the bit vector is zero.
- `int bv_hamming_weight (const bit_vector_t &x)`
Hamming weight.
- `int bv_hamming_weight (const std::vector< bool > &x)`
Hamming weight.
- `int bv_hamming_distance (const bit_vector_t &x, const bit_vector_t &y)`
Hamming distance between two bit vectors.
- `bit_t bv_dot_product (const bit_vector_t &x, const bit_vector_t &y)`
Dot product.
- `bit_t bv_dot_product (const bit_vector_t &x, const std::vector< bool > &y)`
Dot product.
- `void bv_clear (bit_vector_t &x)`
Clear bit vector.
- `void bv_flip (bit_vector_t &x, int i)`
Flip a single bit.
- `void bv_flip (bit_vector_t &x, const bit_vector_t &mask)`
Flip many bits.
- `void bv_random (bit_vector_t &x)`
Sample a random bit vector.
- `void bv_random (bit_vector_t &x, int k)`
Sample a random bit vector with given Hamming weight.
- `void bv_add (bit_vector_t &dest, const bit_vector_t &src)`
Add two bit vectors.

- void `bv_add` (`bit_vector_t` &dest, const `bit_vector_t` &x, const `bit_vector_t` &y)
Add two bit vectors.
- void `bv_to_vector_bool` (`std::vector< bool >` &y, const `bit_vector_t` &x)
Convert a bit vector to a bool vector.
- void `bv_from_vector_bool` (`bit_vector_t` &x, const `std::vector< bool >` &y)
Convert a bool vector to a bit vector.
- `std::size_t` `bv_to_size_type` (const `bit_vector_t` &x)
Convert a small bit vector to a size_t.
- `std::size_t` `bv_to_size_type` (const `bit_vector_t` &x, int start, int stop)
Convert a slice of a small bit vector to a size_t.
- void `bv_from_size_type` (`bit_vector_t` &x, `std::size_t` u)
Convert a size_t to a small bit vector.
- `bit_vector_t` `bv_from_string` (const `std::string` &str)
Read a bit vector from a string.
- `bit_vector_t` `bv_from_stream` (`std::istream` &stream)
Read a bit vector from a stream.

Types and functions related to permutations

- using `permutation_t` = `std::vector< int >`
Permutation type
- bool `perm_is_valid` (const `permutation_t` &permutation)
Check that a vector represents a permutation.
- void `perm_identity` (`permutation_t` &s)
Identity permutation.
- void `perm_shuffle` (`permutation_t` &s)
Shuffle a permutation.
- void `perm_random` (`permutation_t` &s)
Sample a random permutation.
- void `perm_display` (const `permutation_t` &permutation, `std::ostream` &stream)
Display a permutation.

Types and functions related to sparse bit vectors

Output and input-output function parameters appear at the beginning of the parameter list.

Input object parameters are passed by const reference.

- using `sparse_bit_vector_t` = `std::vector< int >`
Sparse bit vector.
- bool `sbv_is_valid` (const `sparse_bit_vector_t` &sbv)
Check that a sparse bit vector is valid.
- bool `sbv_is_valid` (const `sparse_bit_vector_t` &sbv, int n)
Check that a sparse bit vector is valid.
- void `sbv_flip` (`bit_vector_t` &x, const `sparse_bit_vector_t` &sbv)
Flip many bits of a bit vector.
- void `sbv_display` (const `sparse_bit_vector_t` &v, `std::ostream` &stream)
Display sparse bit vector.
- `sparse_bit_vector_t` `sbv_from_bv` (const `bit_vector_t` &bv)
Convert a bit vector to a sparse bit vector.

4.1.1 Detailed Description

top-level HNCO namespace

4.1.2 Typedef Documentation

4.1.2.1 `sparse_bit_vector_t`

```
using sparse_bit_vector_t = std::vector<int>
```

Sparse bit vector.

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

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

Definition at line 50 of file `sparse-bit-vector.hh`.

4.1.3 Function Documentation

4.1.3.1 `bm_add_columns()`

```
void bm_add_columns (
    bit_matrix_t & M,
    int dest,
    int src )
```

Add two columns.

Equivalent to `dest = dest + src`.

Parameters

<i>M</i>	Bit matrix
<i>dest</i>	Destination column
<i>src</i>	Source column

Warning

M is modified by the function.

Definition at line 187 of file `bit-matrix.cc`.

4.1.3.2 `bm_add_rows()`

```
void bm_add_rows (
    bit_matrix_t & M,
    int dest,
    int src )
```

Add two rows.

Equivalent to `dest = dest + src`.

Parameters

<i>M</i>	Bit matrix
<i>dest</i>	Destination row
<i>src</i>	Source row

Definition at line 178 of file bit-matrix.cc.

4.1.3.3 `bm_identity()` [1/2]

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

Set a matrix to the identity matrix.

Precondition

`bm_is_square(M)`

Definition at line 39 of file bit-matrix.cc.

4.1.3.4 `bm_identity()` [2/2]

```
bit_matrix_t bm_identity (
    int n )
```

Make an identity bit matrix.

Parameters

<i>n</i>	Dimension
----------	-----------

Returns

An order n identity matrix

Definition at line 50 of file bit-matrix.cc.

4.1.3.5 bm_invert()

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

Invert a bit matrix.

Parameters

M	Bit matrix
N	Inverse bit matrix

Precondition

```
bm_is_square(M)
bm_is_square(N)
bm_num_rows(M) == bm_num_rows(N)
```

Returns

true if M is invertible

Warning

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

Definition at line 316 of file bit-matrix.cc.

4.1.3.6 bm_multiply()

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

Multiply a bit matrix and a bit vector.

Computes $y = Mx$.

Parameters

y	Output bit vector
M	Bit matrix
x	Bit vector

Definition at line 360 of file bit-matrix.cc.

4.1.3.7 `bm_rank()`

```
int bm_rank (
    const bit_matrix_t & A )
```

Compute the rank of a matrix.

Precondition

A must be in row echelon form.

Definition at line 244 of file bit-matrix.cc.

4.1.3.8 `bm_row_echelon_form()`

```
void bm_row_echelon_form (
    bit_matrix_t & A )
```

Compute a row echelon form of a matrix.

Warning

A is modified by the function.

Definition at line 213 of file bit-matrix.cc.

4.1.3.9 `bm_set_column()`

```
void bm_set_column (
    bit_matrix_t & M,
    int j,
    const bit_vector_t & bv )
```

Set column.

Set a column to a given bit vector.

Parameters

M	Bit matrix
j	Column index
bv	Bit vector

Precondition

`bm_num_rows(M) == bv.size()`

Definition at line 202 of file bit-matrix.cc.

4.1.3.10 `bm_solve()`

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

Solve a linear system.

Solve the linear equation $Ax = b$.

Parameters

A	Matrix
b	Right hand side

Precondition

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

Returns

true if the system has a unique solution

Warning

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

Definition at line 262 of file bit-matrix.cc.

4.1.3.11 `bm_solve_upper_triangular()`

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

Solve a linear system in upper triangular form.

Solve the linear equation $Ax = b$.

Parameters

A	Upper triangular matrix
b	Right hand side

Precondition

```

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

```

Returns

true if the system has a unique solution

Warning

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

Definition at line 295 of file bit-matrix.cc.

4.1.3.12 `bm_transpose()` [1/2]

```

void bm_transpose (
    bit_matrix_t & N,
    const bit_matrix_t & M )

```

Transpose a bit matrix.

Precondition

```

bm_num_columns(N) == bm_num_rows(M)
bm_num_rows(N) == bm_num_columns(M)

```

Definition at line 59 of file bit-matrix.cc.

4.1.3.13 `bm_transpose()` [2/2]

```

bit_matrix_t bm_transpose (
    const bit_matrix_t & M )

```

Transpose a bit matrix.

Parameters

<i>M</i>	Bit matrix
----------	------------

Returns

Transposed bit matrix

Definition at line 73 of file bit-matrix.cc.

4.1.3.14 `bv_add()` [1/2]

```
void bv_add (
    bit_vector_t & dest,
    const bit_vector_t & src )
```

Add two bit vectors.

Equivalent to `dest = dest + src`.

Parameters

<i>dest</i>	Destination bit vector
<i>src</i>	Source bit vector

Warning

Vectors must be of the same size.

Definition at line 124 of file bit-vector.cc.

4.1.3.15 `bv_add()` [2/2]

```
void bv_add (
    bit_vector_t & dest,
    const bit_vector_t & x,
    const bit_vector_t & y )
```

Add two bit vectors.

Equivalent to `dest = x + y`.

Parameters

<i>dest</i>	Destination bit vector
<i>x</i>	First operand
<i>y</i>	Second operand

Warning

Vectors must be of the same size.

Definition at line 133 of file bit-vector.cc.

4.1.3.16 bv_from_size_type()

```
void bv_from_size_type (
    bit_vector_t & x,
    std::size_t u )
```

Convert a size_t to a small bit vector.

Parameters

<i>x</i>	Output bit vector
<i>u</i>	Unsigned integer representing a bit vector

Precondition

`x.size() <= 8 * sizeof(std::size_t)`

Warning

Depending on the size of the output bit vector, some bits might be lost. The original bit vector can be reconstructed only if it is small and the unsigned integer *u* is the result of `bv_to_size_type`.

Definition at line 201 of file bit-vector.cc.

4.1.3.17 bv_from_stream()

```
bit_vector_t bv_from_stream (
    std::istream & stream )
```

Read a bit vector from a stream.

Parameters

<i>stream</i>	Input stream
---------------	--------------

Returns

A `bit_vector_t`

Definition at line 232 of file bit-vector.cc.

4.1.3.18 `bv_from_string()`

```
bit_vector_t bv_from_string (
    const std::string & str )
```

Read a bit vector from a string.

Parameters

<code>str</code>	Input string
------------------	--------------

Returns

A `bit_vector_t`

Definition at line 216 of file `bit-vector.cc`.

4.1.3.19 `bv_from_vector_bool()`

```
void bv_from_vector_bool (
    bit_vector_t & x,
    const std::vector< bool > & y )
```

Convert a bool vector to a bit vector.

Warning

Vectors must be of the same size.

Definition at line 156 of file `bit-vector.cc`.

4.1.3.20 `bv_to_size_type()` [1/2]

```
std::size_t bv_to_size_type (
    const bit_vector_t & x )
```

Convert a small bit vector to a `size_t`.

`x[0]` is the least significant bit.

Parameters

<code>x</code>	Input bit vector
----------------	------------------

Returns

An unsigned integer representing *x*

Precondition

`x.size() <= 8 * sizeof(std::size_t)`

Definition at line 169 of file bit-vector.cc.

4.1.3.21 `bv_to_size_type()` [2/2]

```
std::size_t bv_to_size_type (
    const bit_vector_t & x,
    int start,
    int stop )
```

Convert a slice of a small bit vector to a `size_t`.

`x[start]` is the least significant bit.

`x[stop-1]` is the most significant bit.

Parameters

<i>x</i>	Input bit vector
<i>start</i>	Start bit
<i>stop</i>	Stop bit

Returns

An unsigned integer representing `x[start], ..., x[stop-1]`

Precondition

`start` in `[0, x.size())`

`stop` in `[start+1, x.size())`

`(stop - start) <= 8 * sizeof(std::size_t)`

Definition at line 184 of file bit-vector.cc.

4.1.3.22 `bv_to_vector_bool()`

```
void bv_to_vector_bool (
    std::vector< bool > & y,
    const bit_vector_t & x )
```

Convert a bit vector to a bool vector.

Warning

Vectors must be of the same size.

Definition at line 143 of file bit-vector.cc.

4.1.3.23 ensure()

```
void hnco::ensure (
    bool b,
    const std::string message ) [inline]
```

Ensure that a condition is satisfied or throw a runtime exception.

Parameters

<i>b</i>	Boolean
<i>message</i>	Message to display if the boolean is false

Definition at line 39 of file util.hh.

4.1.3.24 is_in_range() [1/2]

```
bool hnco::is_in_range (
    int i,
    int a,
    int b ) [inline]
```

Check whether an index is in a given range.

Parameters

<i>i</i>	Index
<i>a</i>	Lower bound
<i>b</i>	Upper bound (excluded)

Returns

true if $i \geq a$ and $i < b$

Definition at line 58 of file util.hh.

4.1.3.25 is_in_range() [2/2]

```
bool hnco::is_in_range (
    int i,
    int n ) [inline]
```

Check whether an index is in a given range.

The lower bound is implicit and is equal to 0.

Parameters

<i>i</i>	Index
<i>n</i>	Upper bound (excluded)

Returns

true if $i \geq 0$ and $i < n$

Definition at line 67 of file util.hh.

4.1.3.26 load_from_archive()

```
void hnco::load_from_archive (
    T & object,
    std::string path,
    std::string name )
```

Load from a boost archive.

Parameters

<i>object</i>	Object to load
<i>path</i>	Path of the file
<i>name</i>	Class name

Definition at line 44 of file serialization.hh.

4.1.3.27 perm_identity()

```
void hnco::perm_identity (
    permutation_t & s ) [inline]
```

Identity permutation.

Warning

This function does not set the size of the permutation.

Definition at line 47 of file permutation.hh.

4.1.3.28 perm_random()

```
void hnco::perm_random (
    permutation_t & s ) [inline]
```

Sample a random permutation.

Warning

This function does not set the size of the permutation.

Definition at line 60 of file permutation.hh.

4.1.3.29 save_to_archive()

```
void hnco::save_to_archive (
    const T & object,
    std::string path,
    std::string name )
```

Save to a boost archive.

Parameters

<i>object</i>	Object to save
<i>path</i>	Path of the file
<i>name</i>	Class name

Definition at line 64 of file serialization.hh.

4.1.3.30 sbv_flip()

```
void sbv_flip (
    bit_vector_t & x,
    const sparse_bit_vector_t & sbv )
```

Flip many bits of a bit vector.

Parameters

<i>x</i>	Input-output bit vector
<i>sbv</i>	Bits to flip

Definition at line 54 of file sparse-bit-vector.cc.

4.1.3.31 sbv_is_valid() [1/2]

```
bool sbv_is_valid (
    const sparse_bit_vector_t & sbv )
```

Check that a sparse bit vector is valid.

A sparse bit vector is valid if:

- Its elements are non negative.
- Its elements are sorted in non-descending order.

Definition at line 32 of file sparse-bit-vector.cc.

4.1.3.32 sbv_is_valid() [2/2]

```
bool sbv_is_valid (
    const sparse_bit_vector_t & sbv,
    int n )
```

Check that a sparse bit vector is valid.

A sparse bit vector is valid if:

- Its elements are non negative.
- Its elements are sorted in non-descending order.
- Its elements are valid indices w.r.t. the given dimension.

Parameters

<i>sbv</i>	Input sparse bit vector
<i>n</i>	Dimension

Definition at line 43 of file sparse-bit-vector.cc.

4.2 hngo::algorithm Namespace Reference

Algorithms.

Namespaces

- [fast_efficient_p3](#)
Algorithms from the FastEfficientP3 library.
- [walsh_moment](#)
Algorithms using Walsh moments.

Classes

- class [Algorithm](#)
Abstract search algorithm.
- class [CompleteSearch](#)
Complete search.
- class [Restart](#)
Restart.
- class [Crossover](#)
Crossover
- class [UniformCrossover](#)
Uniform crossover.
- class [BiasedCrossover](#)
Biased crossover.
- class [GeneticAlgorithm](#)
Genetic algorithm.
- class [MuCommaLambdaEa](#)
(mu, lambda) EA.
- class [MuPlusLambdaEa](#)
(mu+lambda) EA.
- class [OnePlusLambdaCommaLambdaGa](#)
(1+(lambda, lambda)) genetic algorithm.
- class [OnePlusOneEa](#)
(1+1) EA.
- class [RandomSelection](#)
Random selection.
- class [UniformSelection](#)
Uniform selection.
- class [TournamentSelection](#)
Tournament selection.
- class [CommaSelection](#)
Comma selection.
- class [PlusSelection](#)
Plus selection.
- class [SelfAdjustingOnePlusOneEa](#)
Self-adjusting (1+1) evolutionary algorithm.
- class [Human](#)
Human

- class [IterativeAlgorithm](#)
Iterative search.
- class [FirstAscentHillClimbing](#)
First ascent hill climbing.
- class [LocalSearchAlgorithm](#)
Local search algorithm.
- class [RandomLocalSearch](#)
Random local search.
- class [RandomWalk](#)
Random walk.
- class [SimulatedAnnealing](#)
Simulated annealing.
- class [SteepestAscentHillClimbing](#)
Steepest ascent hill climbing.
- class [Mimic](#)
Mutual information maximizing input clustering.
- struct [Population](#)
Population
- class [CompactGa](#)
Compact genetic algorithm.
- class [Mmas](#)
Max-min ant system.
- class [NpsPbil](#)
Population-based incremental learning with negative and positive selection.
- class [Pbil](#)
Population-based incremental learning.
- class [PvAlgorithm](#)
Probability vector algorithm.
- class [Umda](#)
Univariate marginal distribution algorithm.
- class [RandomSearch](#)
Random search.

Typedefs

- using [solution_t](#) = std::pair< [bit_vector_t](#), double >
Type of a solution.

Functions

- template<class T >
bool [matrix_is_symmetric](#) (const std::vector< std::vector< T > > &A)
Check for symmetric matrix.
- template<class T >
bool [matrix_is_strictly_lower_triangular](#) (const std::vector< std::vector< T > > &A)
Check for strictly lower triangular matrix.
- template<class T >
bool [matrix_has_diagonal](#) (const std::vector< std::vector< T > > &A, T x)
Check for diagonal elements.

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

Type and functions related to probability vectors

Output and input-output function parameters appear at the beginning of the parameter list.

Output and input-output `pv_t` parameters are passed by reference and must have the right size for the considered function.

Input object parameters are passed by const reference.

- using `pv_t = std::vector< double >`
Probability vector type.
- `double pv_entropy (const pv_t &pv)`
Entropy of a probability vector.
- `void pv_sample (bit_vector_t &x, const pv_t &pv)`
Sample a bit vector.
- `void pv_uniform (pv_t &pv)`
Probability vector of the uniform distribution.
- `void pv_init (pv_t &pv)`
Initialize.
- `void pv_add (pv_t &pv, const bit_vector_t &x)`
Accumulate a bit vector into a probability vector.
- `void pv_average (pv_t &pv, int count)`
Average.
- `template<class T >`
`void pv_update (pv_t &pv, double rate, const T &x)`
Update a probability vector.
- `void pv_update (pv_t &pv, double rate, const pv_t &x, const pv_t &y)`
Update a probability vector.
- `void pv_bound (pv_t &pv, double lower_bound, double upper_bound)`
Bound the elements of a probability vector.

4.2.1 Detailed Description

Algorithms.

4.2.2 Function Documentation

4.2.2.1 pv_add()

```
void pv_add (
    pv_t & pv,
    const bit_vector_t & x )
```

Accumulate a bit vector into a probability vector.

Equivalent to `pv += x`

Parameters

<i>pv</i>	Probability vector
<i>x</i>	Bit vector

Definition at line 58 of file probability-vector.cc.

4.2.2.2 pv_average()

```
void pv_average (
    pv_t & pv,
    int count )
```

Average.

Equivalent to $pv = pv / count$.

Parameters

<i>pv</i>	Probability vector
<i>count</i>	Number of accumulated bit vectors

Definition at line 67 of file probability-vector.cc.

4.2.2.3 pv_bound()

```
void pv_bound (
    pv_t & pv,
    double lower_bound,
    double upper_bound )
```

Bound the elements of a probability vector.

Parameters

<i>pv</i>	Probability vector
<i>lower_bound</i>	Lower bound
<i>upper_bound</i>	Upper bound

Definition at line 82 of file probability-vector.cc.

4.2.2.4 pv_init()

```
void hnco::algorithm::pv_init (
```

```
pv_t & pv ) [inline]
```

Initialize.

All the elements of the probability vector are set to 0.

Parameters

<i>pv</i>	Probability vector
-----------	--------------------

Definition at line 74 of file probability-vector.hh.

4.2.2.5 pv_sample()

```
void pv_sample (
    bit_vector_t & x,
    const pv_t & pv )
```

Sample a bit vector.

Parameters

<i>x</i>	Sampled bit vector
<i>pv</i>	Probability vector

Definition at line 46 of file probability-vector.cc.

4.2.2.6 pv_uniform()

```
void hnco::algorithm::pv_uniform (
    pv_t & pv ) [inline]
```

Probability vector of the uniform distribution.

All the elements of the probability vector are set to 1/2.

Parameters

<i>pv</i>	Probability vector
-----------	--------------------

Definition at line 66 of file probability-vector.hh.

4.2.2.7 `pv_update()` [1/2]

```
void pv_update (
    pv_t & pv,
    double rate,
    const pv_t & x,
    const pv_t & y )
```

Update a probability vector.

Equivalent to `pv += rate(x - y)`

Parameters

<i>pv</i>	Probability vector
<i>rate</i>	Rate
<i>x</i>	Attractor probability vector
<i>y</i>	Repulsor probability vector

Definition at line 73 of file `probability-vector.cc`.

4.2.2.8 `pv_update()` [2/2]

```
void hnco::algorithm::pv_update (
    pv_t & pv,
    double rate,
    const T & x )
```

Update a probability vector.

Equivalent to `pv += rate * (x - pv)`

Parameters

<i>pv</i>	Probability vector
<i>rate</i>	Rate
<i>x</i>	Attractor bit vector

Definition at line 103 of file `probability-vector.hh`.

4.3 `hnco::algorithm::fast_efficient_p3` Namespace Reference

Algorithms from the FastEfficientP3 library.

Classes

- class [Hboa](#)

- Hierarchical Bayesian Optimization Algorithm.*
 - class [HncoEvaluator](#)
Evaluator for HNCO functions.
 - struct [Implementation](#)
Implementation
 - class [Ltga](#)
Linkage Tree Genetic Algorithm.
 - class [ParameterLessPopulationPyramid](#)
Parameter-less Population Pyramid.

4.3.1 Detailed Description

Algorithms from the FastEfficientP3 library.

4.4 `hnco::algorithm::walsh_moment` Namespace Reference

Algorithms using Walsh moments.

Classes

- class [BmPbil](#)
Boltzmann machine PBIL.
- class [LowerTriangularWalshMoment2GibbsSampler](#)
Gibbs sampler with lower triangular Walsh moments.
- class [SymmetricWalshMoment2GibbsSampler](#)
Gibbs sampler with symmetric Walsh moments.
- class [Hea](#)
Herding evolutionary algorithm.
- class [LowerTriangularWalshMoment2Herding](#)
Herding with lower triangular Walsh moment.
- class [SymmetricWalshMoment2Herding](#)
Herding with symmetric Walsh moment.
- struct [LowerTriangularWalshMoment2](#)
Lower triangular Walsh moment.
- struct [SymmetricWalshMoment2](#)
Symmetric Walsh moment.

4.4.1 Detailed Description

Algorithms using Walsh moments.

4.5 `hnco::app` Namespace Reference

Classes for applications.

Classes

- class [AlgorithmFactory](#)
Algorithm factory.
- class [CommandLineAlgorithmFactory](#)
Command line algorithm factory.
- class [CommandLineApplication](#)
Command line application.
- class [DecoratedFunctionFactory](#)
Decorated function factory.
- class [FunctionFactory](#)
Function factory.
- class [CommandLineFunctionFactory](#)
Command line function factory.
- class [HncoOptions](#)
Command line options for hnco.
- class [FfgenOptions](#)
Command line options for ffgen.
- class [MapgenOptions](#)
Command line options for mapgen.

Functions

- `std::ostream & operator<< (std::ostream &stream, const HncoOptions &options)`
Print a header containing the parameter values.
- `std::ostream & operator<< (std::ostream &stream, const FfgenOptions &options)`
Print a header containing the parameter values.
- `std::ostream & operator<< (std::ostream &stream, const MapgenOptions &options)`
Print a header containing the parameter values.

4.5.1 Detailed Description

Classes for applications.

4.6 hnco::exception Namespace Reference

Exceptions.

Classes

- class [LastEvaluation](#)
Last evaluation.
- class [TargetReached](#)
Target reached.

4.6.1 Detailed Description

Exceptions.

4.7 hnc0::function Namespace Reference

Functions defined on bit vectors.

Namespaces

- [controller](#)
Controllers.
- [modifier](#)
Modifiers.

Classes

- class [SummationCancellation](#)
Summation cancellation.
- class [SinusSummationCancellation](#)
Summation cancellation with sinus.
- class [EqualProducts](#)
Equal products.
- class [Factorization](#)
Factorization.
- class [FourPeaks](#)
Four Peaks.
- class [SixPeaks](#)
Six Peaks.
- class [NearestNeighborIsingModel1](#)
Nearest neighbor Ising model in one dimension.
- class [NearestNeighborIsingModel2](#)
Nearest neighbor Ising model in two dimensions.
- class [Jump](#)
Jump.
- class [DeceptiveJump](#)
Deceptive jump.
- class [Labs](#)
Low autocorrelation binary sequences.
- class [LinearFunction](#)
Linear function.
- class [LongPath](#)
Long path.
- class [AbstractMaxSat](#)
Abstract class for MaxSat-like functions.
- class [MaxSat](#)
MAX-SAT.

- class [MaxNae3Sat](#)
Max not-all-equal 3SAT.
- class [NkLandscape](#)
NK landscape.
- class [ParsedMultivariateFunction](#)
Parsed multivariate function.
- class [Partition](#)
Partition.
- class [FunctionPlugin](#)
Function plugin
- class [PythonFunction](#)
Python function.
- class [Qubo](#)
Quadratic unconstrained binary optimization.
- class [Sudoku](#)
Sudoku
- class [OneMax](#)
OneMax.
- class [LeadingOnes](#)
Leading ones.
- class [Needle](#)
Needle in a haystack.
- class [Hiff](#)
Hierarchical if and only if.
- class [Ridge](#)
Ridge.
- class [Plateau](#)
Plateau.
- class [Trap](#)
Trap.
- class [Tsp](#)
Traveling salesman problem.
- class [WalshExpansion1](#)
Walsh expansion of degree 1.
- class [WalshExpansion2](#)
Walsh expansion of degree 2.
- class [WalshExpansion](#)
Walsh expansion.
- struct [ScalarToDouble](#)
Convert a scalar to a double.
- struct [ComplexToDouble](#)
Convert a complex to a double.
- class [Decorator](#)
Function decorator
- class [Function](#)
Function
- class [MultivariateFunctionAdapter](#)
Multivariate function adapter.
- class [PermutationFunctionAdapter](#)
Permutation function adapter.
- class [UniversalFunctionAdapter](#)

- *Universal function adapter.*
class [UniversalFunction](#)
- *Universal function.*
struct [WalshTerm](#)
- *Walsh transform term.*

Functions

- void [compute_walsh_transform](#) ([function::Function](#) *function, std::vector< [function::WalshTerm](#) > &terms)
Compute the Walsh transform of the function.
- void [compute_fast_walsh_transform](#) ([function::Function](#) *function, std::vector< [function::WalshTerm](#) > &terms)
Compute the Walsh transform of the function using a fast Walsh transform.
- bool [bv_is_locally_maximal](#) (const [bit_vector_t](#) &bv, [Function](#) &fn, [neighborhood::NeighborhoodIterator](#) &it)
Check whether a bit vector is locally maximal.
- bool [bv_is_globally_maximal](#) (const [bit_vector_t](#) &bv, [Function](#) &fn)
Check whether a bit vector is globally maximal.

4.7.1 Detailed Description

Functions defined on bit vectors.

4.7.2 Function Documentation

4.7.2.1 [compute_fast_walsh_transform\(\)](#)

```
void compute_fast_walsh_transform (
    function::Function * function,
    std::vector< function::WalshTerm > & terms )
```

Compute the Walsh transform of the function using a fast Walsh transform.

Let f be a fitness function defined on the hypercube $\{0,1\}^n$. Then it can be expressed as $\sum_u c_u \chi_u$ where $c_u = \langle f, \chi_u \rangle$, $\langle f, g \rangle = \frac{1}{2^n} \sum_x f(x)g(x)$, $\chi_u(x) = (-1)^{x \cdot u}$, and $x \cdot u = \sum_i x_i u_i \pmod{2}$. In the respective sums, we have x and u in the hypercube and i in $\{1, \dots, n\}$.

We have dropped the normalizing constant 2^n since we are mostly interested in ratios $|c_u/c_{\max}|$, where c_{\max} is the coefficient with the largest amplitude. It is also helpful to achieve exact computations in the case of functions taking only integer values.

Parameters

<i>function</i>	Function the Walsh transform of which to compute
<i>terms</i>	Vector of non zero terms of the Walsh transform

Warning

The time complexity is exponential in the dimension n . It requires 2^n function evaluations and $n2^n$ additions, which is faster than `compute_walsh_transform`.

The size of the Walsh transform is potentially exponential in the dimension n . For example, if $n = 10$ then the number of terms is at most 1024.

Definition at line 77 of file `function.cc`.

4.7.2.2 compute_walsh_transform()

```
void compute_walsh_transform (
    function::Function * function,
    std::vector< function::WalshTerm > & terms )
```

Compute the Walsh transform of the function.

Let f be a fitness function defined on the hypercube $\{0,1\}^n$. Then it can be expressed as $\sum_u c_u \chi_u$ where $c_u = \langle f, \chi_u \rangle$, $\langle f, g \rangle = \frac{1}{2^n} \sum_x f(x)g(x)$, $\chi_u(x) = (-1)^{x \cdot u}$, and $x \cdot u = \sum_i x_i u_i \pmod{2}$. In the respective sums, we have x and u in the hypercube and i in $\{1, \dots, n\}$.

We have dropped the normalizing constant 2^n since we are mostly interested in ratios $|c_u/c_{\max}|$, where c_{\max} is the coefficient with the largest amplitude. It is also helpful to achieve exact computations in the case of functions taking only integer values.

Parameters

<i>function</i>	Function the Walsh transform of which to compute
<i>terms</i>	Vector of non zero terms of the Walsh transform

Warning

The time complexity is exponential in the dimension n . The computation is done with two nested loops over the hypercube. It requires 2^n function evaluations and 2^{2n} dot products and additions.

The size of the Walsh transform is potentially exponential in the dimension n . For example, if $n = 10$ then the number of terms is at most 1024.

Definition at line 33 of file `function.cc`.

4.8 hnco::function::controller Namespace Reference

Controllers.

Classes

- class [Controller](#)
Function controller.

- class [StopOnTarget](#)
Stop on target.
- class [StopOnMaximum](#)
Stop on maximum.
- class [CallCounter](#)
Call counter.
- class [OnBudgetFunction](#)
Function with a limited number of evaluations.
- class [ProgressTracker](#)
ProgressTracker.
- class [Cache](#)
Cache.

Functions

- `std::ostream & operator<< (std::ostream &stream, const ProgressTracker::Event &event)`
Insert formatted output.

4.8.1 Detailed Description

Controllers.

4.9 hngo::function::modifier Namespace Reference

Modifiers.

Classes

- class [Modifier](#)
Function modifier.
- class [Negation](#)
Negation.
- class [FunctionMapComposition](#)
Composition of a function and a map.
- class [AdditiveGaussianNoise](#)
Additive Gaussian Noise.
- class [ParsedModifier](#)
Parsed modifier.
- class [PriorNoise](#)
Prior noise.

4.9.1 Detailed Description

Modifiers.

4.10 hnco::logging Namespace Reference

Logging.

Classes

- class [LogContext](#)
Log context.
- class [ProgressTrackerContext](#)
Log context for ProgressTracker.
- class [Logger](#)
Logger.

4.10.1 Detailed Description

Logging.

4.11 hnco::map Namespace Reference

Maps.

Classes

- class [Map](#)
Map
- class [Translation](#)
Translation.
- class [Permutation](#)
Permutation.
- class [LinearMap](#)
Linear map.
- class [AffineMap](#)
Affine map.
- class [MapComposition](#)
Map composition.
- class [Injection](#)
Injection.
- class [Projection](#)
Projection.
- class [TsAffineMap](#)
Transvection sequence affine map.
- struct [Transvection](#)
Transvection.

Types and functions related to transvections

Output and input-output function parameters appear at the beginning of the parameter list.

Output and input-output `transvection_sequence_t` parameters are passed by reference.

Input object parameters are passed by const reference.

- using `transvection_sequence_t` = `std::vector< Transvection >`
Transvection sequence.
- bool `transvections_commute` (const `Transvection` &a, const `Transvection` &b)
Check whether two transvections commute.
- bool `transvections_are_disjoint` (const `Transvection` &a, const `Transvection` &b)
Check whether two transvections are disjoint.
- bool `ts_is_valid` (const `transvection_sequence_t` &ts)
Check validity.
- bool `ts_is_valid` (const `transvection_sequence_t` &ts, int n)
Check validity.
- void `ts_display` (const `transvection_sequence_t` &ts, `std::ostream` &stream)
Display a transvection sequence.
- void `ts_random` (`transvection_sequence_t` &ts, int n, int t)
Sample a random transvection sequence.
- void `ts_random_commuting` (`transvection_sequence_t` &ts, int n, int t)
Sample a random sequence of commuting transvections.
- void `ts_random_unique_source` (`transvection_sequence_t` &ts, int n, int t)
Sample a random sequence of transvections with unique source.
- void `ts_random_unique_destination` (`transvection_sequence_t` &ts, int n, int t)
Sample a random sequence of transvections with unique destination.
- void `ts_random_disjoint` (`transvection_sequence_t` &ts, int n, int t)
Sample a random sequence of disjoint transvections.
- void `ts_random_non_commuting` (`transvection_sequence_t` &ts, int n, int t)
Sample a random sequence of non commuting transvections.
- void `ts_multiply` (`bit_vector_t` &x, const `transvection_sequence_t` &ts)
Multiply a vector by a transvection sequence from the left.
- void `ts_multiply` (`bit_matrix_t` &M, const `transvection_sequence_t` &ts)
Multiply a matrix by a transvection sequence from the left.

4.11.1 Detailed Description

Maps.

4.11.2 Typedef Documentation

4.11.2.1 transvection_sequence_t

```
using transvection_sequence_t = std::vector<Transvection>
```

Transvection sequence.

The general linear group of a linear space of dimension n over the finite field F_2 is the group of invertible n by n bit matrices.

Any invertible bit matrix can be expressed as a finite product of transvections.

Finite transvection sequences can then represent all invertible bit matrices.

Definition at line 166 of file transvection.hh.

4.11.3 Function Documentation

4.11.3.1 ts_is_valid() [1/2]

```
bool ts_is_valid (
    const transvection_sequence_t & ts )
```

Check validity.

Parameters

<i>ts</i>	Transvection sequence
-----------	-----------------------

Definition at line 150 of file transvection.cc.

4.11.3.2 ts_is_valid() [2/2]

```
bool ts_is_valid (
    const transvection_sequence_t & ts,
    int n )
```

Check validity.

Parameters

<i>ts</i>	Transvection sequence
<i>n</i>	Dimension

Definition at line 156 of file transvection.cc.

4.11.3.3 ts_multiply() [1/2]

```
void ts_multiply (
    bit_matrix_t & M,
    const transvection_sequence_t & ts )
```

Multiply a matrix by a transvection sequence from the left.

Parameters

<i>ts</i>	Transvection sequence
<i>M</i>	Bit matrix

Precondition

```
ts_is_valid(ts)
ts_is_valid(ts, bm_num_rows(M))
```

Warning

This function modifies the given bit vector.

Definition at line 366 of file transvection.cc.

4.11.3.4 ts_multiply() [2/2]

```
void ts_multiply (
    bit_vector_t & x,
    const transvection_sequence_t & ts )
```

Multiply a vector by a transvection sequence from the left.

Parameters

<i>ts</i>	Transvection sequence
<i>x</i>	Bit vector

Precondition

```
ts_is_valid(ts)
ts_is_valid(ts, x.size())
```

Warning

This function modifies the given bit vector.

Definition at line 356 of file transvection.cc.

4.11.3.5 ts_random()

```
void ts_random (
    transvection_sequence_t & ts,
    int n,
    int t )
```

Sample a random transvection sequence.

Parameters

<i>ts</i>	Transvection sequence
<i>n</i>	Dimension
<i>t</i>	Length of the sequence

Precondition

$n > 1$
 $t \geq 0$

Definition at line 172 of file transvection.cc.

4.11.3.6 ts_random_commuting()

```
void ts_random_commuting (
    transvection_sequence_t & ts,
    int n,
    int t )
```

Sample a random sequence of commuting transvections.

This function ensures that all transvections in the sequence commute.

Parameters

<i>ts</i>	Transvection sequence
<i>n</i>	Dimension
<i>t</i>	Length of the sequence

Precondition

$n > 1$
 $t \geq 0$

Warning

If $t > \text{floor}(n / 2)$ then t is set to $\text{floor}(n / 2)$.

If $t = \text{floor}(n / 2)$ then the space and time complexity of `ts_random_commuting` is quadratic in the dimension n .

Definition at line 183 of file `transvection.cc`.

4.11.3.7 ts_random_disjoint()

```
void ts_random_disjoint (
    transvection_sequence_t & ts,
    int n,
    int t )
```

Sample a random sequence of disjoint transvections.

Two transvections τ_{ij} and τ_{kl} are said to be disjoint if the pairs $\{i,j\}$ and $\{k,l\}$ are disjoint.

If $2t > n$ then the sequence length is set to the largest t such that $2t \leq n$.

Parameters

<i>ts</i>	Transvection sequence
<i>n</i>	Dimension
<i>t</i>	Length of the sequence

Precondition

$n > 1$
 $t \geq 0$

Definition at line 311 of file `transvection.cc`.

4.11.3.8 ts_random_non_commuting()

```
void ts_random_non_commuting (
    transvection_sequence_t & ts,
    int n,
    int t )
```

Sample a random sequence of non commuting transvections.

This function ensures that two consecutive transvections do not commute.

Parameters

<i>ts</i>	Transvection sequence
<i>n</i>	Dimension
<i>t</i>	Length of the sequence

Precondition

$n > 1$
 $t \geq 0$

Definition at line 341 of file transvection.cc.

4.11.3.9 ts_random_unique_destination()

```
void ts_random_unique_destination (
    transvection_sequence_t & ts,
    int n,
    int t )
```

Sample a random sequence of transvections with unique destination.

A transvection sequence with unique destination is such that, for each source, there is a unique destination.

Parameters

<i>ts</i>	Transvection sequence
<i>n</i>	Dimension
<i>t</i>	Length of the sequence

Precondition

$n > 1$
 $t \geq 0$

Definition at line 278 of file transvection.cc.

4.11.3.10 ts_random_unique_source()

```
void ts_random_unique_source (
    transvection_sequence_t & ts,
    int n,
    int t )
```

Sample a random sequence of transvections with unique source.

A transvection sequence with unique source is such that, for each destination, there is a unique source.

Parameters

<i>ts</i>	Transvection sequence
<i>n</i>	Dimension
<i>t</i>	Length of the sequence

Precondition

$$n > 1$$

$$t \geq 0$$

Definition at line 245 of file transvection.cc.

4.12 hnco::multiobjective Namespace Reference

Multiobjective optimization.

Namespaces

- [algorithm](#)
Multiobjective Algorithms.
- [app](#)
Classes for applications.
- [function](#)
Functions defined on bit vectors.

4.12.1 Detailed Description

Multiobjective optimization.

4.13 hnco::multiobjective::algorithm Namespace Reference

Multiobjective Algorithms.

Classes

- class [Algorithm](#)
Abstract multiobjective search algorithm.
- class [IterativeAlgorithm](#)
Iterative algorithm.
- struct [FrontDistancePair](#)
Front-distance pair.
- class [Nsga2](#)
NSGA-II.
- class [Nsga2ParetoFrontComputation](#)
Pareto front computation from the NSGA-II paper.
- struct [Population](#)
Population
- class [TournamentSelection](#)
Tournament selection.

Functions

- bool `operator<` (const `FrontDistancePair` &a, const `FrontDistancePair` &b)
Comparison operator for front-distance pairs.

4.13.1 Detailed Description

Multiobjective Algorithms.

4.13.2 Function Documentation

4.13.2.1 `operator<()`

```
bool hnco::multiobjective::algorithm::operator< (  
    const FrontDistancePair & a,  
    const FrontDistancePair & b ) [inline]
```

Comparison operator for front-distance pairs.

Favors individuals with smaller Pareto front then greater crowding distance.

Definition at line 59 of file `nsga2.hh`.

4.14 `hnco::multiobjective::app` Namespace Reference

Classes for applications.

Classes

- class `AlgorithmFactory`
Algorithm factory.
- class `CommandLineAlgorithmFactory`
Command line algorithm factory.
- class `CommandLineApplication`
Command line application.
- class `FunctionFactory`
Function factory.
- class `CommandLineFunctionFactory`
Command line function factory.
- class `HncoOptions`
Command line options for `hnco-mo`.

Functions

- `std::ostream & operator<< (std::ostream &stream, const HncoOptions &options)`
Print a header containing the parameter values.

4.14.1 Detailed Description

Classes for applications.

4.15 hnco::multiobjective::function Namespace Reference

Functions defined on bit vectors.

Classes

- class `ParsedMultivariateFunction`
Parsed multivariate function.
- class `PythonFunction`
Python function.
- class `Function`
Function
- class `MultivariateFunctionAdapter`
Multivariate function adapter.
- class `UniversalFunctionAdapter`
Universal function adapter.
- class `UniversalFunction`
Universal function.

Typedefs

- using `value_t = std::vector< double >`
Value type.

Functions

- bool `dominates (const value_t &a, const value_t &b)`
Domination relation.
- void `value_display (const value_t &a, std::ostream &stream)`
Display a value.

4.15.1 Detailed Description

Functions defined on bit vectors.

4.15.2 Typedef Documentation

4.15.2.1 value_t

```
using value_t = std::vector<double>
```

Value type.

A value type is the type of the output of a [Function](#) in the context of multiobjective optimization.

Definition at line 42 of file value.hh.

4.15.3 Function Documentation

4.15.3.1 dominates()

```
bool hnco::multiobjective::function::dominates (
    const value_t & a,
    const value_t & b ) [inline]
```

Domination relation.

Parameters

<i>a</i>	First value
<i>b</i>	Second value

Returns

true if a dominates b with respect to minimization

Definition at line 51 of file value.hh.

4.16 hnco::neighborhood Namespace Reference

Neighborhoods for local search.

Classes

- class [NeighborhoodIterator](#)
Neighborhood iterator.

- class [SingleBitFlipIterator](#)
Single bit flip neighborhood iterator.
- class [HammingSphereIterator](#)
Hamming sphere neighborhood iterator.
- class [Neighborhood](#)
Neighborhood.
- class [SingleBitFlip](#)
One bit neighborhood.
- class [MultiBitFlip](#)
Multi bit flip.
- class [StandardBitMutation](#)
Standard bit mutation.
- class [HammingBall](#)
Hamming ball.
- class [HammingSphere](#)
Hamming sphere.

4.16.1 Detailed Description

Neighborhoods for local search.

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

4.17 hnco::random Namespace Reference

Random numbers.

Classes

- struct [Generator](#)
Random number generator.

4.17.1 Detailed Description

Random numbers.

4.18 hnco::representation Namespace Reference

Representations.

Classes

- class [LinearCategoricalRepresentation](#)
Linear categorical representation.
- class [IntegerCategoricalRepresentation](#)
Integer categorical representation.
- class [DyadicComplexRepresentation](#)
Dyadic complex representation.
- class [DyadicFloatRepresentation](#)
Dyadic float representation.
- class [DyadicIntegerRepresentation](#)
Dyadic integer representation.
- class [PermutationRepresentation](#)
Permutation representation.

Functions

- template<class T >
bool [difference_is_safe](#) (T a, T b)
Check whether the difference is safe.

4.18.1 Detailed Description

Representations.

4.18.2 Function Documentation

4.18.2.1 [difference_is_safe\(\)](#)

```
bool hnco::representation::difference_is_safe (
    T a,
    T b )
```

Check whether the difference is safe.

The template parameter T must be an integral type such as int or long.

The difference $b - a$ is safe if it can be represented by the type of a and b, i.e. there is no overflow.

Parameters

<i>a</i>	Smallest value
<i>b</i>	Greatest value

Precondition $a < b$

Definition at line 56 of file integer.hh.

Chapter 5

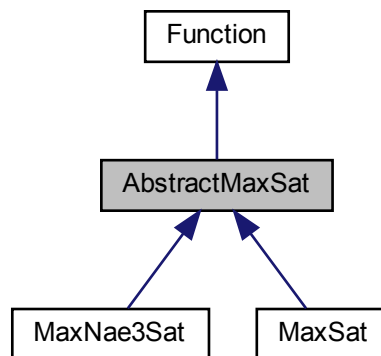
Class Documentation

5.1 AbstractMaxSat Class Reference

Abstract class for MaxSat-like functions.

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

Inheritance diagram for AbstractMaxSat:



Public Member Functions

- [AbstractMaxSat](#) ()
Default constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- void [display](#) (std::ostream &stream) const override
Display the expression.

Load and save instance

- void [load](#) (std::string path)
Load instance.
- void [save](#) (std::string path) const
Save instance.

Protected Member Functions

- void `load_` (std::istream &stream)
Load an instance.
- void `save_` (std::ostream &stream) const
Save an instance.

Protected Attributes

- std::vector< std::vector< int > > `_expression`
Expression.
- int `_num_variables`
Number of variables.

5.1.1 Detailed Description

Abstract class for MaxSat-like functions.

Definition at line 36 of file max-sat.hh.

5.1.2 Member Function Documentation

5.1.2.1 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

<i>path</i>	Path of the instance to load
-------------	------------------------------

Exceptions

<code>std::runtime_error</code>	
---------------------------------	--

Definition at line 87 of file max-sat.hh.

5.1.2.2 load_()

```
void load_ (
    std::istream & stream ) [protected]
```


Load an instance.

Parameters

<i>stream</i>	Input stream
---------------	--------------

Exceptions

<i>std::runtime_error</i>	
---------------------------	--

Definition at line 61 of file max-sat.cc.

5.1.2.3 save()

```
void save (
    std::string path ) const [inline]
```

Save instance.

Parameters

<i>path</i>	Path of the instance to save
-------------	------------------------------

Exceptions

<i>std::runtime_error</i>	
---------------------------	--

Definition at line 99 of file max-sat.hh.

5.1.2.4 save_()

```
void save_ (
    std::ostream & stream ) const [protected]
```

Save an instance.

Parameters

<i>stream</i>	Outputstream
---------------	--------------

Definition at line 153 of file max-sat.cc.

5.1.3 Member Data Documentation

5.1.3.1 `_expression`

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

Expression.

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

Definition at line 47 of file max-sat.hh.

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

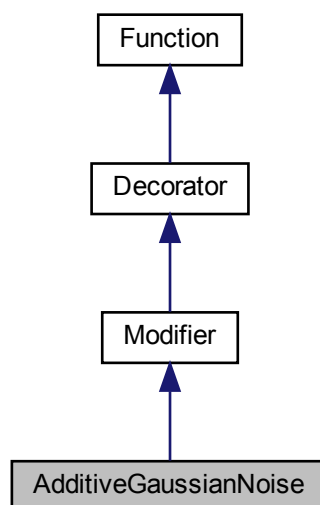
- lib/hnco/functions/collection/max-sat.hh
- lib/hnco/functions/collection/max-sat.cc

5.2 AdditiveGaussianNoise Class Reference

Additive Gaussian Noise.

```
#include <hnco/functions/modifiers/modifier.hh>
```

Inheritance diagram for AdditiveGaussianNoise:



Public Member Functions

- [AdditiveGaussianNoise](#) ([Function](#) *function, double stddev)
Constructor.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Information about the function

- int [get_bv_size](#) () const override
Get bit vector size.

Private Attributes

- `std::normal_distribution< double > _dist`
Normal distribution.

Additional Inherited Members

5.2.1 Detailed Description

Additive Gaussian Noise.

Definition at line 170 of file modifier.hh.

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

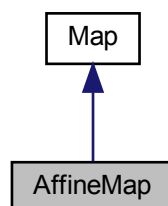
- lib/hnco/functions/modifiers/modifier.hh
- lib/hnco/functions/modifiers/modifier.cc

5.3 AffineMap Class Reference

Affine map.

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

Inheritance diagram for AffineMap:



Public Member Functions

- void [random](#) (int rows, int cols, bool surjective)
Random instance.
- void [map](#) (const [bit_vector_t](#) &input, [bit_vector_t](#) &output) override
Map
- int [get_input_size](#) () const override
Get input size.
- int [get_output_size](#) () const override
Get output size.
- bool [is_surjective](#) () const override
Check for surjective map.
- void [display](#) (std::ostream &stream) const override
Display.

Load and save map

- void [load](#) (std::string path)
Load map.
- void [save](#) (std::string path) const
Save map.

Private Member Functions

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

Private Attributes

- [bit_matrix_t _bm](#)
Bit matrix.
- [bit_vector_t _bv](#)
Translation vector

Friends

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

5.3.1 Detailed Description

Affine map.

An affine map f from F_2^m to F_2^n is defined by $f(x) = Ax + b$, where A is an $n \times m$ bit matrix and b is an n -dimensional bit vector.

Definition at line 331 of file map.hh.

5.3.2 Member Function Documentation

5.3.2.1 is_surjective()

```
bool is_surjective ( ) const [override], [virtual]
```

Check for surjective map.

Returns

true if rank(_bm) == bm_num_rows(_bm)

Reimplemented from [Map](#).

Definition at line 156 of file map.cc.

5.3.2.2 load()

```
void load (
    std::string path ) [inline]
```

Load map.

Parameters

<i>path</i>	Path of the file
-------------	------------------

Exceptions

<i>std::runtime_error</i>	
---------------------------	--

Definition at line 405 of file map.hh.

5.3.2.3 random()

```
void random (
    int rows,
    int cols,
    bool surjective )
```

Random instance.

Parameters

<i>rows</i>	Number of rows
<i>cols</i>	Number of columns
<i>surjective</i>	Flag to ensure a surjective map

Exceptions

<i>std::runtime_error</i>	
---------------------------	--

Definition at line 119 of file map.cc.

5.3.2.4 save()

```
void save (
    std::string path ) const [inline]
```

Save map.

Parameters

<i>path</i>	Path of the file
-------------	------------------

Exceptions

<i>std::runtime_error</i>	
---------------------------	--

Definition at line 412 of file map.hh.

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

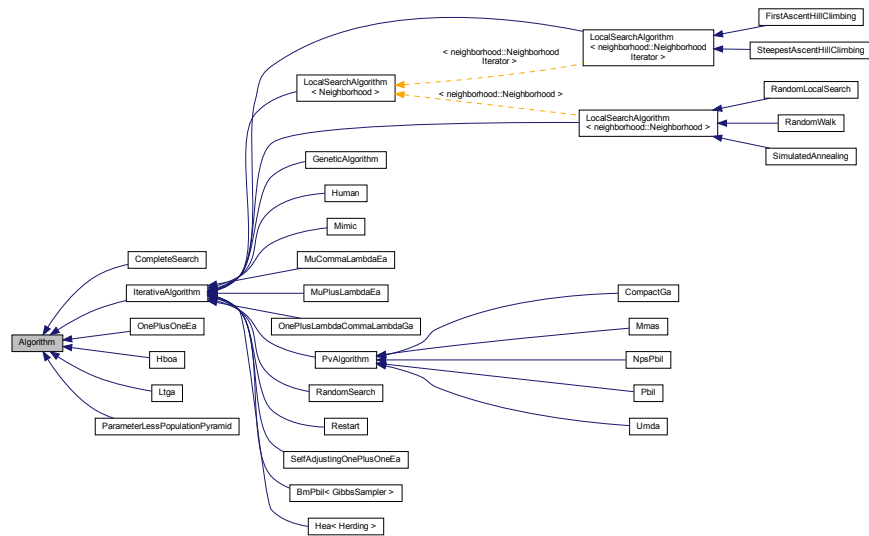
- lib/hnco/maps/map.hh
- lib/hnco/maps/map.cc

5.4 Algorithm Class Reference

Abstract search algorithm.

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

Inheritance diagram for Algorithm:



Public Member Functions

- [Algorithm](#) (int n)
Constructor.
- virtual [~Algorithm](#) ()
Destructor.
- void [set_log_context](#) (logging::LogContext *log_context)
Set the log context.

Optimization

- virtual void [maximize](#) (const std::vector< [function::Function](#) * > &functions)=0
Maximize.
- virtual void [finalize](#) ()
Finalize.
- const [solution_t](#) & [get_solution](#) ()
Get the solution.

Protected Member Functions

- void [set_functions](#) (const std::vector< [function::Function](#) * > &functions)
Set functions.
- int [get_bv_size](#) ()
Get bit vector size.

Managing solution

- void [random_solution](#) ()
Random solution.
- void [set_solution](#) (const [bit_vector_t](#) &bv, double value)

- *Set solution.*
void [set_solution](#) (const [bit_vector_t](#) &bv)
- *Set solution.*
void [update_solution](#) (const [bit_vector_t](#) &bv, double value)
- *Update solution (strict)*
void [update_solution](#) (const [solution_t](#) &s)
- *Update solution (strict)*
void [update_solution](#) (const [bit_vector_t](#) &bv)
- *Update solution (strict).*

Protected Attributes

- [std::vector< function::Function * > _functions](#)
Functions.
- [function::Function * _function](#)
Function.
- [solution_t _solution](#)
Solution.

Parameters

- [logging::LogContext * _log_context](#) = nullptr
Log context.

5.4.1 Detailed Description

Abstract search algorithm.

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

Definition at line 46 of file `algorithm.hh`.

5.4.2 Member Function Documentation

5.4.2.1 finalize()

```
virtual void finalize ( ) [inline], [virtual]
```

Finalize.

Does nothing.

It is usually overridden by algorithms which do not keep `_solution` up-to-date. In case `_function` throws a `LastEvaluation` exception, the algorithm might leave `_solution` in an undefined state. This can be fixed in this member function.

Reimplemented in [RandomLocalSearch](#), [SelfAdjustingOnePlusOneEa](#), [OnePlusOneEa](#), [ParameterLessPopulationPyramid](#), [Ltga](#), and [Hboa](#).

Definition at line 143 of file `algorithm.hh`.

5.4.2.2 set_solution()

```
void set_solution (
    const bit_vector_t & bv ) [protected]
```

Set solution.

Warning

Evaluates the function once.

Definition at line 45 of file algorithm.cc.

5.4.2.3 update_solution()

```
void update_solution (
    const bit_vector_t & bv ) [protected]
```

Update solution (strict).

Warning

Evaluates the function once.

Definition at line 69 of file algorithm.cc.

5.4.3 Member Data Documentation

5.4.3.1 _functions

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

Functions.

Each thread has its own function.

Definition at line 54 of file algorithm.hh.

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

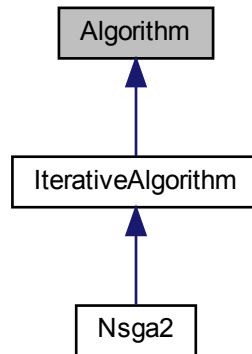
- lib/hnco/algorithms/algorithm.hh
- lib/hnco/algorithms/algorithm.cc

5.5 Algorithm Class Reference

Abstract multiobjective search algorithm.

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

Inheritance diagram for Algorithm:



Public Types

- using [Function](#) = [hnco::multiobjective::function::Function](#)
Function type.

Public Member Functions

- [Algorithm](#) (int n, int num_objectives)
Constructor.
- virtual [~Algorithm](#) ()
Destructor.
- void [set_log_context](#) ([logging::LogContext](#) *log_context)
Set the log context.

Optimization

- virtual void [minimize](#) (const std::vector< [Function](#) * > &functions)=0
Minimize.
- virtual const [Population](#) & [get_solutions](#) ()=0
Get solutions.

Protected Member Functions

- void [set_functions](#) (const std::vector< [Function](#) * > &functions)
Set functions.

Protected Attributes

- `std::vector< Function * > _functions`
Functions.
- `Function * _function`
Function.

Parameters

- `logging::LogContext * _log_context = nullptr`
Log context.

5.5.1 Detailed Description

Abstract multiobjective search algorithm.

All algorithms minimize some given function.

Definition at line 43 of file `algorithm.hh`.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Algorithm()

```
Algorithm (
    int n,
    int num_objectives ) [inline]
```

Constructor.

Parameters

<i>n</i>	Size of bit vectors
<i>num_objectives</i>	Number of objectives

Definition at line 85 of file `algorithm.hh`.

5.5.3 Member Data Documentation

5.5.3.1 _functions

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

Functions.

Each thread has its own function.

Definition at line 56 of file algorithm.hh.

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

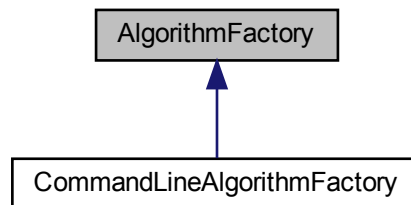
- lib/hnco/multiobjective/algorithms/algorithm.hh

5.6 AlgorithmFactory Class Reference

Algorithm factory.

```
#include <hnco/app/algorithm-factory.hh>
```

Inheritance diagram for AlgorithmFactory:



Public Member Functions

- virtual [hnco::algorithm::Algorithm](#) * [make](#) (int bv_size)=0
Make an algorithm.

5.6.1 Detailed Description

Algorithm factory.

Definition at line 32 of file algorithm-factory.hh.

5.6.2 Member Function Documentation

5.6.2.1 make()

```
virtual hnco::algorithm::Algorithm* make (  
    int bv_size ) [pure virtual]
```

Make an algorithm.

Parameters

<code>bv_size</code>	Bit vector size
----------------------	-----------------

Implemented in [CommandLineAlgorithmFactory](#).

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

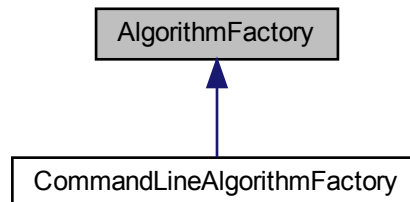
- `lib/hnco/app/algorithm-factory.hh`

5.7 AlgorithmFactory Class Reference

Algorithm factory.

```
#include <hnco/multiobjective/app/algorithm-factory.hh>
```

Inheritance diagram for AlgorithmFactory:



Public Member Functions

- virtual `hnco::multiobjective::algorithm::Algorithm * make` (int `bv_size`, int `num_objectives`)=0
Make an algorithm.

5.7.1 Detailed Description

Algorithm factory.

Definition at line 35 of file `algorithm-factory.hh`.

5.7.2 Member Function Documentation

5.7.2.1 make()

```
virtual hnco::multiobjective::algorithm::Algorithm* make (
    int bv_size,
    int num_objectives ) [pure virtual]
```

Make an algorithm.

Parameters

<code>bv_size</code>	Bit vector size
----------------------	-----------------

Implemented in [CommandLineAlgorithmFactory](#).

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

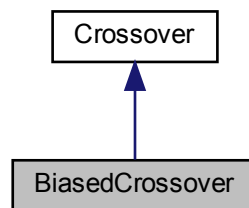
- `lib/hnco/multiobjective/app/algorithm-factory.hh`

5.8 BiasedCrossover Class Reference

Biased crossover.

```
#include <hnco/algorithms/ea/crossover.hh>
```

Inheritance diagram for BiasedCrossover:



Public Member Functions

- [BiasedCrossover](#) ()
Constructor.
- void [recombine](#) (const [bit_vector_t](#) &parent1, const [bit_vector_t](#) &parent2, [bit_vector_t](#) &offspring)
Recombine.
- void [set_bias](#) (double b)
Set bias.

Private Attributes

- `std::bernoulli_distribution` [_bernoulli_dist](#)
Bernoulli distribution.

5.8.1 Detailed Description

Biased crossover.

Definition at line 75 of file crossover.hh.

5.8.2 Member Function Documentation

5.8.2.1 recombine()

```
void recombine (
    const bit\_vector\_t & parent1,
    const bit\_vector\_t & parent2,
    bit\_vector\_t & offspring ) [virtual]
```

Recombine.

Each offspring's bit is copied from second parent with a fixed probability (the crossover bias), from first parent otherwise.

Parameters

<i>parent1</i>	First parent
<i>parent2</i>	Second parent
<i>offspring</i>	Offspring

Implements [Crossover](#).

Definition at line 45 of file crossover.cc.

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

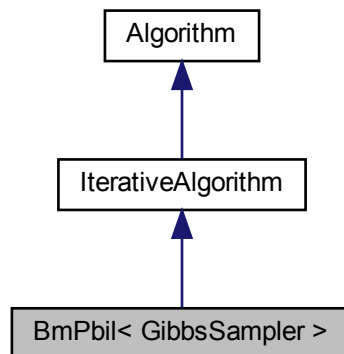
- lib/hnco/algorithms/ea/crossover.hh
- lib/hnco/algorithms/ea/crossover.cc

5.9 BmPbil< GibbsSampler > Class Template Reference

Boltzmann machine PBIL.

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

Inheritance diagram for BmPbil< GibbsSampler >:



Public Types

- enum { [SAMPLING_ASYNCHRONOUS](#) , [SAMPLING_ASYNCHRONOUS_FULL_SCAN](#) , [SAMPLING_SYNCHRONOUS](#) }
- enum { [RESET_NO_RESET](#) , [RESET_ITERATION](#) , [RESET_BIT_VECTOR](#) }

Public Member Functions

- [BmPbil](#) (int n, int population_size)
Constructor.

Setters for parameters

- void [set_selection_size](#) (int x)
Set the selection size.
- void [set_learning_rate](#) (double x)
Set the learning rate.
- void [set_num_gs_steps](#) (int x)
Set the number of gibbs sampler steps.
- void [set_num_gs_cycles](#) (int x)
Set the number of gibbs sampler cycles.
- void [set_negative_positive_selection](#) (bool x)
Set negative and positive selection.
- void [set_sampling](#) (int x)
Set the sampling mode.
- void [set_mc_reset_strategy](#) (int x)
Set the MC reset strategy.

Setters for logging

- void [set_log_norm_infinite](#) (bool x)
Log infinite norm of the model parameters.
- void [set_log_norm_1](#) (bool x)
Log 1-norm of the model parameters.

Protected Member Functions

- void [set_something_to_log](#) ()
Set flag for something to log.
- void [sample](#) (bit_vector_t &x)
Sample a bit vector.
- void [sample_asynchronous](#) ()
Asynchronous sampling.
- void [sample_asynchronous_full_scan](#) ()
Asynchronous sampling with full scan.
- void [sample_synchronous](#) ()
Synchronous sampling.

Loop

- void [init](#) () override
Initialize.
- void [iterate](#) () override
Single iteration.
- void [log](#) () override
Log.

Protected Attributes

- [Population _population](#)
Population.
- GibbsSampler::Moment [_model_parameters](#)
Model parameters.
- GibbsSampler [_gibbs_sampler](#)
Model.
- GibbsSampler::Moment [_walsh_moment_all](#)
Parameters averaged over all individuals.
- GibbsSampler::Moment [_walsh_moment_best](#)
Parameters averaged over selected individuals.
- GibbsSampler::Moment [_walsh_moment_worst](#)
Parameters averaged over negatively selected individuals.
- std::uniform_int_distribution< int > [_choose_bit](#)
Uniform distribution on bit_vector_t components.
- [permutation_t _permutation](#)
Permutation.

Parameters

- int [_selection_size](#) = 1
Selection size (number of selected individuals in the population)
- double [_learning_rate](#) = 1e-3
Learning rate.
- int [_num_gs_steps](#) = 100
Number of gibbs sampler steps.
- int [_num_gs_cycles](#) = 1
Number of gibbs sampler cycles.
- bool [_negative_positive_selection](#) = false

- *Negative and positive selection.*
int `_sampling` = `SAMPLING_ASYNCHRONOUS`
Sampling mode.
- int `_mc_reset_strategy` = `RESET_NO_RESET`
MC reset strategy.

Logging

- bool `_log_norm_infinite` = false
Log infinite norm of the model parameters.
- bool `_log_norm_1` = false
Log 1-norm of the model parameters.

5.9.1 Detailed Description

```
template<class GibbsSampler>
class hnco::algorithm::walsh_moment::BmPbil< GibbsSampler >
```

Boltzmann machine PBIL.

The BM model is slightly different from the one given in the reference below. More precisely, 0/1 variables are mapped to -1/+1 variables as in Walsh analysis.

Reference:

Arnaud Berny. 2002. Boltzmann machine for population-based incremental learning. In ECAI 2002. IOS Press, Lyon.

Definition at line 50 of file bm-pbil.hh.

5.9.2 Member Enumeration Documentation

5.9.2.1 anonymous enum

anonymous enum

Enumerator

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

Definition at line 54 of file bm-pbil.hh.

5.9.2.2 anonymous enum

anonymous enum

Enumerator

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

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

5.9.3 Member Function Documentation

5.9.3.1 set_selection_size()

```
void set_selection_size (
    int x ) [inline]
```

Set the selection size.

The selection size is the number of selected individuals in the population.

Definition at line 309 of file bm-pbil.hh.

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

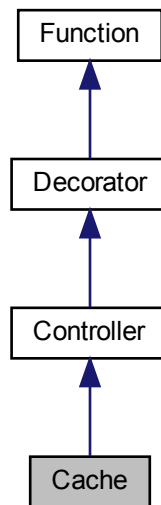
- lib/hnco/algorithms/walsh-moment/bm-pbil.hh

5.10 Cache Class Reference

[Cache.](#)

```
#include <hnco/functions/controllers/controller.hh>
```

Inheritance diagram for Cache:



Public Member Functions

- [Cache](#) ([Function](#) *function)
Constructor.
- bool [provides_incremental_evaluation](#) () const
Check whether the function provides incremental evaluation.
- double [get_lookup_ratio](#) ()
Get lookup ratio.

Evaluation

- double [evaluate](#) (const [bit_vector_t](#) &)
Evaluate a bit vector.

Private Attributes

- std::unordered_map< std::vector< bool >, double > [_cache](#)
Cache.
- std::vector< bool > [_key](#)
Key.
- int [_num_evaluations](#)
Evaluation counter.
- int [_num_lookups](#)
Lookup counter.

Additional Inherited Members

5.10.1 Detailed Description

[Cache](#).

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

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

Definition at line 339 of file `controller.hh`.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 Cache()

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

Constructor.

Parameters

<i>function</i>	Decorated function
-----------------	--------------------

Definition at line 358 of file `controller.hh`.

5.10.3 Member Function Documentation

5.10.3.1 provides_incremental_evaluation()

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

Check whether the function provides incremental evaluation.

Returns

false

Reimplemented from [Controller](#).

Definition at line 367 of file `controller.hh`.

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

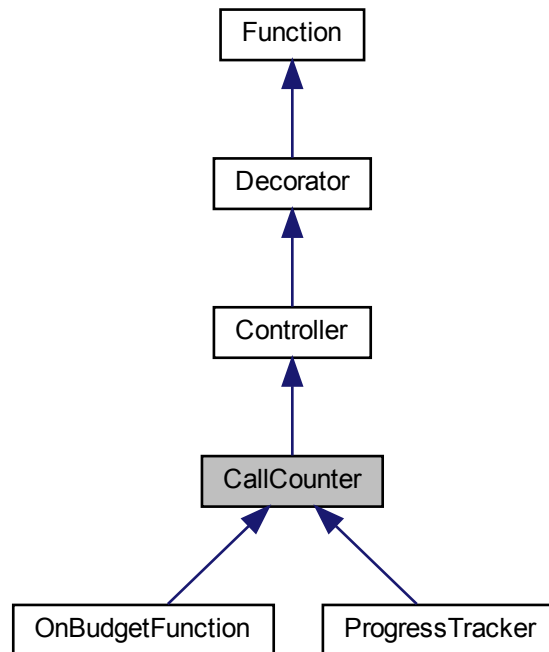
- `lib/hnco/functions/controllers/controller.hh`
- `lib/hnco/functions/controllers/controller.cc`

5.11 CallCounter Class Reference

Call counter.

```
#include <hnco/functions/controllers/controller.hh>
```

Inheritance diagram for CallCounter:



Public Member Functions

- [CallCounter](#) ([Function](#) *function)
Constructor.
- int [get_num_calls](#) ()
Get the number of calls.

Evaluation

- double [evaluate](#) (const [bit_vector_t](#) &)
Evaluate a bit vector.
- double [evaluate_incrementally](#) (const [bit_vector_t](#) &x, double value, const [hnco::sparse_bit_vector_t](#) &flipped_bits)
Incrementally evaluate a bit vector.
- void [update](#) (const [bit_vector_t](#) &x, double value)
Update after a safe evaluation.

Protected Attributes

- `int _num_calls`
Number of calls.

5.11.1 Detailed Description

Call counter.

Definition at line 149 of file controller.hh.

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

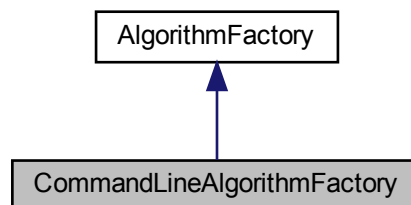
- `lib/hnco/functions/controllers/controller.hh`
- `lib/hnco/functions/controllers/controller.cc`

5.12 CommandLineAlgorithmFactory Class Reference

Command line algorithm factory.

```
#include <hnco/app/algorithm-factory.hh>
```

Inheritance diagram for CommandLineAlgorithmFactory:



Public Member Functions

- `CommandLineAlgorithmFactory` (const `HncoOptions` &options)
Constructor.
- `hnco::algorithm::Algorithm * make` (int bv_size)
Make an algorithm.

Private Attributes

- const `HncoOptions` & `_options`
HNCO options.

5.12.1 Detailed Description

Command line algorithm factory.

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

5.12.2 Member Function Documentation

5.12.2.1 make()

```
Algorithm * make (  
    int bv_size ) [virtual]
```

Make an algorithm.

Parameters

<i>bv_size</i>	Bit vector size
----------------	-----------------

Implements [AlgorithmFactory](#).

Definition at line 81 of file algorithm-factory.cc.

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

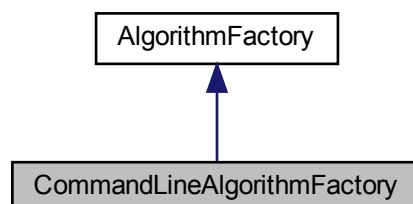
- lib/hnco/app/algorithm-factory.hh
- lib/hnco/app/algorithm-factory.cc

5.13 CommandLineAlgorithmFactory Class Reference

Command line algorithm factory.

```
#include <hnco/multiobjective/app/algorithm-factory.hh>
```

Inheritance diagram for CommandLineAlgorithmFactory:



Public Member Functions

- [CommandLineAlgorithmFactory](#) (const [HncoOptions](#) &options)
Constructor.
- [hnco::multiobjective::algorithm::Algorithm * make](#) (int bv_size, int num_objectives)
Make an algorithm.

Private Attributes

- const [HncoOptions](#) & [_options](#)
HNCO options.

5.13.1 Detailed Description

Command line algorithm factory.

Definition at line 46 of file algorithm-factory.hh.

5.13.2 Member Function Documentation

5.13.2.1 make()

```
Algorithm * make (
    int bv_size,
    int num_objectives ) [virtual]
```

Make an algorithm.

Parameters

<i>bv_size</i>	Bit vector size
----------------	-----------------

Implements [AlgorithmFactory](#).

Definition at line 33 of file algorithm-factory.cc.

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

- lib/hnco/multiobjective/app/algorithm-factory.hh
- lib/hnco/multiobjective/app/algorithm-factory.cc

5.14 CommandLineApplication Class Reference

Command line application.

```
#include <hnco/app/application.hh>
```

Public Member Functions

- [CommandLineApplication](#) (const [HncoOptions](#) &options, [FunctionFactory](#) &function_factory, [AlgorithmFactory](#) &algorithm_factory)
Constructor.
- void [run](#) ()
Run the application.

Private Member Functions

- void [init](#) ()
Initialization.
- void [make_functions](#) ()
Make all functions.
- void [load_solution](#) ()
Load a solution.
- void [print_information](#) ()
Print information about the function.
- void [make_algorithm](#) ()
Make algorithm.
- void [maximize](#) ()
Maximize the function.
- void [print_results](#) (double total_time, bool target_reached)
Print results.
- void [manage_solution](#) (const [bit_vector_t](#) &bv)
Manage solution.

Private Attributes

- const [HncoOptions](#) & [_options](#)
HNCO options.
- [DecoratedFunctionFactory](#) [_decorated_function_factory](#)
Decorated function factory.
- [AlgorithmFactory](#) & [_algorithm_factory](#)
Algorithm factory.
- std::vector< [function::Function](#) * > [_fns](#)
All functions.
- [function::Function](#) * [_fn](#) = nullptr
Main function.
- [hnco::algorithm::Algorithm](#) * [_algorithm](#) = nullptr
Algorithm.
- [logging::ProgressTrackerContext](#) * [_log_context](#) = nullptr
Log context.

5.14.1 Detailed Description

Command line application.

Definition at line 34 of file application.hh.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 CommandLineApplication()

```
CommandLineApplication (
    const HncoOptions & options,
    FunctionFactory & function_factory,
    AlgorithmFactory & algorithm_factory ) [inline]
```

Constructor.

Parameters

<i>options</i>	HNCO options
<i>function_factory</i>	Function factory
<i>algorithm_factory</i>	Algorithm factory

Definition at line 89 of file application.hh.

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

- lib/hnco/app/application.hh
- lib/hnco/app/application.cc

5.15 CommandLineApplication Class Reference

Command line application.

```
#include <hnco/multiojective/app/application.hh>
```

Public Member Functions

- [CommandLineApplication](#) (const [HncoOptions](#) &options, [FunctionFactory](#) &function_factory, [AlgorithmFactory](#) &algorithm_factory)
Constructor.
- void [run](#) ()
Run the application.

Private Member Functions

- void `init` ()
Initialization.
- void `make_functions` ()
Make all functions.
- void `print_information` ()
Print information about the function.
- void `make_algorithm` ()
Make algorithm.
- void `minimize` ()
Minimize objective functions.
- void `manage_solutions` ()
Manage solutions.

Private Attributes

- const `HncoOptions` & `_options`
HNCO options.
- `FunctionFactory` & `_function_factory`
Function factory.
- `AlgorithmFactory` & `_algorithm_factory`
Algorithm factory.
- `std::vector< hnco::multiobjective::function::Function * >` `_fns`
All functions.
- `hnco::multiobjective::function::Function *` `_fn` = nullptr
Main function.
- `hnco::multiobjective::algorithm::Algorithm *` `_algorithm` = nullptr
Algorithm.
- `logging::ProgressTrackerContext *` `_log_context` = nullptr
Log context.

5.15.1 Detailed Description

Command line application.

Definition at line 37 of file `application.hh`.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 CommandLineApplication()

```
CommandLineApplication (
    const HncoOptions & options,
    FunctionFactory & function_factory,
    AlgorithmFactory & algorithm_factory ) [inline]
```

Constructor.

Parameters

<i>options</i>	HNCO options
<i>function_factory</i>	Function factory
<i>algorithm_factory</i>	Algorithm factory

Definition at line 86 of file application.hh.

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

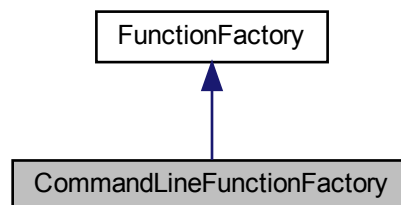
- lib/hnco/multiobjective/app/application.hh
- lib/hnco/multiobjective/app/application.cc

5.16 CommandLineFunctionFactory Class Reference

Command line function factory.

```
#include <hnco/app/function-factory.hh>
```

Inheritance diagram for CommandLineFunctionFactory:



Public Member Functions

- [CommandLineFunctionFactory](#) (const [HncoOptions](#) &options)
Constructor.
- [hnco::function::Function * make](#) ()
Make a function.

Private Attributes

- const [HncoOptions](#) & [_options](#)
HNCO options.

5.16.1 Detailed Description

Command line function factory.

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

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

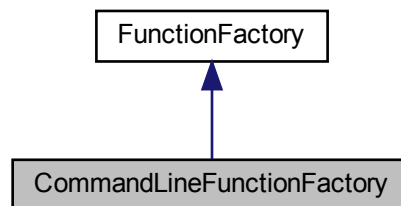
- lib/hnco/app/function-factory.hh
- lib/hnco/app/function-factory.cc

5.17 CommandLineFunctionFactory Class Reference

Command line function factory.

```
#include <hnco/multiobjective/app/function-factory.hh>
```

Inheritance diagram for CommandLineFunctionFactory:



Public Member Functions

- [CommandLineFunctionFactory](#) (const [HncoOptions](#) &options)
Constructor.
- [hnco::multiobjective::function::Function * make](#) ()
Make a function.

Private Attributes

- const [HncoOptions](#) & [_options](#)
HNCO options.

5.17.1 Detailed Description

Command line function factory.

Definition at line 44 of file function-factory.hh.

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

- lib/hnco/multiobjective/app/function-factory.hh
- lib/hnco/multiobjective/app/function-factory.cc

5.18 CommaSelection Class Reference

Comma selection.

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

Public Member Functions

- [CommaSelection](#) ([Population](#) &parents, [Population](#) &offsprings)
Constructor.
- void [select](#) ()
Apply selection.

Private Attributes

- [Population](#) & [_parents](#)
Parent population.
- [Population](#) & [_offsprings](#)
Offspring population.

5.18.1 Detailed Description

Comma selection.

Used as selection for replacement in evolutionary algorithms.

Definition at line 38 of file selection.hh.

5.18.2 Constructor & Destructor Documentation

5.18.2.1 CommaSelection()

```
CommaSelection (  
    Population & parents,  
    Population & offsprings ) [inline]
```

Constructor.

Parameters

<i>parents</i>	Parent population
<i>offsprings</i>	Offspring population

Definition at line 53 of file selection.hh.

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

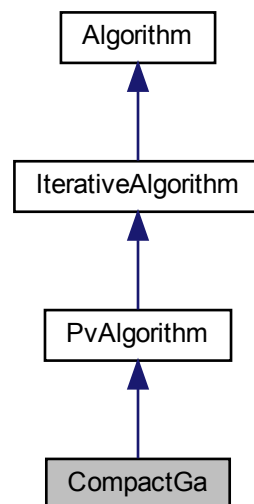
- lib/hnco/algorithms/ea/selection.hh

5.19 CompactGa Class Reference

Compact genetic algorithm.

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

Inheritance diagram for CompactGa:



Public Member Functions

- [CompactGa](#) (int n)
Constructor.

Setters

- void [set_learning_rate](#) (double x)
Set the learning rate.

Protected Member Functions

Loop

- void `init()` override
Initialize.
- void `iterate()` override
Single iteration.

Protected Attributes

- `std::vector< bit_vector_t > _candidates`
Candidates.

Parameters

- double `_learning_rate` = 1e-3
Learning rate.

5.19.1 Detailed Description

Compact genetic algorithm.

Reference:

Georges R. Harik, Fernando G. Lobo, and David E. Goldberg. 1999. The Compact Genetic Algorithm. IEEE Trans. on Evolutionary Computation 3, 4 (November 1999), 287–297.

Definition at line 41 of file compact-ga.hh.

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

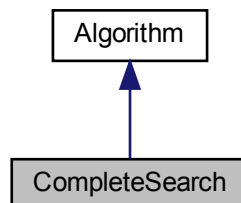
- lib/hnco/algorithms/pv/compact-ga.hh
- lib/hnco/algorithms/pv/compact-ga.cc

5.20 CompleteSearch Class Reference

Complete search.

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

Inheritance diagram for CompleteSearch:



Public Member Functions

- [CompleteSearch](#) (int n)
Constructor.
- void [maximize](#) (const std::vector< [function::Function](#) * > &functions)
Maximize.

Additional Inherited Members

5.20.1 Detailed Description

Complete search.

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

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

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

5.21 ComplexToDouble< T > Struct Template Reference

Convert a complex to a double.

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

Public Types

- using [codomain_type](#) = std::complex< T >
Codomain type.

Public Member Functions

- double [operator\(\)](#) (std::complex< T > z)
Convert to double.

5.21.1 Detailed Description

```
template<class T>
struct hnco::function::ComplexToDouble< T >
```

Convert a complex to a double.

Definition at line 44 of file converter.hh.

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

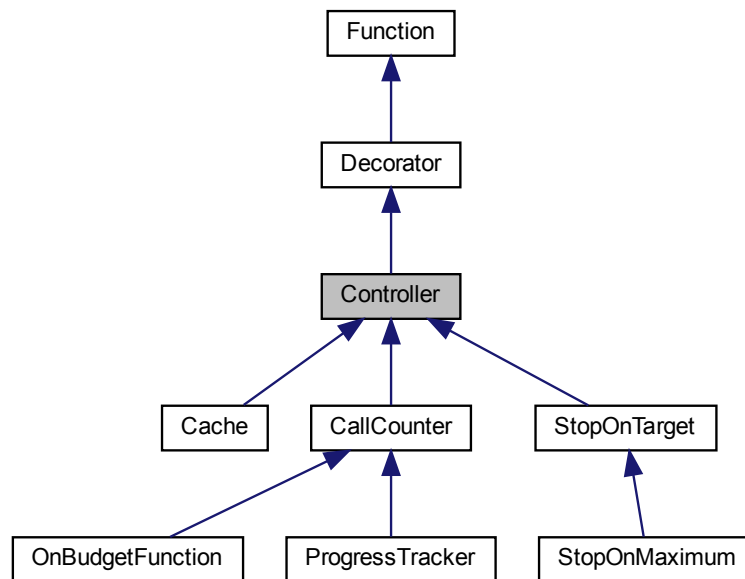
- lib/hnco/functions/converter.hh

5.22 Controller Class Reference

[Function](#) controller.

```
#include <hnco/functions/controllers/controller.hh>
```

Inheritance diagram for Controller:



Public Member Functions

- [Controller](#) ([Function](#) *function)
Constructor.

Information about the function

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

Evaluation

- double [evaluate_safely](#) (const [bit_vector_t](#) &x)
Safely evaluate a bit vector.

Additional Inherited Members

5.22.1 Detailed Description

[Function](#) controller.

Definition at line 42 of file controller.hh.

5.22.2 Member Function Documentation

5.22.2.1 provides_incremental_evaluation()

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

Check whether the function provides incremental evaluation.

Returns

true if the decorated function does

Reimplemented from [Function](#).

Reimplemented in [Cache](#).

Definition at line 66 of file controller.hh.

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

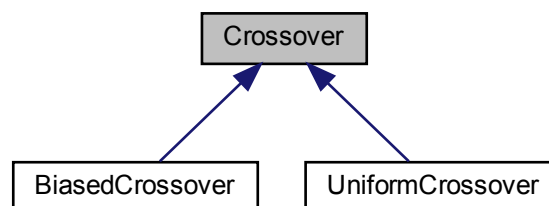
- lib/hnco/functions/controllers/controller.hh

5.23 Crossover Class Reference

Crossover

```
#include <hnco/algorithms/ea/crossover.hh>
```

Inheritance diagram for Crossover:



Public Member Functions

- virtual [~Crossover](#) ()
Destructor.
- virtual void [recombine](#) (const [bit_vector_t](#) &parent1, const [bit_vector_t](#) &parent2, [bit_vector_t](#) &offspring)=0
Recombine.

5.23.1 Detailed Description

Crossover

Definition at line 35 of file crossover.hh.

5.23.2 Member Function Documentation

5.23.2.1 recombine()

```
virtual void recombine (
    const bit\_vector\_t & parent1,
    const bit\_vector\_t & parent2,
    bit\_vector\_t & offspring ) [pure virtual]
```

Recombine.

The offspring is the crossover of two parents.

Parameters

<i>parent1</i>	First parent
<i>parent2</i>	Second parent
<i>offspring</i>	Offspring

Implemented in [BiasedCrossover](#), and [UniformCrossover](#).

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

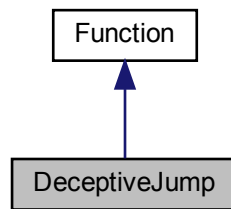
- lib/hnco/algorithms/ea/crossover.hh

5.24 DeceptiveJump Class Reference

Deceptive jump.

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

Inheritance diagram for DeceptiveJump:



Public Member Functions

- [DeceptiveJump](#) (int bv_size, int gap)
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- bool [has_known_maximum](#) () const override
Check for a known maximum.
- double [get_maximum](#) () const override
Get the global maximum.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Private Attributes

- int [_bv_size](#)
Bit vector size.
- int [_gap](#)
Gap.

5.24.1 Detailed Description

Deceptive jump.

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

Reference:

Thomas Jansen, Analyzing Evolutionary Algorithms. Springer, 2013.

Definition at line 85 of file jump.hh.

5.24.2 Member Function Documentation

5.24.2.1 get_maximum()

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

Get the global maximum.

Returns

`_bv_size + _gap`

Reimplemented from [Function](#).

Definition at line 108 of file jump.hh.

5.24.2.2 has_known_maximum()

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

Check for a known maximum.

Returns

`true`

Reimplemented from [Function](#).

Definition at line 104 of file jump.hh.

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

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

5.25 DecoratedFunctionFactory Class Reference

Decorated function factory.

```
#include <hnco/app/decorated-function-factory.hh>
```

Public Member Functions

- [DecoratedFunctionFactory](#) (const [HncoOptions](#) &options, [FunctionFactory](#) &function_factory)
Constructor.
- [hnco::function::Function](#) * [make_function_modifier](#) ()
Make a function modifier.
- [hnco::function::Function](#) * [make_function_controller](#) ([hnco::function::Function](#) *function)
Make a function controller.
- [hnco::map::Map](#) * [get_map](#) ()
Get map.
- [hnco::function::controller::ProgressTracker](#) * [get_tracker](#) ()
Get tracker controller.
- [hnco::function::controller::Cache](#) * [get_cache](#) ()
Get Cache controller.
- [hnco::function::controller::StopOnTarget](#) * [get_stop_on_target](#) ()
Get StopOnTarget controller.

Private Member Functions

- [hnco::function::Function](#) * [make_function](#) ()
Make a function.

Private Attributes

- const [HncoOptions](#) & [_options](#)
HNCO options.
- [FunctionFactory](#) & [_function_factory](#)
Factory function.
- [hnco::map::Map](#) * [_map](#) = nullptr
Map.
- [hnco::function::controller::ProgressTracker](#) * [_tracker](#) = nullptr
Tracker controller.
- [hnco::function::controller::Cache](#) * [_cache](#) = nullptr
Cache controller.
- [hnco::function::controller::StopOnTarget](#) * [_stop_on_target](#) = nullptr
StopOnTarget controller.

5.25.1 Detailed Description

Decorated function factory.

Definition at line 35 of file decorated-function-factory.hh.

5.25.2 Member Function Documentation

5.25.2.1 [make_function_controller\(\)](#)

```
Function * make\_function\_controller (
    hnco::function::Function * function )
```

Make a function controller.

Parameters

<i>function</i>	Decorated function
-----------------	--------------------

Definition at line 254 of file decorated-function-factory.cc.

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

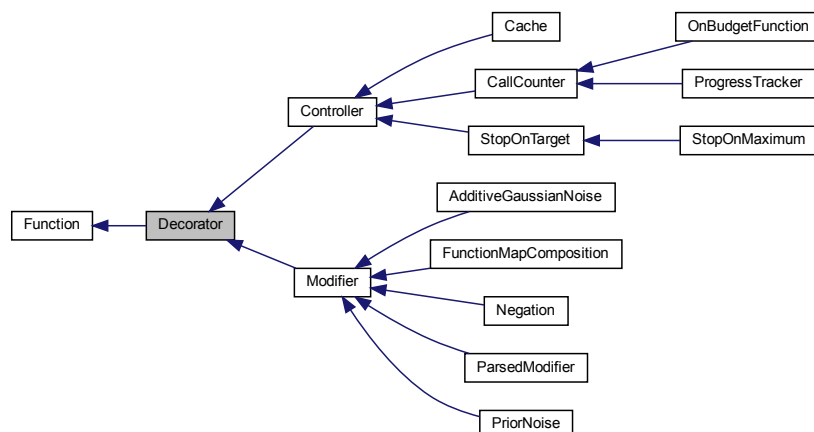
- lib/hnco/app/decorated-function-factory.hh
- lib/hnco/app/decorated-function-factory.cc

5.26 Decorator Class Reference

Function decorator

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

Inheritance diagram for Decorator:



Public Member Functions

- [Decorator](#) ([Function](#) *function)
Constructor.

Display

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

Protected Attributes

- [Function](#) * `_function`
Decorated function.

5.26.1 Detailed Description

Function decorator

Definition at line 34 of file `decorator.hh`.

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

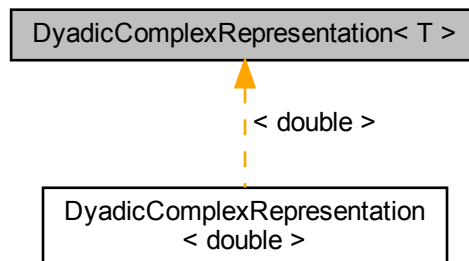
- `lib/hnco/functions/decorator.hh`

5.27 DyadicComplexRepresentation< T > Class Template Reference

Dyadic complex representation.

```
#include <hnco/representations/complex.hh>
```

Inheritance diagram for DyadicComplexRepresentation< T >:



Public Types

- using `domain_type` = `std::complex< T >`
Domain type.

Public Member Functions

- [DyadicComplexRepresentation](#) ([DyadicFloatRepresentation](#)< T > real_part, [DyadicFloatRepresentation](#)< T > imaginary_part)
Constructor.
- [DyadicComplexRepresentation](#) ([DyadicFloatRepresentation](#)< T > rep)
Constructor.
- int [size](#) () const
Size of the representation.
- [domain_type unpack](#) (const [bit_vector_t](#) &bv, int start)
Unpack bit vector into a value.
- void [display](#) (std::ostream &stream) const
Display.

Private Attributes

- [DyadicFloatRepresentation](#)< T > [_real_part](#)
Representation of the real part.
- [DyadicFloatRepresentation](#)< T > [_imaginary_part](#)
Representation of the imaginary part.

5.27.1 Detailed Description

```
template<class T>
class hnco::representation::DyadicComplexRepresentation< T >
```

Dyadic complex representation.

Definition at line 46 of file complex.hh.

5.27.2 Constructor & Destructor Documentation

5.27.2.1 DyadicComplexRepresentation() [1/2]

```
DyadicComplexRepresentation (
    DyadicFloatRepresentation< T > real_part,
    DyadicFloatRepresentation< T > imaginary_part ) [inline]
```

Constructor.

Parameters

<i>real_part</i>	Representation of real part
<i>imaginary_part</i>	Representation of imaginary part

Definition at line 64 of file complex.hh.

5.27.2.2 DyadicComplexRepresentation() [2/2]

```
DyadicComplexRepresentation (
    DyadicFloatRepresentation< T > rep ) [inline]
```

Constructor.

Parameters

<i>rep</i>	Representation of both real and imaginary parts
------------	---

Definition at line 73 of file complex.hh.

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

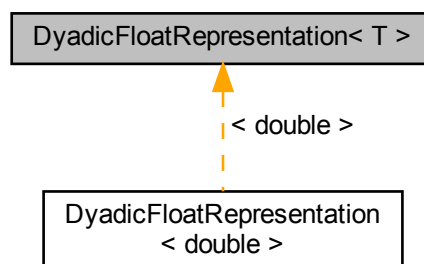
- lib/hnco/representations/complex.hh

5.28 DyadicFloatRepresentation< T > Class Template Reference

Dyadic float representation.

```
#include <hnco/representations/float.hh>
```

Inheritance diagram for DyadicFloatRepresentation< T >:



Public Types

- using `domain_type` = T
Domain type.

Public Member Functions

- [DyadicFloatRepresentation](#) (T lower_bound, T upper_bound, int num_bits)
Constructor.
- [DyadicFloatRepresentation](#) (T lower_bound, T upper_bound, T precision)
Constructor.
- int [size](#) () const
Size of the representation.
- [domain_type unpack](#) (const [bit_vector_t](#) &bv, int start)
Unpack bit vector into a value.
- void [display](#) (std::ostream &stream) const
Display.

Private Member Functions

- T [affine_transformation](#) (T x)
Affine transformation.
- void [compute_lengths](#) (int num_bits)
Compute lengths.

Private Attributes

- std::vector< T > [_lengths](#)
Lengths of dyadic intervals.
- T [_lower_bound](#)
Lower bound of the interval.
- T [_length](#)
Length of the interval.

5.28.1 Detailed Description

```
template<class T>
class hnco::representation::DyadicFloatRepresentation< T >
```

Dyadic float representation.

Definition at line 44 of file float.hh.

5.28.2 Constructor & Destructor Documentation

5.28.2.1 DyadicFloatRepresentation() [1/2]

```
DyadicFloatRepresentation (
    T lower_bound,
    T upper_bound,
    int num_bits ) [inline]
```

Constructor.

The represented interval is [lower_bound, upper_bound).

Parameters

<i>lower_bound</i>	Lower bound of the interval
<i>upper_bound</i>	Upper bound of the interval
<i>num_bits</i>	Number of bits per float number

Definition at line 87 of file float.hh.

5.28.2.2 DyadicFloatRepresentation() [2/2]

```
DyadicFloatRepresentation (
    T lower_bound,
    T upper_bound,
    T precision ) [inline]
```

Constructor.

The represented interval is [lower_bound, upper_bound).

Parameters

<i>lower_bound</i>	Lower bound of the interval
<i>upper_bound</i>	Upper bound of the interval
<i>precision</i>	Precision

Definition at line 105 of file float.hh.

5.28.3 Member Function Documentation

5.28.3.1 compute_lengths()

```
void compute_lengths (
    int num_bits ) [inline], [private]
```

Compute lengths.

Parameters

<i>num_bits</i>	Number of bits per float number
-----------------	---------------------------------

Definition at line 62 of file float.hh.

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

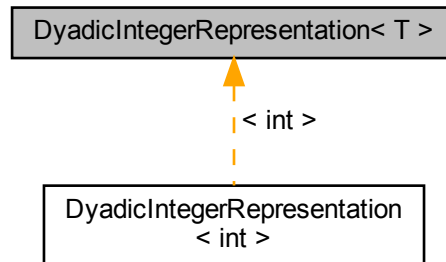
- lib/hnco/representations/float.hh

5.29 DyadicIntegerRepresentation< T > Class Template Reference

Dyadic integer representation.

```
#include <hnco/representations/integer.hh>
```

Inheritance diagram for DyadicIntegerRepresentation< T >:



Public Types

- using `domain_type` = T
Domain type.

Public Member Functions

- `DyadicIntegerRepresentation` (T lower_bound, T upper_bound, int num_bits)
Constructor.
- `DyadicIntegerRepresentation` (T lower_bound, T upper_bound)
Constructor.
- int `size` () const
Size of the representation.
- `domain_type` `unpack` (const `bit_vector_t` &bv, int start)
Unpack bit vector into a value.
- void `display` (std::ostream &stream) const
Display.

Private Member Functions

- void `set_num_bits_complete` (T lower_bound, T upper_bound)
The the number of bits of a complete representation.

Private Attributes

- `int _num_bits`
Number of bits.
- `int _num_bits_complete`
Number of bits for a complete representation.
- `T _lower_bound`
Lower bound of the interval.
- `T _upper_bound`
Upper bound of the interval.

5.29.1 Detailed Description

```
template<class T>
class hnco::representation::DyadicIntegerRepresentation< T >
```

Dyadic integer representation.

Definition at line 78 of file integer.hh.

5.29.2 Constructor & Destructor Documentation

5.29.2.1 DyadicIntegerRepresentation() [1/2]

```
DyadicIntegerRepresentation (
    T lower_bound,
    T upper_bound,
    int num_bits ) [inline]
```

Constructor.

The represented interval is [lower_bound..upper_bound].

Parameters

<i>num_bits</i>	Number of bits per integer
<i>lower_bound</i>	Lower bound of the interval
<i>upper_bound</i>	Upper bound of the interval

Definition at line 115 of file integer.hh.

5.29.2.2 DyadicIntegerRepresentation() [2/2]

```
DyadicIntegerRepresentation (
```



```
T lower_bound,
T upper_bound )    [inline]
```

Constructor.

The represented interval is [lower_bound..upper_bound].

Parameters

<i>lower_bound</i>	Lower bound of the interval
<i>upper_bound</i>	Upper bound of the interval

Definition at line 135 of file integer.hh.

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

- lib/hnco/representations/integer.hh

5.30 PermutationRepresentation::Element Struct Reference

[Element.](#)

```
#include <hnco/representations/permutation.hh>
```

Public Attributes

- int [index](#)
Index.
- int [value](#)
Value.

5.30.1 Detailed Description

[Element.](#)

Definition at line 44 of file permutation.hh.

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

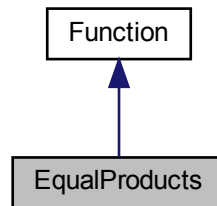
- lib/hnco/representations/permutation.hh

5.31 EqualProducts Class Reference

Equal products.

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

Inheritance diagram for EqualProducts:



Public Member Functions

- [EqualProducts](#) ()
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Instance generators

- template<class Generator >
void [generate](#) (int n, Generator generator)
Instance generator.
- void [random](#) (int n)
Random instance.

Load and save instance

- void [load](#) (std::string path)
Load instance.
- void [save](#) (std::string path) const
Save instance.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
Serialize.

Private Attributes

- `std::vector< double > _numbers`
Numbers.

Friends

- class `boost::serialization::access`

5.31.1 Detailed Description

Equal products.

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

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

Reference:

S. Baluja and S. Davies. 1997. Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space. Technical Report CMU- CS-97-107. Carnegie-Mellon University.

Definition at line 59 of file equal-products.hh.

5.31.2 Member Function Documentation

5.31.2.1 generate()

```
void generate (
    int n,
    Generator generator ) [inline]
```

Instance generator.

Parameters

<i>n</i>	Size of bit vectors
<i>generator</i>	Number generator

Definition at line 91 of file equal-products.hh.

5.31.2.2 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

<i>path</i>	Path of the instance to load
-------------	------------------------------

Exceptions

<i>std::runtime_error</i>	
---------------------------	--

Definition at line 124 of file equal-products.hh.

5.31.2.3 random()

```
void random (
    int n ) [inline]
```

Random instance.

The weights are sampled from the uniform distribution on [0,1).

Parameters

<i>n</i>	Size of bit vector
----------	--------------------

Definition at line 106 of file equal-products.hh.

5.31.2.4 save()

```
void save (
    std::string path ) const [inline]
```

Save instance.

Parameters

<i>path</i>	Path of the instance to save
-------------	------------------------------

Exceptions

<code>std::runtime_error</code>

Definition at line 131 of file equal-products.hh.

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

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

5.32 ProgressTracker::Event Struct Reference

Event

```
#include <hnco/functions/controllers/controller.hh>
```

Public Attributes

- int `num_evaluations`
Number of evaluations.
- double `value`
Value.

5.32.1 Detailed Description

Event

Definition at line 231 of file controller.hh.

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

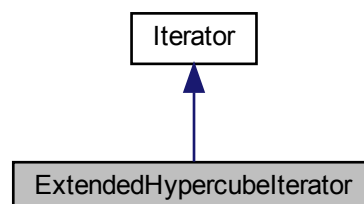
- lib/hnco/functions/controllers/controller.hh

5.33 ExtendedHypercubeliterator Class Reference

Extended Hypercube iterator.

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

Inheritance diagram for ExtendedHypercubeliterator:



Public Member Functions

- [ExtendedHypercubeIterator](#) (int n)
Constructor.
- bool [has_next](#) () override
Has next bit vector.
- const [bit_vector_t](#) & [next](#) () override
Next bit vector.

Additional Inherited Members

5.33.1 Detailed Description

Extended Hypercube iterator.

Similar to Hypercube. In dimension 0, an [HypercubeIterator](#) does not contain any element. However, in dimension 0, an [ExtendedHypercubeIterator](#) contains a unique element which is the vector of size 0. An [ExtendedHypercubeIterator](#) is helpful when the enumerated vectors are seen as prefixes or suffixes hence can be empty. This is used, in particular, in `compute_fast_walsh_transform`.

Definition at line 97 of file `iterator.hh`.

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

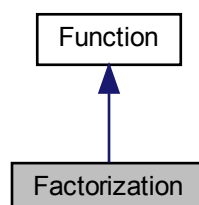
- `lib/hnco/iterator.hh`
- `lib/hnco/iterator.cc`

5.34 Factorization Class Reference

Factorization.

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

Inheritance diagram for Factorization:



Public Member Functions

- [Factorization](#) ()
Constructor.
- [Factorization](#) (const std::string number)
Constructor.
- [~Factorization](#) ()
Destructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- void [display](#) (std::ostream &stream) const override
Display.
- void [describe](#) (const [bit_vector_t](#) &x, std::ostream &stream) override
Describe a bit vector.

Load and save instance

- void [load](#) (std::string path)
Load instance.

Private Member Functions

- void [init](#) ()
Init GMP data structures.
- void [clear](#) ()
Clear GMP data structures.
- void [set_number](#) (const std::string number)
Set number.
- void [convert](#) (const [bit_vector_t](#) &x)
Convert a bit vector into two numbers.

Private Attributes

- [mpz_t _number](#)
Number to factorize.
- [mpz_t _first_factor](#)
First factor.
- [mpz_t _second_factor](#)
Second factor.
- [mpz_t _product](#)
Product.
- std::string [_first_factor_string](#)
First factor in binary form.
- std::string [_second_factor_string](#)
Secon factor in binary form.
- size_t [_number_size](#)
Number size in bits.
- size_t [_first_factor_size](#)
First factor size in bits.
- size_t [_second_factor_size](#)
Second factor size in bits.
- int [_bv_size](#)
Bit vector size.

5.34.1 Detailed Description

Factorization.

Reference:

Torbjörn Granlund and the GMP development team. 2012. GNU MP: The GNU Multiple Precision Arithmetic Library (5.0.5 ed.).

<http://gmplib.org/>.

Definition at line 29 of file factorization.hh.

5.34.2 Constructor & Destructor Documentation

5.34.2.1 Factorization()

```
Factorization (
    const std::string number ) [inline]
```

Constructor.

Parameters

<i>number</i>	Number to factorize written in decimal form
---------------	---

Definition at line 82 of file factorization.hh.

5.34.3 Member Function Documentation

5.34.3.1 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

The file referenced by the path is a text file which contains exactly one natural number written in base 10 without any space

Parameters

<i>path</i>	Path of the instance to load
-------------	------------------------------

Exceptions

<code>std::runtime_error</code>	
---------------------------------	--

Definition at line 102 of file factorization.hh.

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

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

5.35 FfgenOptions Class Reference

Command line options for ffgen.

```
#include <ffgen-options.hh>
```

Public Member Functions

- [FfgenOptions](#) (int argc, char *argv[])
Constructor.
- int [get_bv_size](#) () const
Get bv_size.
- void [set_bv_size](#) (int x)
Set bv_size.
- bool [set_bv_size](#) () const
Get set-flag for bv_size.
- double [get_coupling_constant](#) () const
Get coupling_constant.
- void [set_coupling_constant](#) (double x)
Set coupling_constant.
- bool [set_coupling_constant](#) () const
Get set-flag for coupling_constant.
- double [get_ep_upper_bound](#) () const
Get ep_upper_bound.
- void [set_ep_upper_bound](#) (double x)
Set ep_upper_bound.
- bool [set_ep_upper_bound](#) () const
Get set-flag for ep_upper_bound.
- double [get_field_constant](#) () const
Get field_constant.
- void [set_field_constant](#) (double x)
Set field_constant.
- bool [set_field_constant](#) () const
Get set-flag for field_constant.
- int [get_function](#) () const
Get function.
- void [set_function](#) (int x)

- int [get_nn2_generator](#) () const
Get nn2_generator.
- void [set_nn2_generator](#) (int x)
Set nn2_generator.
- bool [set_nn2_generator](#) () const
Get set-flag for nn2_generator.
- int [get_nn2_num_columns](#) () const
Get nn2_num_columns.
- void [set_nn2_num_columns](#) (int x)
Set nn2_num_columns.
- bool [set_nn2_num_columns](#) () const
Get set-flag for nn2_num_columns.
- int [get_nn2_num_rows](#) () const
Get nn2_num_rows.
- void [set_nn2_num_rows](#) (int x)
Set nn2_num_rows.
- bool [set_nn2_num_rows](#) () const
Get set-flag for nn2_num_rows.
- int [get_part_upper_bound](#) () const
Get part_upper_bound.
- void [set_part_upper_bound](#) (int x)
Set part_upper_bound.
- bool [set_part_upper_bound](#) () const
Get set-flag for part_upper_bound.
- std::string [get_path](#) () const
Get path.
- void [set_path](#) (std::string x)
Set path.
- bool [set_path](#) () const
Get set-flag for path.
- int [get_seed](#) () const
Get seed.
- void [set_seed](#) (int x)
Set seed.
- bool [set_seed](#) () const
Get set-flag for seed.
- double [get_stddev](#) () const
Get stddev.
- void [set_stddev](#) (double x)
Set stddev.
- bool [set_stddev](#) () const
Get set-flag for stddev.
- int [get_sudoku_num_empty_cells](#) () const
Get sudoku_num_empty_cells.
- void [set_sudoku_num_empty_cells](#) (int x)
Set sudoku_num_empty_cells.
- bool [set_sudoku_num_empty_cells](#) () const
Get set-flag for sudoku_num_empty_cells.
- int [get_walsh2_generator](#) () const
Get walsh2_generator.
- void [set_walsh2_generator](#) (int x)

- *Set walsh2_generator.*
- bool [set_walsh2_generator](#) () const
Get set-flag for walsh2_generator.
- double [get_walsh2_ising_alpha](#) () const
Get walsh2_ising_alpha.
- void [set_walsh2_ising_alpha](#) (double x)
Set walsh2_ising_alpha.
- bool [set_walsh2_ising_alpha](#) () const
Get set-flag for walsh2_ising_alpha.
- int [get_walsh_num_features](#) () const
Get walsh_num_features.
- void [set_walsh_num_features](#) (int x)
Set walsh_num_features.
- bool [set_walsh_num_features](#) () const
Get set-flag for walsh_num_features.
- bool [with_ms_planted_solution](#) () const
Get ms_planted_solution.
- void [set_ms_planted_solution](#) ()
Set ms_planted_solution.
- bool [with_periodic_boundary_conditions](#) () const
Get periodic_boundary_conditions.
- void [set_periodic_boundary_conditions](#) ()
Set periodic_boundary_conditions.

Private Member Functions

- void [print_help](#) (std::ostream &stream) const
Print help message.
- void [print_version](#) (std::ostream &stream) const
Print version.

Private Attributes

- std::string [_exec_name](#)
Name of the executable.
- std::string [_version](#)
Name Version.
- int [_bv_size](#)
Size of bit vectors.
- bool [_opt_bv_size](#)
- double [_coupling_constant](#)
Coupling constant.
- bool [_opt_coupling_constant](#)
- double [_ep_upper_bound](#)
Upper bound of numbers.
- bool [_opt_ep_upper_bound](#)
- double [_field_constant](#)
Field constant.
- bool [_opt_field_constant](#)
- int [_function](#)

Type of function.

- bool **_opt_function**
- double [_lin_distance](#)

Common distance of arithmetic progression.

- bool **_opt_lin_distance**
- int [_lin_generator](#)

Type of LinearFunction generator.

- bool **_opt_lin_generator**
- double [_lin_initial_weight](#)

Initial weight.

- bool **_opt_lin_initial_weight**
- double [_lin_ratio](#)

Common ratio of geometric progression.

- bool **_opt_lin_ratio**
- int [_ms_num_clauses](#)

Number of clauses.

- bool **_opt_ms_num_clauses**
- int [_ms_num_literals_per_clause](#)

Number of literals per clause.

- bool **_opt_ms_num_literals_per_clause**
- int [_nk_k](#)

Each bit is connected to k other bits.

- bool **_opt_nk_k**
- int [_nn1_generator](#)

Type of NearestNeighborIsingModel1 generator.

- bool **_opt_nn1_generator**
- int [_nn2_generator](#)

Type of NearestNeighborIsingModel2 generator.

- bool **_opt_nn2_generator**
- int [_nn2_num_columns](#)

Number of columns.

- bool **_opt_nn2_num_columns**
- int [_nn2_num_rows](#)

Number of rows.

- bool **_opt_nn2_num_rows**
- int [_part_upper_bound](#)

Upper bound of numbers.

- bool **_opt_part_upper_bound**
- std::string [_path](#)

Path (relative or absolute) of a function file.

- bool **_opt_path**
- int [_seed](#)

Seed for the random number generator.

- bool **_opt_seed**
- double [_stddev](#)

Standard deviation.

- bool **_opt_stddev**
- int [_sudoku_num_empty_cells](#)

Number of empty cells.

- bool **_opt_sudoku_num_empty_cells**
- int [_walsh2_generator](#)

Type of WalshExpansion2 generator.

- bool **_opt_walsh2_generator**
- double [_walsh2_ising_alpha](#)
Dyson-Ising: exponential decay parameter for long range interactions.
- bool **_opt_walsh2_ising_alpha**
- int [_walsh_num_features](#)
Number of features.
- bool **_opt_walsh_num_features**
- bool [_ms_planted_solution](#)
Generate an instance with a planted solution.
- bool [_periodic_boundary_conditions](#)
Periodic boundary conditions.

Friends

- std::ostream & [operator<<](#) (std::ostream &, const [FfgenOptions](#) &)
Print a header containing the parameter values.

5.35.1 Detailed Description

Command line options for ffgen.

Definition at line 11 of file ffgen-options.hh.

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

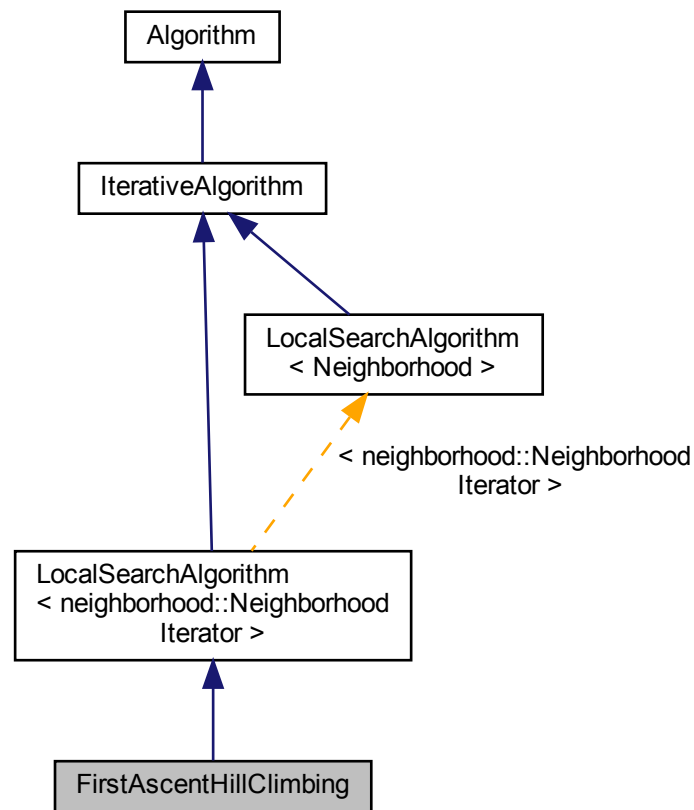
- app/ffgen-options.hh
- app/ffgen-options.cc

5.36 FirstAscentHillClimbing Class Reference

First ascent hill climbing.

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

Inheritance diagram for FirstAscentHillClimbing:



Public Member Functions

- [FirstAscentHillClimbing](#) (int n, [neighborhood::NeighborhoodIterator](#) *neighborhood)
Constructor.

Protected Member Functions

- void [iterate](#) () override
Single iteration.

Additional Inherited Members

5.36.1 Detailed Description

First ascent hill climbing.

Definition at line 34 of file first-ascent-hill-climbing.hh.

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

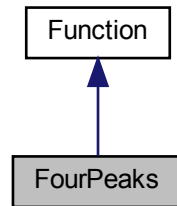
- lib/hnco/algorithms/ls/first-ascent-hill-climbing.hh
- lib/hnco/algorithms/ls/first-ascent-hill-climbing.cc

5.37 FourPeaks Class Reference

Four Peaks.

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

Inheritance diagram for FourPeaks:



Public Member Functions

- [FourPeaks](#) (int bv_size, int threshold)
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- bool [has_known_maximum](#) () const override
Check for a known maximum.
- double [get_maximum](#) () const override
Get the global maximum.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Private Attributes

- int [_bv_size](#)
Bit vector size.
- int [_threshold](#)
Threshold.
- int [_maximum](#)
Maximum.

5.37.1 Detailed Description

Four Peaks.

It is defined by

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

where:

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

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

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

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

Reference:

S. Baluja and R. Caruana. 1995. Removing the genetics from the standard genetic algorithm. In Proceedings of the 12th Annual Conference on Machine Learning. 38–46.

Definition at line 60 of file four-peaks.hh.

5.37.2 Member Function Documentation

5.37.2.1 get_maximum()

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

Get the global maximum.

Returns

$2 * \text{_bv_size} - \text{_threshold} - 1$

Reimplemented from [Function](#).

Definition at line 88 of file four-peaks.hh.

5.37.2.2 has_known_maximum()

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

Check for a known maximum.

Returns

true

Reimplemented from [Function](#).

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

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

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

5.38 FrontDistancePair Struct Reference

Front-distance pair.

```
#include <hnco/multiobjective/algorithms/nsga2.hh>
```

Public Attributes

- int [pareto_front](#)
Pareto front.
- double [crowding_distance](#)
Crowding distance.

5.38.1 Detailed Description

Front-distance pair.

A front-distance pair measures the quality of an individual within a population.

Definition at line 44 of file nsga2.hh.

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

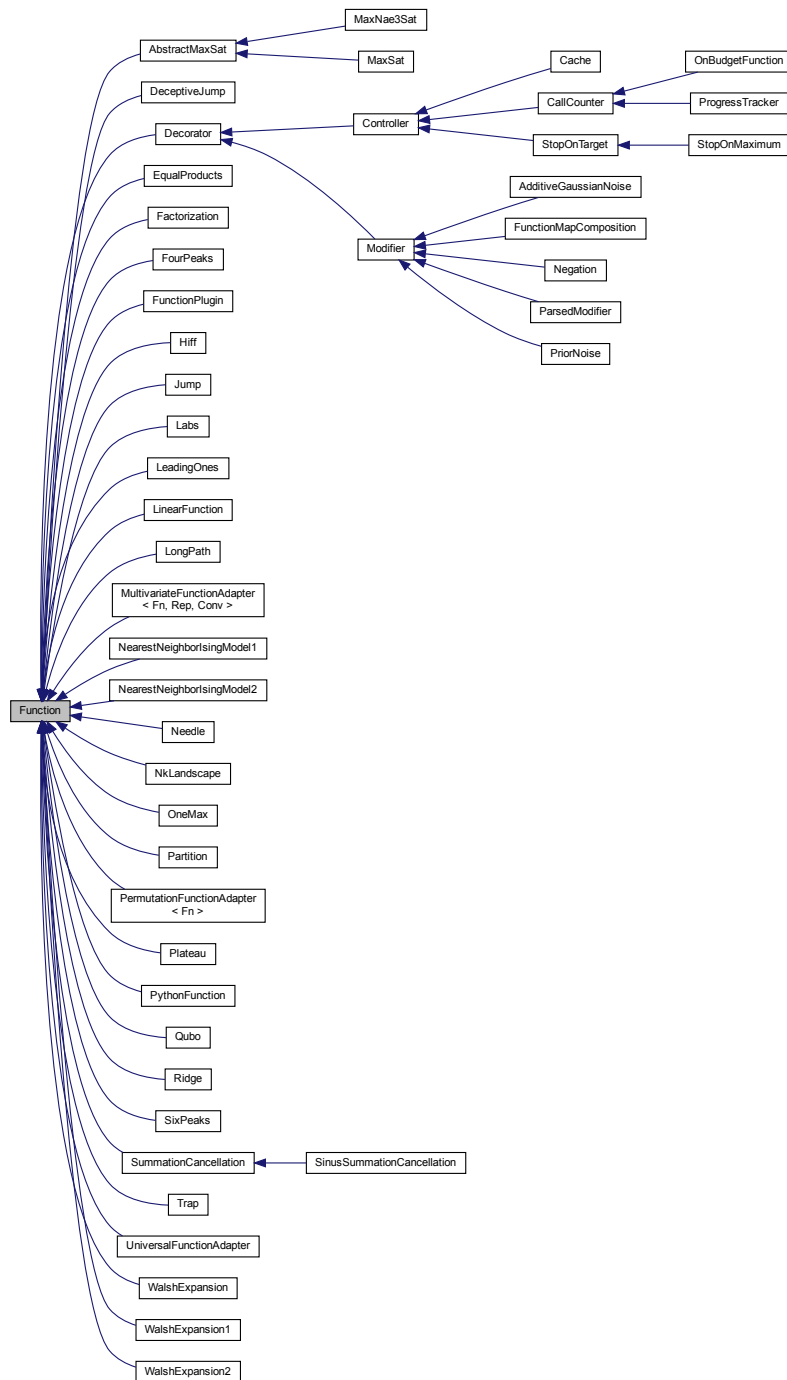
- lib/hnco/multiobjective/algorithms/nsga2.hh

5.39 Function Class Reference

Function

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

Inheritance diagram for Function:



Public Member Functions

- virtual [~Function](#) ()
Destructor.

Information about the function

- virtual int [get_bv_size](#) () const =0
Get bit vector size.
- virtual double [get_maximum](#) () const
Get the global maximum.
- virtual bool [has_known_maximum](#) () const
Check for a known maximum.
- virtual bool [provides_incremental_evaluation](#) () const
Check whether the function provides incremental evaluation.

Evaluation

- virtual double [evaluate](#) (const [bit_vector_t](#) &)=0
Evaluate a bit vector.
- virtual double [evaluate_incrementally](#) (const [bit_vector_t](#) &x, double value, const [sparse_bit_vector_t](#) &flipped_bits)
Incrementally evaluate a bit vector.
- virtual double [evaluate_safely](#) (const [bit_vector_t](#) &x)
Safely evaluate a bit vector.
- virtual void [update](#) (const [bit_vector_t](#) &x, double value)
Update states after a safe evaluation.

Display

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

5.39.1 Detailed Description

Function

Definition at line 45 of file function.hh.

5.39.2 Member Function Documentation

5.39.2.1 describe()

```
virtual void describe (
    const bit\_vector\_t & x,
    std::ostream & stream ) [inline], [virtual]
```

Describe a bit vector.

The member function [Function::describe](#) is not declared const for the same reason [Function::evaluate](#) is not: it might need to decode the given bit vector hence use some pre-allocated memory buffer.

Reimplemented in [FunctionMapComposition](#), [Decorator](#), [Partition](#), [Factorization](#), [UniversalFunctionAdapter](#), [PermutationFunctionAdapter< Fn >](#), and [MultivariateFunctionAdapter< Fn, Rep, Conv >](#).

Definition at line 134 of file `function.hh`.

5.39.2.2 evaluate()

```
virtual double evaluate (
    const bit\_vector\_t & ) [pure virtual]
```

Evaluate a bit vector.

This member function is not declared const and is not supposed to be thread-safe. In particular, in order to evaluate a bit vector, it might require some data member to store temporary results. In case of parallel evaluation, there should be a copy of the function per thread, as is done in `Population::evaluate_in_parallel`.

Implemented in [SinusSummationCancellation](#), [SummationCancellation](#), [UniversalFunctionAdapter](#), [PermutationFunctionAdapter< Fn >](#), [MultivariateFunctionAdapter< Fn, Rep, Conv >](#), [PriorNoise](#), [ParsedModifier](#), [AdditiveGaussianNoise](#), [FunctionMapComposition](#), [Negation](#), [WalshExpansion](#), [WalshExpansion2](#), [WalshExpansion1](#), [Plateau](#), [Ridge](#), [Hiff](#), [Needle](#), [LeadingOnes](#), [OneMax](#), [Qubo](#), [Partition](#), [NkLandscape](#), [MaxNae3Sat](#), [MaxSat](#), [LinearFunction](#), [Labs](#), [DeceptiveJump](#), [Jump](#), [NearestNeighborsingModel2](#), [NearestNeighborsingModel1](#), [SixPeaks](#), [FourPeaks](#), [Factorization](#), [EqualProducts](#), [Cache](#), [ProgressTracker](#), [OnBudgetFunction](#), [CallCounter](#), [StopOnTarget](#), [Trap](#), [PythonFunction](#), [FunctionPlugin](#), and [LongPath](#).

5.39.2.3 evaluate_incrementally()

```
virtual double evaluate_incrementally (
    const bit\_vector\_t & x,
    double value,
    const sparse\_bit\_vector\_t & flipped_bits ) [inline], [virtual]
```

Incrementally evaluate a bit vector.

Exceptions

<code>std::runtime_error</code>	
---------------------------------	--

Reimplemented in [Negation](#), [ProgressTracker](#), [OnBudgetFunction](#), [CallCounter](#), [StopOnTarget](#), [NearestNeighborIsingModel2](#), [NearestNeighborIsingModel1](#), [WalshExpansion1](#), [OneMax](#), and [LinearFunction](#).

Definition at line 95 of file function.hh.

5.39.2.4 `evaluate_safely()`

```
virtual double evaluate_safely (
    const bit\_vector\_t & x ) [inline], [virtual]
```

Safely evaluate a bit vector.

Must neither throw any exception nor update global states (e.g. maximum) in function controllers. It is used in `Population::evaluate_in_parallel` inside a OMP parallel for loop.

By default, calls `evaluate`.

Reimplemented in [Controller](#).

Definition at line 109 of file function.hh.

5.39.2.5 `get_maximum()`

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

Get the global maximum.

Exceptions

<code>std::runtime_error</code>	
---------------------------------	--

Reimplemented in [PriorNoise](#), [FunctionMapComposition](#), [WalshExpansion1](#), [Plateau](#), [Ridge](#), [Hiff](#), [Needle](#), [LeadingOnes](#), [OneMax](#), [LinearFunction](#), [DeceptiveJump](#), [Jump](#), [SixPeaks](#), [FourPeaks](#), [SummationCancellation](#), [Controller](#), [Trap](#), and [LongPath](#).

Definition at line 61 of file function.hh.

5.39.2.6 `provides_incremental_evaluation()`

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

Check whether the function provides incremental evaluation.

Returns

false

Reimplemented in [PriorNoise](#), [Negation](#), [WalshExpansion1](#), [OneMax](#), [LinearFunction](#), [NearestNeighborIsingModel2](#), [NearestNeighborIsingModel1](#), [Cache](#), and [Controller](#).

Definition at line 71 of file function.hh.

5.39.2.7 update()

```
virtual void update (
    const bit\_vector\_t & x,
    double value ) [inline], [virtual]
```

Update states after a safe evaluation.

By default, does nothing.

Reimplemented in [ProgressTracker](#), [OnBudgetFunction](#), [CallCounter](#), and [StopOnTarget](#).

Definition at line 115 of file function.hh.

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

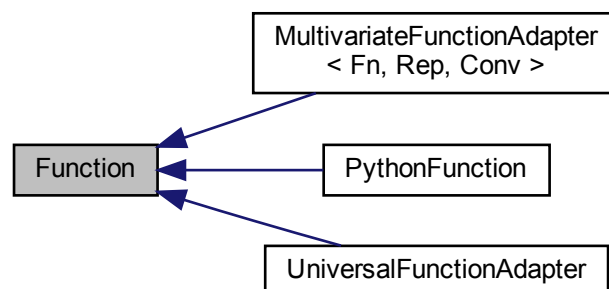
- [lib/hnco/functions/function.hh](#)

5.40 Function Class Reference

Function

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

Inheritance diagram for Function:



Public Member Functions

- virtual [~Function](#) ()
Destructor.

Information about the function

- virtual int [get_bv_size](#) () const =0
Get bit vector size.
- virtual int [get_output_size](#) () const =0
Get output size (number of objectives)

Evaluation

- virtual void [evaluate](#) (const [bit_vector_t](#) &bv, [value_t](#) &value)=0
Evaluate a bit vector.

Display

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

5.40.1 Detailed Description

Function

Definition at line 41 of file function.hh.

5.40.2 Member Function Documentation

5.40.2.1 describe()

```
virtual void describe (
    const bit\_vector\_t & x,
    std::ostream & stream ) [inline], [virtual]
```

Describe a bit vector.

The member function [describe\(\)](#) is not declared const for the same reason [evaluate\(\)](#) is not: it might need to decode the given bit vector hence use some pre-allocated memory buffer.

Reimplemented in [UniversalFunctionAdapter](#), and [MultivariateFunctionAdapter< Fn, Rep, Conv >](#).

Definition at line 95 of file function.hh.

5.40.2.2 evaluate()

```
virtual void evaluate (
    const bit\_vector\_t & bv,
    value\_t & value ) [pure virtual]
```

Evaluate a bit vector.

This member function is not declared const and is not supposed to be thread-safe. In particular, in order to evaluate a bit vector, it might require some data member to store temporary results. In case of parallel evaluation, there should be a copy of the function per thread, as is done in `Population::evaluate_in_parallel()`.

Parameters

<i>bv</i>	Bit vector to evaluate
<i>value</i>	Output value

Implemented in [UniversalFunctionAdapter](#), [MultivariateFunctionAdapter< Fn, Rep, Conv >](#), and [PythonFunction](#).

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

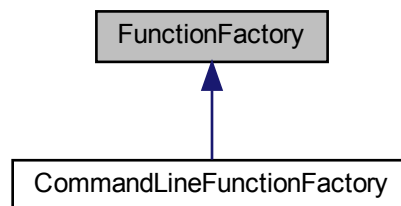
- `lib/hnco/multiobjective/functions/function.hh`

5.41 FunctionFactory Class Reference

Function factory.

```
#include <hnco/app/function-factory.hh>
```

Inheritance diagram for FunctionFactory:



Public Member Functions

- virtual `hnco::function::Function * make ()=0`
Make a function.

5.41.1 Detailed Description

Function factory.

Definition at line 33 of file `function-factory.hh`.

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

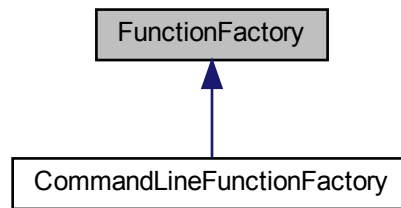
- `lib/hnco/app/function-factory.hh`

5.42 FunctionFactory Class Reference

Function factory.

```
#include <hnco/multiobjective/app/function-factory.hh>
```

Inheritance diagram for FunctionFactory:



Public Member Functions

- virtual `hnco::multiobjective::function::Function * make ()=0`
Make a function.

5.42.1 Detailed Description

Function factory.

Definition at line 36 of file function-factory.hh.

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

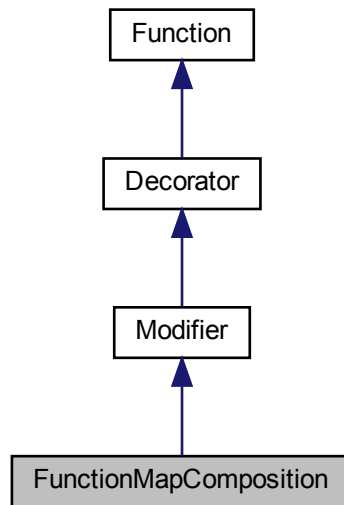
- lib/hnco/multiobjective/app/function-factory.hh

5.43 FunctionMapComposition Class Reference

Composition of a function and a map.

```
#include <hnco/functions/modifiers/modifier.hh>
```

Inheritance diagram for FunctionMapComposition:



Public Member Functions

- [FunctionMapComposition](#) ([Function](#) *function, [hnco::map::Map](#) *map)
Constructor.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Information about the function

- int [get_bv_size](#) () const override
Get bit vector size.
- double [get_maximum](#) () const override
Get the global maximum.
- bool [has_known_maximum](#) () const override
Check for a known maximum.

Display

- void [describe](#) (const [bit_vector_t](#) &x, [std::ostream](#) &stream) override
Describe a bit vector.

Private Attributes

- [hnco::map::Map](#) * [_map](#)
Map.
- [bit_vector_t](#) [_bv](#)
Image of bit vectors under the map.

Additional Inherited Members

5.43.1 Detailed Description

Composition of a function and a map.

Definition at line 100 of file modifier.hh.

5.43.2 Constructor & Destructor Documentation

5.43.2.1 FunctionMapComposition()

```
FunctionMapComposition (
    Function * function,
    hnco::map::Map * map ) [inline]
```

Constructor.

Precondition

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

Exceptions

<code>std::runtime_error</code>	
---------------------------------	--

Definition at line 115 of file modifier.hh.

5.43.3 Member Function Documentation

5.43.3.1 get_maximum()

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

Get the global maximum.

Exceptions

<code>std::runtime_error</code>	
---------------------------------	--

Reimplemented from [Function](#).

Definition at line 135 of file modifier.hh.

5.43.3.2 has_known_maximum()

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

Check for a known maximum.

Returns

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

Reimplemented from [Function](#).

Definition at line 145 of file modifier.hh.

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

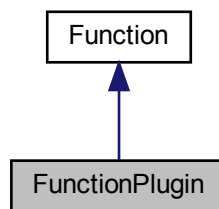
- lib/hnco/functions/modifiers/modifier.hh
- lib/hnco/functions/modifiers/modifier.cc

5.44 FunctionPlugin Class Reference

Function plugin

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

Inheritance diagram for FunctionPlugin:



Public Member Functions

- [FunctionPlugin](#) (int bv_size, std::string path, std::string name)
Constructor.
- [~FunctionPlugin](#) ()
Destructor.
- int [get_bv_size](#) () const
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &)
Evaluate a bit vector.

Private Types

- using `extern_function_t` = `double (*)(const bit_t *, size_t)`
Type of an extern function.

Private Attributes

- int `_bv_size`
Bit vector size.
- void * `_handle`
Handle returned by `dlopen`.
- `extern_function_t` `_extern_function`
Extern function.

5.44.1 Detailed Description

Function plugin

Definition at line 34 of file `plugin.hh`.

5.44.2 Constructor & Destructor Documentation

5.44.2.1 FunctionPlugin()

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

Constructor.

Parameters

<i>bv_size</i>	Size of bit vectors
<i>path</i>	Path to a shared library
<i>name</i>	Name of a function of the shared library

Definition at line 35 of file `plugin.cc`.

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

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

5.45 Generator Struct Reference

Random number generator.

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

Static Public Member Functions

- static void `set_seed` (unsigned n)
Set seed.
- static void `set_seed` ()
Set seed.
- static void `reset` ()
Reset engine.
- static double `uniform` ()
Sample random number with uniform distribution.
- static double `normal` ()
Sample random number with normal distribution.
- static bool `bernoulli` ()
Sample random number with Bernoulli distribution.

Static Public Attributes

- static std::mt19937 `engine`
Mersenne Twister engine.
- static unsigned `seed` = std::mt19937::default_seed
Seed.

5.45.1 Detailed Description

Random number generator.

Definition at line 34 of file random.hh.

5.45.2 Member Function Documentation

5.45.2.1 `reset()`

```
void reset ( ) [static]
```

Reset engine.

Using static member seed.

Definition at line 45 of file random.cc.

5.45.2.2 set_seed()

```
void set_seed ( ) [static]
```

Set seed.

Uses `std::chrono::system_clock`.

Definition at line 39 of file `random.cc`.

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

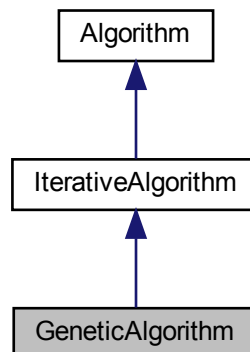
- `lib/hnco/random.hh`
- `lib/hnco/random.cc`

5.46 GeneticAlgorithm Class Reference

Genetic algorithm.

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

Inheritance diagram for GeneticAlgorithm:



Public Member Functions

- [GeneticAlgorithm](#) (int n, int mu)
Constructor.

Setters

- void [set_mutation_rate](#) (double p)
Set the mutation rate.
- void [set_crossover_probability](#) (double p)
Set the crossover probability.
- void [set_tournament_size](#) (int n)
Set the tournament size.
- void [set_allow_no_mutation](#) (bool b)
Set the flag `_allow_no_mutation`.

Protected Member Functions

Loop

- void `init` () override
Initialize.
- void `iterate` () override
Single iteration.

Protected Attributes

- `Population _parents`
Parents.
- `Population _offsprings`
Offsprings.
- `CommaSelection _comma_selection`
Comma selection.
- `TournamentSelection _tournament_selection`
Tournament selection.
- `neighborhood::StandardBitMutation _mutation`
Mutation operator.
- `std::bernoulli_distribution _do_crossover`
Do crossover.
- `UniformCrossover _crossover`
Uniform crossover.

Parameters

- double `_mutation_rate`
Mutation rate.
- double `_crossover_probability` = 0.5
Crossover probability.
- int `_tournament_size` = 10
Tournament size.
- bool `_allow_no_mutation` = false
Allow no mutation.

5.46.1 Detailed Description

Genetic algorithm.

- Tournament selection for reproduction
- Uniform crossover
- Standard bit mutation
- (mu, mu) selection (offspring population replaces parent population)

Reference:

J. H. Holland. 1975. Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor.

Definition at line 53 of file genetic-algorithm.hh.

5.46.2 Constructor & Destructor Documentation

5.46.2.1 GeneticAlgorithm()

```
GeneticAlgorithm (
    int n,
    int mu ) [inline]
```

Constructor.

Parameters

<i>n</i>	Size of bit vectors
<i>mu</i>	Population size

Definition at line 115 of file genetic-algorithm.hh.

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

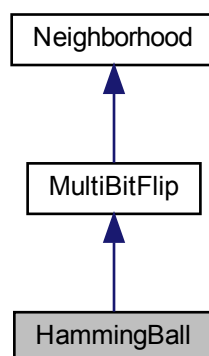
- lib/hnco/algorithms/ea/genetic-algorithm.hh
- lib/hnco/algorithms/ea/genetic-algorithm.cc

5.47 HammingBall Class Reference

Hamming ball.

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

Inheritance diagram for HammingBall:



Public Member Functions

- [HammingBall](#) (int n, int r)
Constructor.

Private Member Functions

- void [sample_bits](#) ()
Sample bits.

Private Attributes

- std::uniform_int_distribution< int > [_choose_k](#)
Choose the distance to the center.

Additional Inherited Members

5.47.1 Detailed Description

Hamming ball.

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

Definition at line 302 of file neighborhood.hh.

5.47.2 Constructor & Destructor Documentation

5.47.2.1 HammingBall()

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

Constructor.

Parameters

<i>n</i>	Size of bit vectors
<i>r</i>	Radius of the ball

Definition at line 318 of file neighborhood.hh.

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

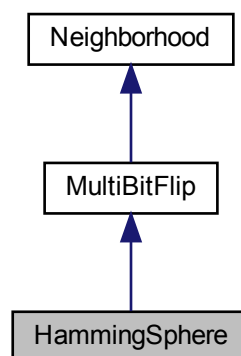
- `lib/hnco/neighborhoods/neighborhood.hh`
- `lib/hnco/neighborhoods/neighborhood.cc`

5.48 HammingSphere Class Reference

Hamming sphere.

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

Inheritance diagram for HammingSphere:



Public Member Functions

- `HammingSphere` (int n, int r)
Constructor.
- void `set_radius` (int r)
Set radius.

Private Member Functions

- void `sample_bits` ()
Sample bits.

Private Attributes

- int `_radius`
Radius of the sphere.

Additional Inherited Members

5.48.1 Detailed Description

Hamming sphere.

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

Definition at line 334 of file neighborhood.hh.

5.48.2 Constructor & Destructor Documentation

5.48.2.1 HammingSphere()

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

Constructor.

Parameters

n	Size of bit vectors
r	Radius of the sphere

Definition at line 350 of file neighborhood.hh.

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

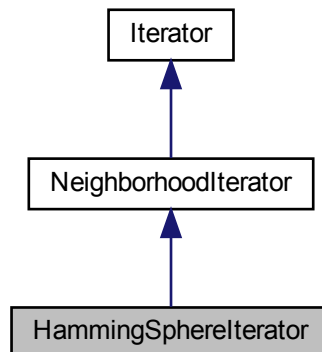
- lib/hnco/neighborhoods/neighborhood.hh
- lib/hnco/neighborhoods/neighborhood.cc

5.49 HammingSphereIterator Class Reference

Hamming sphere neighborhood iterator.

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

Inheritance diagram for HammingSphereIterator:



Public Member Functions

- [HammingSphereIterator](#) (int n, int r)
Constructor.
- bool [has_next](#) () override
Has next bit vector.
- const [bit_vector_t](#) & [next](#) () override
Next bit vector.

Private Attributes

- int [_radius](#)
Radius of the ball.
- [sparse_bit_vector_t](#) [_bit_indexes](#)
Bit indexes.

Additional Inherited Members

5.49.1 Detailed Description

Hamming sphere neighborhood iterator.

The Hamming sphere iterator is implemented using an array of indexes which indicate the bits to flip in the given origin.

For example, in dimension $n = 4$ and with radius = 2, the sequence of indexes is as follows (assuming indexes start at 1):

- 12 (first state, bits 1 and 2 are flipped)

- 13
- 14
- 23 (last index cannot be increased, first index is increased and second index is reset)
- 24
- 34

Reference: https://en.wikipedia.org/wiki/Combination#Enumerating_k-combinations

Definition at line 96 of file neighborhood-iterator.hh.

5.49.2 Constructor & Destructor Documentation

5.49.2.1 HammingSphereIterator()

```
HammingSphereIterator (
    int n,
    int r )
```

Constructor.

Parameters

n	Size of bit vectors
r	Radius of Hamming Ball

Definition at line 72 of file neighborhood-iterator.cc.

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

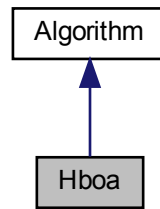
- lib/hnco/neighborhoods/neighborhood-iterator.hh
- lib/hnco/neighborhoods/neighborhood-iterator.cc

5.50 Hboa Class Reference

Hierarchical Bayesian Optimization Algorithm.

```
#include <hnco/algorithms/fast-efficient-p3/hboa.hh>
```

Inheritance diagram for Hboa:



Public Member Functions

- [Hboa](#) (int n)
Constructor.
- [~Hboa](#) ()
Destructor.
- void [maximize](#) (const std::vector< [function::Function](#) * > &functions)
Maximize.
- void [finalize](#) ()
Finalize.
- void [set_population_size](#) (int n)
Set population size.

Private Attributes

- [Implementation](#) * [_pimpl](#)
Pointer to implementation.
- int [_population_size](#) = 10
Population size.

Additional Inherited Members

5.50.1 Detailed Description

Hierarchical Bayesian Optimization Algorithm.

[Implementation](#) of the Hierarchical Bayesian Optimization Algorithm and helper classes based on the publication: Pelikan, M. and Goldberg, D. (2006). Hierarchical bayesian optimization algorithm. In Scalable Optimization via Probabilistic Modeling, volume 33 of Studies in Computational Intelligence, pages 63–90. Springer Berlin Heidelberg.

Author: Brian W. Goldman

Integrated into HNCO by Arnaud Berny

Definition at line 48 of file hboa.hh.

5.50.2 Member Data Documentation

5.50.2.1 _pimpl

```
Implementation* _pimpl [private]
```

Pointer to implementation.

The main motivation for this pattern is to avoid including declarations from `fast_efficient_p3` into the global namespace.

A raw pointer is used instead of a `unique_ptr` because the latter will not compile with `pybind11`.

Definition at line 59 of file `hboa.hh`.

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

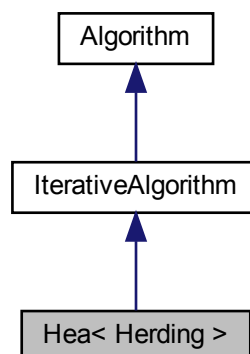
- `lib/hnco/algorithms/fast-efficient-p3/hboa.hh`
- `lib/hnco/algorithms/fast-efficient-p3/hboa.cc`

5.51 Hea< Herding > Class Template Reference

Herding evolutionary algorithm.

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

Inheritance diagram for Hea< Herding >:



Public Member Functions

- [Hea](#) (int n, int population_size)
Constructor.

Setters

- void [set_margin](#) (double x)
Set the moment margin.
- void [set_selection_size](#) (int x)
Set the selection size.
- void [set_reset_period](#) (int x)
Set the reset period.
- void [set_learning_rate](#) (double x)
Set the learning rate.
- void [set_bound_moment](#) (bool x)
Set the bound moment after update.
- void [set_randomize_bit_order](#) (bool b)
Randomize bit order.

Setters for logging

- void [set_log_herding_error](#) (bool b)
Log herding error (moment discrepancy)
- void [set_log_target_norm](#) (bool b)
Log target 2-norm (distance to uniform moment)
- void [set_log_delta_norm](#) (bool b)
Log delta (moment increment) 2-norm.
- void [set_log_target](#) (bool b)
Log target moment as a symmetric matrix.

Private Member Functions

Loop

- void [init](#) () override
Initialization.
- void [iterate](#) () override
Single iteration.
- void [set_something_to_log](#) ()
Set flag for something to log.
- void [log](#) () override
Log.

Private Attributes

- Herding::Moment [_target](#)
Target moment.
- Herding::Moment [_selection](#)
Moment of selected individuals.
- [algorithm::Population](#) [_population](#)
Population
- Herding [_herding](#)
Herding.

- double `_herding_error`
Herding error (moment discrepancy)
- double `_target_norm`
Target 2-norm (distance to uniform moment)
- double `_delta_norm`
Delta (moment increment) 2-norm.

Parameters

- double `_margin`
Moment margin.
- int `_selection_size` = 1
Selection size.
- int `_reset_period` = 0
Reset period.
- double `_learning_rate` = 1e-4
Learning rate.
- bool `_bound_moment` = false
Bound moment after update.

Logging

- bool `_log_herding_error` = false
Log herding error (moment discrepancy)
- bool `_log_target_norm` = false
Log target 2-norm (distance to uniform moment)
- bool `_log_delta_norm` = false
Log delta 2-norm (moment increment)
- bool `_log_target` = false
Log target moment as a symmetric matrix.

Additional Inherited Members

5.51.1 Detailed Description

```
template<class Herding>
class hnco::algorithm::walsh_moment::Hea< Herding >
```

Herding evolutionary algorithm.

Reference:

Arnaud Berny. 2015. Herding Evolutionary Algorithm. In Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO Companion '15). ACM, New York, NY, USA, 1355–1356.

Definition at line 47 of file hea.hh.

5.51.2 Constructor & Destructor Documentation

5.51.2.1 Hea()

```
Hea (
    int n,
    int population_size ) [inline]
```

Constructor.

Parameters

<i>n</i>	Size of bit vectors
<i>population_size</i>	Population size

`_margin` is initialized to $1 / n$.

Definition at line 200 of file `hea.hh`.

5.51.3 Member Function Documentation

5.51.3.1 `set_reset_period()`

```
void set_reset_period (
    int x ) [inline]
```

Set the reset period.

Parameters

<i>x</i>	Reset period
----------	--------------

$x \leq 0$ means no reset.

Definition at line 229 of file `hea.hh`.

5.51.3.2 `set_selection_size()`

```
void set_selection_size (
    int x ) [inline]
```

Set the selection size.

The selection size is the number of selected individuals in the population.

Definition at line 221 of file `hea.hh`.

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

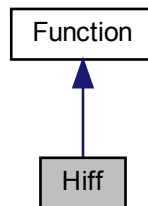
- `lib/hnco/algorithms/walsh-moment/hea.hh`

5.52 Hiff Class Reference

Hierarchical if and only if.

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

Inheritance diagram for Hiff:



Public Member Functions

- `Hiff` (int `bv_size`)
Constructor.
- int `get_bv_size` () const override
Get bit vector size.
- double `evaluate` (const `bit_vector_t` &) override
Evaluate a bit vector.
- bool `has_known_maximum` () const override
Check for a known maximum.
- double `get_maximum` () const override
Get the global maximum.

Private Attributes

- int `_bv_size`
Bit vector size.
- int `_depth`
Tree depth.

5.52.1 Detailed Description

Hierarchical if and only if.

Reference:

Thomas Jansen, *Analyzing Evolutionary Algorithms*. Springer, 2013.

Definition at line 170 of file `theory.hh`.

5.52.2 Member Function Documentation

5.52.2.1 `get_maximum()`

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

Get the global maximum.

Returns

$(i + 1) * 2^i$ where $2^i = _bv_size$

Reimplemented from [Function](#).

Definition at line 195 of file theory.hh.

5.52.2.2 `has_known_maximum()`

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

Check for a known maximum.

Returns

true

Reimplemented from [Function](#).

Definition at line 191 of file theory.hh.

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

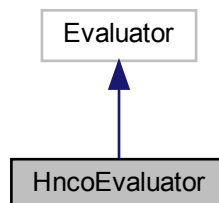
- lib/hnco/functions/collection/theory.hh
- lib/hnco/functions/collection/theory.cc

5.53 HncoEvaluator Class Reference

Evaluator for HNCO functions.

```
#include <hnco/algorithms/fast-efficient-p3/hnco-evaluator.hh>
```

Inheritance diagram for HncoEvaluator:



Public Member Functions

- [HncoEvaluator](#) ([hnco::function::Function](#) *function)
Constructor.
- float [evaluate](#) (const std::vector< bool > &x)
Evaluate a bit vector.

Private Attributes

- [hnco::function::Function](#) * [_function](#)
HNCO function.
- [hnco::bit_vector_t](#) [_bv](#)
Argument of HNCO function.

5.53.1 Detailed Description

Evaluator for HNCO functions.

Definition at line 36 of file hnco-evaluator.hh.

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

- lib/hnco/algorithms/fast-efficient-p3/hnco-evaluator.hh

5.54 HncoOptions Class Reference

Command line options for hnco.

```
#include <hnco/app/hnco-options.hh>
```

Public Member Functions

- [HncoOptions](#) (int argc, char *argv[])
Constructor.
- int [get_algorithm](#) () const
Get algorithm.
- void [set_algorithm](#) (int x)
Set algorithm.
- bool [set_algorithm](#) () const
Get set-flag for algorithm.
- int [get_bm_mc_reset_strategy](#) () const
Get bm_mc_reset_strategy.
- void [set_bm_mc_reset_strategy](#) (int x)
Set bm_mc_reset_strategy.
- bool [set_bm_mc_reset_strategy](#) () const
Get set-flag for bm_mc_reset_strategy.
- int [get_bm_num_gs_cycles](#) () const

- *Get bm_num_gs_cycles.*
 void [set_bm_num_gs_cycles](#) (int x)
- *Set bm_num_gs_cycles.*
 bool [set_bm_num_gs_cycles](#) () const
- *Get set-flag for bm_num_gs_cycles.*
 int [get_bm_num_gs_steps](#) () const
- *Get bm_num_gs_steps.*
 void [set_bm_num_gs_steps](#) (int x)
- *Set bm_num_gs_steps.*
 bool [set_bm_num_gs_steps](#) () const
- *Get set-flag for bm_num_gs_steps.*
 int [get_bm_sampling](#) () const
- *Get bm_sampling.*
 void [set_bm_sampling](#) (int x)
- *Set bm_sampling.*
 bool [set_bm_sampling](#) () const
- *Get set-flag for bm_sampling.*
 int [get_budget](#) () const
- *Get budget.*
 void [set_budget](#) (int x)
- *Set budget.*
 bool [set_budget](#) () const
- *Get set-flag for budget.*
 int [get_bv_size](#) () const
- *Get bv_size.*
 void [set_bv_size](#) (int x)
- *Set bv_size.*
 bool [set_bv_size](#) () const
- *Get set-flag for bv_size.*
 std::string [get_description_path](#) () const
- *Get description_path.*
 void [set_description_path](#) (std::string x)
- *Set description_path.*
 bool [set_description_path](#) () const
- *Get set-flag for description_path.*
 double [get_ea_crossover_bias](#) () const
- *Get ea_crossover_bias.*
 void [set_ea_crossover_bias](#) (double x)
- *Set ea_crossover_bias.*
 bool [set_ea_crossover_bias](#) () const
- *Get set-flag for ea_crossover_bias.*
 double [get_ea_crossover_probability](#) () const
- *Get ea_crossover_probability.*
 void [set_ea_crossover_probability](#) (double x)
- *Set ea_crossover_probability.*
 bool [set_ea_crossover_probability](#) () const
- *Get set-flag for ea_crossover_probability.*
 int [get_ea_lambda](#) () const
- *Get ea_lambda.*
 void [set_ea_lambda](#) (int x)
- *Set ea_lambda.*

- bool [set_ea_lambda](#) () const
Get set-flag for ea_lambda.
- int [get_ea_mu](#) () const
Get ea_mu.
- void [set_ea_mu](#) (int x)
Set ea_mu.
- bool [set_ea_mu](#) () const
Get set-flag for ea_mu.
- double [get_ea_mutation_rate](#) () const
Get ea_mutation_rate.
- void [set_ea_mutation_rate](#) (double x)
Set ea_mutation_rate.
- bool [set_ea_mutation_rate](#) () const
Get set-flag for ea_mutation_rate.
- double [get_ea_mutation_rate_max](#) () const
Get ea_mutation_rate_max.
- void [set_ea_mutation_rate_max](#) (double x)
Set ea_mutation_rate_max.
- bool [set_ea_mutation_rate_max](#) () const
Get set-flag for ea_mutation_rate_max.
- double [get_ea_mutation_rate_min](#) () const
Get ea_mutation_rate_min.
- void [set_ea_mutation_rate_min](#) (double x)
Set ea_mutation_rate_min.
- bool [set_ea_mutation_rate_min](#) () const
Get set-flag for ea_mutation_rate_min.
- double [get_ea_success_ratio](#) () const
Get ea_success_ratio.
- void [set_ea_success_ratio](#) (double x)
Set ea_success_ratio.
- bool [set_ea_success_ratio](#) () const
Get set-flag for ea_success_ratio.
- int [get_ea_tournament_size](#) () const
Get ea_tournament_size.
- void [set_ea_tournament_size](#) (int x)
Set ea_tournament_size.
- bool [set_ea_tournament_size](#) () const
Get set-flag for ea_tournament_size.
- double [get_ea_update_strength](#) () const
Get ea_update_strength.
- void [set_ea_update_strength](#) (double x)
Set ea_update_strength.
- bool [set_ea_update_strength](#) () const
Get set-flag for ea_update_strength.
- std::string [get_expression](#) () const
Get expression.
- void [set_expression](#) (std::string x)
Set expression.
- bool [set_expression](#) () const
Get set-flag for expression.
- std::string [get_fn_name](#) () const

- Get fn_name.*

 - void [set_fn_name](#) (std::string x)

Set fn_name.
- bool [set_fn_name](#) () const

Get set-flag for fn_name.
- int [get_fn_num_traps](#) () const

Get fn_num_traps.
- void [set_fn_num_traps](#) (int x)

Set fn_num_traps.
- bool [set_fn_num_traps](#) () const

Get set-flag for fn_num_traps.
- int [get_fn_prefix_length](#) () const

Get fn_prefix_length.
- void [set_fn_prefix_length](#) (int x)

Set fn_prefix_length.
- bool [set_fn_prefix_length](#) () const

Get set-flag for fn_prefix_length.
- int [get_fn_threshold](#) () const

Get fn_threshold.
- void [set_fn_threshold](#) (int x)

Set fn_threshold.
- bool [set_fn_threshold](#) () const

Get set-flag for fn_threshold.
- std::string [get_fp_expression](#) () const

Get fp_expression.
- void [set_fp_expression](#) (std::string x)

Set fp_expression.
- bool [set_fp_expression](#) () const

Get set-flag for fp_expression.
- double [get_fp_lower_bound](#) () const

Get fp_lower_bound.
- void [set_fp_lower_bound](#) (double x)

Set fp_lower_bound.
- bool [set_fp_lower_bound](#) () const

Get set-flag for fp_lower_bound.
- int [get_fp_num_bits](#) () const

Get fp_num_bits.
- void [set_fp_num_bits](#) (int x)

Set fp_num_bits.
- bool [set_fp_num_bits](#) () const

Get set-flag for fp_num_bits.
- double [get_fp_precision](#) () const

Get fp_precision.
- void [set_fp_precision](#) (double x)

Set fp_precision.
- bool [set_fp_precision](#) () const

Get set-flag for fp_precision.
- double [get_fp_upper_bound](#) () const

Get fp_upper_bound.
- void [set_fp_upper_bound](#) (double x)

Set fp_upper_bound.

- bool [set_fp_upper_bound](#) () const
Get set-flag for fp_upper_bound.
- int [get_function](#) () const
Get function.
- void [set_function](#) (int x)
Set function.
- bool [set_function](#) () const
Get set-flag for function.
- int [get_he_a_reset_period](#) () const
Get hea_reset_period.
- void [set_he_a_reset_period](#) (int x)
Set hea_reset_period.
- bool [set_he_a_reset_period](#) () const
Get set-flag for hea_reset_period.
- double [get_learning_rate](#) () const
Get learning_rate.
- void [set_learning_rate](#) (double x)
Set learning_rate.
- bool [set_learning_rate](#) () const
Get set-flag for learning_rate.
- int [get_map](#) () const
Get map.
- void [set_map](#) (int x)
Set map.
- bool [set_map](#) () const
Get set-flag for map.
- int [get_map_input_size](#) () const
Get map_input_size.
- void [set_map_input_size](#) (int x)
Set map_input_size.
- bool [set_map_input_size](#) () const
Get set-flag for map_input_size.
- std::string [get_map_path](#) () const
Get map_path.
- void [set_map_path](#) (std::string x)
Set map_path.
- bool [set_map_path](#) () const
Get set-flag for map_path.
- int [get_map_ts_length](#) () const
Get map_ts_length.
- void [set_map_ts_length](#) (int x)
Set map_ts_length.
- bool [set_map_ts_length](#) () const
Get set-flag for map_ts_length.
- int [get_map_ts_sampling_mode](#) () const
Get map_ts_sampling_mode.
- void [set_map_ts_sampling_mode](#) (int x)
Set map_ts_sampling_mode.
- bool [set_map_ts_sampling_mode](#) () const
Get set-flag for map_ts_sampling_mode.
- int [get_neighborhood](#) () const

- *Get neighborhood.*
- void [set_neighborhood](#) (int x)
Set neighborhood.
- bool [set_neighborhood](#) () const
Get set-flag for neighborhood.
- int [get_neighborhood_iterator](#) () const
Get neighborhood_iterator.
- void [set_neighborhood_iterator](#) (int x)
Set neighborhood_iterator.
- bool [set_neighborhood_iterator](#) () const
Get set-flag for neighborhood_iterator.
- double [get_noise_stddev](#) () const
Get noise_stddev.
- void [set_noise_stddev](#) (double x)
Set noise_stddev.
- bool [set_noise_stddev](#) () const
Get set-flag for noise_stddev.
- int [get_num_iterations](#) () const
Get num_iterations.
- void [set_num_iterations](#) (int x)
Set num_iterations.
- bool [set_num_iterations](#) () const
Get set-flag for num_iterations.
- int [get_num_threads](#) () const
Get num_threads.
- void [set_num_threads](#) (int x)
Set num_threads.
- bool [set_num_threads](#) () const
Get set-flag for num_threads.
- std::string [get_path](#) () const
Get path.
- void [set_path](#) (std::string x)
Set path.
- bool [set_path](#) () const
Get set-flag for path.
- double [get_pn_mutation_rate](#) () const
Get pn_mutation_rate.
- void [set_pn_mutation_rate](#) (double x)
Set pn_mutation_rate.
- bool [set_pn_mutation_rate](#) () const
Get set-flag for pn_mutation_rate.
- int [get_pn_neighborhood](#) () const
Get pn_neighborhood.
- void [set_pn_neighborhood](#) (int x)
Set pn_neighborhood.
- bool [set_pn_neighborhood](#) () const
Get set-flag for pn_neighborhood.
- int [get_pn_radius](#) () const
Get pn_radius.
- void [set_pn_radius](#) (int x)
Set pn_radius.

- bool [set_pn_radius](#) () const
Get set-flag for pn_radius.
- int [get_population_size](#) () const
Get population_size.
- void [set_population_size](#) (int x)
Set population_size.
- bool [set_population_size](#) () const
Get set-flag for population_size.
- int [get_pv_log_num_components](#) () const
Get pv_log_num_components.
- void [set_pv_log_num_components](#) (int x)
Set pv_log_num_components.
- bool [set_pv_log_num_components](#) () const
Get set-flag for pv_log_num_components.
- int [get_radius](#) () const
Get radius.
- void [set_radius](#) (int x)
Set radius.
- bool [set_radius](#) () const
Get set-flag for radius.
- int [get_rep_categorical_representation](#) () const
Get rep_categorical_representation.
- void [set_rep_categorical_representation](#) (int x)
Set rep_categorical_representation.
- bool [set_rep_categorical_representation](#) () const
Get set-flag for rep_categorical_representation.
- int [get_rep_num_additional_bits](#) () const
Get rep_num_additional_bits.
- void [set_rep_num_additional_bits](#) (int x)
Set rep_num_additional_bits.
- bool [set_rep_num_additional_bits](#) () const
Get set-flag for rep_num_additional_bits.
- std::string [get_results_path](#) () const
Get results_path.
- void [set_results_path](#) (std::string x)
Set results_path.
- bool [set_results_path](#) () const
Get set-flag for results_path.
- int [get_rls_patience](#) () const
Get rls_patience.
- void [set_rls_patience](#) (int x)
Set rls_patience.
- bool [set_rls_patience](#) () const
Get set-flag for rls_patience.
- double [get_sa_beta_ratio](#) () const
Get sa_beta_ratio.
- void [set_sa_beta_ratio](#) (double x)
Set sa_beta_ratio.
- bool [set_sa_beta_ratio](#) () const
Get set-flag for sa_beta_ratio.
- double [get_sa_initial_acceptance_probability](#) () const

- Get sa_initial_acceptance_probability.*
 • void [set_sa_initial_acceptance_probability](#) (double x)
Set sa_initial_acceptance_probability.
- bool [set_sa_initial_acceptance_probability](#) () const
Get set-flag for sa_initial_acceptance_probability.
- int [get_sa_num_transitions](#) () const
Get sa_num_transitions.
- void [set_sa_num_transitions](#) (int x)
Set sa_num_transitions.
- bool [set_sa_num_transitions](#) () const
Get set-flag for sa_num_transitions.
- int [get_sa_num_trials](#) () const
Get sa_num_trials.
- void [set_sa_num_trials](#) (int x)
Set sa_num_trials.
- bool [set_sa_num_trials](#) () const
Get set-flag for sa_num_trials.
- unsigned [get_seed](#) () const
Get seed.
- void [set_seed](#) (unsigned x)
Set seed.
- bool [set_seed](#) () const
Get set-flag for seed.
- int [get_selection_size](#) () const
Get selection_size.
- void [set_selection_size](#) (int x)
Set selection_size.
- bool [set_selection_size](#) () const
Get set-flag for selection_size.
- std::string [get_solution_path](#) () const
Get solution_path.
- void [set_solution_path](#) (std::string x)
Set solution_path.
- bool [set_solution_path](#) () const
Get set-flag for solution_path.
- double [get_target](#) () const
Get target.
- void [set_target](#) (double x)
Set target.
- bool [set_target](#) () const
Get set-flag for target.
- bool [with_additive_gaussian_noise](#) () const
Get additive_gaussian_noise.
- void [set_additive_gaussian_noise](#) ()
Set additive_gaussian_noise.
- bool [with_bm_log_norm_1](#) () const
Get bm_log_norm_1.
- void [set_bm_log_norm_1](#) ()
Set bm_log_norm_1.
- bool [with_bm_log_norm_infinite](#) () const
Get bm_log_norm_infinite.

- void [set_bm_log_norm_infinite](#) ()
Set bm_log_norm_infinite.
- bool [with_bm_negative_positive_selection](#) () const
Get bm_negative_positive_selection.
- void [set_bm_negative_positive_selection](#) ()
Set bm_negative_positive_selection.
- bool [with_cache](#) () const
Get cache.
- void [set_cache](#) ()
Set cache.
- bool [with_cache_budget](#) () const
Get cache_budget.
- void [set_cache_budget](#) ()
Set cache_budget.
- bool [with_concrete_solution](#) () const
Get concrete_solution.
- void [set_concrete_solution](#) ()
Set concrete_solution.
- bool [with_ea_allow_no_mutation](#) () const
Get ea_allow_no_mutation.
- void [set_ea_allow_no_mutation](#) ()
Set ea_allow_no_mutation.
- bool [with_ea_log_mutation_rate](#) () const
Get ea_log_mutation_rate.
- void [set_ea_log_mutation_rate](#) ()
Set ea_log_mutation_rate.
- bool [with_fn_display](#) () const
Get fn_display.
- void [set_fn_display](#) ()
Set fn_display.
- bool [with_fn_get_bv_size](#) () const
Get fn_get_bv_size.
- void [set_fn_get_bv_size](#) ()
Set fn_get_bv_size.
- bool [with_fn_get_maximum](#) () const
Get fn_get_maximum.
- void [set_fn_get_maximum](#) ()
Set fn_get_maximum.
- bool [with_fn_has_known_maximum](#) () const
Get fn_has_known_maximum.
- void [set_fn_has_known_maximum](#) ()
Set fn_has_known_maximum.
- bool [with_fn_provides_incremental_evaluation](#) () const
Get fn_provides_incremental_evaluation.
- void [set_fn_provides_incremental_evaluation](#) ()
Set fn_provides_incremental_evaluation.
- bool [with_fn_walsh_transform](#) () const
Get fn_walsh_transform.
- void [set_fn_walsh_transform](#) ()
Set fn_walsh_transform.
- bool [with_heuristic_bound_moment](#) () const

- *Get hea_bound_moment.*
 • void [set_hea_bound_moment](#) ()
 Set hea_bound_moment.
- bool [with_hea_log_delta_norm](#) () const
 Get hea_log_delta_norm.
 • void [set_hea_log_delta_norm](#) ()
 Set hea_log_delta_norm.
- bool [with_hea_log_herding_error](#) () const
 Get hea_log_herding_error.
 • void [set_hea_log_herding_error](#) ()
 Set hea_log_herding_error.
- bool [with_hea_log_target](#) () const
 Get hea_log_target.
 • void [set_hea_log_target](#) ()
 Set hea_log_target.
- bool [with_hea_log_target_norm](#) () const
 Get hea_log_target_norm.
 • void [set_hea_log_target_norm](#) ()
 Set hea_log_target_norm.
- bool [with_hea_randomize_bit_order](#) () const
 Get hea_randomize_bit_order.
 • void [set_hea_randomize_bit_order](#) ()
 Set hea_randomize_bit_order.
- bool [with_incremental_evaluation](#) () const
 Get incremental_evaluation.
 • void [set_incremental_evaluation](#) ()
 Set incremental_evaluation.
- bool [with_load_solution](#) () const
 Get load_solution.
 • void [set_load_solution](#) ()
 Set load_solution.
- bool [with_log_improvement](#) () const
 Get log_improvement.
 • void [set_log_improvement](#) ()
 Set log_improvement.
- bool [with_map_display](#) () const
 Get map_display.
 • void [set_map_display](#) ()
 Set map_display.
- bool [with_map_random](#) () const
 Get map_random.
 • void [set_map_random](#) ()
 Set map_random.
- bool [with_map_surjective](#) () const
 Get map_surjective.
 • void [set_map_surjective](#) ()
 Set map_surjective.
- bool [with_mmas_strict](#) () const
 Get mmas_strict.
 • void [set_mmas_strict](#) ()
 Set mmas_strict.

- bool [with_negation](#) () const
Get negation.
- void [set_negation](#) ()
Set negation.
- bool [with_parsed_modifier](#) () const
Get parsed_modifier.
- void [set_parsed_modifier](#) ()
Set parsed_modifier.
- bool [with_pn_allow_no_mutation](#) () const
Get pn_allow_no_mutation.
- void [set_pn_allow_no_mutation](#) ()
Set pn_allow_no_mutation.
- bool [with_print_defaults](#) () const
Get print_defaults.
- void [set_print_defaults](#) ()
Set print_defaults.
- bool [with_print_description](#) () const
Get print_description.
- void [set_print_description](#) ()
Set print_description.
- bool [with_print_header](#) () const
Get print_header.
- void [set_print_header](#) ()
Set print_header.
- bool [with_print_results](#) () const
Get print_results.
- void [set_print_results](#) ()
Set print_results.
- bool [with_print_solution](#) () const
Get print_solution.
- void [set_print_solution](#) ()
Set print_solution.
- bool [with_prior_noise](#) () const
Get prior_noise.
- void [set_prior_noise](#) ()
Set prior_noise.
- bool [with_pv_log_entropy](#) () const
Get pv_log_entropy.
- void [set_pv_log_entropy](#) ()
Set pv_log_entropy.
- bool [with_pv_log_pv](#) () const
Get pv_log_pv.
- void [set_pv_log_pv](#) ()
Set pv_log_pv.
- bool [with_record_evaluation_time](#) () const
Get record_evaluation_time.
- void [set_record_evaluation_time](#) ()
Set record_evaluation_time.
- bool [with_restart](#) () const
Get restart.
- void [set_restart](#) ()

- Set restart.
 - bool [with_rls_strict](#) () const
 - Get rls_strict.
 - void [set_rls_strict](#) ()
 - Set rls_strict.
 - bool [with_rw_log_value](#) () const
 - Get rw_log_value.
 - void [set_rw_log_value](#) ()
 - Set rw_log_value.
 - bool [with_save_description](#) () const
 - Get save_description.
 - void [set_save_description](#) ()
 - Set save_description.
 - bool [with_save_results](#) () const
 - Get save_results.
 - void [set_save_results](#) ()
 - Set save_results.
 - bool [with_save_solution](#) () const
 - Get save_solution.
 - void [set_save_solution](#) ()
 - Set save_solution.
 - bool [with_stop_on_maximum](#) () const
 - Get stop_on_maximum.
 - void [set_stop_on_maximum](#) ()
 - Set stop_on_maximum.
 - bool [with_stop_on_target](#) () const
 - Get stop_on_target.
 - void [set_stop_on_target](#) ()
 - Set stop_on_target.

Private Member Functions

- void [print_help](#) (std::ostream &stream) const
- Print help message.
 - void [print_help_fp](#) (std::ostream &stream) const
 - Print help message for section fp.
 - void [print_help_rep](#) (std::ostream &stream) const
 - Print help message for section rep.
 - void [print_help_pn](#) (std::ostream &stream) const
 - Print help message for section pn.
 - void [print_help_map](#) (std::ostream &stream) const
 - Print help message for section map.
 - void [print_help_ls](#) (std::ostream &stream) const
 - Print help message for section ls.
 - void [print_help_sa](#) (std::ostream &stream) const
 - Print help message for section sa.
 - void [print_help_ea](#) (std::ostream &stream) const
 - Print help message for section ea.
 - void [print_help_eda](#) (std::ostream &stream) const
 - Print help message for section eda.

- void [print_help_he](#)(std::ostream &stream) const
Print help message for section hea.
- void [print_help_bm](#)(std::ostream &stream) const
Print help message for section bm.
- void [print_version](#)(std::ostream &stream) const
Print version.

Private Attributes

- std::string [_exec_name](#)
Name of the executable.
- std::string [_version](#)
Name Version.
- int [_algorithm](#)
Type of algorithm.
- bool [_opt_algorithm](#)
- int [_bm_mc_reset_strategy](#)
Markov chain reset strategy.
- bool [_opt_bm_mc_reset_strategy](#)
- int [_bm_num_gs_cycles](#)
Number of Gibbs sampler cycles per bit vector.
- bool [_opt_bm_num_gs_cycles](#)
- int [_bm_num_gs_steps](#)
Number of Gibbs sampler steps per bit vector.
- bool [_opt_bm_num_gs_steps](#)
- int [_bm_sampling](#)
Sampling mode for the Boltzmann machine.
- bool [_opt_bm_sampling](#)
- int [_budget](#)
Number of allowed function evaluations (≤ 0 means indefinite)
- bool [_opt_budget](#)
- int [_bv_size](#)
Size of bit vectors.
- bool [_opt_bv_size](#)
- std::string [_description_path](#)
Path of the description file.
- bool [_opt_description_path](#)
- double [_ea_crossover_bias](#)
Crossover bias.
- bool [_opt_ea_crossover_bias](#)
- double [_ea_crossover_probability](#)
Crossover probability.
- bool [_opt_ea_crossover_probability](#)
- int [_ea_lambda](#)
Offspring population size.
- bool [_opt_ea_lambda](#)
- int [_ea_mu](#)
Parent population size.
- bool [_opt_ea_mu](#)
- double [_ea_mutation_rate](#)
Mutation rate relative to [bv_size](#) (fixed or initial value)

- `bool _opt_ea_mutation_rate`
- `double _ea_mutation_rate_max`
Maximum mutation rate.
- `bool _opt_ea_mutation_rate_max`
- `double _ea_mutation_rate_min`
Minimum mutation rate.
- `bool _opt_ea_mutation_rate_min`
- `double _ea_success_ratio`
Success rate for for self-adjusting mutation rate.
- `bool _opt_ea_success_ratio`
- `int _ea_tournament_size`
Tournament size.
- `bool _opt_ea_tournament_size`
- `double _ea_update_strength`
Update strength for self-adjusting mutation rate.
- `bool _opt_ea_update_strength`
- `std::string _expression`
Expression of the variable x.
- `bool _opt_expression`
- `std::string _fn_name`
Name of the function in the dynamic library.
- `bool _opt_fn_name`
- `int _fn_num_traps`
Number of traps.
- `bool _opt_fn_num_traps`
- `int _fn_prefix_length`
Prefix length for long path.
- `bool _opt_fn_prefix_length`
- `int _fn_threshold`
Threshold (in bits) for Jump, Four Peaks, and Six Peaks.
- `bool _opt_fn_threshold`
- `std::string _fp_expression`
Expression to parse.
- `bool _opt_fp_expression`
- `double _fp_lower_bound`
Lower bound.
- `bool _opt_fp_lower_bound`
- `int _fp_num_bits`
Number of bits in the dyadic representation of a number.
- `bool _opt_fp_num_bits`
- `double _fp_precision`
Precision of the dyadic representation of a number.
- `bool _opt_fp_precision`
- `double _fp_upper_bound`
Upper bound.
- `bool _opt_fp_upper_bound`
- `int _function`
Type of function.
- `bool _opt_function`
- `int _hea_reset_period`
Reset period (<= 0 means no reset)
- `bool _opt_hea_reset_period`

- double [_learning_rate](#)
Learning rate.
- bool **_opt_learning_rate**
- int [_map](#)
Type of map.
- bool **_opt_map**
- int [_map_input_size](#)
Input size of linear and affine maps.
- bool **_opt_map_input_size**
- std::string [_map_path](#)
Path of a map file.
- bool **_opt_map_path**
- int [_map_ts_length](#)
Transvection sequence length.
- bool **_opt_map_ts_length**
- int [_map_ts_sampling_mode](#)
Transvection sequence sampling mode.
- bool **_opt_map_ts_sampling_mode**
- int [_neighborhood](#)
Type of neighborhood.
- bool **_opt_neighborhood**
- int [_neighborhood_iterator](#)
Type of neighborhood iterator.
- bool **_opt_neighborhood_iterator**
- double [_noise_stddev](#)
Noise standard deviation.
- bool **_opt_noise_stddev**
- int [_num_iterations](#)
Number of iterations (≤ 0 means indefinite)
- bool **_opt_num_iterations**
- int [_num_threads](#)
Number of threads.
- bool **_opt_num_threads**
- std::string [_path](#)
Path of a function file.
- bool **_opt_path**
- double [_pn_mutation_rate](#)
Mutation rate relative to `bv_size`.
- bool **_opt_pn_mutation_rate**
- int [_pn_neighborhood](#)
Type of neighborhood.
- bool **_opt_pn_neighborhood**
- int [_pn_radius](#)
Radius of Hamming ball or sphere.
- bool **_opt_pn_radius**
- int [_population_size](#)
Population size.
- bool **_opt_population_size**
- int [_pv_log_num_components](#)
Number of probability vector components to log.
- bool **_opt_pv_log_num_components**
- int [_radius](#)

- *Radius of Hamming ball or sphere.*
- **bool _opt_radius**
- **int _rep_categorical_representation**
- *Categorical representation.*
- **bool _opt_rep_categorical_representation**
- **int _rep_num_additional_bits**
- *Number of additional bits per element for permutation representation.*
- **bool _opt_rep_num_additional_bits**
- **std::string _results_path**
- *Path of the results file.*
- **bool _opt_results_path**
- **int _rls_patience**
- *Number of consecutive rejected moves before ending the search (≤ 0 means infinite)*
- **bool _opt_rls_patience**
- **double _sa_beta_ratio**
- *Ratio for beta or inverse temperature.*
- **bool _opt_sa_beta_ratio**
- **double _sa_initial_acceptance_probability**
- *Initial acceptance probability.*
- **bool _opt_sa_initial_acceptance_probability**
- **int _sa_num_transitions**
- *Number of accepted transitions before annealing.*
- **bool _opt_sa_num_transitions**
- **int _sa_num_trials**
- *Number of trials to estimate initial inverse temperature.*
- **bool _opt_sa_num_trials**
- **unsigned _seed**
- *Seed for the random number generator.*
- **bool _opt_seed**
- **int _selection_size**
- *Selection size (number of selected individuals)*
- **bool _opt_selection_size**
- **std::string _solution_path**
- *Path of the solution file.*
- **bool _opt_solution_path**
- **double _target**
- *Target.*
- **bool _opt_target**
- **bool _additive_gaussian_noise**
- *Additive Gaussian noise.*
- **bool _bm_log_norm_1**
- *Log 1-norm of the parameters.*
- **bool _bm_log_norm_infinite**
- *Log infinite norm of the parameters.*
- **bool _bm_negative_positive_selection**
- *Negative and positive selection.*
- **bool _cache**
- *Cache function evaluations.*
- **bool _cache_budget**
- *Set cache on budget.*
- **bool _concrete_solution**
- *At the end, print or save the solution in the domain of the concrete function.*

- [bool _ea_allow_no_mutation](#)
Allow no mutation with standard bit mutation.
- [bool _ea_log_mutation_rate](#)
Log mutation rate.
- [bool _fn_display](#)
Display the function and exit.
- [bool _fn_get_bv_size](#)
Print the size of bit vectors.
- [bool _fn_get_maximum](#)
If the maximum is known then print it and exit with status 0 else exit with status 1.
- [bool _fn_has_known_maximum](#)
Does the function have a known maximum?
- [bool _fn_provides_incremental_evaluation](#)
Does the function provide incremental evaluation?
- [bool _fn_walsh_transform](#)
Compute the Walsh transform of the function.
- [bool _hea_bound_moment](#)
Bound moment after update.
- [bool _hea_log_delta_norm](#)
Log delta (moment increment) 2-norm.
- [bool _hea_log_herding_error](#)
Log herding error (moment discrepancy)
- [bool _hea_log_target](#)
Log target moment as a symmetric matrix.
- [bool _hea_log_target_norm](#)
Log target 2-norm (distance to uniform moment)
- [bool _hea_randomize_bit_order](#)
Randomize bit order.
- [bool _incremental_evaluation](#)
Incremental evaluation.
- [bool _load_solution](#)
Load a solution from a file.
- [bool _log_improvement](#)
Log improvement.
- [bool _map_display](#)
Display the map and exit.
- [bool _map_random](#)
Sample a random map.
- [bool _map_surjective](#)
Ensure that the sampled linear or affine map is surjective.
- [bool _mmas_strict](#)
Strict (>) max-min ant system.
- [bool _negation](#)
Negation (hence minimization) of the function.
- [bool _parsed_modifier](#)
Parsed modifier.
- [bool _pn_allow_no_mutation](#)
Allow no mutation with standard bit mutation.
- [bool _print_defaults](#)
Print the default parameters and exit.
- [bool _print_description](#)

- *Print a description of the solution.*
- [bool _print_header](#)
At the beginning, print the header.
- [bool _print_results](#)
Print results.
- [bool _print_solution](#)
Print the solution.
- [bool _prior_noise](#)
Prior noise.
- [bool _pv_log_entropy](#)
Log entropy of probability vector.
- [bool _pv_log_pv](#)
Log probability vector.
- [bool _record_evaluation_time](#)
Record evaluation time.
- [bool _restart](#)
Restart any algorithm an indefinite number of times.
- [bool _rls_strict](#)
Strict (>) random local search.
- [bool _rw_log_value](#)
Log bit vector value during random walk.
- [bool _save_description](#)
At the end, save a description of the solution in a file.
- [bool _save_results](#)
At the end, save results in a file.
- [bool _save_solution](#)
At the end, save the solution in a file.
- [bool _stop_on_maximum](#)
Stop on maximum.
- [bool _stop_on_target](#)
Stop on target.

Friends

- `std::ostream & operator<< (std::ostream &, const HncoOptions &)`
Print a header containing the parameter values.

5.54.1 Detailed Description

Command line options for hnco.

Definition at line 11 of file hnco-options.hh.

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

- lib/hnco/app/hnco-options.hh
- lib/hnco/app/hnco-options.cc

5.55 HncoOptions Class Reference

Command line options for hnco-mo.

```
#include <hnco/multiobjective/app/hnco-mo-options.hh>
```

Public Member Functions

- [HncoOptions](#) (int argc, char *argv[])
Constructor.
- int [get_algorithm](#) () const
Get algorithm.
- void [set_algorithm](#) (int x)
Set algorithm.
- bool [set_algorithm](#) () const
Get set-flag for algorithm.
- int [get_bv_size](#) () const
Get bv_size.
- void [set_bv_size](#) (int x)
Set bv_size.
- bool [set_bv_size](#) () const
Get set-flag for bv_size.
- std::string [get_description_path](#) () const
Get description_path.
- void [set_description_path](#) (std::string x)
Set description_path.
- bool [set_description_path](#) () const
Get set-flag for description_path.
- double [get_ea_crossover_probability](#) () const
Get ea_crossover_probability.
- void [set_ea_crossover_probability](#) (double x)
Set ea_crossover_probability.
- bool [set_ea_crossover_probability](#) () const
Get set-flag for ea_crossover_probability.
- int [get_ea_mu](#) () const
Get ea_mu.
- void [set_ea_mu](#) (int x)
Set ea_mu.
- bool [set_ea_mu](#) () const
Get set-flag for ea_mu.
- double [get_ea_mutation_rate](#) () const
Get ea_mutation_rate.
- void [set_ea_mutation_rate](#) (double x)
Set ea_mutation_rate.
- bool [set_ea_mutation_rate](#) () const
Get set-flag for ea_mutation_rate.
- int [get_ea_tournament_size](#) () const
Get ea_tournament_size.
- void [set_ea_tournament_size](#) (int x)
Set ea_tournament_size.

- bool [set_ea_tournament_size](#) () const
Get set-flag for ea_tournament_size.
- std::string [get_fn_name](#) () const
Get fn_name.
- void [set_fn_name](#) (std::string x)
Set fn_name.
- bool [set_fn_name](#) () const
Get set-flag for fn_name.
- std::string [get_fp_expression](#) () const
Get fp_expression.
- void [set_fp_expression](#) (std::string x)
Set fp_expression.
- bool [set_fp_expression](#) () const
Get set-flag for fp_expression.
- double [get_fp_lower_bound](#) () const
Get fp_lower_bound.
- void [set_fp_lower_bound](#) (double x)
Set fp_lower_bound.
- bool [set_fp_lower_bound](#) () const
Get set-flag for fp_lower_bound.
- int [get_fp_num_bits](#) () const
Get fp_num_bits.
- void [set_fp_num_bits](#) (int x)
Set fp_num_bits.
- bool [set_fp_num_bits](#) () const
Get set-flag for fp_num_bits.
- double [get_fp_precision](#) () const
Get fp_precision.
- void [set_fp_precision](#) (double x)
Set fp_precision.
- bool [set_fp_precision](#) () const
Get set-flag for fp_precision.
- double [get_fp_upper_bound](#) () const
Get fp_upper_bound.
- void [set_fp_upper_bound](#) (double x)
Set fp_upper_bound.
- bool [set_fp_upper_bound](#) () const
Get set-flag for fp_upper_bound.
- int [get_function](#) () const
Get function.
- void [set_function](#) (int x)
Set function.
- bool [set_function](#) () const
Get set-flag for function.
- int [get_num_iterations](#) () const
Get num_iterations.
- void [set_num_iterations](#) (int x)
Set num_iterations.
- bool [set_num_iterations](#) () const
Get set-flag for num_iterations.
- int [get_num_threads](#) () const

- Get num_threads.*

 - void [set_num_threads](#) (int x)

Set num_threads.
- bool [set_num_threads](#) () const

Get set-flag for num_threads.
- std::string [get_path](#) () const

Get path.
- void [set_path](#) (std::string x)

Set path.
- bool [set_path](#) () const

Get set-flag for path.
- int [get_rep_categorical_representation](#) () const

Get rep_categorical_representation.
- void [set_rep_categorical_representation](#) (int x)

Set rep_categorical_representation.
- bool [set_rep_categorical_representation](#) () const

Get set-flag for rep_categorical_representation.
- int [get_rep_num_additional_bits](#) () const

Get rep_num_additional_bits.
- void [set_rep_num_additional_bits](#) (int x)

Set rep_num_additional_bits.
- bool [set_rep_num_additional_bits](#) () const

Get set-flag for rep_num_additional_bits.
- std::string [get_results_path](#) () const

Get results_path.
- void [set_results_path](#) (std::string x)

Set results_path.
- bool [set_results_path](#) () const

Get set-flag for results_path.
- unsigned [get_seed](#) () const

Get seed.
- void [set_seed](#) (unsigned x)

Set seed.
- bool [set_seed](#) () const

Get set-flag for seed.
- std::string [get_solution_path](#) () const

Get solution_path.
- void [set_solution_path](#) (std::string x)

Set solution_path.
- bool [set_solution_path](#) () const

Get set-flag for solution_path.
- bool [with_ea_allow_no_mutation](#) () const

Get ea_allow_no_mutation.
- void [set_ea_allow_no_mutation](#) ()

Set ea_allow_no_mutation.
- bool [with_fn_display](#) () const

Get fn_display.
- void [set_fn_display](#) ()

Set fn_display.
- bool [with_fn_get_bv_size](#) () const

Get fn_get_bv_size.

- void [set_fn_get_bv_size](#) ()
Set fn_get_bv_size.
- bool [with_fn_get_output_size](#) () const
Get fn_get_output_size.
- void [set_fn_get_output_size](#) ()
Set fn_get_output_size.
- bool [with_print_defaults](#) () const
Get print_defaults.
- void [set_print_defaults](#) ()
Set print_defaults.
- bool [with_print_description](#) () const
Get print_description.
- void [set_print_description](#) ()
Set print_description.
- bool [with_print_header](#) () const
Get print_header.
- void [set_print_header](#) ()
Set print_header.
- bool [with_print_pareto_front](#) () const
Get print_pareto_front.
- void [set_print_pareto_front](#) ()
Set print_pareto_front.

Private Member Functions

- void [print_help](#) (std::ostream &stream) const
Print help message.
- void [print_help_fp](#) (std::ostream &stream) const
Print help message for section fp.
- void [print_help_rep](#) (std::ostream &stream) const
Print help message for section rep.
- void [print_help_ea](#) (std::ostream &stream) const
Print help message for section ea.
- void [print_version](#) (std::ostream &stream) const
Print version.

Private Attributes

- std::string [_exec_name](#)
Name of the executable.
- std::string [_version](#)
Name Version.
- int [_algorithm](#)
Type of algorithm.
- bool [_opt_algorithm](#)
- int [_bv_size](#)
Size of bit vectors.
- bool [_opt_bv_size](#)
- std::string [_description_path](#)

Path of the description file.

- bool **_opt_description_path**
- double [_ea_crossover_probability](#)

Crossover probability.

- bool **_opt_ea_crossover_probability**
- int [_ea_mu](#)

Parent population size.

- bool **_opt_ea_mu**
- double [_ea_mutation_rate](#)

Mutation rate relative to `bv_size`.

- bool **_opt_ea_mutation_rate**
- int [_ea_tournament_size](#)

Tournament size.

- bool **_opt_ea_tournament_size**
- std::string [_fn_name](#)

Name of the function in the dynamic library.

- bool **_opt_fn_name**
- std::string [_fp_expression](#)

Expression to parse.

- bool **_opt_fp_expression**
- double [_fp_lower_bound](#)

Lower bound.

- bool **_opt_fp_lower_bound**
- int [_fp_num_bits](#)

Number of bits in the dyadic representation of a number.

- bool **_opt_fp_num_bits**
- double [_fp_precision](#)

Precision of the dyadic representation of a number.

- bool **_opt_fp_precision**
- double [_fp_upper_bound](#)

Upper bound.

- bool **_opt_fp_upper_bound**
- int [_function](#)

Type of function.

- bool **_opt_function**
- int [_num_iterations](#)

Number of iterations.

- bool **_opt_num_iterations**
- int [_num_threads](#)

Number of threads.

- bool **_opt_num_threads**
- std::string [_path](#)

Path of a function file.

- bool **_opt_path**
- int [_rep_categorical_representation](#)

Categorical representation.

- bool **_opt_rep_categorical_representation**
- int [_rep_num_additional_bits](#)

Number of additional bits per element for permutation representation.

- bool **_opt_rep_num_additional_bits**
- std::string [_results_path](#)

Path of the results file.

- `bool _opt_results_path`
- `unsigned _seed`
Seed for the random number generator.
- `bool _opt_seed`
- `std::string _solution_path`
Path of the solution file.
- `bool _opt_solution_path`
- `bool _ea_allow_no_mutation`
Allow no mutation with standard bit mutation.
- `bool _fn_display`
Display the function and exit.
- `bool _fn_get_bv_size`
Print the size of bit vectors.
- `bool _fn_get_output_size`
Print the number of objectives.
- `bool _print_defaults`
Print the default parameters and exit.
- `bool _print_description`
Print a description of the solution.
- `bool _print_header`
At the beginning, print the header.
- `bool _print_pareto_front`
Print the Pareto front.

Friends

- `std::ostream & operator<< (std::ostream &, const HncoOptions &)`
Print a header containing the parameter values.

5.55.1 Detailed Description

Command line options for hnco-mo.

Definition at line 12 of file hnco-mo-options.hh.

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

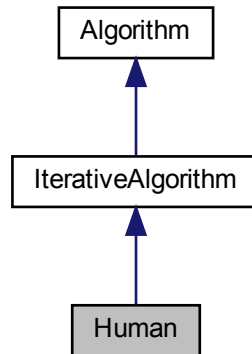
- `lib/hnco/multiobjective/app/hnco-mo-options.hh`
- `lib/hnco/multiobjective/app/hnco-mo-options.cc`

5.56 Human Class Reference

Human

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

Inheritance diagram for Human:



Public Member Functions

- [Human](#) (int n)
Constructor.

Protected Member Functions

- void [parse_bit_vector](#) ()
Parse bit vector.

Loop

- void [init](#) () override
Initialize.
- void [iterate](#) () override
Single iteration.

Protected Attributes

- [bit_vector_t_candidate](#)
Candidate.

5.56.1 Detailed Description

Human

Definition at line 32 of file human.hh.

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

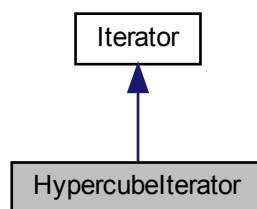
- lib/hnco/algorithms/human.hh
- lib/hnco/algorithms/human.cc

5.57 Hypercubeliterator Class Reference

Hypercube iterator.

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

Inheritance diagram for Hypercubeliterator:



Public Member Functions

- [Hypercubeliterator](#) (int n)
Constructor.
- bool [has_next](#) () override
Has next bit vector.
- const [bit_vector_t](#) & [next](#) () override
Next bit vector.

Additional Inherited Members

5.57.1 Detailed Description

Hypercube iterator.

Implemented as a simple binary adder.

Definition at line 69 of file iterator.hh.

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

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

5.58 Implementation Struct Reference

Implementation

```
#include <hnco/algorithms/fast-efficient-p3/implementation.hh>
```

Public Attributes

- Configuration [configuration](#)
Configuration.
- std::shared_ptr< [HncoEvaluator](#) > [evaluator](#)
Evaluator.
- std::shared_ptr< Middle_Layer > [middle_layer](#)
Middle layer.

5.58.1 Detailed Description

Implementation

Definition at line 37 of file implementation.hh.

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

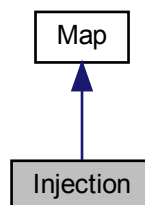
- lib/hnco/algorithms/fast-efficient-p3/implementation.hh

5.59 Injection Class Reference

Injection.

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

Inheritance diagram for Injection:



Public Member Functions

- [Injection](#) (const std::vector< int > &bit_positions, int output_size)
Constructor.
- void [map](#) (const [bit_vector_t](#) &input, [bit_vector_t](#) &output) override
Map
- int [get_input_size](#) () const override
Get input size.
- int [get_output_size](#) () const override
Get output size.
- bool [is_surjective](#) () const override
Check for surjective map.

Private Attributes

- std::vector< int > [_bit_positions](#)
Bit positions.
- int [_output_size](#)
Output size.

5.59.1 Detailed Description

Injection.

An injection copies the bits of input x to given positions of output y.

Let $I = \{i_1, i_2, \dots, i_m\}$ be a subset of $\{1, 2, \dots, n\}$.

An injection f from F_2^m to F_2^n , where $n \geq m$, is defined by $f(x) = y$, where, for all $j \in \{1, 2, \dots, m\}$, $y_{i_j} = x_j$.

If f is a projection and g is an injection with the same bit positions then their composition $f \circ g$ is the identity.

Definition at line 493 of file map.hh.

5.59.2 Constructor & Destructor Documentation

5.59.2.1 Injection()

```
Injection (
    const std::vector< int > & bit_positions,
    int output_size )
```

Constructor.

The input size of the map is given by the size of bit_positions.

Parameters

| | |
|----------------------|--|
| <i>bit_positions</i> | Bit positions in the output to where input bits are copied |
| <i>output_size</i> | Output size |

Precondition

`output_size >= bit_positions.size()`

Definition at line 176 of file map.cc.

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

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

5.60 IntegerCategoricalRepresentation Class Reference

Integer categorical representation.

```
#include <hnco/representations/categorical.hh>
```

Public Types

- using `domain_type` = `std::size_t`
Domain type.

Public Member Functions

- `IntegerCategoricalRepresentation` (int num_categories)
Constructor.
- int `size` () const
Size of the representation.
- `domain_type unpack` (const `bit_vector_t` &bv, int start)
Unpack bit vector into a category.
- void `display` (std::ostream &stream) const
Display.

Private Attributes

- int `_num_categories`
Number of categories.
- int `_num_bits`
Number of bits.

5.60.1 Detailed Description

Integer categorical representation.

Definition at line 142 of file categorical.hh.

5.60.2 Constructor & Destructor Documentation

5.60.2.1 IntegerCategoricalRepresentation()

```
IntegerCategoricalRepresentation (
    int num_categories ) [inline]
```

Constructor.

Parameters

| | |
|-----------------------|----------------------|
| <i>num_categories</i> | Number of categories |
|-----------------------|----------------------|

Definition at line 159 of file categorical.hh.

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

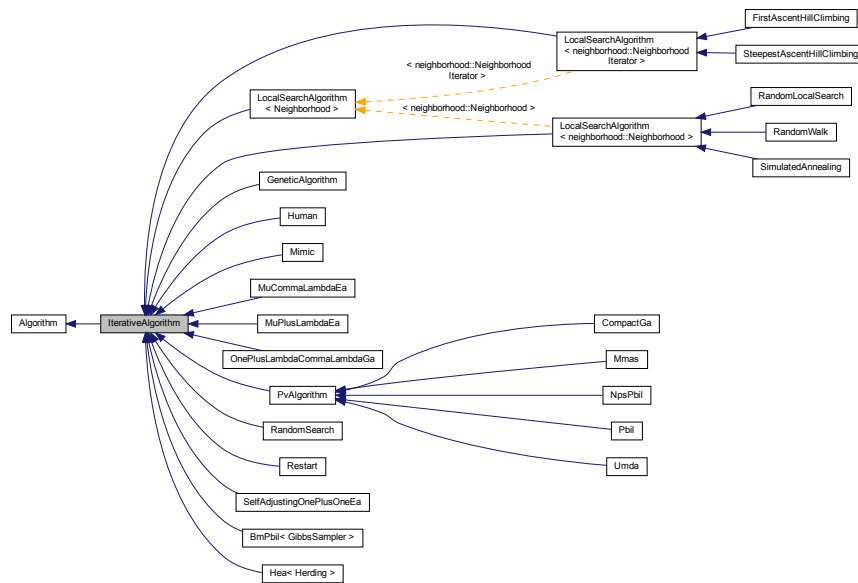
- lib/hnco/representations/categorical.hh

5.61 IterativeAlgorithm Class Reference

Iterative search.

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

Inheritance diagram for IterativeAlgorithm:



Public Member Functions

- [IterativeAlgorithm](#) (int n)
Constructor.

Optimization

- void [maximize](#) (const std::vector< [function::Function](#) * > &functions) override
Maximize.

Setters

- void [set_num_iterations](#) (int x)
Set the number of iterations.

Protected Member Functions

Loop

- virtual void [init](#) ()
Initialize.
- virtual void [iterate](#) ()=0
Single iteration.
- virtual void [log](#) ()
Log.
- virtual void [loop](#) () final
Loop.

Protected Attributes

- `int _iteration`
Current iteration.
- `bool _last_iteration = false`
Last iteration.
- `bool _something_to_log = false`
Something to log.

Parameters

- `int _num_iterations = 0`
Number of iterations.

5.61.1 Detailed Description

Iterative search.

Definition at line 32 of file `iterative-algorithm.hh`.

5.61.2 Constructor & Destructor Documentation

5.61.2.1 IterativeAlgorithm()

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

Constructor.

Parameters

| | |
|----------------|---------------------|
| <code>n</code> | Size of bit vectors |
|----------------|---------------------|

Definition at line 83 of file `iterative-algorithm.hh`.

5.61.3 Member Function Documentation

5.61.3.1 loop()

```
void loop ( ) [final], [protected], [virtual]
```

Loop.

Calls [init\(\)](#) then enter the main loop which, at each iteration, calls [iterate\(\)](#) then [log\(\)](#) only if `_something_to_log` is true.

Definition at line 28 of file `iterative-algorithm.cc`.

5.61.3.2 `maximize()`

```
void maximize (
    const std::vector< function::Function * > & functions ) [override], [virtual]
```

Maximize.

Calls [set_functions\(\)](#) then loop.

Implements [Algorithm](#).

Definition at line 53 of file `iterative-algorithm.cc`.

5.61.3.3 `set_num_iterations()`

```
void set_num_iterations (
    int x ) [inline]
```

Set the number of iterations.

Parameters

| | |
|----------------|----------------------|
| <code>x</code> | Number of iterations |
|----------------|----------------------|

Warning

`x <= 0` means indefinite

Definition at line 109 of file `iterative-algorithm.hh`.

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

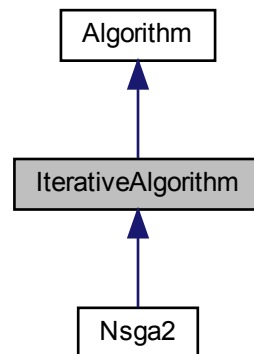
- `lib/hnco/algorithms/iterative-algorithm.hh`
- `lib/hnco/algorithms/iterative-algorithm.cc`

5.62 IterativeAlgorithm Class Reference

Iterative algorithm.

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

Inheritance diagram for IterativeAlgorithm:



Public Member Functions

- [IterativeAlgorithm](#) (int n, int num_objectives)
Constructor.

Optimization

- void [minimize](#) (const std::vector< [Function](#) * > &functions) override
Minimize.

Setters

- void [set_num_iterations](#) (int n)
Set the number of iterations.

Protected Member Functions

Loop

- virtual void [init](#) ()
Initialize.
- virtual void [iterate](#) ()=0
Single iteration.
- virtual void [log](#) ()
Log.
- virtual void [finalize](#) ()
Finalize.
- virtual void [loop](#) () final
Loop.

Protected Attributes

- `int _iteration`
Current iteration.
- `bool _last_iteration = false`
Last iteration.
- `bool _something_to_log = false`
Something to log.

Parameters

- `int _num_iterations = 0`
Number of iterations.

Additional Inherited Members

5.62.1 Detailed Description

Iterative algorithm.

Definition at line 33 of file `iterative-algorithm.hh`.

5.62.2 Constructor & Destructor Documentation

5.62.2.1 IterativeAlgorithm()

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

Constructor.

Parameters

| | |
|-----------------------|----------------------|
| <i>n</i> | Size of bit vectors |
| <i>num_objectives</i> | Number of objectives |

Definition at line 87 of file `iterative-algorithm.hh`.

5.62.3 Member Function Documentation

5.62.3.1 loop()

```
void loop ( ) [final], [protected], [virtual]
```

Loop.

Calls [init\(\)](#) then enter the main loop which, at each iteration, calls [iterate\(\)](#) then [log\(\)](#) only if `_something_to_log` is true.

Definition at line 28 of file `iterative-algorithm.cc`.

5.62.3.2 minimize()

```
void minimize (
    const std::vector< Function * > & functions ) [override], [virtual]
```

Minimize.

Calls [set_functions\(\)](#) then loop.

Implements [Algorithm](#).

Definition at line 43 of file `iterative-algorithm.cc`.

5.62.3.3 set_num_iterations()

```
void set_num_iterations (
    int n ) [inline]
```

Set the number of iterations.

Parameters

| | |
|----------|----------------------|
| <i>n</i> | Number of iterations |
|----------|----------------------|

Warning

`n <= 0` means indefinite

Definition at line 113 of file `iterative-algorithm.hh`.

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

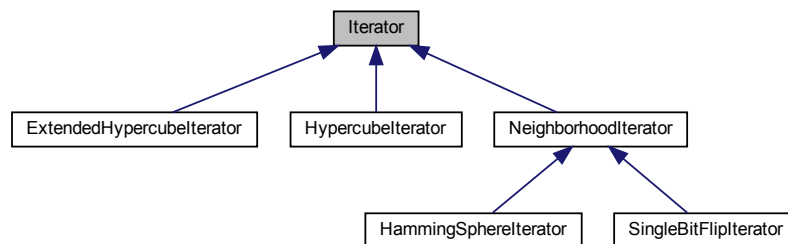
- `lib/hnco/multiobjective/algorithms/iterative-algorithm.hh`
- `lib/hnco/multiobjective/algorithms/iterative-algorithm.cc`

5.63 Iterator Class Reference

Iterator over bit vectors

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

Inheritance diagram for Iterator:



Public Member Functions

- `Iterator` (int n)
Constructor.
- `virtual ~Iterator` ()
Destructor.
- `virtual void init` ()
Initialization.
- `virtual bool has_next` ()=0
Has next bit vector.
- `virtual const bit_vector_t & next` ()=0
Next bit vector.

Protected Attributes

- `bit_vector_t _current`
Current bit vector.
- `bool _initial_state` = true
Flag for initial state.

5.63.1 Detailed Description

Iterator over bit vectors

Definition at line 34 of file iterator.hh.

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

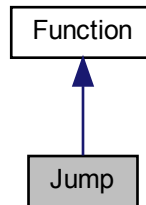
- lib/hnco/iterator.hh

5.64 Jump Class Reference

Jump.

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

Inheritance diagram for Jump:



Public Member Functions

- [Jump](#) (int bv_size, int gap)
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- bool [has_known_maximum](#) () const override
Check for a known maximum.
- double [get_maximum](#) () const override
Get the global maximum.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Private Attributes

- int [_bv_size](#)
Bit vector size.
- int [_gap](#)
Gap.

5.64.1 Detailed Description

Jump.

Reference:

H. Mühlenbein and T. Mahnig. 2001. Evolutionary Algorithms: From Recombination to Search Distributions. In Theoretical Aspects of Evolutionary Computing, Leila Kallel, Bart Naudts, and Alex Rogers (Eds.). Springer Berlin Heidelberg, 135–174.

Definition at line 41 of file jump.hh.

5.64.2 Member Function Documentation

5.64.2.1 `get_maximum()`

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

Get the global maximum.

Returns

`_bv_size`

Reimplemented from [Function](#).

Definition at line 64 of file `jump.hh`.

5.64.2.2 `has_known_maximum()`

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

Check for a known maximum.

Returns

`true`

Reimplemented from [Function](#).

Definition at line 60 of file `jump.hh`.

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

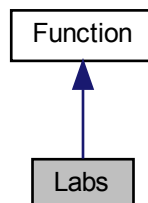
- `lib/hnco/functions/collection/jump.hh`
- `lib/hnco/functions/collection/jump.cc`

5.65 Labs Class Reference

Low autocorrelation binary sequences.

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

Inheritance diagram for Labs:



Public Member Functions

- [Labs](#) (int n)
Constructor.
- void [set_merit_factor_flag](#) (bool b)
Set merit factor flag.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Protected Member Functions

- double [compute_autocorrelation](#) (const [bit_vector_t](#) &)
Compute autocorrelation.

Protected Attributes

- std::vector< int > [_sequence](#)
Binary sequence written using 1 and -1.
- bool [_merit_factor_flag](#) = false
Merit factor flag.

5.65.1 Detailed Description

Low autocorrelation binary sequences.

Reference:

S Mertens. 1996. Exhaustive search for low-autocorrelation binary sequences. Journal of Physics A: Mathematical and General 29, 18 (1996), L473.

<http://stacks.iop.org/0305-4470/29/i=18/a=005>

If [_merit_factor_flag](#) is true then the function returns $n / (2 * \text{autocorrelation})$ else it returns $-\text{autocorrelation}$.

Definition at line 44 of file labs.hh.

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

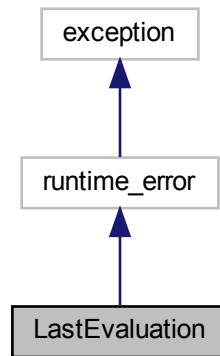
- lib/hnco/functions/collection/labs.hh
- lib/hnco/functions/collection/labs.cc

5.66 LastEvaluation Class Reference

Last evaluation.

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

Inheritance diagram for LastEvaluation:



5.66.1 Detailed Description

Last evaluation.

Definition at line 33 of file exception.hh.

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

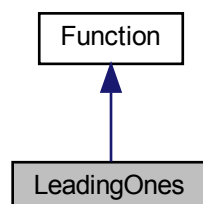
- lib/hnco/exception.hh

5.67 LeadingOnes Class Reference

Leading ones.

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

Inheritance diagram for LeadingOnes:



Public Member Functions

- [LeadingOnes](#) (int bv_size)
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- bool [has_known_maximum](#) () const override
Check for a known maximum.
- double [get_maximum](#) () const override
Get the global maximum.

Private Attributes

- int [_bv_size](#)
Bit vector size.

5.67.1 Detailed Description

Leading ones.

Reference:

Thomas Jansen, Analyzing Evolutionary Algorithms. Springer, 2013.

Definition at line 100 of file theory.hh.

5.67.2 Member Function Documentation

5.67.2.1 [get_maximum\(\)](#)

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

Get the global maximum.

Returns

[_bv_size](#)

Reimplemented from [Function](#).

Definition at line 123 of file theory.hh.

5.67.2.2 has_known_maximum()

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

Check for a known maximum.

Returns

true

Reimplemented from [Function](#).

Definition at line 119 of file theory.hh.

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

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

5.68 LinearCategoricalRepresentation Class Reference

Linear categorical representation.

```
#include <hnco/representations/categorical.hh>
```

Public Types

- using [domain_type](#) = std::size_t
Domain type.

Public Member Functions

- [LinearCategoricalRepresentation](#) (int num_categories)
Constructor.
- int [size](#) () const
Size of the representation.
- [domain_type unpack](#) (const [bit_vector_t](#) &bv, int start)
Unpack bit vector into a category.
- void [display](#) (std::ostream &stream) const
Display.

Private Attributes

- `int _num_categories`
Number of categories.
- `int _nrows`
Number of rows.
- `int _ncols`
Number of columns.
- `bit_matrix_t _A`
Linear code as a bit matrix.
- `bit_vector_t _y`
Output category.
- `bit_vector_t _x`
Input bit vector.

5.68.1 Detailed Description

Linear categorical representation.

Definition at line 43 of file categorical.hh.

5.68.2 Constructor & Destructor Documentation

5.68.2.1 LinearCategoricalRepresentation()

```
LinearCategoricalRepresentation (
    int num_categories ) [inline]
```

Constructor.

Parameters

| | |
|-----------------------------|----------------------|
| <code>num_categories</code> | Number of categories |
|-----------------------------|----------------------|

Definition at line 72 of file categorical.hh.

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

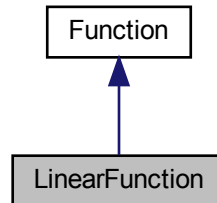
- `lib/hnco/representations/categorical.hh`

5.69 LinearFunction Class Reference

Linear function.

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

Inheritance diagram for LinearFunction:



Public Member Functions

- [LinearFunction](#) ()

Constructor.

Instance generators

- `template<class Generator >`
void [generate](#) (int n, Generator generator)
Instance generator.
- void [random](#) (int n)
Random instance.

Load and save instance

- void [load](#) (std::string path)
Load instance.
- void [save](#) (std::string path) const
Save instance.

Evaluation

- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- double [evaluate_incrementally](#) (const [bit_vector_t](#) &x, double v, const [hnco::sparse_bit_vector_t](#) &flipped_bits) override
Incrementally evaluate a bit vector.

Information about the function

- int [get_bv_size](#) () const override
Get bit vector size.
- double [get_maximum](#) () const override
Get the global maximum.
- bool [has_known_maximum](#) () const override
Check for a known maximum.
- bool [provides_incremental_evaluation](#) () const override
Check whether the function provides incremental evaluation.
- void [display](#) (std::ostream &stream) const override
Display.

Private Member Functions

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

Private Attributes

- `std::vector< double > _weights`
Weights.

Friends

- class `boost::serialization::access`

5.69.1 Detailed Description

Linear function.

Definition at line 39 of file linear-function.hh.

5.69.2 Member Function Documentation

5.69.2.1 `generate()`

```
void generate (
    int n,
    Generator generator ) [inline]
```

Instance generator.

Parameters

| | |
|------------------|---------------------|
| <i>n</i> | Size of bit vectors |
| <i>generator</i> | Weight generator |

Definition at line 70 of file linear-function.hh.

5.69.2.2 `has_known_maximum()`

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

Check for a known maximum.

Returns

true

Reimplemented from [Function](#).

Definition at line 135 of file linear-function.hh.

5.69.2.3 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to load |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 99 of file linear-function.hh.

5.69.2.4 provides_incremental_evaluation()

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

Check whether the function provides incremental evaluation.

Returns

true

Reimplemented from [Function](#).

Definition at line 140 of file linear-function.hh.

5.69.2.5 random()

```
void random (
    int n ) [inline]
```

Random instance.

The weights are sampled from the normal distribution.

Parameters

| | |
|----------|---------------------|
| <i>n</i> | Size of bit vectors |
|----------|---------------------|

Definition at line 82 of file linear-function.hh.

5.69.2.6 save()

```
void save (
    std::string path ) const [inline]
```

Save instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to save |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

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

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

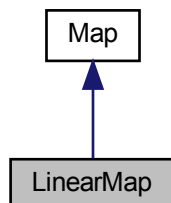
- lib/hnco/functions/collection/linear-function.hh
- lib/hnco/functions/collection/linear-function.cc

5.70 LinearMap Class Reference

Linear map.

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

Inheritance diagram for LinearMap:



Public Member Functions

- void [random](#) (int rows, int cols, bool surjective)
Random instance.
- void [map](#) (const [bit_vector_t](#) &input, [bit_vector_t](#) &output) override
Map
- int [get_input_size](#) () const override
Get input size.
- int [get_output_size](#) () const override
Get output size.
- bool [is_surjective](#) () const override
Check for surjective map.

Load and save map

- void [load](#) (std::string path)
Load map.
- void [save](#) (std::string path) const
Save map.

Private Member Functions

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

Private Attributes

- [bit_matrix_t_bm](#)
Bit matrix.

Friends

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

5.70.1 Detailed Description

Linear map.

A linear map f from F_2^m to F_2^n is defined by $f(x) = Ax$, where A is an $n \times m$ bit matrix.

Definition at line 248 of file map.hh.

5.70.2 Member Function Documentation

5.70.2.1 is_surjective()

```
bool is_surjective ( ) const [override], [virtual]
```

Check for surjective map.

Returns

true if rank(_bm) == bm_num_rows(_bm)

Reimplemented from [Map](#).

Definition at line 110 of file map.cc.

5.70.2.2 load()

```
void load (
    std::string path ) [inline]
```

Load map.

Parameters

| | |
|-------------|------------------|
| <i>path</i> | Path of the file |
|-------------|------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 311 of file map.hh.

5.70.2.3 random()

```
void random (
    int rows,
    int cols,
    bool surjective )
```

Random instance.

Parameters

| | |
|-------------------|---------------------------------|
| <i>rows</i> | Number of rows |
| <i>cols</i> | Number of columns |
| <i>surjective</i> | Flag to ensure a surjective map |

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 81 of file map.cc.

5.70.2.4 save()

```
void save (
    std::string path ) const [inline]
```

Save map.

Parameters

| | |
|-------------|------------------|
| <i>path</i> | Path of the file |
|-------------|------------------|

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 318 of file map.hh.

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

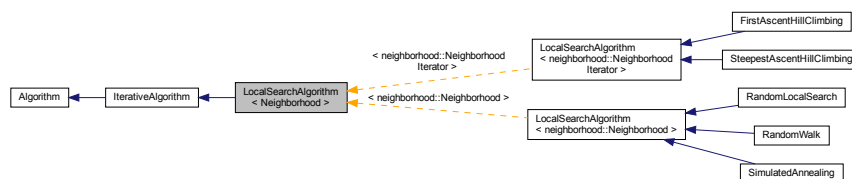
- lib/hnco/maps/map.hh
- lib/hnco/maps/map.cc

5.71 LocalSearchAlgorithm< Neighborhood > Class Template Reference

Local search algorithm.

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

Inheritance diagram for LocalSearchAlgorithm< Neighborhood >:



Public Member Functions

- [LocalSearchAlgorithm](#) (int n, Neighborhood *neighborhood)
Constructor.

Setters

- void [set_random_initialization](#) (bool b)
Set random initialization.
- void [set_starting_point](#) (const [bit_vector_t](#) &x)
Set the starting point.

Protected Member Functions

Loop

- void [init](#) () override
Initialize.

Protected Attributes

- [bit_vector_t _starting_point](#)
Starting point.
- Neighborhood * [_neighborhood](#)
Neighborhood.

Parameters

- bool [_random_initialization](#) = true
Random initialization.

5.71.1 Detailed Description

```
template<class Neighborhood>
class hnco::algorithm::LocalSearchAlgorithm< Neighborhood >
```

Local search algorithm.

Definition at line 33 of file local-search-algorithm.hh.

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

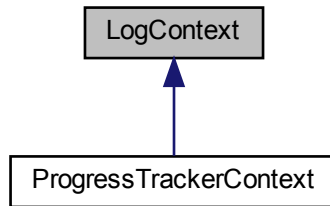
- lib/hnco/algorithms/ls/local-search-algorithm.hh

5.72 LogContext Class Reference

Log context.

```
#include <hnco/logging/log-context.hh>
```

Inheritance diagram for LogContext:



Public Member Functions

- virtual std::string [to_string](#) ()=0
Get context.

5.72.1 Detailed Description

Log context.

A log context gives an algorithm more information about what is going on during optimization than what can be gained through its function. In particular, its function may not be a function controller. Information is provided through a log context in the form of a string.

Definition at line 41 of file log-context.hh.

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

- lib/hnco/logging/log-context.hh

5.73 Logger Class Reference

Logger.

```
#include <hnco/logging/logger.hh>
```

Public Member Functions

- [Logger](#) ()
Default constructor.
- [Logger](#) ([LogContext](#) *context)
Constructor.
- std::ostringstream & [line](#) ()
Get the line.
- virtual [~Logger](#) ()
Destructor.

Static Public Member Functions

- static std::ostream & [stream](#) ()
Get the stream.
- static void [set_stream](#) (std::ostream *stream)
Set the stream.

Private Attributes

- std::ostringstream [_line](#)
Line.

Static Private Attributes

- static std::ostream * [_stream](#) = &std::cout
Output stream.

5.73.1 Detailed Description

Logger.

Simple logger inspired by the Log class published in Dr. Dobb's:

<https://www.drdobbs.com/cpp/logging-in-c/201804215>

Definition at line 43 of file logger.hh.

5.73.2 Constructor & Destructor Documentation

5.73.2.1 Logger()

```
Logger (
    LogContext * context ) [inline]
```

Constructor.

The constructor converts the context to a string which it writes at the beginning of the line.

Parameters

| | |
|----------------|-------------|
| <i>context</i> | Log context |
|----------------|-------------|

Definition at line 69 of file logger.hh.

5.73.2.2 ~Logger()

```
virtual ~Logger ( ) [inline], [virtual]
```

Destructor.

Send the line to the output stream and add an end of line.

Definition at line 81 of file logger.hh.

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

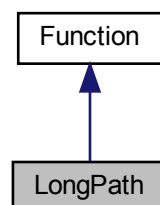
- lib/hnco/logging/logger.hh
- lib/hnco/logging/logger.cc

5.74 LongPath Class Reference

Long path.

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

Inheritance diagram for LongPath:



Public Member Functions

- [LongPath](#) (int bv_size, int prefix_length)
Constructor.
- double [evaluate](#) (const [bit_vector_t](#) &)
Evaluate a bit vector.

Information about the function

- int [get_bv_size](#) () const
Get bit vector size.
- bool [has_known_maximum](#) () const
Check for a known maximum.
- double [get_maximum](#) () const
Get the global maximum.

Private Attributes

- int [_bv_size](#)
Bit vector size.
- int [_prefix_length](#)
Prefix length.

5.74.1 Detailed Description

Long path.

Long paths have been introduced by Jeffrey Horn, David E. Goldberg, and Kalyanmoy Deb. Here we mostly follow the definition given by Thomas Jansen (see references below).

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

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

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

To help with the detection of maximum, we have dropped the constant n^2 whose sole purpose was to make the function non negative.

References:

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

Thomas Jansen, Analyzing Evolutionary Algorithms. Springer, 2013.

Definition at line 62 of file long-path.hh.

5.74.2 Member Function Documentation

5.74.2.1 get_maximum()

```
double get_maximum ( ) const [virtual]
```

Get the global maximum.

Let n be the bit vector size and k the prefix length which must divide n . Then the maximum is $k2^{n/k} - k + 1$.

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Reimplemented from [Function](#).

Definition at line 62 of file long-path.cc.

5.74.2.2 has_known_maximum()

```
bool has_known_maximum ( ) const [virtual]
```

Check for a known maximum.

Let n be the bit vector size and k the prefix length which must divide n .

We have to check that the maximum can be represented exactly as a double, that is, it must be lower or equal to 2^{53} . We are a little bit more conservative with the following test.

If $\log_2(k) + n/k \leq 53$ then returns true else returns false.

Reimplemented from [Function](#).

Definition at line 52 of file long-path.cc.

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

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

5.75 LowerTriangularWalshMoment2 Struct Reference

Lower triangular Walsh moment.

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

Public Member Functions

- [LowerTriangularWalshMoment2](#) (int n)
Constructor.
- void [display](#) (std::ostream &stream)
Display Walsh moment.
- void [init](#) ()
Initialize Walsh moment.
- void [add](#) (const [bit_vector_t](#) &bv)
Add a bit vector to a Walsh moment.
- void [average](#) (int count)
Average each Walsh moment.
- void [update](#) (const [LowerTriangularWalshMoment2](#) &wm, double rate)
Update a Walsh moment.
- void [update](#) (const [LowerTriangularWalshMoment2](#) &wm1, const [LowerTriangularWalshMoment2](#) &wm2, double rate)
Update a Walsh moment.
- void [scaled_difference](#) (double lambda, const [LowerTriangularWalshMoment2](#) &wm1, const [LowerTriangularWalshMoment2](#) &wm2)
Compute a scaled difference between two moments.
- void [bound](#) (double margin)
Bound Walsh moment.
- double [norm_1](#) () const
1-norm of the Walsh moment
- double [norm_2](#) () const
2-norm of the Walsh moment
- double [norm_infinite](#) () const
infinite-norm of the Walsh moment
- double [distance](#) (const [LowerTriangularWalshMoment2](#) &wm) const
distance between the Walsh moment and another Walsh moment

Public Attributes

- std::vector< double > [first_moment](#)
First moment.
- std::vector< std::vector< double > > [second_moment](#)
Second moment.

5.75.1 Detailed Description

Lower triangular Walsh moment.

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

5.75.2 Constructor & Destructor Documentation

5.75.2.1 LowerTriangularWalshMoment2()

```
LowerTriangularWalshMoment2 (
    int n )
```

Constructor.

Parameters

| | |
|----------|--------------------|
| <i>n</i> | Size of bit vector |
|----------|--------------------|

Definition at line 32 of file walsh-moment.cc.

5.75.3 Member Function Documentation

5.75.3.1 bound()

```
void bound (
    double margin )
```

Bound Walsh moment.

Ensure that the distance from each Walsh moment to the -1/1 bounds is greater or equal to the given margin.

Parameters

| | |
|---------------|-------------------------------|
| <i>margin</i> | Distance from the -1/1 bounds |
|---------------|-------------------------------|

Definition at line 161 of file walsh-moment.cc.

5.75.3.2 display()

```
void display (
    std::ostream & stream )
```

Display Walsh moment.

A [LowerTriangularWalshMoment2](#) is displayed as a full symmetric matrix with diagonal entries equal to first moments and off-diagonal entries equal to second moments.

Definition at line 43 of file walsh-moment.cc.

5.75.3.3 scaled_difference()

```
void scaled_difference (
    double lambda,
    const LowerTriangularWalshMoment2 & wm1,
    const LowerTriangularWalshMoment2 & wm2 )
```

Compute a scaled difference between two moments.

This member function implements:

$\text{self} = \text{lambda} * \text{wm1} - \text{wm2}$

It is mostly useful in herding ([Hea](#)).

Parameters

| | |
|---------------|---------------------|
| <i>lambda</i> | Scale |
| <i>wm1</i> | First Walsh moment |
| <i>wm2</i> | Second Walsh moment |

Definition at line 142 of file walsh-moment.cc.

5.75.3.4 update() [1/2]

```
void update (
    const LowerTriangularWalshMoment2 & wm,
    double rate )
```

Update a Walsh moment.

This member function implements:

```
self += rate * (wm1 - self)
```

Parameters

| | |
|-------------|---------------------|
| <i>wm</i> | Target Walsh moment |
| <i>rate</i> | Learning rate |

Postcondition

For all i , $\text{is_in_interval}(\text{first_moment}[i], -1, 1)$

For all $j < i$, $\text{is_in_interval}(\text{second_moment}[i][j], -1, 1)$

Definition at line 104 of file walsh-moment.cc.

5.75.3.5 update() [2/2]

```
void update (
    const LowerTriangularWalshMoment2 & wm1,
    const LowerTriangularWalshMoment2 & wm2,
    double rate )
```

Update a Walsh moment.

This member function implements:

```
self += rate * (wm1 - wm2)
```

The resulting entries are not necessarily those of a Walsh moment, that is

$\text{is_in_interval}(\text{first_moment}[i], -1, 1)$ or

$\text{is_in_interval}(\text{second_moment}[i][j], -1, 1)$

might fail for some i, j .

Parameters

| | |
|-------------|--------------------------------|
| <i>wm1</i> | Target Walsh moment |
| <i>wm2</i> | Walsh moment to move away from |
| <i>rate</i> | Learning rate |

Definition at line 122 of file walsh-moment.cc.

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

- lib/hnco/algorithms/walsh-moment/walsh-moment.hh
- lib/hnco/algorithms/walsh-moment/walsh-moment.cc

5.76 LowerTriangularWalshMoment2GibbsSampler Class Reference

Gibbs sampler with lower triangular Walsh moments.

```
#include <hnco/algorithms/walsh-moment/gibbs-sampler.hh>
```

Public Types

- using [Moment](#) = [LowerTriangularWalshMoment2](#)
Walsh moment type.

Public Member Functions

- [LowerTriangularWalshMoment2GibbsSampler](#) (int n, const [LowerTriangularWalshMoment2](#) &mp)
Constructor.
- void [init](#) ()
Initialize.
- void [update](#) (int i)
Update state.
- void [update_sync](#) ()
Update state synchronously.
- const [bit_vector_t](#) & [get_state](#) ()
Get the state of the Gibbs sampler.

Private Attributes

- const [LowerTriangularWalshMoment2](#) & [_model_parameters](#)
Model parameters.
- [bit_vector_t](#) [_state](#)
State of the Gibbs sampler.
- [pv_t](#) [_pv](#)
Probability vector for synchronous Gibbs sampling.

5.76.1 Detailed Description

Gibbs sampler with lower triangular Walsh moments.

Definition at line 38 of file gibbs-sampler.hh.

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

- lib/hnco/algorithms/walsh-moment/gibbs-sampler.hh
- lib/hnco/algorithms/walsh-moment/gibbs-sampler.cc

5.77 LowerTriangularWalshMoment2Herding Class Reference

Herding with lower triangular Walsh moment.

```
#include <hnco/algorithms/walsh-moment/herding.hh>
```

Public Types

- using [Moment](#) = [LowerTriangularWalshMoment2](#)
Walsh moment type.

Public Member Functions

- [LowerTriangularWalshMoment2Herding](#) (int n)
Constructor.
- void [init](#) ()
Initialization.
- void [sample](#) (const [LowerTriangularWalshMoment2](#) &target, [bit_vector_t](#) &x)
Sample a bit vector.
- double [error](#) (const [LowerTriangularWalshMoment2](#) &target)
Compute the error.

Getters

- const [LowerTriangularWalshMoment2](#) & [get_delta](#) () const
Get delta.

Setters

- void [set_randomize_bit_order](#) (bool x)
Randomize bit order.

Protected Attributes

- [LowerTriangularWalshMoment2 _delta](#)
Delta moment.
- [LowerTriangularWalshMoment2 _count](#)
Counter moment.
- [LowerTriangularWalshMoment2 _error](#)
Error moment.
- [permutation_t _permutation](#)
Permutation.
- [int _time](#)
Time.

Parameters

- [bool _randomize_bit_order](#) = false
Randomize bit order.

5.77.1 Detailed Description

Herdng with lower triangular Walsh moment.

Definition at line 37 of file herding.hh.

5.77.2 Constructor & Destructor Documentation

5.77.2.1 LowerTriangularWalshMoment2Herdng()

```
LowerTriangularWalshMoment2Herdng (
    int n ) [inline]
```

Constructor.

Parameters

| | |
|----------|---------------------|
| <i>n</i> | Size of bit vectors |
|----------|---------------------|

Definition at line 74 of file herding.hh.

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

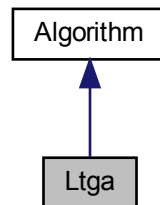
- lib/hnco/algorithms/walsh-moment/herding.hh
- lib/hnco/algorithms/walsh-moment/herding.cc

5.78 Ltga Class Reference

Linkage Tree Genetic Algorithm.

```
#include <hnco/algorithms/fast-efficient-p3/ltga.hh>
```

Inheritance diagram for Ltga:



Public Member Functions

- [Ltga](#) (int n)
Constructor.
- [~Ltga](#) ()
Destructor.
- void [maximize](#) (const std::vector< [function::Function](#) * > &functions)
Maximize.
- void [finalize](#) ()
Finalize.
- void [set_population_size](#) (int n)
Set population size.

Private Attributes

- [Implementation](#) * [_pimpl](#)
Pointer to implementation.
- int [_population_size](#) = 10
Population size.

Additional Inherited Members

5.78.1 Detailed Description

Linkage Tree Genetic Algorithm.

[Implementation](#) of the Linkage Tree Genetic Algorithm Designed to match the variant in the paper: "Hierarchical problem solving with the linkage tree genetic algorithm" by D. Thierens and P. A. N. Bosman

Author: Brian W. Goldman

Integrated into HNCO by Arnaud Berny

Definition at line 47 of file ltga.hh.

5.78.2 Member Data Documentation

5.78.2.1 `_pimpl`

`Implementation* _pimpl [private]`

Pointer to implementation.

The main motivation for this pattern is to avoid including declarations from `fast_efficient_p3` into the global namespace.

A raw pointer is used instead of a `unique_ptr` because the latter will not compile with `pybind11`.

Definition at line 57 of file `ltga.hh`.

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

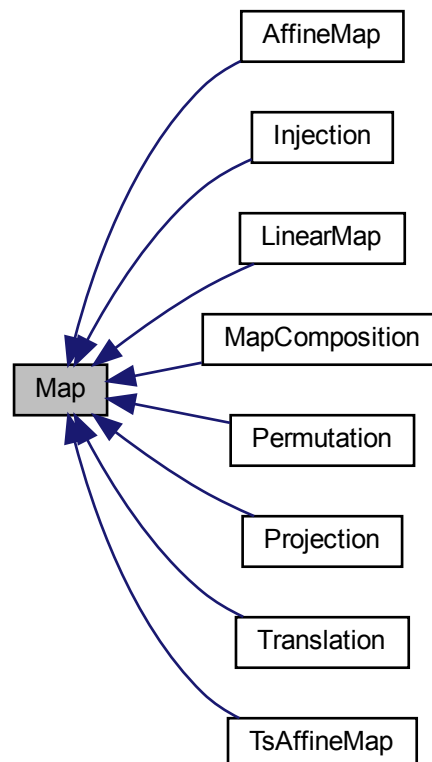
- `lib/hnco/algorithms/fast-efficient-p3/ltga.hh`
- `lib/hnco/algorithms/fast-efficient-p3/ltga.cc`

5.79 Map Class Reference

Map

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

Inheritance diagram for Map:



Public Member Functions

- virtual `~Map ()`
Destructor.
- virtual void `map` (const `bit_vector_t` &input, `bit_vector_t` &output)=0
Map
- virtual int `get_input_size` () const =0
Get input size.
- virtual int `get_output_size` () const =0
Get output size.
- virtual bool `is_surjective` () const
Check for surjective map.
- virtual void `display` (std::ostream &stream) const
Display.

5.79.1 Detailed Description

Map

Definition at line 46 of file map.hh.

5.79.2 Member Function Documentation

5.79.2.1 `is_surjective()`

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

Check for surjective map.

Returns

false

Reimplemented in [TsAffineMap](#), [Projection](#), [Injection](#), [MapComposition](#), [AffineMap](#), [LinearMap](#), [Permutation](#), and [Translation](#).

Definition at line 66 of file map.hh.

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

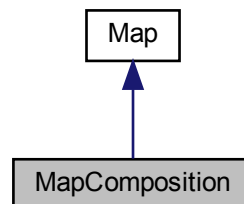
- lib/hnco/maps/map.hh

5.80 MapComposition Class Reference

Map composition.

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

Inheritance diagram for MapComposition:



Public Member Functions

- [MapComposition](#) ()
Default constructor.
- [MapComposition](#) ([Map](#) *outer, [Map](#) *inner)
Constructor.
- void [map](#) (const [bit_vector_t](#) &input, [bit_vector_t](#) &output) override
Map
- int [get_input_size](#) () const override
Get input size.
- int [get_output_size](#) () const override
Get output size.
- bool [is_surjective](#) () const override
Check for surjective map.

Private Attributes

- [Map](#) * [_outer](#)
Outer map.
- [Map](#) * [_inner](#)
Inner map.
- [bit_vector_t](#) [_bv](#)
Temporary bit vector.

5.80.1 Detailed Description

Map composition.

The resulting composition f is defined for all bit vector x by $f(x) = \text{outer}(\text{inner}(x))$.

Definition at line 424 of file map.hh.

5.80.2 Constructor & Destructor Documentation

5.80.2.1 MapComposition()

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

Constructor.

Parameters

| | |
|--------------|-----------|
| <i>outer</i> | outer map |
| <i>inner</i> | inner map |

Precondition

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

Definition at line 448 of file map.hh.

5.80.3 Member Function Documentation

5.80.3.1 is_surjective()

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

Check for surjective map.

Returns

true if both maps are surjective

Reimplemented from [Map](#).

Definition at line 472 of file map.hh.

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

- lib/hnco/maps/map.hh

5.81 MapgenOptions Class Reference

Command line options for mapgen.

```
#include <mapgen-options.hh>
```

Public Member Functions

- [MapgenOptions](#) (int argc, char *argv[])
Constructor.
- int [get_input_size](#) () const
Get input_size.
- void [set_input_size](#) (int x)
Set input_size.
- bool [set_input_size](#) () const
Get set-flag for input_size.
- int [get_map](#) () const
Get map.
- void [set_map](#) (int x)
Set map.
- bool [set_map](#) () const
Get set-flag for map.
- int [get_output_size](#) () const
Get output_size.
- void [set_output_size](#) (int x)
Set output_size.
- bool [set_output_size](#) () const
Get set-flag for output_size.
- std::string [get_path](#) () const
Get path.
- void [set_path](#) (std::string x)
Set path.
- bool [set_path](#) () const
Get set-flag for path.
- int [get_seed](#) () const
Get seed.
- void [set_seed](#) (int x)
Set seed.
- bool [set_seed](#) () const
Get set-flag for seed.
- int [get_ts_length](#) () const
Get ts_length.
- void [set_ts_length](#) (int x)
Set ts_length.
- bool [set_ts_length](#) () const
Get set-flag for ts_length.
- int [get_ts_sampling_mode](#) () const
Get ts_sampling_mode.
- void [set_ts_sampling_mode](#) (int x)
Set ts_sampling_mode.

- bool [set_ts_sampling_mode](#) () const
Get set-flag for ts_sampling_mode.
- bool [with_surjective](#) () const
Get surjective.
- void [set_surjective](#) ()
Set surjective.

Private Member Functions

- void [print_help](#) (std::ostream &stream) const
Print help message.
- void [print_version](#) (std::ostream &stream) const
Print version.

Private Attributes

- std::string [_exec_name](#)
Name of the executable.
- std::string [_version](#)
Name Version.
- int [_input_size](#)
Input bit vector size.
- bool [_opt_input_size](#)
- int [_map](#)
Type of map.
- bool [_opt_map](#)
- int [_output_size](#)
Output bit vector size.
- bool [_opt_output_size](#)
- std::string [_path](#)
Path (relative or absolute) of a map file.
- bool [_opt_path](#)
- int [_seed](#)
Seed for the random number generator.
- bool [_opt_seed](#)
- int [_ts_length](#)
Transvection sequence length.
- bool [_opt_ts_length](#)
- int [_ts_sampling_mode](#)
Transvection sequence sampling mode.
- bool [_opt_ts_sampling_mode](#)
- bool [_surjective](#)
Ensure that the sampled linear or affine map is surjective.

Friends

- std::ostream & [operator<<](#) (std::ostream &, const [MapgenOptions](#) &)
Print a header containing the parameter values.

5.81.1 Detailed Description

Command line options for mapgen.

Definition at line 11 of file mapgen-options.hh.

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

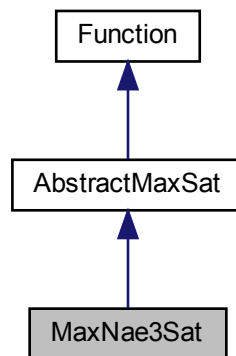
- app/mapgen-options.hh
- app/mapgen-options.cc

5.82 MaxNae3Sat Class Reference

Max not-all-equal 3SAT.

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

Inheritance diagram for MaxNae3Sat:



Public Member Functions

- [MaxNae3Sat](#) ()
Default constructor.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- void [load](#) (std::string path)
Load instance.

Additional Inherited Members

5.82.1 Detailed Description

Max not-all-equal 3SAT.

Reference:

Christos M. Papadimitriou. 1994. Computational complexity. Addison-Wesley, Reading, Massachusetts.

Definition at line 162 of file max-sat.hh.

5.82.2 Member Function Documentation

5.82.2.1 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to load |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 177 of file max-sat.hh.

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

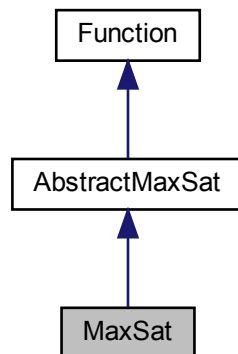
- lib/hnco/functions/collection/max-sat.hh
- lib/hnco/functions/collection/max-sat.cc

5.83 MaxSat Class Reference

MAX-SAT.

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

Inheritance diagram for MaxSat:



Public Member Functions

- [MaxSat](#) ()
Default constructor.
- void [random](#) (int n, int k, int c)
Random instance.
- void [random](#) (const [bit_vector_t](#) &solution, int k, int c)
Random instance with satisfiable expression.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Additional Inherited Members

5.83.1 Detailed Description

MAX-SAT.

Reference:

Christos M. Papadimitriou. 1994. Computational complexity. Addison-Wesley, Reading, Massachusetts.

Definition at line 119 of file max-sat.hh.

5.83.2 Member Function Documentation

5.83.2.1 random() [1/2]

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

Random instance with satisfiable expression.

Warning

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

Parameters

| | |
|-----------------|-------------------------------|
| <i>solution</i> | Solution |
| <i>k</i> | Number of literals per clause |
| <i>c</i> | Number of clauses |

Definition at line 218 of file max-sat.cc.

5.83.2.2 random() [2/2]

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

Random instance.

Parameters

| | |
|----------|-------------------------------|
| <i>n</i> | Size of bit vectors |
| <i>k</i> | Number of literals per clause |
| <i>c</i> | Number of clauses |

Definition at line 190 of file max-sat.cc.

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

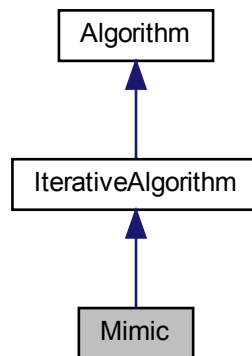
- lib/hnco/functions/collection/max-sat.hh
- lib/hnco/functions/collection/max-sat.cc

5.84 Mimic Class Reference

Mutual information maximizing input clustering.


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

Inheritance diagram for Mimic:



Public Member Functions

- [Mimic](#) (int n, int population_size)
Constructor.

Setters

- void [set_selection_size](#) (int selection_size)
Set the selection size.

Protected Member Functions

- void [sample](#) ([bit_vector_t](#) &bv)
Sample a bit vector.
- void [compute_conditional_entropy](#) (int index)
Compute conditional entropy.
- void [update_model](#) ()
Update model.

Loop

- void [init](#) () override
Initialize.
- void [iterate](#) () override
Single iteration.

Protected Attributes

- [Population _population](#)
Population.
- [permutation_t _permutation](#)
Permutation.
- `std::array< pv_t, 2 > _parameters`
Model parameters.
- [pv_t _mean](#)
Mean of selected bit vectors.
- `std::vector< double > _entropies`
Conditional entropies.
- `std::array< std::array< int, 2 >, 2 > _table`
Contingency table.
- `double _lower_bound`
Lower bound of probability.
- `double _upper_bound`
Upper bound of probability.

Parameters

- `int _selection_size`
Selection size.

5.84.1 Detailed Description

Mutual information maximizing input clustering.

This implementation differs from the algorithm described in the reference below in that it constrains all probabilities (marginal and conditional) to stay away from the values 0 and 1 by a fixed margin equal to $1/n$, as usually done in algorithms such as [Pbil](#) or [Umda](#).

Reference:

Jeremy S. De Bonet and Charles L. Isbell and Jr. and Paul Viola, MIMIC: Finding Optima by Estimating Probability Densities, in Advances in Neural Information Processing Systems, 1996, MIT Press.

Definition at line 52 of file `mimic.hh`.

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

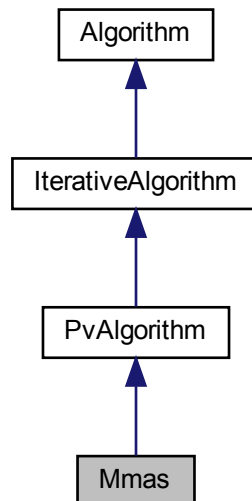
- `lib/hnco/algorithms/mimic.hh`
- `lib/hnco/algorithms/mimic.cc`

5.85 Mmas Class Reference

Max-min ant system.

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

Inheritance diagram for Mmas:



Public Member Functions

- [Mmas](#) (int n)
Constructor.

Setters

- void [set_compare](#) (std::function< bool(double, double)> x)
Set the binary operator for comparing evaluations.
- void [set_learning_rate](#) (double x)
Set the learning rate.

Protected Member Functions

Loop

- void [init](#) () override
Initialize.
- void [iterate](#) () override
Single iteration.

Protected Attributes

- `bit_vector_t_x`
Candidate solution.

Parameters

- `std::function< bool(double, double)> _compare = std::greater_equal<double>()`
Binary operator for comparing evaluations.
- `double _learning_rate = 1e-3`
Learning rate.

5.85.1 Detailed Description

Max-min ant system.

Reference:

Thomas Stützle and Holger H. Hoos. 2000. MAX-MIN Ant System. *Future Generation Computer Systems* 16, 8 (2000), 889–914.

Definition at line 42 of file `mmas.hh`.

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

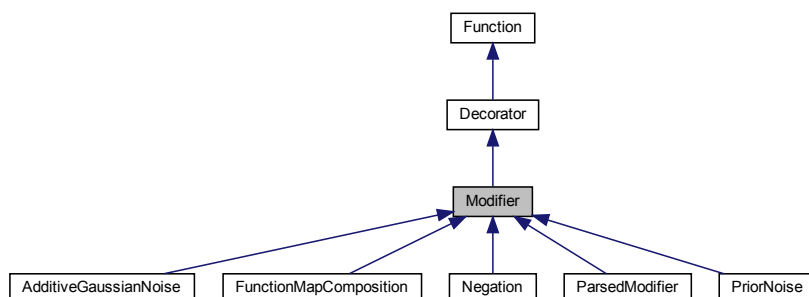
- `lib/hnco/algorithms/pv/mmas.hh`
- `lib/hnco/algorithms/pv/mmas.cc`

5.86 Modifier Class Reference

[Function](#) modifier.

```
#include <hnco/functions/modifiers/modifier.hh>
```

Inheritance diagram for `Modifier`:



Public Member Functions

- [Modifier](#) ([Function](#) *function)
Constructor.

Additional Inherited Members

5.86.1 Detailed Description

[Function](#) modifier.

Definition at line 39 of file modifier.hh.

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

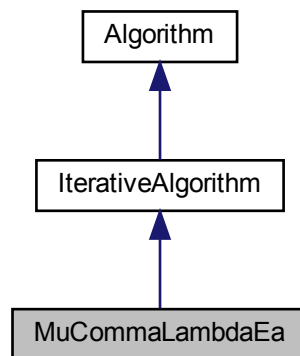
- lib/hnco/functions/modifiers/modifier.hh

5.87 MuCommaLambdaEa Class Reference

(mu, lambda) EA.

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

Inheritance diagram for MuCommaLambdaEa:



Public Member Functions

- [MuCommaLambdaEa](#) (int n, int mu, int lambda)
Constructor.

Setters

- void [set_mutation_rate](#) (double p)
Set the mutation rate.
- void [set_allow_no_mutation](#) (bool b)
Set the flag _allow_no_mutation.

Protected Member Functions

Loop

- void `init` () override
Initialize.
- void `iterate` () override
Single iteration.

Protected Attributes

- `Population _parents`
Parents.
- `Population _offsprings`
Offsprings.
- `CommaSelection _comma_selection`
Comma selection.
- `neighborhood::StandardBitMutation _mutation`
Mutation operator.
- `std::uniform_int_distribution< int > _select_parent`
Select parent.

Parameters

- double `_mutation_rate`
Mutation rate.
- bool `_allow_no_mutation` = false
Allow no mutation.

5.87.1 Detailed Description

(mu, lambda) EA.

Reference:

Thomas Jansen, Analyzing Evolutionary Algorithms. Springer, 2013.

Definition at line 43 of file mu-comma-lambda-ea.hh.

5.87.2 Constructor & Destructor Documentation

5.87.2.1 MuCommaLambdaEa()

```
MuCommaLambdaEa (
    int n,
    int mu,
    int lambda ) [inline]
```

Constructor.

Parameters

| | |
|---------------|---------------------------|
| <i>n</i> | Size of bit vectors |
| <i>mu</i> | Parent population size |
| <i>lambda</i> | Offspring population size |

Definition at line 94 of file mu-commma-lambda-ea.hh.

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

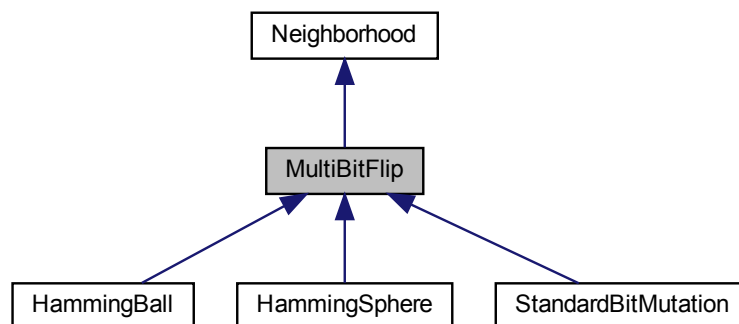
- lib/hnco/algorithms/ea/mu-commma-lambda-ea.hh
- lib/hnco/algorithms/ea/mu-commma-lambda-ea.cc

5.88 MultiBitFlip Class Reference

Multi bit flip.

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

Inheritance diagram for MultiBitFlip:



Public Member Functions

- [MultiBitFlip](#) (int n)
Constructor.

Protected Member Functions

- void [bernoulli_trials](#) (int k)
Sample a given number of bits using Bernoulli trials.
- void [rejection_sampling](#) (int k)
Sample a given number of bits using rejection sampling.

Additional Inherited Members

5.88.1 Detailed Description

Multi bit flip.

Definition at line 185 of file neighborhood.hh.

5.88.2 Constructor & Destructor Documentation

5.88.2.1 MultiBitFlip()

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

Constructor.

Parameters

| | |
|-----|---------------------|
| n | Size of bit vectors |
|-----|---------------------|

Definition at line 208 of file neighborhood.hh.

5.88.3 Member Function Documentation

5.88.3.1 bernoulli_trials()

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

Sample a given number of bits using Bernoulli trials.

Parameters

| | |
|-----|--------------------------|
| k | Number of bits to sample |
|-----|--------------------------|

Definition at line 34 of file neighborhood.cc.

5.88.3.2 rejection_sampling()

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

Sample a given number of bits using rejection sampling.

Parameters

| | |
|----------|--------------------------|
| <i>k</i> | Number of bits to sample |
|----------|--------------------------|

Definition at line 52 of file neighborhood.cc.

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

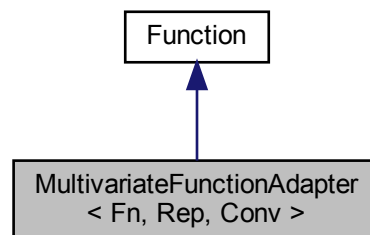
- lib/hnco/neighborhoods/neighborhood.hh
- lib/hnco/neighborhoods/neighborhood.cc

5.89 MultivariateFunctionAdapter< Fn, Rep, Conv > Class Template Reference

Multivariate function adapter.

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

Inheritance diagram for MultivariateFunctionAdapter< Fn, Rep, Conv >:



Public Member Functions

- [MultivariateFunctionAdapter](#) (Fn *fn, std::vector< Rep > reps)
Constructor.

Information about the function

- int [get_bv_size](#) () const override

Get bit vector size.

Evaluation

- double `evaluate` (const `bit_vector_t` &bv) override
Evaluate.

Display

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

Private Member Functions

- void `unpack` (const `bit_vector_t` &bv)
Unpack a bit vector into values.

Private Attributes

- Fn * `_function`
Multivariate function.
- std::vector< Rep > `_representations`
Representations.
- std::vector< typename Rep::domain_type > `_variables`
Variables.
- Conv `_converter`
Converter from codomain to double.

5.89.1 Detailed Description

```
template<class Fn, class Rep, class Conv>
class hnco::function::MultivariateFunctionAdapter< Fn, Rep, Conv >
```

Multivariate function adapter.

The purpose of this class is to build a regular hnco function from an arbitrary multivariate function. This is achieved using a composition:

- Representations (Rep): hypercube -> domain
- Multivariate function (Fn): product of domains -> codomain
- Converter (Conv): codomain -> double

Definition at line 46 of file multivariate-function-adapter.hh.

5.89.2 Constructor & Destructor Documentation

5.89.2.1 MultivariateFunctionAdapter()

```
MultivariateFunctionAdapter (
    Fn * fn,
    std::vector< Rep > reps ) [inline]
```

Constructor.

Parameters

| | |
|-------------|-----------------------|
| <i>fn</i> | Multivariate function |
| <i>reps</i> | Representations |

Definition at line 86 of file multivariate-function-adapter.hh.

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

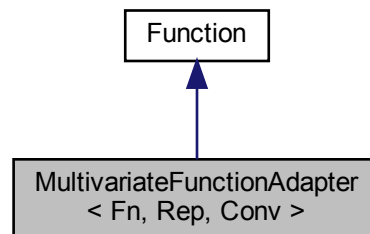
- lib/hnco/functions/multivariate-function-adapter.hh

5.90 MultivariateFunctionAdapter< Fn, Rep, Conv > Class Template Reference

Multivariate function adapter.

```
#include <hnco/multiobjective/functions/multivariate-function-adapter.hh>
```

Inheritance diagram for MultivariateFunctionAdapter< Fn, Rep, Conv >:



Public Member Functions

- [MultivariateFunctionAdapter](#) (Fn *fn, std::vector< Rep > reps)
Constructor.

Information about the function

- int [get_bv_size](#) () const override
Get bit vector size.
- int [get_output_size](#) () const override
Get output size (number of objectives)

Evaluation

- void [evaluate](#) (const [bit_vector_t](#) &bv, [value_t](#) &value) override
Evaluate.

Display

- void [display](#) (std::ostream &stream) const override
Display.
- void [describe](#) (const [bit_vector_t](#) &bv, std::ostream &stream) override
Describe a bit vector.

Private Member Functions

- void `unpack` (const `bit_vector_t` &bv)
Unpack a bit vector into variables.

Private Attributes

- Fn * `_function`
Multivariate function.
- `std::vector< Rep > _representations`
Representations.
- `std::vector< typename Rep::domain_type > _variables`
Variables.
- `std::vector< typename Fn::codomain_type > _codomain_value`
Codomain value.
- Conv `_converter`
Converter from codomain to double.

5.90.1 Detailed Description

```
template<class Fn, class Rep, class Conv>
class hnco::multiobjective::function::MultivariateFunctionAdapter< Fn, Rep, Conv >
```

Multivariate function adapter.

The purpose of this class is to build a regular hnco function from an arbitrary multivariate function. This is achieved using a composition:

- Representations (Rep): hypercube -> domain
- Multivariate function (Fn): product of domains -> product of codomains
- Converter (Conv): codomain -> double

Definition at line 48 of file multivariate-function-adapter.hh.

5.90.2 Constructor & Destructor Documentation

5.90.2.1 MultivariateFunctionAdapter()

```
MultivariateFunctionAdapter (
    Fn * fn,
    std::vector< Rep > reps ) [inline]
```

Constructor.

Parameters

| | |
|-------------|-----------------------|
| <i>fn</i> | Multivariate function |
| <i>reps</i> | Representations |

Definition at line 91 of file multivariate-function-adapter.hh.

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

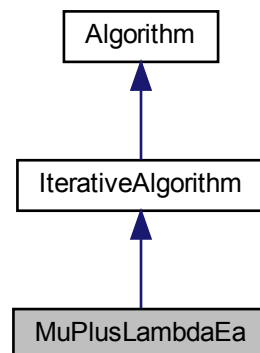
- lib/hnco/multiobjective/functions/multivariate-function-adapter.hh

5.91 MuPlusLambdaEa Class Reference

(mu+lambda) EA.

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

Inheritance diagram for MuPlusLambdaEa:



Public Member Functions

- [MuPlusLambdaEa](#) (int n, int mu, int lambda)
Constructor.

Setters

- void [set_mutation_rate](#) (double p)
Set the mutation rate.
- void [set_allow_no_mutation](#) (bool b)
Set the flag _allow_no_mutation.

Protected Member Functions

Loop

- void `init` () override
Initialize.
- void `iterate` () override
Single iteration.

Protected Attributes

- `Population _parents`
Parents.
- `Population _offsprings`
Offsprings.
- `PlusSelection _plus_selection`
Plus selection.
- `neighborhood::StandardBitMutation _mutation`
Mutation operator.
- `std::uniform_int_distribution< int > _select_parent`
Select parent.

Parameters

- double `_mutation_rate`
Mutation rate.
- bool `_allow_no_mutation` = false
Allow no mutation.

5.91.1 Detailed Description

(mu+lambda) EA.

Reference:

Thomas Jansen, Analyzing Evolutionary Algorithms. Springer, 2013.

Definition at line 43 of file mu-plus-lambda-ea.hh.

5.91.2 Constructor & Destructor Documentation

5.91.2.1 MuPlusLambdaEa()

```
MuPlusLambdaEa (
    int n,
    int mu,
    int lambda ) [inline]
```

Constructor.

Parameters

| | |
|---------------|---------------------------|
| <i>n</i> | Size of bit vectors |
| <i>mu</i> | Parent population size |
| <i>lambda</i> | Offspring population size |

Definition at line 94 of file mu-plus-lambda-ea.hh.

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

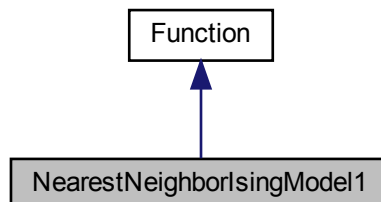
- lib/hnco/algorithms/ea/mu-plus-lambda-ea.hh
- lib/hnco/algorithms/ea/mu-plus-lambda-ea.cc

5.92 NearestNeighborIsingModel1 Class Reference

Nearest neighbor Ising model in one dimension.

```
#include <hnco/functions/collection/ising/nearest-neighbor-ising-model-1.↵
hh>
```

Inheritance diagram for NearestNeighborIsingModel1:



Public Member Functions

- [NearestNeighborIsingModel1](#) ()
Constructor.
- void [set_periodic_boundary_conditions](#) (bool x)
Set periodic boundary conditions.

Instance generators

- template<class CouplingGen , class FieldGen >
void [generate](#) (int n, CouplingGen coupling_gen, FieldGen field_gen)
Instance generator.
- void [random](#) (int n)
Random instance.

Load and save instance

- void [load](#) (std::string path)
Load instance.
- void [save](#) (std::string path) const
Save instance.

Evaluation

- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- double [evaluate_incrementally](#) (const [bit_vector_t](#) &x, double v, const [sparse_bit_vector_t](#) &flipped_bits) override
Incrementally evaluate a bit vector.

Information about the function

- int [get_bv_size](#) () const override
Get bit vector size.
- bool [provides_incremental_evaluation](#) () const override
Check whether the function provides incremental evaluation.
- void [display](#) (std::ostream &stream) const override
Display.

Private Member Functions

- template<class Archive >
void [save](#) (Archive &ar, const unsigned int version) const
Save.
- template<class Archive >
void [load](#) (Archive &ar, const unsigned int version)
Load.
- void [resize](#) (int n)
Resize data structures.

Private Attributes

- std::vector< double > [_coupling](#)
Coupling with nearest neighbor to the right.
- std::vector< double > [_field](#)
External field.
- [bit_vector_t](#) [_flipped_bits](#)
Flipped bits.
- bool [_periodic_boundary_conditions](#) = false
Periodic boundary conditions.

Friends

- class [boost::serialization::access](#)

5.92.1 Detailed Description

Nearest neighbor Ising model in one dimension.

Its expression is of the form

$$f(x) = \sum_i J_{i,i+1}(1 - 2x_i)(1 - 2x_{i+1}) + \sum_i h_i(1 - 2x_i)$$

or equivalently

$$f(x) = \sum_i J_{i,i+1}(-1)^{x_i+x_{i+1}} + \sum_i h_i(-1)^{x_i}$$

where $J_{i,i+1}$ is the interaction between adjacent sites i and $i+1$ and h_i is the external magnetic field interacting with site i .

In the case of periodic boundary conditions, the sum $i + 1$ is mod n .

Since we are maximizing f or minimizing $-f$, the expression of f is compatible with what can be found in physics textbooks.

It should be noted that such an Ising model can be represented by a Walsh expansion of degree 2, that is [WalshExpansion2](#).

Reference: https://en.wikipedia.org/wiki/Ising_model

Definition at line 63 of file nearest-neighbor-ising-model-1.hh.

5.92.2 Member Function Documentation

5.92.2.1 evaluate()

```
double evaluate (
    const bit_vector_t & s ) [override], [virtual]
```

Evaluate a bit vector.

Complexity: $O(n)$

Implements [Function](#).

Definition at line 44 of file nearest-neighbor-ising-model-1.cc.

5.92.2.2 generate()

```
void generate (
    int n,
    CouplingGen coupling_gen,
    FieldGen field_gen ) [inline]
```

Instance generator.

Parameters

| | |
|---------------------|--------------------------|
| <i>n</i> | Size of bit vectors |
| <i>coupling_gen</i> | Coupling generator |
| <i>field_gen</i> | External field generator |

Definition at line 124 of file nearest-neighbor-ising-model-1.hh.

5.92.2.3 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to load |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 158 of file nearest-neighbor-ising-model-1.hh.

5.92.2.4 provides_incremental_evaluation()

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

Check whether the function provides incremental evaluation.

Returns

true

Reimplemented from [Function](#).

Definition at line 199 of file nearest-neighbor-ising-model-1.hh.

5.92.2.5 random()

```
void random (
    int n ) [inline]
```

Random instance.

The weights are sampled from the normal distribution.

Parameters

| | |
|----------|--------------------|
| <i>n</i> | Size of bit vector |
|----------|--------------------|

Definition at line 140 of file nearest-neighbor-ising-model-1.hh.

5.92.2.6 save()

```
void save (
    std::string path ) const [inline]
```

Save instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to save |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 165 of file nearest-neighbor-ising-model-1.hh.

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

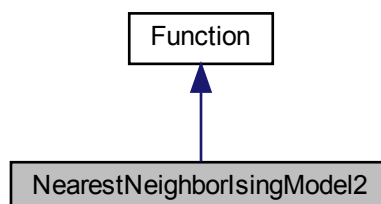
- lib/hnco/functions/collection/ising/nearest-neighbor-ising-model-1.hh
- lib/hnco/functions/collection/ising/nearest-neighbor-ising-model-1.cc

5.93 NearestNeighborIsingModel2 Class Reference

Nearest neighbor Ising model in two dimensions.

```
#include <hnco/functions/collection/ising/nearest-neighbor-ising-model-2.↵
hh>
```

Inheritance diagram for NearestNeighborIsingModel2:



Public Member Functions

- [NearestNeighborIsingModel2](#) ()
Constructor.
- void [set_periodic_boundary_conditions](#) (bool x)
Set periodic boundary conditions.

Instance generators

- template<class CouplingGen , class FieldGen >
void [generate](#) (int num_rows, int num_columns, CouplingGen coupling_gen, FieldGen field_gen)
Instance generator.
- void [random](#) (int num_rows, int num_columns)
Random instance.

Load and save instance

- void [load](#) (std::string path)
Load instance.
- void [save](#) (std::string path) const
Save instance.

Evaluation

- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- double [evaluate_incrementally](#) (const [bit_vector_t](#) &x, double v, const [sparse_bit_vector_t](#) &flipped_bits) override
Incrementally evaluate a bit vector.

Information about the function

- int [get_bv_size](#) () const override
Get bit vector size.
- bool [provides_incremental_evaluation](#) () const override
Check whether the function provides incremental evaluation.
- void [display](#) (std::ostream &stream) const override
Display.

Private Member Functions

- template<class Archive >
void [save](#) (Archive &ar, const unsigned int version) const
Save.
- template<class Archive >
void [load](#) (Archive &ar, const unsigned int version)
Load.
- void [resize](#) (int num_rows, int num_columns)
Resize data structures.

Private Attributes

- `std::vector< std::vector< double > > _coupling_right`
Coupling with nearest neighbor to the right.
- `std::vector< std::vector< double > > _coupling_below`
Coupling with nearest neighbor below.
- `std::vector< std::vector< double > > _field`
External field.
- `bit_vector_t _flipped_bits`
Flipped bits.
- `bool _periodic_boundary_conditions = false`
Periodic boundary conditions.

Friends

- class `boost::serialization::access`

5.93.1 Detailed Description

Nearest neighbor Ising model in two dimensions.

We are considering a rectangular lattice in which each site has (at most) four neighbors (left, right, above, below).

The expression of the function is of the form

$$f(x) = \sum_{(i,j)} J_{ij}(1 - 2x_i)(1 - 2x_j) + \sum_i h_i(1 - 2x_i)$$

or equivalently

$$f(x) = \sum_{(i,j)} J_{ij}(-1)^{x_i+x_j} + \sum_i h_i(-1)^{x_i}$$

where the first sum is over adjacent sites (i, j), J_{ij} is the interaction between adjacent sites i and j, and h_i is the external magnetic field interacting with site i.

Since we are maximizing f or minimizing -f, the expression of f is compatible with what can be found in physics textbooks.

It should be noted that such an Ising model can be represented by a Walsh expansion of degree 2, that is [WalshExpansion2](#).

Reference: https://en.wikipedia.org/wiki/Ising_model

Definition at line 65 of file nearest-neighbor-ising-model-2.hh.

5.93.2 Member Function Documentation

5.93.2.1 evaluate()

```
double evaluate (
    const bit_vector_t & s ) [override], [virtual]
```

Evaluate a bit vector.

Complexity: $O(n)$

Implements [Function](#).

Definition at line 47 of file nearest-neighbor-ising-model-2.cc.

5.93.2.2 generate()

```
void generate (
    int num_rows,
    int num_columns,
    CouplingGen coupling_gen,
    FieldGen field_gen ) [inline]
```

Instance generator.

Parameters

| | |
|---------------------|--------------------------|
| <i>num_rows</i> | Number of rows |
| <i>num_columns</i> | Number of columns |
| <i>coupling_gen</i> | Coupling generator |
| <i>field_gen</i> | External field generator |

Definition at line 132 of file nearest-neighbor-ising-model-2.hh.

5.93.2.3 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to load |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 170 of file nearest-neighbor-ising-model-2.hh.

5.93.2.4 provides_incremental_evaluation()

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

Check whether the function provides incremental evaluation.

Returns

true

Reimplemented from [Function](#).

Definition at line 216 of file nearest-neighbor-ising-model-2.hh.

5.93.2.5 random()

```
void random (
    int num_rows,
    int num_columns ) [inline]
```

Random instance.

The weights are sampled from the normal distribution.

Parameters

| | |
|--------------------|-------------------|
| <i>num_rows</i> | Number of rows |
| <i>num_columns</i> | Number of columns |

Definition at line 152 of file nearest-neighbor-ising-model-2.hh.

5.93.2.6 save()

```
void save (
    std::string path ) const [inline]
```

Save instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to save |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 177 of file nearest-neighbor-ising-model-2.hh.

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

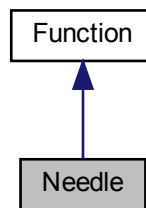
- lib/hnco/functions/collection/ising/nearest-neighbor-ising-model-2.hh
- lib/hnco/functions/collection/ising/nearest-neighbor-ising-model-2.cc

5.94 Needle Class Reference

Needle in a haystack.

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

Inheritance diagram for Needle:



Public Member Functions

- [Needle](#) (int bv_size)
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- bool [has_known_maximum](#) () const override
Check for a known maximum.
- double [get_maximum](#) () const override
Get the global maximum.

Private Attributes

- int [_bv_size](#)
Bit vector size.

5.94.1 Detailed Description

Needle in a haystack.

Reference:

Thomas Jansen, Analyzing Evolutionary Algorithms. Springer, 2013.

Definition at line 135 of file theory.hh.

5.94.2 Member Function Documentation

5.94.2.1 `get_maximum()`

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

Get the global maximum.

Returns

1

Reimplemented from [Function](#).

Definition at line 158 of file theory.hh.

5.94.2.2 `has_known_maximum()`

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

Check for a known maximum.

Returns

true

Reimplemented from [Function](#).

Definition at line 154 of file theory.hh.

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

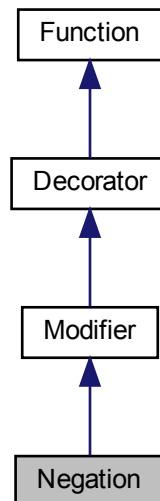
- lib/hnco/functions/collection/theory.hh
- lib/hnco/functions/collection/theory.cc

5.95 Negation Class Reference

[Negation](#).

```
#include <hnco/functions/modifiers/modifier.hh>
```

Inheritance diagram for Negation:



Public Member Functions

- [Negation](#) ([Function](#) *function)
Constructor.

Information about the function

- int [get_bv_size](#) () const override
Get bit vector size.
- bool [provides_incremental_evaluation](#) () const override
Check whether the function provides incremental evaluation.

Evaluation

- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- double [evaluate_incrementally](#) (const [bit_vector_t](#) &x, double value, const [hnco::sparse_bit_vector_t](#) &flipped_bits) override
Incrementally evaluate a bit vector.

Additional Inherited Members

5.95.1 Detailed Description

[Negation](#).

Use cases:

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

Definition at line 60 of file modifier.hh.

5.95.2 Member Function Documentation

5.95.2.1 `provides_incremental_evaluation()`

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

Check whether the function provides incremental evaluation.

Returns

true

Reimplemented from [Function](#).

Definition at line 79 of file modifier.hh.

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

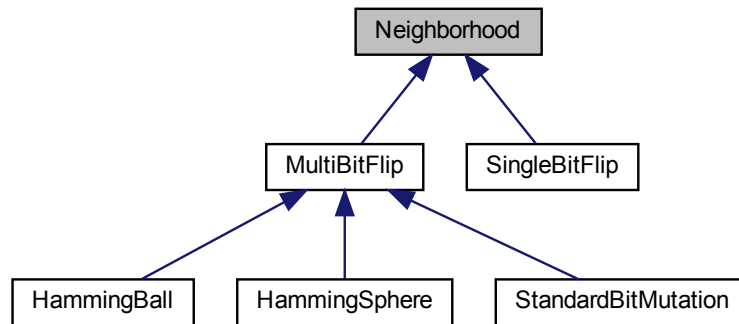
- lib/hnco/functions/modifiers/modifier.hh
- lib/hnco/functions/modifiers/modifier.cc

5.96 Neighborhood Class Reference

Neighborhood.

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

Inheritance diagram for Neighborhood:



Public Member Functions

- [Neighborhood](#) (int n)
Constructor.
- virtual [~Neighborhood](#) ()
Destructor.
- virtual void [set_origin](#) (const [bit_vector_t](#) &x)
Set the origin.
- virtual const [bit_vector_t](#) & [get_origin](#) () const
Get the origin.
- virtual const [bit_vector_t](#) & [get_candidate](#) () const
Get the candidate bit vector.
- virtual const [sparse_bit_vector_t](#) & [get_flipped_bits](#) () const
Get flipped bits.
- virtual void [propose](#) ()
Propose a candidate bit vector.
- virtual void [keep](#) ()
Keep the candidate bit vector.
- virtual void [forget](#) ()
Forget the candidate bit vector.
- virtual void [mutate](#) ([bit_vector_t](#) &bv)
Mutate.
- virtual void [map](#) (const [bit_vector_t](#) &input, [bit_vector_t](#) &output)
Map.

Protected Member Functions

- virtual void [sample_bits](#) ()=0
Sample bits.

Protected Attributes

- [bit_vector_t _origin](#)
Origin of the neighborhood.
- [bit_vector_t _candidate](#)
candidate bit vector
- `std::uniform_int_distribution< int > _index_dist`
Index distribution.
- [sparse_bit_vector_t _flipped_bits](#)
Flipped bits.

5.96.1 Detailed Description

Neighborhood.

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

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

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

A [Neighborhood](#) class can also behave as a mutation operator through the member functions `mutate` and `map`.

Definition at line 61 of file `neighborhood.hh`.

5.96.2 Constructor & Destructor Documentation

5.96.2.1 Neighborhood()

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

Constructor.

Parameters

| | |
|----------|---------------------|
| <i>n</i> | Size of bit vectors |
|----------|---------------------|

Definition at line 86 of file neighborhood.hh.

5.96.3 Member Function Documentation

5.96.3.1 map()

```
virtual void map (
    const bit\_vector\_t & input,
    bit\_vector\_t & output ) [inline], [virtual]
```

Map.

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

Parameters

| | |
|---------------|-------------------|
| <i>input</i> | Input bit vector |
| <i>output</i> | Output bit vector |

Definition at line 151 of file neighborhood.hh.

5.96.3.2 mutate()

```
virtual void mutate (
    bit\_vector\_t & bv ) [inline], [virtual]
```

Mutate.

In-place mutation of the bit vector.

Parameters

| | |
|-----------|----------------------|
| <i>bv</i> | Bit vector to mutate |
|-----------|----------------------|

Definition at line 137 of file neighborhood.hh.

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

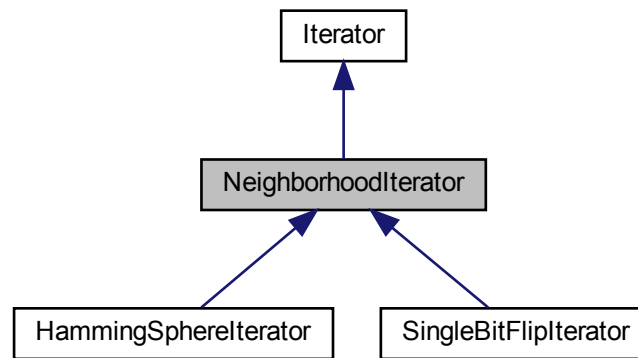
- lib/hnco/neighborhoods/neighborhood.hh

5.97 NeighborhoodIterator Class Reference

Neighborhood iterator.

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

Inheritance diagram for NeighborhoodIterator:



Public Member Functions

- [NeighborhoodIterator](#) (int n)
Constructor.
- virtual void [set_origin](#) (const [bit_vector_t](#) &x)
Set origin.

Additional Inherited Members

5.97.1 Detailed Description

Neighborhood iterator.

A neighborhood iterator allows to iterate over bit vectors in the neighborhood of a given origin. The origin itself should not belong to the neighborhood.

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

5.97.2 Constructor & Destructor Documentation

5.97.2.1 NeighborhoodIterator()

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

Constructor.

Parameters

| | |
|-----|---------------------|
| n | Size of bit vectors |
|-----|---------------------|

Definition at line 47 of file neighborhood-iterator.hh.

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

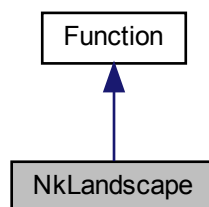
- lib/hnco/neighborhoods/neighborhood-iterator.hh
- lib/hnco/neighborhoods/neighborhood-iterator.cc

5.98 NkLandscape Class Reference

NK landscape.

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

Inheritance diagram for NkLandscape:



Public Member Functions

- [NkLandscape](#) ()
Default constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- void [display](#) (std::ostream &stream) const override
Display.

Instance generators

- template<class Generator >
void [generate](#) (int n, int k, Generator generator)
Instance generator.
- void [random](#) (int n, int k)
Random instance.

Load and save instance

- void [load](#) (std::string path)
Load instance.
- void [save](#) (std::string path) const
Save instance.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
Serialize.
- void [random_structure](#) (int n, int k)
Random structue.

Private Attributes

- std::vector< std::vector< int > > [_neighbors](#)
Bit neighbors.
- std::vector< std::vector< double > > [_partial_functions](#)
Partial functions.

Friends

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

5.98.1 Detailed Description

NK landscape.

Reference:

S. A. Kauffman. 1993. The origins of order: self-organisation and selection in evolution. Oxford University Press.

Definition at line 45 of file nk-landscape.hh.

5.98.2 Member Function Documentation

5.98.2.1 generate()

```
void generate (
    int n,
    int k,
    Generator generator ) [inline]
```

Instance generator.

Parameters

| | |
|------------------|---------------------------------------|
| <i>n</i> | Size of bit vector |
| <i>k</i> | Number of neighbors per bit |
| <i>generator</i> | Generator for partial function values |

Definition at line 89 of file nk-landscape.hh.

5.98.2.2 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to load |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 126 of file nk-landscape.hh.

5.98.2.3 random()

```
void random (
    int n,
    int k ) [inline]
```

Random instance.

Partial function values are sampled from the normal distribution.

Parameters

| | |
|----------|-----------------------------|
| <i>n</i> | Size of bit vector |
| <i>k</i> | Number of neighbors per bit |

Definition at line 107 of file nk-landscape.hh.

5.98.2.4 random_structure()

```
void random_structure (
    int n,
    int k ) [private]
```

Random structue.

Parameters

| | |
|----------|-----------------------------|
| <i>n</i> | Size of bit vector |
| <i>k</i> | Number of neighbors per bit |

Definition at line 32 of file nk-landscape.cc.

5.98.2.5 save()

```
void save (
    std::string path ) const [inline]
```

Save instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to save |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 133 of file nk-landscape.hh.

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

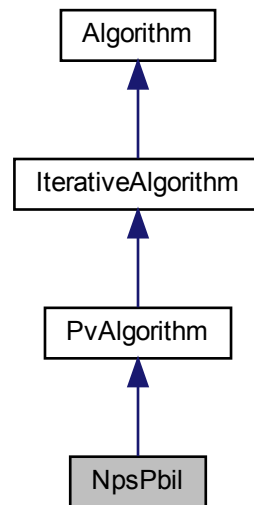
- lib/hnco/functions/collection/nk-landscape.hh
- lib/hnco/functions/collection/nk-landscape.cc

5.99 NpsPbil Class Reference

Population-based incremental learning with negative and positive selection.

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

Inheritance diagram for NpsPbil:



Public Member Functions

- [NpsPbil](#) (int n, int population_size)
Constructor.

Setters

- void [set_selection_size](#) (int x)
Set the selection size.
- void [set_learning_rate](#) (double x)
Set the learning rate.

Protected Member Functions

Loop

- void [init](#) () override
Initialize.
- void [iterate](#) () override
Single iteration.

Protected Attributes

- `Population _population`
Population.
- `pv_t _mean_best`
Mean of best individuals.
- `pv_t _mean_worst`
Mean of worst individuals.

Parameters

- `int _selection_size = 1`
Selection size.
- `double _learning_rate = 1e-3`
Learning rate.

5.99.1 Detailed Description

Population-based incremental learning with negative and positive selection.

Reference:

Arnaud Berny. 2001. Extending selection learning toward fixed-length d-ary strings. In *Artificial Evolution (Lecture Notes in Computer Science)*, P. Collet and others (Eds.). Springer, Le Creusot.

Definition at line 42 of file `nps-pbil.hh`.

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

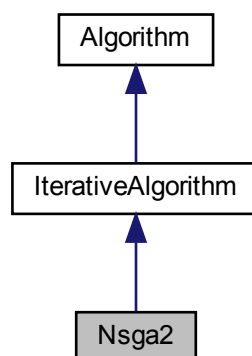
- `lib/hnco/algorithms/pv/nps-pbil.hh`
- `lib/hnco/algorithms/pv/nps-pbil.cc`

5.100 Nsga2 Class Reference

NSGA-II.

```
#include <hnco/multiobjective/algorithms/nsga2.hh>
```

Inheritance diagram for Nsga2:



Public Member Functions

- [Nsga2](#) (int n, int num_objectives, int population_size)
Constructor.
- const [Population](#) & [get_solutions](#) () override
Get solutions.

Setters

- void [set_tournament_size](#) (int n)
Set the tournament size.
- void [set_mutation_rate](#) (double p)
Set the mutation rate.
- void [set_allow_no_mutation](#) (bool b)
Set the flag _allow_no_mutation.
- void [set_crossover_probability](#) (double p)
Set the crossover probability.

Protected Member Functions

Loop

- void [init](#) () override
Initialize.
- void [iterate](#) () override
Single iteration.
- void [finalize](#) () override
Finalize.
- void [log](#) () override
Log.

Protected Attributes

- [Population _parents](#)
Parent population.
- [Population _offsprings](#)
Offspring population.
- [Population _full_population](#)
Full population.
- [Population _solutions](#)
Solutions.
- [neighborhood::StandardBitMutation _mutation](#)
Mutation operator.
- [std::bernoulli_distribution _do_crossover](#)
Do crossover.
- [hnco::algorithm::UniformCrossover _crossover](#)
Uniform crossover.
- [Nsga2ParetoFrontComputation _pareto_front_computation](#)
Pareto front computation.
- [std::vector< int > _pareto_fronts](#)
Pareto fronts.
- [std::vector< double > _crowding_distances](#)

Crowding distances.

- [hnco::permutation_t _permutation](#)

Permutation relative to Pareto front.

- `std::vector< FrontDistancePair > _front_distance_pairs`

Front distance pairs.

- [TournamentSelection< \[FrontDistancePair\]\(#\), std::less< \[FrontDistancePair\]\(#\) > > _selection_by_front_distance_pair](#)

Selection by front distance pairs.

Parameters

- `int _tournament_size = 2`

Tournament size.

- `double _mutation_rate`

Mutation rate.

- `bool _allow_no_mutation = false`

Allow no mutation.

- `double _crossover_probability = 0.8`

Crossover probability.

Additional Inherited Members

5.100.1 Detailed Description

NSGA-II.

NSGA-II is a (mu+mu) evolutionary algorithm for multiobjective optimization.

Deb, Agrawal, Pratap, and Meyarivan, "A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II", Parallel Problem Solving from Nature PPSN VI, 2000, Springer Berlin Heidelberg.

https://link.springer.com/chapter/10.1007/3-540-45356-3_83

Definition at line 81 of file nsga2.hh.

5.100.2 Constructor & Destructor Documentation

5.100.2.1 Nsga2()

```
Nsga2 (
    int n,
    int num_objectives,
    int population_size ) [inline]
```

Constructor.

Parameters

| | |
|------------------------|----------------------|
| <i>n</i> | Size of bit vectors |
| <i>num_objectives</i> | Number of objectives |
| <i>population_size</i> | Population size |

Definition at line 169 of file nsga2.hh.

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

- lib/hnco/multiobjective/algorithms/nsga2.hh
- lib/hnco/multiobjective/algorithms/nsga2.cc

5.101 Nsga2ParetoFrontComputation Class Reference

Pareto front computation from the NSGA-II paper.

```
#include <hnco/multiobjective/algorithms/pareto-front-computation.hh>
```

Public Member Functions

- [Nsga2ParetoFrontComputation](#) ([Population](#) &population)
Constructor.
- void [compute](#) (std::vector< int > &pareto_fronts)
Compute Pareto fronts.

Private Member Functions

- bool [is_non_dominated](#) (int i)
Check that a value is non dominated.

Private Attributes

- const [Population](#) & [_population](#)
Population
- std::vector< int > [_pool](#)
Pool of values to consider for inclusion in the Pareto front.
- std::vector< int > [_next_pool](#)
Next pool of values.
- std::unordered_set< int > [_non_dominated](#)
Non dominated values.
- std::vector< int > [_dominated](#)
Dominated values.

5.101.1 Detailed Description

Pareto front computation from the NSGA-II paper.

Definition at line 40 of file pareto-front-computation.hh.

5.101.2 Member Function Documentation

5.101.2.1 compute()

```
void compute (
    std::vector< int > &pareto_fronts ) [inline]
```

Compute Pareto fronts.

Parameters

| | |
|----------------------|----------------------------------|
| <i>pareto_fronts</i> | Pareto fronts (output parameter) |
|----------------------|----------------------------------|

Definition at line 89 of file pareto-front-computation.hh.

5.101.2.2 is_non_dominated()

```
bool is_non_dominated (  
    int i ) [inline], [private]
```

Check that a value is non dominated.

Check that no value in the non dominated set dominates the considered value.

Parameters

| | |
|----------|--------------------|
| <i>i</i> | Index of the value |
|----------|--------------------|

Definition at line 67 of file pareto-front-computation.hh.

5.101.3 Member Data Documentation**5.101.3.1 _dominated**

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

Dominated values.

To be removed from the non dominated ones.

Definition at line 58 of file pareto-front-computation.hh.

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

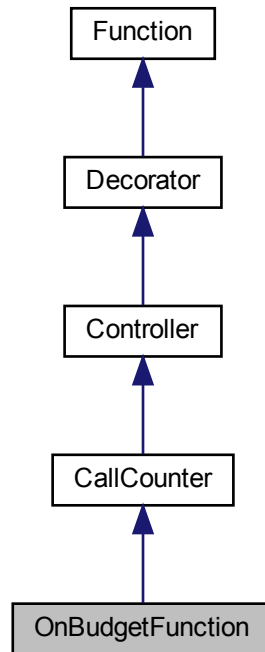
- lib/hnco/multiobjective/algorithms/pareto-front-computation.hh

5.102 OnBudgetFunction Class Reference

[Function](#) with a limited number of evaluations.

```
#include <hnco/functions/controllers/controller.hh>
```

Inheritance diagram for OnBudgetFunction:



Public Member Functions

- [OnBudgetFunction](#) ([Function](#) *function, int budget)

Constructor.

Evaluation

- double [evaluate](#) (const [bit_vector_t](#) &)
Evaluate a bit vector.
- double [evaluate_incrementally](#) (const [bit_vector_t](#) &x, double value, const [hnco::sparse_bit_vector_t](#) &flipped_bits)
Incrementally evaluate a bit vector.
- void [update](#) (const [bit_vector_t](#) &x, double value)
Update after a safe evaluation.

Private Attributes

- int [_budget](#)
Budget.

Additional Inherited Members

5.102.1 Detailed Description

[Function](#) with a limited number of evaluations.

Definition at line 186 of file controller.hh.

5.102.2 Member Function Documentation

5.102.2.1 `evaluate()`

```
double evaluate (
    const bit\_vector\_t & x ) [virtual]
```

Evaluate a bit vector.

Exceptions

| | |
|-----------------------|--|
| <i>LastEvaluation</i> | |
|-----------------------|--|

Reimplemented from [CallCounter](#).

Definition at line 97 of file controller.cc.

5.102.2.2 `evaluate_incrementally()`

```
double evaluate_incrementally (
    const bit\_vector\_t & x,
    double value,
    const hnco::sparse\_bit\_vector\_t & flipped_bits ) [virtual]
```

Incrementally evaluate a bit vector.

Exceptions

| | |
|-----------------------|--|
| <i>LastEvaluation</i> | |
|-----------------------|--|

Reimplemented from [CallCounter](#).

Definition at line 106 of file controller.cc.

5.102.2.3 update()

```
void update (
    const bit\_vector\_t & x,
    double value ) [virtual]
```

Update after a safe evaluation.

Exceptions

| | |
|-----------------------|--|
| <i>LastEvaluation</i> | |
|-----------------------|--|

Reimplemented from [CallCounter](#).

Definition at line 115 of file controller.cc.

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

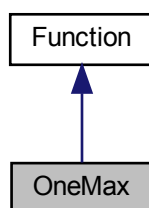
- lib/hnco/functions/controllers/controller.hh
- lib/hnco/functions/controllers/controller.cc

5.103 OneMax Class Reference

OneMax.

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

Inheritance diagram for OneMax:



Public Member Functions

- [OneMax](#) (int bv_size)
Constructor.

Information about the function

- int [get_bv_size](#) () const override
Get bit vector size.
- double [get_maximum](#) () const override
Get the global maximum.
- bool [has_known_maximum](#) () const override
Check for a known maximum.
- bool [provides_incremental_evaluation](#) () const override
Check whether the function provides incremental evaluation.
- void [display](#) (std::ostream &stream) const override
Display.

Evaluation

- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- double [evaluate_incrementally](#) (const [bit_vector_t](#) &x, double v, const [hnco::sparse_bit_vector_t](#) &flipped_bits) override
Incrementally evaluate a bit vector.

Private Attributes

- int [_bv_size](#)
Bit vector size.

5.103.1 Detailed Description

OneMax.

References:

Heinz Mühlenbein, "How genetic algorithms really work: I. mutation and hillclimbing", in Proc. 2nd Int. Conf. on Parallel Problem Solving from Nature, 1992

Thomas Jansen, Analyzing Evolutionary Algorithms. Springer, 2013.

Definition at line 41 of file theory.hh.

5.103.2 Member Function Documentation

5.103.2.1 `get_maximum()`

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

Get the global maximum.

Returns

`_bv_size`

Reimplemented from [Function](#).

Definition at line 61 of file theory.hh.

5.103.2.2 `has_known_maximum()`

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

Check for a known maximum.

Returns

`true`

Reimplemented from [Function](#).

Definition at line 65 of file theory.hh.

5.103.2.3 `provides_incremental_evaluation()`

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

Check whether the function provides incremental evaluation.

Returns

`true`

Reimplemented from [Function](#).

Definition at line 70 of file theory.hh.

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

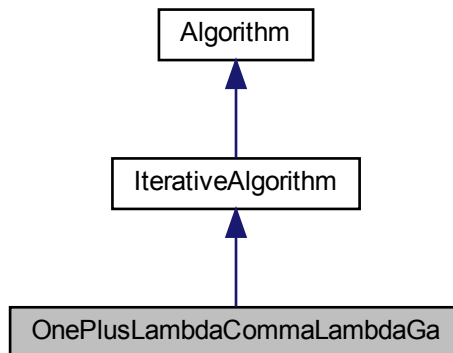
- `lib/hnco/functions/collection/theory.hh`
- `lib/hnco/functions/collection/theory.cc`

5.104 OnePlusLambdaCommaLambdaGa Class Reference

(1+(lambda, lambda)) genetic algorithm.

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

Inheritance diagram for OnePlusLambdaCommaLambdaGa:



Public Member Functions

- [OnePlusLambdaCommaLambdaGa](#) (int n, int lambda)
Constructor.

Setters

- void [set_mutation_rate](#) (double p)
Set the mutation rate.
- void [set_crossover_bias](#) (double x)
Set the crossover bias.

Protected Member Functions

Loop

- void [init](#) () override
Initialize.
- void [iterate](#) () override
Single iteration.

Protected Attributes

- [Population _offsprings](#)
Offsprings.
- `std::binomial_distribution< int > _radius_dist`
Radius distribution.
- [neighborhood::HammingSphere _mutation](#)
Mutation operator.
- [bit_vector_t _parent](#)
Parent.
- [BiasedCrossover _crossover](#)
Biased crossover.

Parameters

- `double _mutation_rate`
Mutation rate.
- `double _crossover_bias`
Crossover bias.

5.104.1 Detailed Description

(1+(lambda, lambda)) genetic algorithm.

Reference:

Benjamin Doerr, Carola Doerr, and Franziska Ebel. 2015. From black-box complexity to designing new genetic algorithms. Theoretical Computer Science 567 (2015), 87–104.

Definition at line 49 of file one-plus-lambda-comma-lambda-ga.hh.

5.104.2 Constructor & Destructor Documentation

5.104.2.1 OnePlusLambdaCommaLambdaGa()

```
OnePlusLambdaCommaLambdaGa (
    int n,
    int lambda ) [inline]
```

Constructor.

By default, `_mutation_rate` is set to `lambda / n` and `_crossover_bias` to `1 / lambda`.

Parameters

| | |
|---------------|---------------------------|
| <i>n</i> | Size of bit vectors |
| <i>lambda</i> | Offspring population size |

Definition at line 102 of file one-plus-lambda-comma-lambda-ga.hh.

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

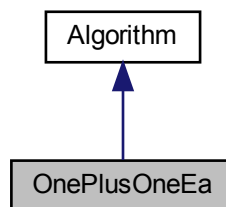
- lib/hnco/algorithms/ea/one-plus-lambda-comma-lambda-ga.hh
- lib/hnco/algorithms/ea/one-plus-lambda-comma-lambda-ga.cc

5.105 OnePlusOneEa Class Reference

(1+1) EA.

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

Inheritance diagram for OnePlusOneEa:



Public Member Functions

- [OnePlusOneEa](#) (int n)
Constructor.
- void [maximize](#) (const std::vector< [function::Function](#) * > &functions) override
Maximize.
- void [finalize](#) () override
Finalize.

Setters

- void [set_num_iterations](#) (int x)
Set the number of iterations.
- void [set_mutation_rate](#) (double p)
Set the mutation rate.
- void [set_allow_no_mutation](#) (bool b)
Set the flag_allow_no_mutation.
- void [set_incremental_evaluation](#) (bool x)
Set incremental evaluation.

Private Attributes

- [neighborhood::StandardBitMutation _neighborhood](#)
Neighborhood.
- [RandomLocalSearch _rls](#)
Random local search.

Parameters

- `int _num_iterations = 0`
Number of iterations.
- `double _mutation_rate`
Mutation rate.
- `bool _allow_no_mutation = false`
Allow no mutation.
- `bool _incremental_evaluation = false`
Incremental evaluation.

Additional Inherited Members

5.105.1 Detailed Description

(1+1) EA.

(1+1) EA is implemented as a [RandomLocalSearch](#) with a [StandardBitMutation](#) neighborhood and infinite patience. Thus the class [OnePlusOneEa](#) is derived from [Algorithm](#) instead of [IterativeAlgorithm](#).

Reference:

Thomas Jansen, Analyzing Evolutionary Algorithms. Springer, 2013.

Definition at line 45 of file one-plus-one-ea.hh.

5.105.2 Constructor & Destructor Documentation

5.105.2.1 OnePlusOneEa()

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

Constructor.

Parameters

| | |
|----------|---------------------|
| <i>n</i> | Size of bit vectors |
|----------|---------------------|

`_mutation_rate` is initialized to $1 / n$.

Definition at line 80 of file one-plus-one-ea.hh.

5.105.3 Member Function Documentation

5.105.3.1 set_num_iterations()

```
void set_num_iterations (
    int x ) [inline]
```

Set the number of iterations.

Parameters

| | |
|---|----------------------|
| x | Number of iterations |
|---|----------------------|

x <= 0 means indefinite

Definition at line 111 of file one-plus-one-ea.hh.

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

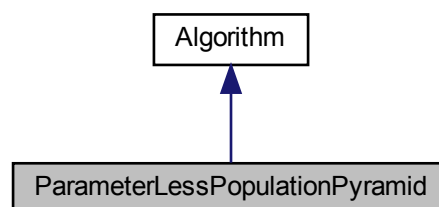
- lib/hnco/algorithms/ea/one-plus-one-ea.hh

5.106 ParameterLessPopulationPyramid Class Reference

Parameter-less Population Pyramid.

```
#include <hnco/algorithms/fast-efficient-p3/p3.hh>
```

Inheritance diagram for ParameterLessPopulationPyramid:



Public Member Functions

- [ParameterLessPopulationPyramid](#) (int n)
Constructor.
- [~ParameterLessPopulationPyramid](#) ()
Destructor.
- void [maximize](#) (const std::vector< [function::Function](#) * > &functions)
Maximize.
- void [finalize](#) ()
Finalize.

Private Attributes

- [Implementation](#) * [_pimpl](#)
Pointer to implementation.

Additional Inherited Members

5.106.1 Detailed Description

Parameter-less Population Pyramid.

Implementation of the Parameter-less Population Pyramid (P3 for short).

Author: Brian W. Goldman

Reference:

"Fast and Efficient Black Box Optimization using the Parameter-less Population Pyramid" by B. W. Goldman and W. F. Punch

Integrated into HNCO by Arnaud Berny

Definition at line 53 of file p3.hh.

5.106.2 Member Data Documentation

5.106.2.1 [_pimpl](#)

```
Implementation* \_pimpl [private]
```

Pointer to implementation.

The main motivation for this pattern is to avoid including declarations from [fast_efficient_p3](#) into the global namespace.

A raw pointer is used instead of a `unique_ptr` because the latter will not compile with `pybind11`.

Definition at line 64 of file p3.hh.

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

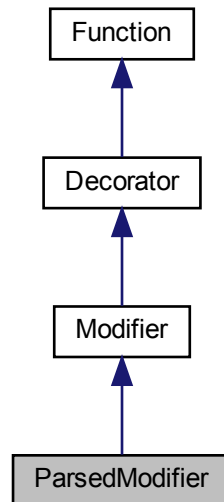
- lib/hnco/algorithms/fast-efficient-p3/p3.hh
- lib/hnco/algorithms/fast-efficient-p3/p3.cc

5.107 ParsedModifier Class Reference

Parsed modifier.

```
#include <hnco/functions/modifiers/parsed-modifier.hh>
```

Inheritance diagram for ParsedModifier:



Public Member Functions

- [ParsedModifier](#) ([Function](#) *function, std::string expression)
Constructor.

Information about the function

- int [get_bv_size](#) () const override
Get bit vector size.

Evaluation

- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Private Attributes

- FunctionParser [_parser](#)
Function parser.
- double [_values](#) [1]
Array of values.

Additional Inherited Members

5.107.1 Detailed Description

Parsed modifier.

Let f be the original function. Then the modified function is equivalent to $g \circ f$, where g is a real function defined by an expression $g(x)$ provided as a string.

Definition at line 40 of file `parsed-modifier.hh`.

5.107.2 Constructor & Destructor Documentation

5.107.2.1 ParsedModifier()

```
ParsedModifier (
    Function * function,
    std::string expression )
```

Constructor.

Parameters

| | |
|-------------------|---------------------|
| <i>function</i> | Decorated function |
| <i>expression</i> | Expression to parse |

Definition at line 31 of file `parsed-modifier.cc`.

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

- `lib/hnco/functions/modifiers/parsed-modifier.hh`
- `lib/hnco/functions/modifiers/parsed-modifier.cc`

5.108 ParsedMultivariateFunction< Parser > Class Template Reference

Parsed multivariate function.

```
#include <hnco/functions/collection/parsed-multivariate-function.hh>
```

Public Types

- using `domain_type` = typename `Parser::value_type`
Domain type.
- using `codomain_type` = typename `Parser::value_type`
Codomain type.

Public Member Functions

- [ParsedMultivariateFunction](#) (std::string expression)
Constructor.
- void [display](#) (std::ostream &stream) const
Display the problem.
- [codomain_type evaluate](#) (const std::vector< [domain_type](#) > &x)
Evaluate.
- void [describe](#) (const std::vector< [domain_type](#) > &x, std::ostream &stream)
Describe a solution.
- int [get_num_variables](#) ()
Get the number of variables.

Private Attributes

- Parser [_fparser](#)
Function parser.
- std::vector< std::string > [_variable_names](#)
Variable names.
- std::string [_expression](#)
Expression.

5.108.1 Detailed Description

```
template<class Parser>
class hnco::function::ParsedMultivariateFunction< Parser >
```

Parsed multivariate function.

Uses the C++ library "Function Parser" (fparser):

<http://warp.povusers.org/FunctionParser/fparser.html>

Warning

The function string syntax depends on the chosen parser.

Definition at line 48 of file parsed-multivariate-function.hh.

5.108.2 Constructor & Destructor Documentation

5.108.2.1 ParsedMultivariateFunction()

```
ParsedMultivariateFunction (
    std::string expression ) [inline]
```

Constructor.

Parameters

| | |
|-------------------|---------------------|
| <i>expression</i> | Expression to parse |
|-------------------|---------------------|

Definition at line 71 of file parsed-multivariate-function.hh.

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

- lib/hnco/functions/collection/parsed-multivariate-function.hh

5.109 ParsedMultivariateFunction< Parser > Class Template Reference

Parsed multivariate function.

```
#include <hnco/multiobjective/functions/collection/parsed-multivariate-function.hh>
```

Public Types

- using [domain_type](#) = typename Parser::value_type
Domain type.
- using [codomain_type](#) = typename Parser::value_type
Codomain type.

Public Member Functions

- [ParsedMultivariateFunction](#) (std::string expression)
Constructor.
- int [get_num_variables](#) () const
Get the number of variables.
- int [get_output_size](#) () const
Get output size (number of objectives)
- void [evaluate](#) (const std::vector< [domain_type](#) > &x, std::vector< [codomain_type](#) > &values)
Evaluate.
- void [display](#) (std::ostream &stream) const
Display the problem.
- void [describe](#) (const std::vector< [domain_type](#) > &x, std::ostream &stream)
Describe a solution.

Private Attributes

- std::vector< std::string > [_expressions](#)
Expressions.
- std::vector< Parser > [_parsers](#)
Function parsers
- std::vector< std::vector< std::string > > [_names](#)
Variable names.
- std::vector< std::vector< [domain_type](#) > > [_variables](#)
Variables.
- std::vector< std::vector< int > > [_lookup_tables](#)
Lookup tables.
- std::unordered_map< std::string, int > [_index_of](#)
Index of variable.

5.109.1 Detailed Description

```
template<class Parser>
class hnco::multiobjective::function::ParsedMultivariateFunction< Parser >
```

Parsed multivariate function.

Uses the C++ library "Function Parser" (fparser):

<http://warp.povusers.org/FunctionParser/fparser.html>

Warning

The function string syntax depends on the chosen parser.

Definition at line 53 of file parsed-multivariate-function.hh.

5.109.2 Constructor & Destructor Documentation

5.109.2.1 ParsedMultivariateFunction()

```
ParsedMultivariateFunction (
    std::string expression ) [inline]
```

Constructor.

An expression is a list of sub expressions separated by double colons (::). Each sub expression defines a multivariate function.

Parameters

| | |
|-------------------|---------------------|
| <i>expression</i> | Expression to parse |
|-------------------|---------------------|

Definition at line 105 of file parsed-multivariate-function.hh.

5.109.3 Member Data Documentation

5.109.3.1 _index_of

```
std::unordered_map<std::string, int> _index_of [private]
```

Index of variable.

In the vector of variables to be evaluated.

Definition at line 93 of file parsed-multivariate-function.hh.

5.109.3.2 `_lookup_tables`

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

Lookup tables.

Indexed by parser index.

Definition at line 87 of file `parsed-multivariate-function.hh`.

5.109.3.3 `_names`

```
std::vector<std::vector<std::string> > _names [private]
```

Variable names.

Indexed by parser index.

Definition at line 75 of file `parsed-multivariate-function.hh`.

5.109.3.4 `_variables`

```
std::vector<std::vector<domain_type> > _variables [private]
```

Variables.

Indexed by parser index.

Definition at line 81 of file `parsed-multivariate-function.hh`.

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

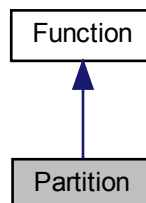
- `lib/hnco/multiobjective/functions/collection/parsed-multivariate-function.hh`

5.110 Partition Class Reference

Partition.

```
#include <hnco/functions/collection/partition.hh>
```

Inheritance diagram for Partition:



Public Member Functions

- [Partition](#) ()
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Instance generators

- template<class Generator >
void [generate](#) (int n, Generator generator)
Instance generator.
- void [random](#) (int n, int upper_bound)
Random instance.

Load and save instance

- void [load](#) (std::string path)
Load instance.
- void [save](#) (std::string path) const
Save instance.

Display

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

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
Serialize.

Private Attributes

- std::vector< int > [_numbers](#)
Multiset of positive integers.

Friends

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

5.110.1 Detailed Description

Partition.

Partition a finite multiset of positive integers into two subsets such that the sum of numbers in the first subset is the closest to the sum of numbers in the second subset.

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

Definition at line 52 of file partition.hh.

5.110.2 Member Function Documentation

5.110.2.1 generate()

```
void generate (
    int n,
    Generator generator ) [inline]
```

Instance generator.

Parameters

| | |
|------------------|---------------------|
| <i>n</i> | Size of bit vectors |
| <i>generator</i> | Number generator |

Definition at line 84 of file partition.hh.

5.110.2.2 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to load |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 120 of file partition.hh.

5.110.2.3 random()

```
void random (
    int n,
    int upper_bound ) [inline]
```

Random instance.

The numbers are sampled from the uniform distribution on [1..upper_bound].

Parameters

| | |
|--------------------|----------------------------------|
| <i>n</i> | Size of bit vector |
| <i>upper_bound</i> | Upper bound of positive integers |

Definition at line 100 of file partition.hh.

5.110.2.4 save()

```
void save (
    std::string path ) const [inline]
```

Save instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to save |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 127 of file partition.hh.

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

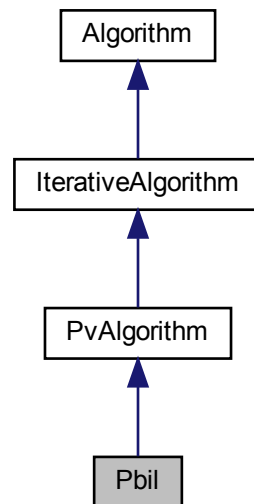
- lib/hnco/functions/collection/partition.hh
- lib/hnco/functions/collection/partition.cc

5.111 Pbil Class Reference

Population-based incremental learning.

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

Inheritance diagram for Pbil:



Public Member Functions

- [Pbil](#) (int n, int population_size)
Constructor.

Setters

- void [set_selection_size](#) (int x)
Set the selection size.
- void [set_learning_rate](#) (double x)
Set the learning rate.

Protected Member Functions

Loop

- void [init](#) () override
Initialize.
- void [iterate](#) () override
Single iteration.

Protected Attributes

- [Population _population](#)
Population.
- [pv_t _mean](#)
Mean of selected bit vectors.

Parameters

- `int _selection_size = 1`
Selection size.
- `double _learning_rate = 1e-3`
Learning rate.

5.111.1 Detailed Description

Population-based incremental learning.

Reference:

S. Baluja and R. Caruana. 1995. Removing the genetics from the standard genetic algorithm. In Proceedings of the 12th Annual Conference on Machine Learning. 38–46.

Definition at line 42 of file pbil.hh.

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

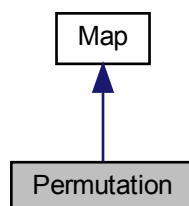
- `lib/hnco/algorithms/pv/pbil.hh`
- `lib/hnco/algorithms/pv/pbil.cc`

5.112 Permutation Class Reference

Permutation.

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

Inheritance diagram for Permutation:



Public Member Functions

- void `random` (int n)
Random instance.
- void `map` (const `bit_vector_t` &input, `bit_vector_t` &output) override
Map
- int `get_input_size` () const override
Get input size.
- int `get_output_size` () const override
Get output size.
- bool `is_surjective` () const override
Check for surjective map.

Load and save map

- void `load` (std::string path)
Load map.
- void `save` (std::string path) const
Save map.

Private Member Functions

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

Private Attributes

- `permutation_t_permutation`
Permutation.

Friends

- class `boost::serialization::access`

5.112.1 Detailed Description

Permutation.

A permutation is a linear map f from F_2^n to itself defined by $f(x) = y$, where $y_i = x_{\sigma_i}$ and σ is a permutation of $0, 1, \dots, n - 1$.

Definition at line 167 of file map.hh.

5.112.2 Member Function Documentation

5.112.2.1 `is_surjective()`

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

Check for surjective map.

Returns

true

Reimplemented from [Map](#).

Definition at line 218 of file map.hh.

5.112.2.2 `load()`

```
void load (
    std::string path ) [inline]
```

Load map.

Parameters

| | |
|-------------|------------------|
| <i>path</i> | Path of the file |
|-------------|------------------|

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 229 of file map.hh.

5.112.2.3 `save()`

```
void save (
    std::string path ) const [inline]
```

Save map.

Parameters

| | |
|-------------|------------------|
| <i>path</i> | Path of the file |
|-------------|------------------|

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 236 of file map.hh.

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

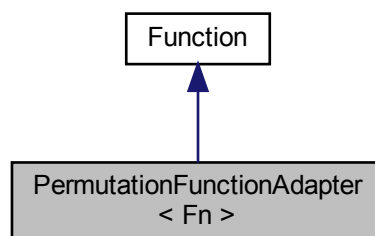
- lib/hnco/maps/map.hh
- lib/hnco/maps/map.cc

5.113 PermutationFunctionAdapter< Fn > Class Template Reference

Permutation function adapter.

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

Inheritance diagram for PermutationFunctionAdapter< Fn >:



Public Member Functions

- [PermutationFunctionAdapter](#) (Fn *fn, [representation::PermutationRepresentation](#) rep)
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &bv) override
Evaluate.
- void [display](#) (std::ostream &stream) const override
Display.
- void [describe](#) (const [bit_vector_t](#) &bv, std::ostream &stream) override
Describe a bit vector.

Private Member Functions

- void `unpack` (const `bit_vector_t` &bv)
Unpack a bit vector into a permutation.

Private Attributes

- `Fn * _function`
Permutation function.
- `representation::PermutationRepresentation _representation`
Permutation representation.
- `permutation_t _permutation`
Permutation.

5.113.1 Detailed Description

```
template<class Fn>
class hnco::function::PermutationFunctionAdapter< Fn >
```

Permutation function adapter.

The purpose of this class is to build a regular hnco function from an arbitrary function over permutations. This is achieved using a permutation representation.

Definition at line 42 of file permutation-function-adapter.hh.

5.113.2 Constructor & Destructor Documentation

5.113.2.1 PermutationFunctionAdapter()

```
PermutationFunctionAdapter (
    Fn * fn,
    representation::PermutationRepresentation rep ) [inline]
```

Constructor.

Parameters

| | |
|------------|----------------------------|
| <i>fn</i> | Multivariate function |
| <i>rep</i> | Permutation representation |

Definition at line 66 of file permutation-function-adapter.hh.

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

- lib/hnco/functions/permutation-function-adapter.hh

5.114 PermutationRepresentation Class Reference

Permutation representation.

```
#include <hnco/representations/permutation.hh>
```

Classes

- struct [Element](#)
Element.

Public Member Functions

- [PermutationRepresentation](#) (int num_elements, int num_additional_bits)
Constructor.
- int [get_num_elements](#) () const
Get number of elements.
- int [size](#) () const
Size of the representation.
- void [unpack](#) (const [bit_vector_t](#) &bv, int start, [hnco::permutation_t](#) &permutation)
Unpack bit vector into a permutation.
- void [display](#) (std::ostream &stream) const
Display.

Private Attributes

- std::vector< [Element](#) > [_elements](#)
Elements.
- int [_num_bits](#)
Number of bits per element.
- int [_representation_size](#)
Representation size.

5.114.1 Detailed Description

Permutation representation.

Definition at line 39 of file permutation.hh.

5.114.2 Constructor & Destructor Documentation

5.114.2.1 PermutationRepresentation()

```
PermutationRepresentation (  
    int num_elements,  
    int num_additional_bits ) [inline]
```

Constructor.

Each element is represented by an integer encoded using $\text{std::ceil}(\text{std::log}(\text{num_elements}) / \text{std::log}(2)) + \text{num_additional_bits}$.

Parameters

| | |
|----------------------------|---------------------------------------|
| <i>num_elements</i> | Number of elements |
| <i>num_additional_bits</i> | Number of additional bits per element |

Definition at line 73 of file permutation.hh.

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

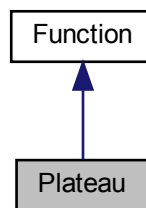
- lib/hnco/representations/permutation.hh

5.115 Plateau Class Reference

Plateau.

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

Inheritance diagram for Plateau:



Public Member Functions

- [Plateau](#) (int bv_size)
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- bool [has_known_maximum](#) () const override
Check for a known maximum.
- double [get_maximum](#) () const override
Get the global maximum.

Private Attributes

- int [_bv_size](#)
Bit vector size.

5.115.1 Detailed Description

Plateau.

Reference:

Thomas Jansen, Analyzing Evolutionary Algorithms. Springer, 2013.

Definition at line 242 of file theory.hh.

5.115.2 Member Function Documentation

5.115.2.1 `get_maximum()`

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

Get the global maximum.

Returns

`_bv_size + 2`

Reimplemented from [Function](#).

Definition at line 265 of file theory.hh.

5.115.2.2 `has_known_maximum()`

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

Check for a known maximum.

Returns

`true`

Reimplemented from [Function](#).

Definition at line 261 of file theory.hh.

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

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

5.116 PlusSelection Class Reference

Plus selection.

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

Public Member Functions

- [PlusSelection](#) ([Population](#) &parents, [Population](#) &offsprings)
Constructor.
- void [select](#) ()
Apply selection.

Private Attributes

- [Population](#) & [_parents](#)
Parent population.
- [Population](#) & [_offsprings](#)
Offspring population.
- [Population](#) [_pool](#)
Union of parent and offspring population.

5.116.1 Detailed Description

Plus selection.

Used as selection for replacement in evolutionary algorithms.

Definition at line 78 of file selection.hh.

5.116.2 Constructor & Destructor Documentation

5.116.2.1 PlusSelection()

```
PlusSelection (  
    Population & parents,  
    Population & offsprings ) [inline]
```

Constructor.

Parameters

| | |
|-------------------|----------------------|
| <i>parents</i> | Parent population |
| <i>offsprings</i> | Offspring population |

Definition at line 96 of file selection.hh.

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

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

5.117 Population Struct Reference

Population

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

Public Types

- using [Function](#) = [hnco::function::Function](#)
Function type

Public Member Functions

- [Population](#) (int population_size, int n)
Constructor.
- int [get_size](#) () const
Get population size.
- int [get_bv_size](#) () const
Get bit vector size.
- void [random](#) ()
Sample a random population.

Get sorted bit vectors

- [bit_vector_t](#) & [get_best_bv](#) ()
Get best bit vector.
- [bit_vector_t](#) & [get_best_bv](#) (int i)
Get best bit vector.
- [bit_vector_t](#) & [get_worst_bv](#) (int i)
Get worst bit vector.

Get sorted values

- double [get_best_value](#) () const
Get best value.
- double [get_best_value](#) (int i) const
Get best value.

Evaluation and sorting

- void [evaluate](#) ([Function](#) *function)
Evaluate the population.
- void [evaluate_in_parallel](#) (const std::vector< [Function](#) * > &functions)
Evaluate the population in parallel.
- void [sort](#) ()
Sort the population.
- void [partial_sort](#) (int selection_size)
Partially sort the population.
- std::pair< int, int > [get_equivalent_bvs](#) (int index)
Get equivalent bit vectors.

Public Attributes

- `std::vector< bit_vector_t > bvs`
Bit vectors.
- `std::vector< double > values`
Values.
- `hnco::permutation_t permutation`
Permutation.

5.117.1 Detailed Description

Population

Definition at line 41 of file `population.hh`.

5.117.2 Constructor & Destructor Documentation

5.117.2.1 Population()

```
Population (
    int population_size,
    int n ) [inline]
```

Constructor.

Parameters

| | |
|------------------------------|---------------------------------|
| <code>population_size</code> | Population size |
| <code>n</code> | Bit vector size |

Definition at line 60 of file `population.hh`.

5.117.3 Member Function Documentation

5.117.3.1 `get_best_bv()` [1/2]

```
bit\_vector\_t& get_best_bv ( ) [inline]
```

Get best bit vector.

Precondition

The population must be sorted.

Definition at line 90 of file `population.hh`.

5.117.3.2 `get_best_bv()` [2/2]

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

Get best bit vector.

Parameters

| | |
|----------|--------------------------------|
| <i>i</i> | Index in the sorted population |
|----------|--------------------------------|

Precondition

The population must be sorted.

Definition at line 97 of file population.hh.

5.117.3.3 `get_best_value()` [1/2]

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

Get best value.

Precondition

The population must be sorted.

Definition at line 118 of file population.hh.

5.117.3.4 `get_best_value()` [2/2]

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

Get best value.

Parameters

| | |
|----------|--------------------------------|
| <i>i</i> | Index in the sorted population |
|----------|--------------------------------|

Precondition

The population must be sorted.

Definition at line 125 of file population.hh.

5.117.3.5 `get_equivalent_bvs()`

```
std::pair< int, int > get_equivalent_bvs (
    int index )
```

Get equivalent bit vectors.

This member function returns a pair of ints (a, b) such that,

- for all i in $[0, a)$, $f(\text{get_best_bv}(i)) > f(\text{get_best_bv}(\text{index}))$
- for all i in $[a, b)$, $f(\text{get_best_bv}(i)) = f(\text{get_best_bv}(\text{index}))$
- for all i in $[b, \text{size})$, $f(\text{get_best_bv}(i)) < f(\text{get_best_bv}(\text{index}))$

Put another way, the range $[a, b)$ is the equivalence class of index , where two indices i and j are equivalent if $f(\text{get_best_bv}(i)) = f(\text{get_best_bv}(j))$.

Parameters

| | |
|--------------|---|
| <i>index</i> | Bit vector's index in the sorted population |
|--------------|---|

Precondition

The population must be sorted.

Definition at line 77 of file `population.cc`.

5.117.3.6 `get_worst_bv()`

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

Get worst bit vector.

Parameters

| | |
|----------|---|
| <i>i</i> | Reversed index in the sorted population |
|----------|---|

Precondition

The population must be sorted.

Definition at line 104 of file `population.hh`.

5.117.3.7 `partial_sort()`

```
void partial_sort (
    int selection_size ) [inline]
```

Partially sort the population.

Only the permutation is sorted using the order defined by $i < j$ if $\text{values}[i] > \text{values}[j]$. Before sorting, the permutation is shuffled to break ties randomly.

Parameters

| | |
|-----------------------------|---|
| <code>selection_size</code> | Sort the best <code>selection_size</code> individuals |
|-----------------------------|---|

Definition at line 158 of file `population.hh`.

5.117.3.8 `sort()`

```
void sort ( ) [inline]
```

Sort the population.

Only the permutation is sorted using the order defined by $i < j$ if $\text{values}[i] > \text{values}[j]$. Before sorting, the permutation is shuffled to break ties randomly.

Definition at line 146 of file `population.hh`.

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

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

5.118 Population Struct Reference

Population

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

Public Types

- using `Function` = `hnco::multiobjective::function::Function`
Function type
- using `value_t` = `hnco::multiobjective::function::value_t`
Value type.

Public Member Functions

- [Population](#) ()=default
Default constructor.
- [Population](#) (int population_size, int n, int num_objectives)
Constructor.
- int [get_size](#) () const
Get the population size.
- void [resize](#) (int population_size, int n, int num_objectives)
Resize the population.
- void [shrink](#) (int population_size)
Shrink the population.
- void [random](#) ()
Sample a random population.
- void [evaluate](#) ([Function](#) *function)
Evaluate a population.
- void [evaluate_in_parallel](#) (const std::vector< [Function](#) * > &functions)
Evaluate a population in parallel.

Public Attributes

- std::vector< [bit_vector_t](#) > [bvs](#)
Bit vectors.
- std::vector< [value_t](#) > [values](#)
Values.

5.118.1 Detailed Description

Population

Definition at line 36 of file population.hh.

5.118.2 Constructor & Destructor Documentation

5.118.2.1 Population()

```
Population (
    int population_size,
    int n,
    int num_objectives ) [inline]
```

Constructor.

Parameters

| | |
|------------------------|----------------------|
| <i>population_size</i> | Population size |
| <i>n</i> | Size of bit vectors |
| <i>num_objectives</i> | Number of objectives |

Definition at line 59 of file population.hh.

5.118.3 Member Function Documentation

5.118.3.1 `resize()`

```
void resize (
    int population_size,
    int n,
    int num_objectives ) [inline]
```

Resize the population.

Parameters

| | |
|------------------------|----------------------|
| <i>population_size</i> | Population size |
| <i>n</i> | Size of bit vectors |
| <i>num_objectives</i> | Number of objectives |

Definition at line 80 of file population.hh.

5.118.3.2 `shrink()`

```
void shrink (
    int population_size ) [inline]
```

Shrink the population.

Parameters

| | |
|------------------------|-----------------|
| <i>population_size</i> | Population size |
|------------------------|-----------------|

Precondition

`population_size < size()`

Definition at line 99 of file population.hh.

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

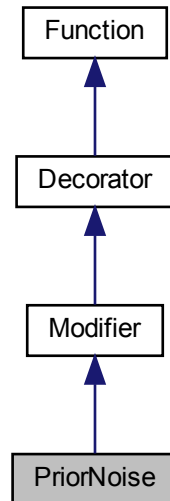
- lib/hnco/multiobjective/algorithms/population.hh
- lib/hnco/multiobjective/algorithms/population.cc

5.119 PriorNoise Class Reference

Prior noise.

```
#include <hnco/functions/modifiers/prior-noise.hh>
```

Inheritance diagram for PriorNoise:



Public Member Functions

- [PriorNoise](#) ([Function](#) *fn, [neighborhood::Neighborhood](#) *nh)
Constructor.

Information about the function

- int [get_bv_size](#) () const override
Get bit vector size.
- double [get_maximum](#) () const override
Get the global maximum.
- bool [has_known_maximum](#) () const override
Check for a known maximum.
- bool [provides_incremental_evaluation](#) () const override
Check whether the function provides incremental evaluation.

Evaluation

- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Private Attributes

- [neighborhood::Neighborhood](#) * [_neighborhood](#)
Neighborhood.
- [bit_vector_t](#) [_noisy_bv](#)
Noisy bit vector.

Additional Inherited Members

5.119.1 Detailed Description

Prior noise.

Definition at line 37 of file prior-noise.hh.

5.119.2 Member Function Documentation

5.119.2.1 `get_maximum()`

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

Get the global maximum.

Delegation is questionable here.

Reimplemented from [Function](#).

Definition at line 69 of file prior-noise.hh.

5.119.2.2 `has_known_maximum()`

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

Check for a known maximum.

Delegation is questionable here.

Reimplemented from [Function](#).

Definition at line 75 of file prior-noise.hh.

5.119.2.3 provides_incremental_evaluation()

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

Check whether the function provides incremental evaluation.

Returns

false

Reimplemented from [Function](#).

Definition at line 79 of file prior-noise.hh.

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

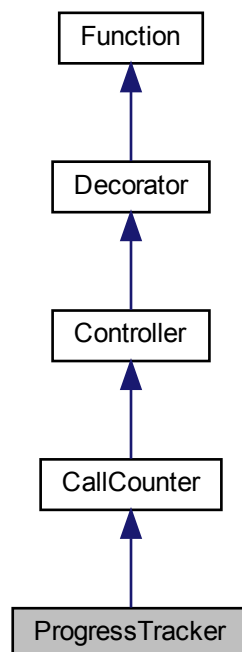
- lib/hnco/functions/modifiers/prior-noise.hh
- lib/hnco/functions/modifiers/prior-noise.cc

5.120 ProgressTracker Class Reference

[ProgressTracker](#).

```
#include <hnco/functions/controllers/controller.hh>
```

Inheritance diagram for ProgressTracker:



Classes

- struct [Event](#)
Event

Public Member Functions

- [ProgressTracker](#) ([Function](#) *function)
Constructor.

Evaluation

- double [evaluate](#) (const [bit_vector_t](#) &)
Evaluate a bit vector.
- double [evaluate_incrementally](#) (const [bit_vector_t](#) &x, double value, const [hnco::sparse_bit_vector_t](#) &flipped_bits)
Incrementally evaluate a bit vector.
- void [update](#) (const [bit_vector_t](#) &x, double value)
Update after a safe evaluation.

Get information

- const [Event](#) & [get_last_improvement](#) ()
Get the last improvement.
- double [get_evaluation_time](#) ()
Get evaluation time.

Setters

- void [set_log_improvement](#) (bool x)
Log improvement.
- void [set_record_evaluation_time](#) (bool b)
Record evaluation time.

Protected Member Functions

- void [update_last_improvement](#) (double value)
Update last improvement.

Protected Attributes

- [Event_last_improvement](#)
Last improvement.
- [StopWatch_stop_watch](#)
Stop watch.

Parameters

- bool [_log_improvement](#) = false
Log improvement.
- bool [_record_evaluation_time](#) = false
Record evaluation time.

5.120.1 Detailed Description

[ProgressTracker](#).

A [ProgressTracker](#) is a [CallCounter](#) which keeps track the last improvement, that is its value and the number of evaluations needed to reach it.

Definition at line 226 of file controller.hh.

5.120.2 Member Function Documentation

5.120.2.1 `get_last_improvement()`

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

Get the last improvement.

Warning

If `_last_improvement.num_evaluations` is zero then `_function` has never been called. The [Event](#) returned by `get_last_improvement` has therefore no meaning.

Definition at line 302 of file controller.hh.

5.120.3 Member Data Documentation

5.120.3.1 `_record_evaluation_time`

```
bool _record_evaluation_time = false [protected]
```

Record evaluation time.

Only relevant for [ProgressTracker::evaluate](#).

Definition at line 260 of file controller.hh.

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

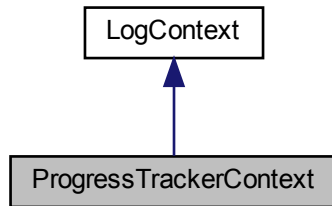
- lib/hnco/functions/controllers/controller.hh
- lib/hnco/functions/controllers/controller.cc

5.121 ProgressTrackerContext Class Reference

Log context for ProgressTracker.

```
#include <hnco/logging/log-context.hh>
```

Inheritance diagram for ProgressTrackerContext:



Public Member Functions

- [ProgressTrackerContext](#) ([hnco::function::controller::ProgressTracker](#) *pt)
Constructor.
- [std::string to_string](#) ()
Get context.

Private Attributes

- [hnco::function::controller::ProgressTracker](#) * _pt
Progress tracker.

5.121.1 Detailed Description

Log context for ProgressTracker.

Definition at line 50 of file log-context.hh.

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

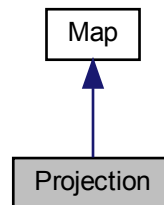
- lib/hnco/logging/log-context.hh

5.122 Projection Class Reference

Projection.

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

Inheritance diagram for Projection:



Public Member Functions

- [Projection](#) (const std::vector< int > &bit_positions, int input_size)
Constructor.
- void [map](#) (const [bit_vector_t](#) &input, [bit_vector_t](#) &output) override
Map
- int [get_input_size](#) () const override
Get input size.
- int [get_output_size](#) () const override
Get output size.
- bool [is_surjective](#) () const override
Check for surjective map.

Private Attributes

- std::vector< int > [_bit_positions](#)
Bit positions.
- int [_input_size](#)
Input size.

5.122.1 Detailed Description

Projection.

The projection y of a bit vector x is x where we have dropped a given set of components.

Let $I = \{i_1, i_2, \dots, i_m\}$ be a subset of $\{1, 2, \dots, n\}$.

A projection f from F_2^n to F_2^m , where $n \geq m$, is defined by $f(x) = y$, where, for all $j \in \{1, 2, \dots, m\}$, $y_j = x_{i_j}$.

If f is a projection and g is an injection with the same bit positions then their composition $f \circ g$ is the identity.

Definition at line 549 of file map.hh.

5.122.2 Constructor & Destructor Documentation

5.122.2.1 Projection()

```
Projection (
    const std::vector< int > & bit_positions,
    int input_size )
```

Constructor.

The output size of the map is given by the size of *bit_positions*.

Parameters

| | |
|----------------------|--|
| <i>bit_positions</i> | Bit positions in the input from where output bits are copied |
| <i>input_size</i> | Input size |

Precondition

`input_size >= bit_positions.size()`

Definition at line 196 of file `map.cc`.

5.122.3 Member Function Documentation

5.122.3.1 is_surjective()

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

Check for surjective map.

Returns

`true`

Reimplemented from [Map](#).

Definition at line 587 of file `map.hh`.

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

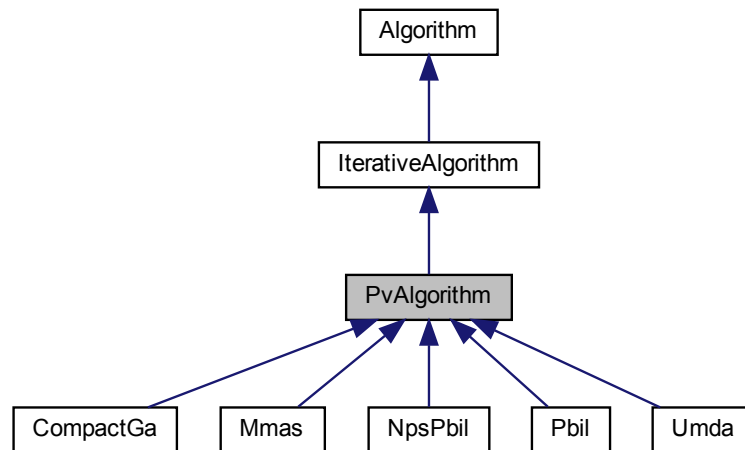
- `lib/hnco/maps/map.hh`
- `lib/hnco/maps/map.cc`

5.123 PvAlgorithm Class Reference

Probability vector algorithm.

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

Inheritance diagram for PvAlgorithm:



Public Member Functions

- [PvAlgorithm](#) (int n)
Constructor.

Setters for logging

- void [set_log_entropy](#) (bool x)
Log entropy.
- void [set_log_num_components](#) (int x)
Set the number of probability vector components to log.
- void [set_log_pv](#) (bool x)
Log probability vector.

Protected Member Functions

- void [set_something_to_log](#) ()
Set flag for something to log.

Loop

- void [log](#) () override
Log.

Protected Attributes

- `pv_t _pv`
Probability vector.
- `double _lower_bound`
Lower bound of probability.
- `double _upper_bound`
Upper bound of probability.

Logging

- `bool _log_entropy = false`
Log entropy.
- `bool _log_pv = false`
Log probability vector.
- `int _log_num_components = 5`
Number of probability vector components to log.

5.123.1 Detailed Description

Probability vector algorithm.

Definition at line 33 of file pv-algorithm.hh.

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

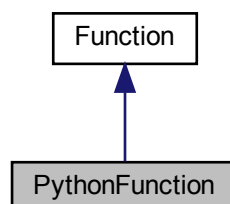
- lib/hnco/algorithms/pv/pv-algorithm.hh
- lib/hnco/algorithms/pv/pv-algorithm.cc

5.124 PythonFunction Class Reference

Python function.

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

Inheritance diagram for PythonFunction:



Public Member Functions

- [PythonFunction](#) (std::string path, std::string name)
Constructor.
- [~PythonFunction](#) ()
Destructor.
- int [get_bv_size](#) () const
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &)
Evaluate a bit vector.

Private Attributes

- pybind11::object [_scope](#)
Module.
- [Function](#) * [_function](#)
Function.

5.124.1 Detailed Description

Python function.

Uses pybind11.

The constructor initializes the python interpreter and the destructor finalizes it.

The python code must import the hnco module (built separately) to allow for communication between C++ and python. It must also define a derived class that inherits [Function](#) and an instance of it.

Definition at line 47 of file python-function.hh.

5.124.2 Constructor & Destructor Documentation

5.124.2.1 PythonFunction()

```
PythonFunction (
    std::string path,
    std::string name )
```

Constructor.

Parameters

| | |
|-------------|--|
| <i>path</i> | Path of the python file |
| <i>name</i> | Name of the Function instance defined in the python file |

Definition at line 32 of file python-function.cc.

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

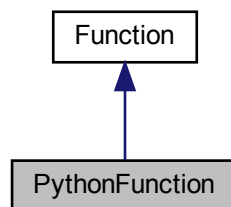
- lib/hnco/functions/collection/python-function.hh
- lib/hnco/functions/collection/python-function.cc

5.125 PythonFunction Class Reference

Python function.

```
#include <hnco/multiobjective/functions/collection/python-function.hh>
```

Inheritance diagram for PythonFunction:



Public Member Functions

- [PythonFunction](#) (std::string path, std::string name)
Constructor.
- [~PythonFunction](#) ()
Destructor.
- int [get_bv_size](#) () const
Get bit vector size.
- int [get_output_size](#) () const
Get output size (number of objectives)
- void [evaluate](#) (const [bit_vector_t](#) &bv, [value_t](#) &value)
Evaluate a bit vector.

Private Attributes

- pybind11::object [_scope](#)
Module.
- [Function](#) * [_function](#)
Function.

5.125.1 Detailed Description

Python function.

Uses pybind11.

The constructor initializes the python interpreter and the destructor finalizes it.

The python code must import the hnco module (built separately) to allow for communication between C++ and python. It must also define a derived class that inherits [Function](#) and an instance of it.

Definition at line 48 of file python-function.hh.

5.125.2 Constructor & Destructor Documentation

5.125.2.1 PythonFunction()

```
PythonFunction (
    std::string path,
    std::string name )
```

Constructor.

Parameters

| | |
|-------------|--|
| <i>path</i> | Path of the python file |
| <i>name</i> | Name of the Function instance defined in the python file |

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

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

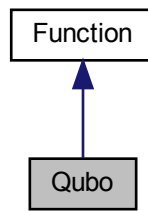
- lib/hnco/multiobjective/functions/collection/python-function.hh
- lib/hnco/multiobjective/functions/collection/python-function.cc

5.126 Qubo Class Reference

Quadratic unconstrained binary optimization.

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

Inheritance diagram for Qubo:



Public Member Functions

- [Qubo](#) ()
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Load and save instance

- void [load](#) (std::string path)
Load instance.

Private Member Functions

- void [load](#) (std::istream &stream)
Load an instance.

Private Attributes

- std::vector< std::vector< double > > [_q](#)
Matrix.

5.126.1 Detailed Description

Quadratic unconstrained binary optimization.

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

[Qubo](#) is the problem addressed by qbsolv. Here is its description as given on github:

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

There are some differences between [WalshExpansion2](#) and [Qubo](#):

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

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

References:

Michael Booth, Steven P. Reinhardt, and Aidan Roy. 2017. Partitioning Optimization Problems for Hybrid Classical/Quantum Execution. Technical Report. D-Wave.

<https://github.com/dwavesystems/qbsolv>

<http://people.brunel.ac.uk/~mastjjb/jeb/orlib/bqpinfo.html>

Definition at line 74 of file qubo.hh.

5.126.2 Member Function Documentation

5.126.2.1 load() [1/2]

```
void load (
    std::istream & stream ) [private]
```

Load an instance.

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 37 of file qubo.cc.

5.126.2.2 load() [2/2]

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

| | |
|-------------------|------------------------------|
| <code>path</code> | Path of the instance to load |
|-------------------|------------------------------|

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 105 of file qubo.hh.

5.126.3 Member Data Documentation

5.126.3.1 `_q`

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

Matrix.

n x n upper triangular matrix.

Definition at line 82 of file qubo.hh.

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

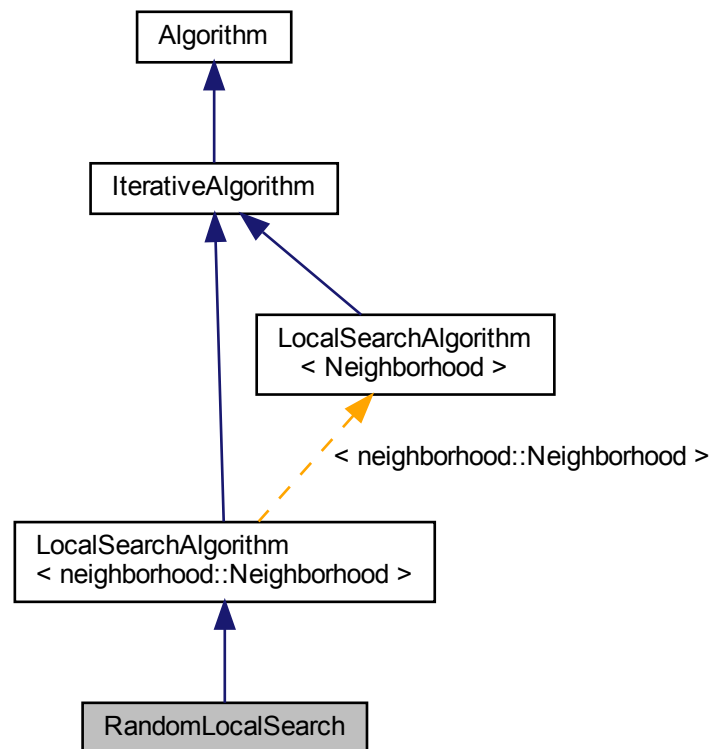
- lib/hnco/functions/collection/qubo.hh
- lib/hnco/functions/collection/qubo.cc

5.127 RandomLocalSearch Class Reference

Random local search.

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

Inheritance diagram for RandomLocalSearch:



Public Member Functions

- [RandomLocalSearch](#) (int n, [neighborhood::Neighborhood](#) *neighborhood)
Constructor.
- void [finalize](#) () override
Finalize.

Setters

- void [set_compare](#) (std::function< bool(double, double)> x)
Set the binary operator for comparing evaluations.
- void [set_patience](#) (int x)
Set patience.
- void [set_incremental_evaluation](#) (bool x)
Set incremental evaluation.

Protected Member Functions

- void [iterate_full](#) ()
Single iteration with full evaluation.

- void `iterate_incremental` ()
Single iteration with incremental evaluation.

Loop

- void `init` () override
Initialize.
- void `iterate` () override
Single iteration.

Protected Attributes

- int `_num_failures`
Number of failure.

Parameters

- `std::function< bool(double, double)> _compare` = `std::greater_equal<double>()`
Binary operator for comparing evaluations.
- int `_patience` = 50
Patience.
- bool `_incremental_evaluation` = false
Incremental evaluation.

5.127.1 Detailed Description

Random local search.

Definition at line 36 of file random-local-search.hh.

5.127.2 Member Function Documentation

5.127.2.1 `set_patience()`

```
void set_patience (
    int x ) [inline]
```

Set patience.

Number of consecutive rejected moves before ending the search.

Parameters

| | |
|----------------|----------|
| <code>x</code> | Patience |
|----------------|----------|

If `x <= 0` then patience is considered infinite.

Definition at line 104 of file random-local-search.hh.

5.127.3 Member Data Documentation

5.127.3.1 `_patience`

```
int _patience = 50 [protected]
```

Patience.

Number of consecutive rejected moves before ending the search.

Definition at line 55 of file random-local-search.hh.

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

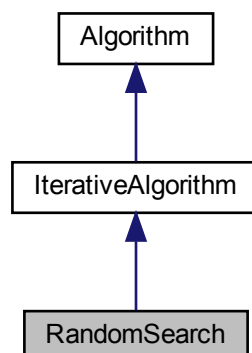
- lib/hnco/algorithms/ls/random-local-search.hh
- lib/hnco/algorithms/ls/random-local-search.cc

5.128 RandomSearch Class Reference

Random search.

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

Inheritance diagram for RandomSearch:



Public Member Functions

- [RandomSearch](#) (int n)
Constructor.

Protected Member Functions

Loop

- void [init](#) () override
Initialize.
- void [iterate](#) () override
Single iteration.

Protected Attributes

- [bit_vector_t _candidate](#)
Candidate.

5.128.1 Detailed Description

Random search.

Definition at line 31 of file random-search.hh.

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

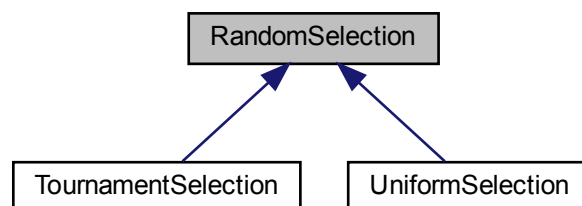
- lib/hnco/algorithms/random-search.hh
- lib/hnco/algorithms/random-search.cc

5.129 RandomSelection Class Reference

Random selection.

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

Inheritance diagram for RandomSelection:



Public Member Functions

- [RandomSelection](#) (const [Population](#) &population)
Constructor.
- virtual void [init](#) ()
Initialize.
- virtual const [bit_vector_t](#) & [select](#) ()=0
Select an individual in the population.

Protected Attributes

- const [Population](#) & [_population](#)
Population to select from

5.129.1 Detailed Description

Random selection.

Used as selection for reproduction in evolutionary algorithms.

Definition at line 39 of file random-selection.hh.

5.129.2 Constructor & Destructor Documentation

5.129.2.1 RandomSelection()

```
RandomSelection (
    const Population & population ) [inline]
```

Constructor.

Parameters

| | |
|-------------------|---------------------------|
| <i>population</i> | Population to select from |
|-------------------|---------------------------|

Definition at line 52 of file random-selection.hh.

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

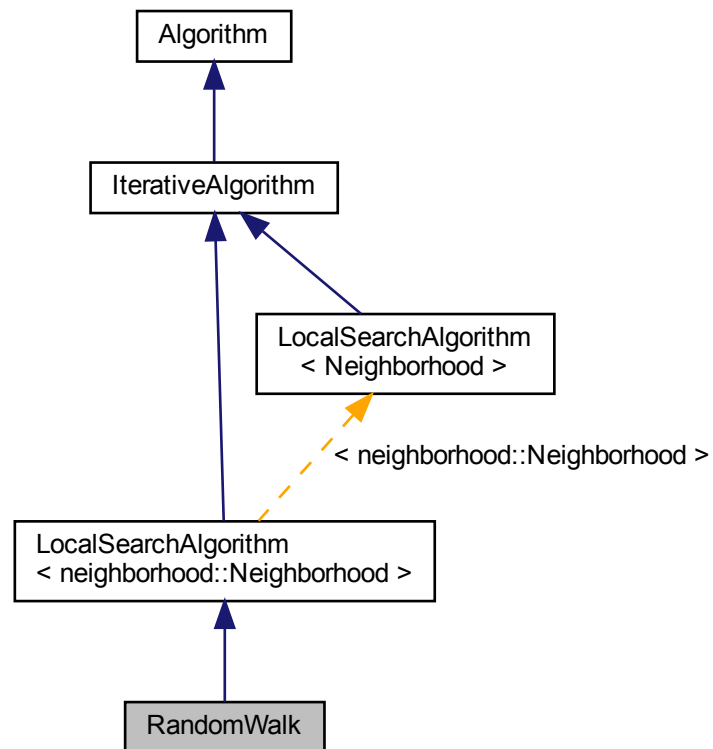
- lib/hnco/algorithms/ea/random-selection.hh

5.130 RandomWalk Class Reference

Random walk.

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

Inheritance diagram for RandomWalk:



Public Member Functions

- [RandomWalk](#) (int n, [neighborhood::Neighborhood](#) *neighborhood)
Constructor.

Setters

- void [set_incremental_evaluation](#) (bool x)
Set incremental evaluation.
- void [set_log_value](#) ()
Set log.

Protected Member Functions

- void [iterate_full](#) ()
Single iteration with full evaluation.
- void [iterate_incremental](#) ()
Single iteration with incremental evaluation.

Loop

- void `iterate` () override
Single iteration.
- void `log` () override
Log.

Protected Attributes

- double `_value`
Value of the last visited bit vector.

Parameters

- bool `_incremental_evaluation` = false
Incremental evaluation.

5.130.1 Detailed Description

Random walk.

The algorithm simply performs a random walk on the graph implicitly given by the neighborhood. At each iteration, the chosen neighbor does not depend on its evaluation. However optimization takes place as in random search, that is the best visited bit vector is remembered.

Definition at line 41 of file random-walk.hh.

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

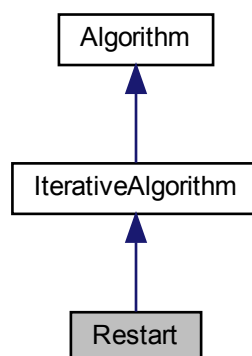
- lib/hnco/algorithms/ls/random-walk.hh
- lib/hnco/algorithms/ls/random-walk.cc

5.131 Restart Class Reference

Restart.

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

Inheritance diagram for Restart:



Public Member Functions

- [Restart](#) (int n, [Algorithm](#) *algorithm)
Constructor.

Protected Member Functions

Loop

- void [iterate](#) () override
Single iteration.

Protected Attributes

- [Algorithm](#) * [_algorithm](#)
Algorithm.

5.131.1 Detailed Description

Restart.

Restart an [Algorithm](#) an indefinite number of times. Should be used in conjunction with OnBudgetFunction or StopOnMaximum.

Definition at line 38 of file restart.hh.

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

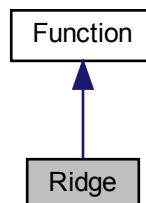
- lib/hnco/algorithms/decorators/restart.hh
- lib/hnco/algorithms/decorators/restart.cc

5.132 Ridge Class Reference

Ridge.

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

Inheritance diagram for Ridge:



Public Member Functions

- [Ridge](#) (int bv_size)
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- bool [has_known_maximum](#) () const override
Check for a known maximum.
- double [get_maximum](#) () const override
Get the global maximum.

Private Attributes

- int [_bv_size](#)
Bit vector size.

5.132.1 Detailed Description

Ridge.

Reference:

Thomas Jansen, Analyzing Evolutionary Algorithms. Springer, 2013.

Definition at line 207 of file theory.hh.

5.132.2 Member Function Documentation

5.132.2.1 [get_maximum\(\)](#)

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

Get the global maximum.

Returns

$2 * \text{_bv_size}$

Reimplemented from [Function](#).

Definition at line 230 of file theory.hh.

5.132.2.2 `has_known_maximum()`

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

Check for a known maximum.

Returns

true

Reimplemented from [Function](#).

Definition at line 226 of file theory.hh.

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

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

5.133 `ScalarToDouble< T >` Struct Template Reference

Convert a scalar to a double.

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

Public Types

- using `codomain_type` = T
Codomain type.

Public Member Functions

- double `operator()` (T x)
Convert to double.

5.133.1 Detailed Description

```
template<class T>
struct hnco::function::ScalarToDouble< T >
```

Convert a scalar to a double.

Definition at line 32 of file converter.hh.

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

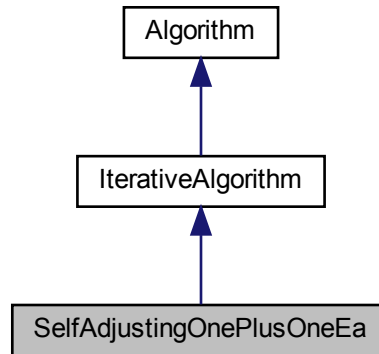
- lib/hnco/functions/converter.hh

5.134 SelfAdjustingOnePlusOneEa Class Reference

Self-adjusting (1+1) evolutionary algorithm.

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

Inheritance diagram for SelfAdjustingOnePlusOneEa:



Public Member Functions

- [SelfAdjustingOnePlusOneEa](#) (int n)
Constructor.
- void [finalize](#) () override
Finalize.

Setters

- void [set_mutation_rate_init](#) (double p)
Set the initial mutation rate.
- void [set_mutation_rate_min](#) (double p)
Set the minimum mutation rate.
- void [set_mutation_rate_max](#) (double p)
Set the maximum mutation rate.
- void [set_update_strength](#) (double x)
Set update strength.
- void [set_success_ratio](#) (double x)
Set success ratio.
- void [set_allow_no_mutation](#) (bool b)
Allow no mutation.
- void [set_incremental_evaluation](#) (bool b)
Turn on incremental evaluation.

Setters for logging

- void [set_log_mutation_rate](#) (bool b)
Log mutation rate.

Private Member Functions

- void `iterate_full` ()
Single iteration with full evaluation.
- void `iterate_incremental` ()
Single iteration with incremental evaluation.
- void `set_something_to_log` ()
Set flag for something to log.

Loop

- void `init` () override
Initialize.
- void `iterate` () override
Single iteration.
- void `log` () override
Log.

Private Attributes

- `neighborhood::StandardBitMutation _mutation`
Mutation operator.
- double `_mutation_rate`
Mutation rate.
- double `_coefficient`
Update strength to the power the success rate.

Parameters

- double `_mutation_rate_init`
Initial mutation rate.
- double `_mutation_rate_min`
Minimum mutation rate.
- double `_mutation_rate_max` = 1
Maximum mutation rate.
- double `_success_ratio` = 4
Success ratio.
- double `_update_strength`
Update strength.
- bool `_allow_no_mutation` = false
Allow no mutation.
- bool `_incremental_evaluation` = false
Incremental evaluation.

Logging

- bool `_log_mutation_rate` = false
Log mutation rate.

Additional Inherited Members

5.134.1 Detailed Description

Self-adjusting (1+1) evolutionary algorithm.

Reference: Benjamin Doerr, Carola Doerr, and Johannes Lengler. 2019. Self-adjusting mutation rates with provably optimal success rules. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '19). Association for Computing Machinery, New York, NY, USA, 1479–1487. <https://doi.org/10.1145/3321707.3321733>

Definition at line 41 of file self-adjusting-one-plus-one-ea.hh.

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

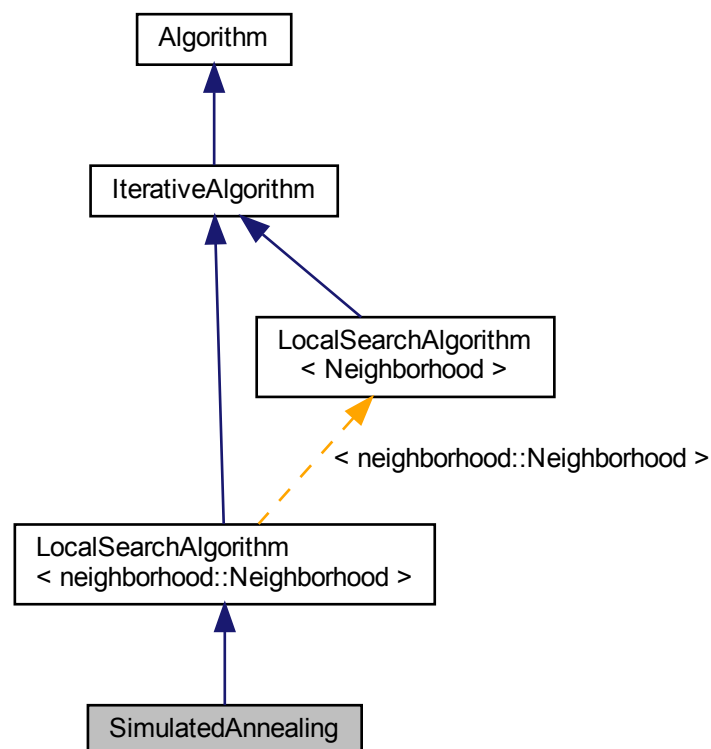
- lib/hnco/algorithms/ea/self-adjusting-one-plus-one-ea.hh
- lib/hnco/algorithms/ea/self-adjusting-one-plus-one-ea.cc

5.135 SimulatedAnnealing Class Reference

Simulated annealing.

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

Inheritance diagram for SimulatedAnnealing:



Public Member Functions

- [SimulatedAnnealing](#) (int n, [neighborhood::Neighborhood](#) *neighborhood)
Constructor.

Setters

- void [set_num_transitions](#) (int x)
Set the number of accepted transitions before annealing.
- void [set_num_trials](#) (int x)
Set the Number of trials.
- void [set_initial_acceptance_probability](#) (double x)
Set the initial acceptance probability.
- void [set_beta_ratio](#) (double x)
Set ratio for beta.

Protected Member Functions

- void [init_beta](#) ()
Initialize beta.

Loop

- void [init](#) () override
Initialize.
- void [iterate](#) () override
Single iteration.

Protected Attributes

- double [_beta](#)
Inverse temperature.
- double [_current_value](#)
Current value.
- int [_transitions](#)
Number of accepted transitions.

Parameters

- int [_num_transitions](#) = 50
Number of accepted transitions before annealing.
- int [_num_trials](#) = 100
Number of trials.
- double [_initial_acceptance_probability](#) = 0.6
Initial acceptance probability.
- double [_beta_ratio](#) = 1.2
Ratio for beta.

5.135.1 Detailed Description

Simulated annealing.

Reference:

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. Science 220, 4598 (May 1983), 671–680.

Definition at line 42 of file simulated-annealing.hh.

5.135.2 Member Function Documentation

5.135.2.1 init_beta()

```
void init_beta ( ) [protected]
```

Initialize beta.

Requires (2 * _num_trials) evaluations. This should be taken into account when using OnBudgetFunction.

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

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

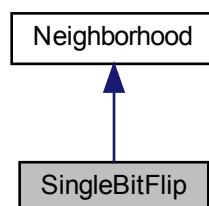
- lib/hnco/algorithms/ls/simulated-annealing.hh
- lib/hnco/algorithms/ls/simulated-annealing.cc

5.136 SingleBitFlip Class Reference

One bit neighborhood.

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

Inheritance diagram for SingleBitFlip:



Public Member Functions

- [SingleBitFlip](#) (int n)
Constructor.

Private Member Functions

- void [sample_bits](#) ()
Sample bits.

Additional Inherited Members

5.136.1 Detailed Description

One bit neighborhood.

Definition at line 163 of file neighborhood.hh.

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

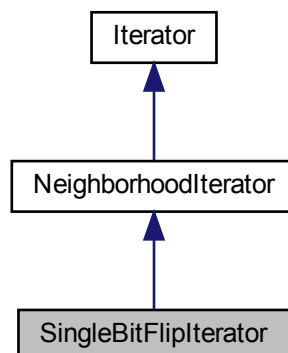
- lib/hnco/neighborhoods/neighborhood.hh

5.137 SingleBitFlipIterator Class Reference

Single bit flip neighborhood iterator.

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

Inheritance diagram for SingleBitFlipIterator:



Public Member Functions

- [SingleBitFlipIterator](#) (int n)
Constructor.
- bool [has_next](#) () override
Has next bit vector.
- const [bit_vector_t](#) & [next](#) () override
Next bit vector.

Private Attributes

- [size_t](#) [_index](#)
Index of the last flipped bit.

Additional Inherited Members

5.137.1 Detailed Description

Single bit flip neighborhood iterator.

Definition at line 56 of file neighborhood-iterator.hh.

5.137.2 Constructor & Destructor Documentation

5.137.2.1 SingleBitFlipIterator()

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

Constructor.

Parameters

| | |
|----------|---------------------|
| <i>n</i> | Size of bit vectors |
|----------|---------------------|

Definition at line 68 of file neighborhood-iterator.hh.

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

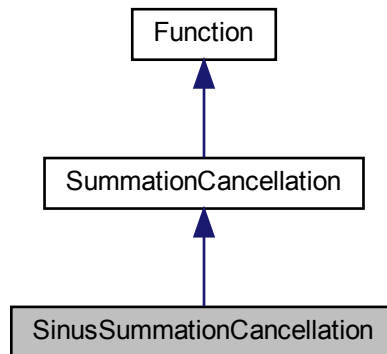
- lib/hnco/neighborhoods/neighborhood-iterator.hh
- lib/hnco/neighborhoods/neighborhood-iterator.cc

5.138 SinusSummationCancellation Class Reference

Summation cancellation with sinus.

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

Inheritance diagram for SinusSummationCancellation:



Public Member Functions

- [SinusSummationCancellation](#) (int n)
Constructor.
- double [evaluate](#) (const [bit_vector_t](#) &x) override
Evaluate a bit vector.

Additional Inherited Members

5.138.1 Detailed Description

Summation cancellation with sinus.

Reference:

M. Sebag and M. Schoenauer. 1997. A society of hill-climbers. In Proc. IEEE Int. Conf. on Evolutionary Computation. Indianapolis, 319–324.

Definition at line 101 of file cancellation.hh.

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

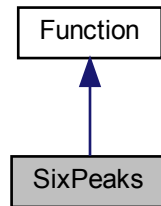
- lib/hnco/functions/collection/cancellation.hh
- lib/hnco/functions/collection/cancellation.cc

5.139 SixPeaks Class Reference

Six Peaks.

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

Inheritance diagram for SixPeaks:



Public Member Functions

- [SixPeaks](#) (int bv_size, int threshold)
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- bool [has_known_maximum](#) () const override
Check for a known maximum.
- double [get_maximum](#) () const override
Get the global maximum.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Private Attributes

- int [_bv_size](#)
Bit vector size.
- int [_threshold](#)
Threshold.
- int [_maximum](#)
Maximum.

5.139.1 Detailed Description

Six Peaks.

It is defined by

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

where:

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

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

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

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

Reference:

J. S. De Bonet, C. L. Isbell, and P. Viola. 1996. MIMIC: finding optima by estimating probability densities. In *Advances in Neural Information Processing Systems*. Vol. 9. MIT Press, Denver.

Definition at line 128 of file four-peaks.hh.

5.139.2 Member Function Documentation

5.139.2.1 `get_maximum()`

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

Get the global maximum.

Returns

$2 * _bv_size - _threshold - 1$

Reimplemented from [Function](#).

Definition at line 156 of file four-peaks.hh.

5.139.2.2 has_known_maximum()

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

Check for a known maximum.

Returns

true

Reimplemented from [Function](#).

Definition at line 152 of file four-peaks.hh.

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

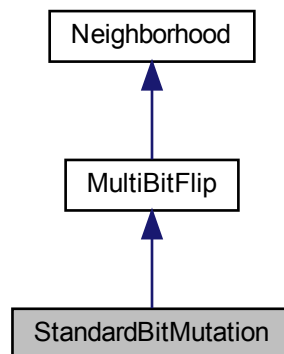
- lib/hnco/functions/collection/four-peaks.hh
- lib/hnco/functions/collection/four-peaks.cc

5.140 StandardBitMutation Class Reference

Standard bit mutation.

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

Inheritance diagram for StandardBitMutation:



Public Member Functions

- [StandardBitMutation](#) (int n)
Constructor.
- [StandardBitMutation](#) (int n, double p)
Constructor.
- void [set_mutation_rate](#) (double p)
Set mutation rate.

Setters

- void [set_allow_no_mutation](#) (bool b)
Set the flag `_allow_no_mutation`.

Private Member Functions

- void [sample_bits](#) ()
Sample bits.
- void [bernoulli_process](#) ()
Bernoulli process.

Private Attributes

- std::bernoulli_distribution [_bernoulli_dist](#)
Bernoulli distribution (biased coin)
- std::binomial_distribution< int > [_binomial_dist](#)
Binomial distribution.
- bool [_rejection_sampling](#) = false
Rejection sampling.

Parameters

- bool [_allow_no_mutation](#) = false
Allow no mutation.

Additional Inherited Members

5.140.1 Detailed Description

Standard bit mutation.

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

Definition at line 222 of file neighborhood.hh.

5.140.2 Constructor & Destructor Documentation

5.140.2.1 StandardBitMutation() [1/2]

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

Constructor.

Parameters

| | |
|-----|---------------------|
| n | Size of bit vectors |
|-----|---------------------|

The Bernoulli probability is set to $1 / n$.

Definition at line 257 of file neighborhood.hh.

5.140.2.2 StandardBitMutation() [2/2]

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

Constructor.

Parameters

| | |
|-----|-----------------------|
| n | Size of bit vectors |
| p | Bernoulli probability |

Definition at line 267 of file neighborhood.hh.

5.140.3 Member Function Documentation

5.140.3.1 set_mutation_rate()

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

Set mutation rate.

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

Definition at line 278 of file neighborhood.hh.

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

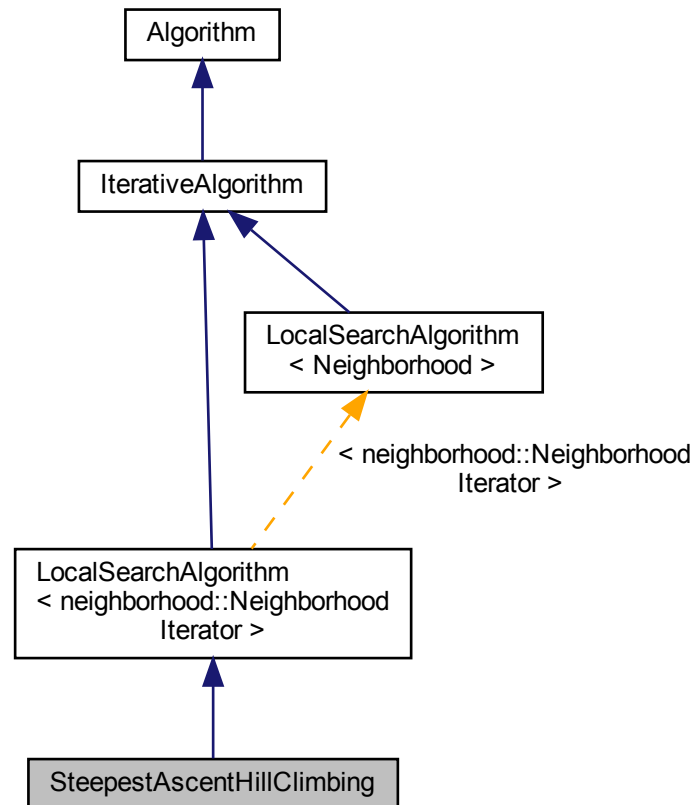
- lib/hnco/neighborhoods/neighborhood.hh
- lib/hnco/neighborhoods/neighborhood.cc

5.141 SteepestAscentHillClimbing Class Reference

Steepest ascent hill climbing.

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

Inheritance diagram for SteepestAscentHillClimbing:



Public Member Functions

- [SteepestAscentHillClimbing](#) (int n, [neighborhood::NeighborhoodIterator](#) *neighborhood)
Constructor.

Protected Member Functions

- void [iterate](#) () override
Single iteration.

Protected Attributes

- `std::vector< bit_vector_t > _candidates`
Potential candidate.

5.141.1 Detailed Description

Steepest ascent hill climbing.

Definition at line 34 of file `steepest-ascent-hill-climbing.hh`.

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

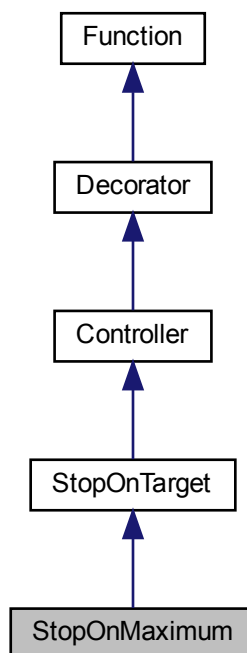
- `lib/hnco/algorithms/ls/steepest-ascent-hill-climbing.hh`
- `lib/hnco/algorithms/ls/steepest-ascent-hill-climbing.cc`

5.142 StopOnMaximum Class Reference

Stop on maximum.

```
#include <hnco/functions/controllers/controller.hh>
```

Inheritance diagram for StopOnMaximum:



Public Member Functions

- [StopOnMaximum](#) ([Function](#) *function)

Constructor.

Additional Inherited Members

5.142.1 Detailed Description

Stop on maximum.

Definition at line 136 of file controller.hh.

5.142.2 Constructor & Destructor Documentation

5.142.2.1 StopOnMaximum()

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

Constructor.

Precondition

function->[has_known_maximum\(\)](#)

Definition at line 143 of file controller.hh.

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

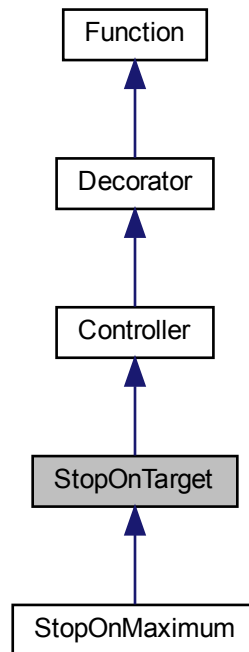
- lib/hnco/functions/controllers/controller.hh

5.143 StopOnTarget Class Reference

Stop on target.

```
#include <hnco/functions/controllers/controller.hh>
```

Inheritance diagram for StopOnTarget:



Public Member Functions

- [StopOnTarget](#) ([Function](#) *function, double target)
Constructor.
- const [algorithm::solution_t](#) & [get_trigger](#) ()
Get trigger.

Evaluation

- double [evaluate](#) (const [bit_vector_t](#) &)
Evaluate a bit vector.
- double [evaluate_incrementally](#) (const [bit_vector_t](#) &x, double value, const [hnco::sparse_bit_vector_t](#) &flipped_bits)
Incrementally evaluate a bit vector.
- void [update](#) (const [bit_vector_t](#) &x, double value)
Update after a safe evaluation.

Private Attributes

- `double _target`
Target.
- `algorithm::solution_t_trigger`
Trigger.

Additional Inherited Members

5.143.1 Detailed Description

Stop on target.

The member function `eval` throws an exception `TargetReached` when the value of its decorated function reaches a given target.

Warning

The target is detected using the greater or equal operator hence the result should be taken with care in case of non integer (floating point) function values.

Definition at line 92 of file `controller.hh`.

5.143.2 Constructor & Destructor Documentation

5.143.2.1 StopOnTarget()

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

Constructor.

Parameters

| | |
|-----------------|--------------------|
| <i>function</i> | Decorated function |
| <i>target</i> | Target |

Definition at line 107 of file `controller.hh`.

5.143.3 Member Function Documentation

5.143.3.1 evaluate()

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

Evaluate a bit vector.

Exceptions

| | |
|----------------------|--|
| <i>TargetReached</i> | |
|----------------------|--|

Implements [Function](#).

Definition at line 33 of file controller.cc.

5.143.3.2 evaluate_incrementally()

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

Incrementally evaluate a bit vector.

Exceptions

| | |
|----------------------|--|
| <i>TargetReached</i> | |
|----------------------|--|

Reimplemented from [Function](#).

Definition at line 46 of file controller.cc.

5.143.3.3 update()

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

Update after a safe evaluation.

Exceptions

| | |
|----------------------|--|
| <i>TargetReached</i> | |
|----------------------|--|

Reimplemented from [Function](#).

Definition at line 59 of file controller.cc.

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

- lib/hnco/functions/controllers/controller.hh
- lib/hnco/functions/controllers/controller.cc

5.144 Stopwatch Class Reference

Stop watch.

```
#include <hnco/stop-watch.hh>
```

Public Member Functions

- void [start](#) ()
Start.
- void [stop](#) ()
Stop.
- double [get_total_time](#) ()
Get total time.
- void [reset](#) ()
Reset.

Private Attributes

- double [_total_time](#) = 0
Total time.
- clock_t [_start](#)
Start time.

5.144.1 Detailed Description

Stop watch.

Definition at line 31 of file stop-watch.hh.

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

- lib/hnco/stop-watch.hh

5.145 Sudoku Class Reference

Sudoku

```
#include <hnco/functions/collection/sudoku.hh>
```

Public Types

- using `domain_type` = `std::size_t`
Domain type.
- using `codomain_type` = `double`
Codomain type.

Public Member Functions

- `Sudoku` ()
Default constructor.
- void `random` (int c)
Random instance.
- int `get_num_variables` ()
Get the number of variables.
- void `display` (std::ostream &stream) const
Display the problem.
- void `describe` (const std::vector< `domain_type` > &x, std::ostream &stream)
Describe a solution.
- double `evaluate` (const std::vector< `domain_type` > &x)
Evaluate a solution.

Private Member Functions

- void `write_variables` (const std::vector< `domain_type` > &x)
Write variables.

Private Attributes

- std::vector< std::vector< char > > `_problem_instance`
Problem instance.
- std::vector< std::vector< `domain_type` > > `_candidate`
Candidate.
- std::vector< int > `_counts`
Counts.
- int `_num_variables`
Number of variables.

Load and save instance

- void `load_` (std::istream &stream)
Load an instance.
- void `save_` (std::ostream &stream) const
Save an instance.
- void `load` (std::string path)
Load instance.
- void `save` (std::string path) const
Save instance.

5.145.1 Detailed Description

Sudoku

Definition at line 34 of file sudoku.hh.

5.145.2 Member Function Documentation

5.145.2.1 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to load |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 100 of file sudoku.hh.

5.145.2.2 load_()

```
void load_ (
    std::istream & stream ) [private]
```

Load an instance.

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 57 of file sudoku.cc.

5.145.2.3 random()

```
void random (
    int c )
```

Random instance.

Parameters

| | |
|----------|-----------------------|
| <i>c</i> | Number of empty cells |
|----------|-----------------------|

Definition at line 96 of file sudoku.cc.

5.145.2.4 save()

```
void save (
    std::string path ) const [inline]
```

Save instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to save |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 112 of file sudoku.hh.

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

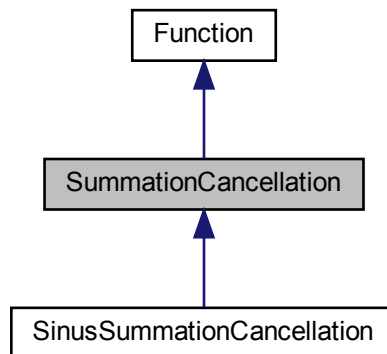
- lib/hnco/functions/collection/sudoku.hh
- lib/hnco/functions/collection/sudoku.cc

5.146 SummationCancellation Class Reference

Summation cancellation.

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

Inheritance diagram for SummationCancellation:



Public Member Functions

- [SummationCancellation](#) (int n)
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- bool [has_known_maximum](#) () const override
Check for a known maximum.
- double [get_maximum](#) () const override
Get the global maximum.
- double [evaluate](#) (const [bit_vector_t](#) &x) override
Evaluate a bit vector.

Protected Member Functions

- void [convert](#) (const [bit_vector_t](#) &x)
Convert a bit vector into a real vector.

Protected Attributes

- int [_bv_size](#)
Bit vector size.
- std::vector< double > [_buffer](#)
Buffer.

5.146.1 Detailed Description

Summation cancellation.

Encoding of a signed integer:

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

Reference:

S. Baluja and S. Davies. 1997. Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space. Technical Report CMU- CS-97-107. Carnegie-Mellon University.

Definition at line 46 of file cancellation.hh.

5.146.2 Constructor & Destructor Documentation

5.146.2.1 SummationCancellation()

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

Constructor.

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

Parameters

| | |
|-----|------------------------|
| n | Size of the bit vector |
|-----|------------------------|

Definition at line 68 of file cancellation.hh.

5.146.3 Member Function Documentation

5.146.3.1 has_known_maximum()

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

Check for a known maximum.

Returns

true

Reimplemented from [Function](#).

Definition at line 81 of file cancellation.hh.

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

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

5.147 SymmetricWalshMoment2 Struct Reference

Symmetric Walsh moment.

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

Public Member Functions

- [SymmetricWalshMoment2](#) (int n)
Constructor.
- void [display](#) (std::ostream &stream)
Display Walsh moment.
- void [init](#) ()
Initialize Walsh moment.
- void [add](#) (const [bit_vector_t](#) &bv)
Add a bit vector to a Walsh moment.
- void [average](#) (int count)
Average each Walsh moment.
- void [update](#) (const [SymmetricWalshMoment2](#) &wm, double rate)
Update a Walsh moment.
- void [update](#) (const [SymmetricWalshMoment2](#) &wm1, const [SymmetricWalshMoment2](#) &wm2, double rate)
Update a Walsh moment.
- void [scaled_difference](#) (double lambda, const [SymmetricWalshMoment2](#) &wm1, const [SymmetricWalshMoment2](#) &wm2)
Compute a scaled difference between two moments.
- void [bound](#) (double margin)
Bound Walsh moment.
- double [norm_1](#) () const
1-norm of the Walsh moment
- double [norm_2](#) () const
2-norm of the Walsh moment
- double [norm_infinite](#) () const
infinite-norm of the Walsh moment
- double [distance](#) (const [SymmetricWalshMoment2](#) &wm) const
distance between the Walsh moment and another Walsh moment

Public Attributes

- `std::vector< double > first_moment`
First moment.
- `std::vector< std::vector< double > > second_moment`
Second moment.

5.147.1 Detailed Description

Symmetric Walsh moment.

Definition at line 144 of file walsh-moment.hh.

5.147.2 Constructor & Destructor Documentation

5.147.2.1 SymmetricWalshMoment2()

```
SymmetricWalshMoment2 (
    int n )
```

Constructor.

Parameters

| | |
|----------|--------------------|
| <i>n</i> | Size of bit vector |
|----------|--------------------|

Definition at line 236 of file walsh-moment.cc.

5.147.3 Member Function Documentation

5.147.3.1 average()

```
void average (
    int count )
```

Average each Walsh moment.

Postcondition

```
matrix_is_symmetric(second_moment)
```

Definition at line 296 of file walsh-moment.cc.

5.147.3.2 bound()

```
void bound (
    double margin )
```

Bound Walsh moment.

Ensure that the distance from each Walsh moment to the -1/1 bounds is greater or equal to the given margin.

Parameters

| | |
|---------------|-------------------------------|
| <i>margin</i> | Distance from the -1/1 bounds |
|---------------|-------------------------------|

Definition at line 378 of file walsh-moment.cc.

5.147.3.3 display()

```
void display (
    std::ostream & stream )
```

Display Walsh moment.

A [SymmetricWalshMoment2](#) is displayed as a full symmetric matrix with diagonal entries equal to first moments and off-diagonal entries equal to second moments.

Definition at line 247 of file walsh-moment.cc.

5.147.3.4 scaled_difference()

```
void scaled_difference (
    double lambda,
    const SymmetricWalshMoment2 & wm1,
    const SymmetricWalshMoment2 & wm2 )
```

Compute a scaled difference between two moments.

This member function implements:

$\text{self} = \text{lambda} * \text{wm1} - \text{wm2}$

It is mostly useful in herding ([Hea](#)).

Parameters

| | |
|---------------|---------------------|
| <i>lambda</i> | Scale |
| <i>wm1</i> | First Walsh moment |
| <i>wm2</i> | Second Walsh moment |

Definition at line 357 of file walsh-moment.cc.

5.147.3.5 update() [1/2]

```
void update (
    const SymmetricWalshMoment2 & wm,
    double rate )
```

Update a Walsh moment.

This member function implements:

`self += rate * (wm1 - self)`

Parameters

| | |
|-------------|---------------------|
| <i>wm</i> | Target Walsh moment |
| <i>rate</i> | Learning rate |

Postcondition

For all i , `is_in_interval(first_moment[i], -1, 1)`

For all $i \neq j$, `is_in_interval(second_moment[i][j], -1, 1)`

`matrix_is_symmetric(second_moment)`

Definition at line 314 of file walsh-moment.cc.

5.147.3.6 update() [2/2]

```
void update (
    const SymmetricWalshMoment2 & wm1,
    const SymmetricWalshMoment2 & wm2,
    double rate )
```

Update a Walsh moment.

This member function implements:

`self += rate * (wm1 - wm2)`

The resulting entries are not necessarily those of a Walsh moment, that is

`is_in_interval(first_moment[i], -1, 1)` or

`is_in_interval(second_moment[i][j], -1, 1)`

might fail for some $i \neq j$.

Parameters

| | |
|-------------|--------------------------------|
| <i>wm1</i> | Target Walsh moment |
| <i>wm2</i> | Walsh moment to move away from |
| <i>rate</i> | Learning rate |

Definition at line 335 of file walsh-moment.cc.

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

- lib/hnco/algorithms/walsh-moment/walsh-moment.hh
- lib/hnco/algorithms/walsh-moment/walsh-moment.cc

5.148 SymmetricWalshMoment2GibbsSampler Class Reference

Gibbs sampler with symmetric Walsh moments.

```
#include <hnco/algorithms/walsh-moment/gibbs-sampler.hh>
```

Public Types

- using [Moment](#) = [SymmetricWalshMoment2](#)
Walsh moment type.

Public Member Functions

- [SymmetricWalshMoment2GibbsSampler](#) (int n, const [SymmetricWalshMoment2](#) &mp)
Constructor.
- void [init](#) ()
Initialize.
- void [update](#) (int i)
Update state.
- void [update_sync](#) ()
Update state synchronously.
- const [bit_vector_t](#) & [get_state](#) ()
Get the state of the Gibbs sampler.

Private Attributes

- const [SymmetricWalshMoment2](#) & [_model_parameters](#)
Model parameters.
- [bit_vector_t](#) [_state](#)
State of the Gibbs sampler.
- [pv_t](#) [_pv](#)
Probability vector for synchronous Gibbs sampling.

5.148.1 Detailed Description

Gibbs sampler with symmetric Walsh moments.

Definition at line 75 of file gibbs-sampler.hh.

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

- lib/hnco/algorithms/walsh-moment/gibbs-sampler.hh
- lib/hnco/algorithms/walsh-moment/gibbs-sampler.cc

5.149 SymmetricWalshMoment2Herding Class Reference

Herding with symmetric Walsh moment.

```
#include <hnco/algorithms/walsh-moment/herding.hh>
```

Public Types

- using [Moment](#) = [SymmetricWalshMoment2](#)
Walsh moment type.

Public Member Functions

- [SymmetricWalshMoment2Herding](#) (int n)
Constructor.
- void [init](#) ()
Initialization.
- void [sample](#) (const [SymmetricWalshMoment2](#) &target, [bit_vector_t](#) &x)
Sample a bit vector.
- double [error](#) (const [SymmetricWalshMoment2](#) &target)
Compute the error.

Getters

- const [SymmetricWalshMoment2](#) & [get_delta](#) () const
Get delta.

Setters

- void [set_randomize_bit_order](#) (bool x)
Randomize bit order.

Protected Attributes

- [SymmetricWalshMoment2_delta](#)
Delta moment.
- [SymmetricWalshMoment2_count](#)
Counter moment.
- [SymmetricWalshMoment2_error](#)
Error moment.
- [permutation_t_permutation](#)
Permutation.
- [int_time](#)
Time.

Parameters

- [bool_randomize_bit_order](#) = false
Randomize bit order.

5.149.1 Detailed Description

Herdling with symmetric Walsh moment.

Definition at line 112 of file herding.hh.

5.149.2 Constructor & Destructor Documentation

5.149.2.1 SymmetricWalshMoment2Herdling()

```
SymmetricWalshMoment2Herdling (
    int n ) [inline]
```

Constructor.

Parameters

| | |
|----------|---------------------|
| <i>n</i> | Size of bit vectors |
|----------|---------------------|

Definition at line 149 of file herding.hh.

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

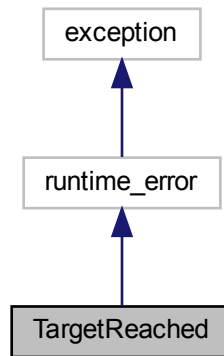
- lib/hnco/algorithms/walsh-moment/herding.hh
- lib/hnco/algorithms/walsh-moment/herding.cc

5.150 TargetReached Class Reference

Target reached.

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

Inheritance diagram for TargetReached:



5.150.1 Detailed Description

Target reached.

Definition at line 40 of file exception.hh.

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

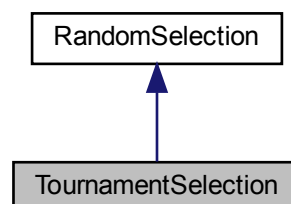
- lib/hnco/exception.hh

5.151 TournamentSelection Class Reference

Tournament selection.

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

Inheritance diagram for TournamentSelection:



Public Member Functions

- [TournamentSelection](#) (const [Population](#) &population)
Constructor.
- void [init](#) () override
Initialize.
- const [bit_vector_t](#) & [select](#) () override
Select an individual in the population.

Setters

- void [set_tournament_size](#) (int n)
Set the tournament size.

Private Attributes

- [hnco::multiobjective::algorithm::TournamentSelection](#)< double, std::greater< double > > [_tournament_selection](#)
Tournament selection.

Parameters

- int [_tournament_size](#) = 2
Tournament size.

Additional Inherited Members

5.151.1 Detailed Description

Tournament selection.

Reuses the [hnco::multiobjective::algorithm::TournamentSelection](#) class.

Definition at line 93 of file random-selection.hh.

5.151.2 Constructor & Destructor Documentation

5.151.2.1 TournamentSelection()

```
TournamentSelection (
    const Population & population ) [inline]
```

Constructor.

Parameters

| | |
|-------------------|---------------------------|
| <i>population</i> | Population to select from |
|-------------------|---------------------------|

Definition at line 115 of file random-selection.hh.

5.151.3 Member Function Documentation

5.151.3.1 select()

```
const bit_vector_t & select ( ) [override], [virtual]
```

Select an individual in the population.

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

Precondition

The population must be evaluated.

Implements [RandomSelection](#).

Definition at line 45 of file random-selection.cc.

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

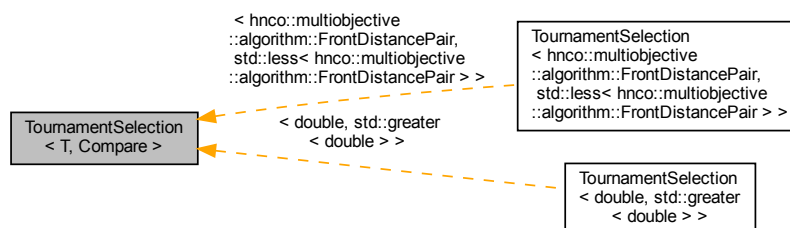
- lib/hnco/algorithms/ea/random-selection.hh
- lib/hnco/algorithms/ea/random-selection.cc

5.152 TournamentSelection< T, Compare > Class Template Reference

Tournament selection.

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

Inheritance diagram for TournamentSelection< T, Compare >:



Public Member Functions

- **TournamentSelection** (const std::vector< [bit_vector_t](#) > &bvs, const std::vector< T > &values)
Constructor.
- void **init** ()
Initialize.
- const [bit_vector_t](#) & **select** ()
Select a bit vector.

Setters

- void **set_tournament_size** (int n)
Set the tournament size.

Private Attributes

- const std::vector< [bit_vector_t](#) > & [_bvs](#)
Bit vectors.
- const std::vector< T > & [_values](#)
Values.
- [hnco::permutation_t](#) [_permutation](#)
Permutation.
- int [_start](#)
Beginning of the slice of permutation used in a tournament round.
- int [_stop](#)
End of the slice of permutation used in a tournament round.
- Compare [_compare](#)
Comparison operator.

Parameters

- int [_tournament_size](#) = 2
Tournament size.

5.152.1 Detailed Description

```
template<typename T, typename Compare>
class hnco::multiobjective::algorithm::TournamentSelection< T, Compare >
```

Tournament selection.

Implement tournament selection without replacement as explained in the reference:

Goldberg, Korb, and Deb, "Messy genetic algorithms: Motivation, analysis, and first results", Complex systems, 1989.

https://www.complex-systems.com/abstracts/v03_i05_a05/

Definition at line 45 of file random-selection.hh.

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

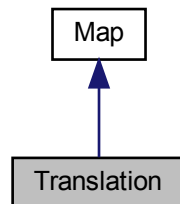
- lib/hnco/multiobjective/algorithms/random-selection.hh

5.153 Translation Class Reference

Translation.

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

Inheritance diagram for Translation:



Public Member Functions

- void [map](#) (const [bit_vector_t](#) &input, [bit_vector_t](#) &output) override
Map
- int [get_input_size](#) () const override
Get input size.
- int [get_output_size](#) () const override
Get output size.
- bool [is_surjective](#) () const override
Check for surjective map.
- void [display](#) (std::ostream &stream) const override
Display.
- void [random](#) (int n)
Random instance.
- void [set_bv](#) (const [bit_vector_t](#) &bv)
Set the translation vector.

Load and save map

- void [load](#) (std::string path)
Load map.
- void [save](#) (std::string path) const
Save map.

Private Member Functions

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

Private Attributes

- [bit_vector_t _bv](#)
Translation vector

Friends

- class `boost::serialization::access`

5.153.1 Detailed Description

Translation.

A translation is an affine map f from F_2y^n to itself defined by $f(x) = x + b$, where b is an n -dimensional bit vector.

Definition at line 80 of file `map.hh`.

5.153.2 Member Function Documentation

5.153.2.1 `is_surjective()`

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

Check for surjective map.

Returns

true

Reimplemented from [Map](#).

Definition at line 122 of file `map.hh`.

5.153.2.2 `load()`

```
void load (
    std::string path ) [inline]
```

Load map.

Parameters

| | |
|-------------|------------------|
| <i>path</i> | Path of the file |
|-------------|------------------|

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 147 of file map.hh.

5.153.2.3 save()

```
void save (
    std::string path ) const [inline]
```

Save map.

Parameters

| | |
|-------------|------------------|
| <i>path</i> | Path of the file |
|-------------|------------------|

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 154 of file map.hh.

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

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

5.154 Transvection Struct Reference

Transvection.

```
#include <hnco/maps/transvection.hh>
```

Public Member Functions

- template<class Archive >
void [save](#) (Archive &ar, const unsigned int version) const
Save.
- template<class Archive >
void [load](#) (Archive &ar, const unsigned int version)
Load.
- bool [is_valid](#) () const
Check validity.
- bool [is_valid](#) (int n) const

- *Check validity.*
- void `display` (std::ostream &stream) const
- *Display transvection.*
- void `random` (int n)
- *Sample a random transvection.*
- void `random_non_commuting` (int n, const `Transvection` &a)
- *Sample a random transvection.*
- void `multiply` (`bit_vector_t` &x) const
- *Multiply a bit vector from the left.*
- void `multiply` (`bit_matrix_t` &M) const
- *Multiply a bit matrix from the left.*

Public Attributes

- int `row_index`
- *Row index.*
- int `column_index`
- *Column index.*

5.154.1 Detailed Description

Transvection.

We only consider transvections defined by matrices $\tau_{ij} = I_n + B_{ij}$, where I_n is the $n \times n$ identity matrix and B_{ij} is the matrix whose (i, j) entry is 1 and other entries are zero. Such a matrix is also sometimes called a shear matrix.

Transvections generate invertible matrices over the finite field F_2 .

Definition at line 63 of file transvection.hh.

5.154.2 Member Function Documentation

5.154.2.1 `is_valid()`

```
bool is_valid (
    int n ) const
```

Check validity.

Parameters

| | |
|----------------|-----------|
| <code>n</code> | Dimension |
|----------------|-----------|

Definition at line 48 of file transvection.cc.

5.154.2.2 multiply() [1/2]

```
void multiply (
    bit_matrix_t & M ) const
```

Multiply a bit matrix from the left.

Parameters

| | |
|----------|------------|
| <i>M</i> | Bit matrix |
|----------|------------|

Precondition

```
is_valid()
is_valid(bm_num_rows(M))
```

Warning

This function modifies the given bit vector.

Definition at line 117 of file transvection.cc.

5.154.2.3 multiply() [2/2]

```
void multiply (
    bit_vector_t & x ) const
```

Multiply a bit vector from the left.

Parameters

| | |
|----------|------------|
| <i>x</i> | Bit vector |
|----------|------------|

Precondition

```
is_valid()
is_valid(x.size())
```

Warning

This function modifies the given bit vector.

Definition at line 105 of file transvection.cc.

5.154.2.4 random()

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

Sample a random transvection.

Parameters

| | |
|-----|-----------|
| n | Dimension |
|-----|-----------|

Precondition

$n > 1$

Definition at line 61 of file transvection.cc.

5.154.2.5 random_non_commuting()

```
void random_non_commuting (
    int n,
    const Transvection & a )
```

Sample a random transvection.

This member function ensures that the sampled transvection does not commute with some given one.

Parameters

| | |
|-----|--------------------|
| n | Dimension |
| a | Given transvection |

Precondition

$n > 1$

Definition at line 77 of file transvection.cc.

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

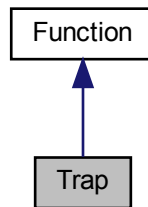
- lib/hnco/maps/transvection.hh
- lib/hnco/maps/transvection.cc

5.155 Trap Class Reference

Trap.

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

Inheritance diagram for Trap:



Public Member Functions

- [Trap](#) (int bv_size, int num_traps)
Constructor.
- int [get_bv_size](#) () const
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &)
Evaluate a bit vector.
- bool [has_known_maximum](#) () const
Check for a known maximum.
- double [get_maximum](#) () const
Get the global maximum.

Private Attributes

- int [_bv_size](#)
Bit vector size.
- int [_num_traps](#)
Number of traps.
- int [_trap_size](#)
Trap size

5.155.1 Detailed Description

Trap.

Reference:

Kalyanmoy Deb and David E. Goldberg. 1993. Analyzing Deception in Trap Functions. In Foundations of Genetic Algorithms 2, L. Darrell Whitley (Ed.). Morgan Kaufmann, San Mateo, CA, 93–108.

Definition at line 43 of file trap.hh.

5.155.2 Constructor & Destructor Documentation

5.155.2.1 Trap()

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

Constructor.

Parameters

| | |
|------------------|-----------------|
| <i>bv_size</i> | Bit vector size |
| <i>num_traps</i> | Number of traps |

Warning

bv_size must be a multiple of *num_traps*

Definition at line 64 of file trap.hh.

5.155.3 Member Function Documentation

5.155.3.1 get_maximum()

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

Get the global maximum.

Returns

_bv_size

Reimplemented from [Function](#).

Definition at line 88 of file trap.hh.

5.155.3.2 has_known_maximum()

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

Check for a known maximum.

Returns

true

Reimplemented from [Function](#).

Definition at line 84 of file trap.hh.

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

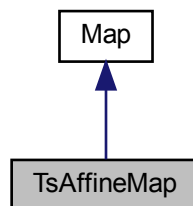
- lib/hnco/functions/collection/trap.hh
- lib/hnco/functions/collection/trap.cc

5.156 TsAffineMap Class Reference

Transvection sequence affine map.

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

Inheritance diagram for TsAffineMap:



Public Types

- enum [SamplingMode](#) {
 [Unconstrained](#) , [CommutingTransvections](#) , [UniqueSource](#) , [UniqueDestination](#) ,
 [DisjointTransvections](#) , [NonCommutingTransvections](#) }
 Sampling mode.

Public Member Functions

- void [random](#) (int n, int t, [SamplingMode](#) mode)
Random instance.
- void [map](#) (const [bit_vector_t](#) &input, [bit_vector_t](#) &output) override
Map
- int [get_input_size](#) () const override
Get input size.
- int [get_output_size](#) () const override
Get output size.
- bool [is_surjective](#) () const override
Check for surjective map.
- void [display](#) (std::ostream &stream) const override
Display.
- void [inverse](#) ()
Inverse.

Load and save map

- void [load](#) (std::string path)
Load map.
- void [save](#) (std::string path) const
Save map.

Private Member Functions

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

Private Attributes

- [transvection_sequence_t_ts](#)
Transvection sequence
- [bit_vector_t_bv](#)
Translation vector

Friends

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

5.156.1 Detailed Description

Transvection sequence affine map.

An affine map f from F_2^m to F_2^n is defined by $f(x) = Ax + b$, where A is an $n \times m$ bit matrix and b is an n -dimensional bit vector.

In [TsAffineMap](#), A is a finite product of transvections represented by a [transvection_sequence_t](#).

Definition at line 601 of file [map.hh](#).

5.156.2 Member Enumeration Documentation

5.156.2.1 SamplingMode

enum [SamplingMode](#)

Sampling mode.

Enumerator

| | |
|---------------------------|---|
| Unconstrained | Unconstrained. |
| CommutingTransvections | Commuting transvections. |
| UniqueSource | Transvection sequence with unique source |
| UniqueDestination | Transvection sequence with unique destination |
| DisjointTransvections | Disjoint transvections. |
| NonCommutingTransvections | Non commuting transvections. |

Definition at line 637 of file map.hh.

5.156.3 Member Function Documentation

5.156.3.1 is_surjective()

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

Check for surjective map.

Returns

true

Reimplemented from [Map](#).

Definition at line 680 of file map.hh.

5.156.3.2 load()

```
void load (
    std::string path ) [inline]
```

Load map.

Parameters

| | |
|-------------|------------------|
| <i>path</i> | Path of the file |
|-------------|------------------|

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 697 of file map.hh.

5.156.3.3 random()

```
void random (
    int n,
    int t,
    SamplingMode mode )
```

Random instance.

Parameters

| | |
|-------------|-------------------------------------|
| <i>n</i> | Dimension |
| <i>t</i> | Length of sequence of transvections |
| <i>mode</i> | Sampling mode |

Definition at line 217 of file map.cc.

5.156.3.4 save()

```
void save (
    std::string path ) const [inline]
```

Save map.

Parameters

| | |
|-------------|------------------|
| <i>path</i> | Path of the file |
|-------------|------------------|

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 704 of file map.hh.

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

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

5.157 Tsp Class Reference

Traveling salesman problem.

```
#include <hnco/functions/collection/tsp.hh>
```

Public Member Functions

- [Tsp](#) ()
Default constructor.
- int [get_num_elements](#) () const
Get the number of elements.
- void [display](#) (std::ostream &stream) const
Display the problem.
- void [describe](#) (const [hnco::permutation_t](#) &permutation, std::ostream &stream)
Describe a solution.
- double [evaluate](#) (const [hnco::permutation_t](#) &permutation)
Evaluate a solution.

Instance generators

- template<class Generator >
void [generate](#) (int n, Generator generator)
Instance generator.
- void [random](#) (int n)
Random instance.

Private Types

- enum { **ATT** , **EUC_2D** }

Private Member Functions

- void [compute_distances](#) ()
- void [compute_distances_att](#) ()
- void [compute_distances_euc_2d](#) ()

Private Attributes

- `std::string _name`
- `std::string _comment`
- `int _num_cities`
- `std::vector< float > _x`
- `std::vector< float > _y`
- `int _edge_weight_type = ATT`
- `std::vector< std::vector< float > > _distances`

Distances.

Load and save instance

- `void load_ (std::istream &stream)`
Load an instance.
- `void load_coordinates (std::istream &stream)`
- `void save_ (std::ostream &stream) const`
Save an instance.
- `void load (std::string path)`
Load instance.
- `void save (std::string path) const`
Save instance.

5.157.1 Detailed Description

Traveling salesman problem.

Source: TSPLIB 95, Gerhard Reinelt

Definition at line 40 of file tsp.hh.

5.157.2 Member Function Documentation

5.157.2.1 generate()

```
void generate (
    int n,
    Generator generator ) [inline]
```

Instance generator.

Parameters

| | |
|------------------|-------------------------|
| <i>n</i> | Number of vertices |
| <i>generator</i> | Generator for distances |

Definition at line 94 of file tsp.hh.

5.157.2.2 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to load |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 129 of file tsp.hh.

5.157.2.3 load_()

```
void load_ (
    std::istream & stream ) [private]
```

Load an instance.

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 32 of file tsp.cc.

5.157.2.4 random()

```
void random (
    int n ) [inline]
```

Random instance.

Distances are sampled from the normal distribution.

Parameters

| | |
|----------|--------------------|
| <i>n</i> | Number of vertices |
|----------|--------------------|

Definition at line 113 of file tsp.hh.

5.157.2.5 save()

```
void save (
    std::string path ) const [inline]
```

Save instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to save |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 141 of file tsp.hh.

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

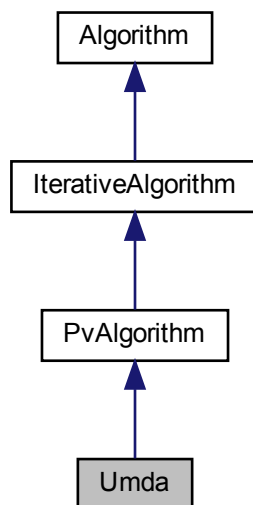
- lib/hnco/functions/collection/tsp.hh
- lib/hnco/functions/collection/tsp.cc

5.158 Umda Class Reference

Univariate marginal distribution algorithm.

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

Inheritance diagram for Umda:



Public Member Functions

- [Umda](#) (int n, int population_size)
Constructor.

Setters

- void [set_selection_size](#) (int x)
Set the selection size.

Protected Member Functions

Loop

- void [init](#) () override
Initialize.
- void [iterate](#) () override
Single iteration.

Protected Attributes

- [Population_population](#)
Population.

Parameters

- int [_selection_size](#) = 1
Selection size.

5.158.1 Detailed Description

Univariate marginal distribution algorithm.

Reference:

H. Mühlenbein. 1997. The equation for response to selection and its use for prediction. *Evolutionary Computation* 5, 3 (1997), 303–346.

Definition at line 41 of file umda.hh.

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

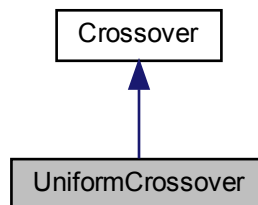
- lib/hnco/algorithms/pv/umda.hh
- lib/hnco/algorithms/pv/umda.cc

5.159 UniformCrossover Class Reference

Uniform crossover.

```
#include <hnco/algorithms/ea/crossover.hh>
```

Inheritance diagram for UniformCrossover:



Public Member Functions

- void `recombine` (const `bit_vector_t` &parent1, const `bit_vector_t` &parent2, `bit_vector_t` &offspring)
Recombine.

5.159.1 Detailed Description

Uniform crossover.

Definition at line 56 of file crossover.hh.

5.159.2 Member Function Documentation

5.159.2.1 recombine()

```
void recombine (
    const bit\_vector\_t & parent1,
    const bit\_vector\_t & parent2,
    bit\_vector\_t & offspring ) [virtual]
```

Recombine.

The offspring is the uniform crossover of two parents.

Parameters

| | |
|------------------|---------------|
| <i>parent1</i> | First parent |
| <i>parent2</i> | Second parent |
| <i>offspring</i> | Offspring |

Implements [Crossover](#).

Definition at line 30 of file `crossover.cc`.

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

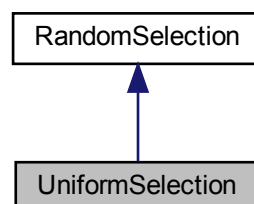
- `lib/hnco/algorithms/ea/crossover.hh`
- `lib/hnco/algorithms/ea/crossover.cc`

5.160 UniformSelection Class Reference

Uniform selection.

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

Inheritance diagram for UniformSelection:



Public Member Functions

- [UniformSelection](#) (const [Population](#) &population)
Constructor.
- const [bit_vector_t](#) & [select](#) () override
Select an individual in the population.

Private Attributes

- std::uniform_int_distribution< int > [_choose_individual](#)
Random index.

Additional Inherited Members

5.160.1 Detailed Description

Uniform selection.

Definition at line 66 of file random-selection.hh.

5.160.2 Constructor & Destructor Documentation

5.160.2.1 UniformSelection()

```
UniformSelection (
    const Population & population ) [inline]
```

Constructor.

Parameters

| | |
|-------------------|---------------------------|
| <i>population</i> | Population to select from |
|-------------------|---------------------------|

Definition at line 78 of file random-selection.hh.

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

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

5.161 UniversalFunction Class Reference

Universal function.

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


Public Member Functions

- virtual [~UniversalFunction](#) ()
Destructor.
- virtual double [evaluate](#) (const [bit_vector_t](#) &boolean_vars, const std::vector< int > &integer_vars, const std::vector< double > &float_vars, const std::vector< std::complex< double >> &complex_vars, const std::vector< int > &categorical_vars, const std::vector< [permutation_t](#) > &permutation_vars)=0
Evaluate the function.
- virtual void [display](#) (std::ostream &stream) const
Display the function.
- virtual void [describe](#) (const [bit_vector_t](#) &boolean_vars, const std::vector< int > &integer_vars, const std::vector< double > &float_vars, const std::vector< std::complex< double >> &complex_vars, const std::vector< int > &categorical_vars, const std::vector< [permutation_t](#) > &permutation_vars, std::ostream &stream)
Describe variables in the context of the function.

5.161.1 Detailed Description

Universal function.

A universal function is a function taking parameters of all types (boolean, integer, float, complex, categorical, permutation) and returning a double.

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

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

- lib/hnco/functions/universal-function.hh

5.162 UniversalFunction Class Reference

Universal function.

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

Public Member Functions

- virtual [~UniversalFunction](#) ()
Destructor.
- virtual int [get_output_size](#) () const =0
Get output size (number of objectives)
- virtual void [evaluate](#) (const [bit_vector_t](#) &boolean_vars, const std::vector< int > &integer_vars, const std::vector< double > &float_vars, const std::vector< std::complex< double >> &complex_vars, const std::vector< int > &categorical_vars, const std::vector< [permutation_t](#) > &permutation_vars, [value_t](#) &value)=0
Evaluate the function.
- virtual void [display](#) (std::ostream &stream) const
Display the function.
- virtual void [describe](#) (const [bit_vector_t](#) &boolean_vars, const std::vector< int > &integer_vars, const std::vector< double > &float_vars, const std::vector< std::complex< double >> &complex_vars, const std::vector< int > &categorical_vars, const std::vector< [permutation_t](#) > &permutation_vars, std::ostream &stream)
Describe variables in the context of the function.

5.162.1 Detailed Description

Universal function.

A universal function is a function taking parameters of all types (boolean, integer, float, complex, categorical, permutation) and returning a double.

Definition at line 43 of file universal-function.hh.

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

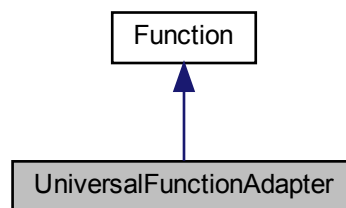
- lib/hnco/multiobjective/functions/universal-function.hh

5.163 UniversalFunctionAdapter Class Reference

Universal function adapter.

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

Inheritance diagram for UniversalFunctionAdapter:



Public Member Functions

- [UniversalFunctionAdapter](#) ([UniversalFunction](#) *fn, int num_boolean_vars, std::vector< [representation::DyadicIntegerRepresentation](#) > integer_reps, std::vector< [representation::DyadicFloatRepresentation](#) < double >> float_reps, std::vector< [representation::DyadicComplexRepresentation](#) < double >> complex_reps, std::vector< [representation::LinearCategoricalRepresentation](#) > categorical_reps, std::vector< [representation::PermutationRepresentation](#) > permutation_reps)
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &bv) override
Evaluate a bit vector.
- void [display](#) (std::ostream &stream) const override
Display.
- void [describe](#) (const [bit_vector_t](#) &bv, std::ostream &stream) override
Describe a bit vector.

Private Member Functions

- void `unpack` (const `bit_vector_t` &bv)
Unpack bit vector into variables.

Private Attributes

- `UniversalFunction` * `_function`
Universal function.
- `std::vector< representation::DyadicIntegerRepresentation< int > >` `_integer_reps`
Integer representations.
- `std::vector< representation::DyadicFloatRepresentation< double > >` `_float_reps`
Float representations.
- `std::vector< representation::DyadicComplexRepresentation< double > >` `_complex_reps`
Complex representations.
- `std::vector< representation::LinearCategoricalRepresentation >` `_categorical_reps`
Categorical representations.
- `std::vector< representation::PermutationRepresentation >` `_permutation_reps`
Permutation representations.
- `bit_vector_t` `_boolean_vars`
Boolean variables.
- `std::vector< int >` `_integer_vars`
Integer variables.
- `std::vector< double >` `_float_vars`
Float variables.
- `std::vector< std::complex< double > >` `_complex_vars`
Complex variables.
- `std::vector< int >` `_categorical_vars`
Categorical variables.
- `std::vector< permutation_t >` `_permutation_vars`
Permutation variables.
- `int` `_bv_size`
Bit vector size.

5.163.1 Detailed Description

Universal function adapter.

A universal function adapter turns a universal function into a regular hnco function defined on bit vectors.

Definition at line 45 of file `universal-function-adapter.hh`.

5.163.2 Constructor & Destructor Documentation

5.163.2.1 UniversalFunctionAdapter()

```

UniversalFunctionAdapter (
    UniversalFunction * fn,
    int num_boolean_vars,
    std::vector< representation::DyadicIntegerRepresentation< int >> integer_reps,
    std::vector< representation::DyadicFloatRepresentation< double >> float_reps,
    std::vector< representation::DyadicComplexRepresentation< double >> complex_reps,
    reps,
    std::vector< representation::LinearCategoricalRepresentation > categorical_reps,
    std::vector< representation::PermutationRepresentation > permutation_reps ) [inline]

```

Constructor.

Parameters

| | |
|-------------------------|-----------------------------|
| <i>fn</i> | Universal function |
| <i>num_boolean_vars</i> | Number of boolean variables |
| <i>integer_reps</i> | Integer representations |
| <i>float_reps</i> | Float representations |
| <i>complex_reps</i> | Complex representations |
| <i>categorical_reps</i> | Categorical representations |
| <i>permutation_reps</i> | Permutation representations |

Replace reps with {} if there is no corresponding variable. For example, if there is no categorical variable,

`UniversalFunctionAdapter(fn, num_boolean_vars, integer_reps, float_reps, complex_reps, {}, permutation_reps)`

Definition at line 132 of file universal-function-adapter.hh.

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

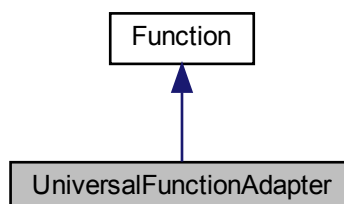
- lib/hnco/functions/universal-function-adapter.hh

5.164 UniversalFunctionAdapter Class Reference

Universal function adapter.

```
#include <hnco/multiobjective/functions/universal-function-adapter.hh>
```

Inheritance diagram for UniversalFunctionAdapter:



Public Member Functions

- [UniversalFunctionAdapter](#) ([UniversalFunction](#) *fn, int num_boolean_vars, std::vector< [representation::DyadicIntegerRepresentation](#)< int >> integer_reps, std::vector< [representation::DyadicFloatRepresentation](#)< double >> float_reps, std::vector< [representation::DyadicComplexRepresentation](#)< double >> complex_reps, std::vector< [representation::LinearCategoricalRepresentation](#)> categorical_reps, std::vector< [representation::PermutationRepresentation](#)> permutation_reps)
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- int [get_output_size](#) () const override
Get output size (number of objectives)
- void [evaluate](#) (const [bit_vector_t](#) &bv, [value_t](#) &value) override
Evaluate a bit vector.
- void [display](#) (std::ostream &stream) const override
Display.
- void [describe](#) (const [bit_vector_t](#) &bv, std::ostream &stream) override
Describe a bit vector.

Private Member Functions

- void [unpack](#) (const [bit_vector_t](#) &bv)
Unpack bit vector into variables.

Private Attributes

- [UniversalFunction](#) * [_function](#)
Universal function.
- std::vector< [representation::DyadicIntegerRepresentation](#)< int >> [_integer_reps](#)
Integer representations.
- std::vector< [representation::DyadicFloatRepresentation](#)< double >> [_float_reps](#)
Float representations.
- std::vector< [representation::DyadicComplexRepresentation](#)< double >> [_complex_reps](#)
Complex representations.
- std::vector< [representation::LinearCategoricalRepresentation](#)> [_categorical_reps](#)
Categorical representations.
- std::vector< [representation::PermutationRepresentation](#)> [_permutation_reps](#)
Permutation representations.
- [bit_vector_t](#) [_boolean_vars](#)
Boolean variables.
- std::vector< int > [_integer_vars](#)
Integer variables.
- std::vector< double > [_float_vars](#)
Float variables.
- std::vector< std::complex< double >> [_complex_vars](#)
Complex variables.
- std::vector< int > [_categorical_vars](#)
Categorical variables.
- std::vector< [permutation_t](#) > [_permutation_vars](#)
Permutation variables.
- int [_bv_size](#)
Bit vector size.

5.164.1 Detailed Description

Universal function adapter.

A universal function adapter turns a universal function into a regular hnco function defined on bit vectors.

Definition at line 46 of file universal-function-adapter.hh.

5.164.2 Constructor & Destructor Documentation

5.164.2.1 UniversalFunctionAdapter()

```
UniversalFunctionAdapter (
    UniversalFunction * fn,
    int num_boolean_vars,
    std::vector< representation::DyadicIntegerRepresentation< int >> integer_reps,
    std::vector< representation::DyadicFloatRepresentation< double >> float_reps,
    std::vector< representation::DyadicComplexRepresentation< double >> complex_reps,
    std::vector< representation::LinearCategoricalRepresentation > categorical_reps,
    std::vector< representation::PermutationRepresentation > permutation_reps ) [inline]
```

Constructor.

Parameters

| | |
|-------------------------|-----------------------------|
| <i>fn</i> | Universal function |
| <i>num_boolean_vars</i> | Number of boolean variables |
| <i>integer_reps</i> | Integer representations |
| <i>float_reps</i> | Float representations |
| <i>complex_reps</i> | Complex representations |
| <i>categorical_reps</i> | Categorical representations |
| <i>permutation_reps</i> | Permutation representations |

Replace reps with {} if there is no corresponding variable. For example, if there is no categorical variable,

`UniversalFunctionAdapter(fn, num_boolean_vars, integer_reps, float_reps, complex_reps, {}, permutation_reps)`

Definition at line 133 of file universal-function-adapter.hh.

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

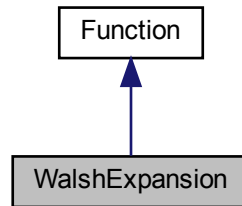
- lib/hnco/multiobjective/functions/universal-function-adapter.hh

5.165 WalshExpansion Class Reference

Walsh expansion.

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

Inheritance diagram for WalshExpansion:



Public Member Functions

- [WalshExpansion](#) ()
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- void [display](#) (std::ostream &stream) const override
Display.
- void [set_terms](#) (const std::vector< [function::WalshTerm](#) > terms)
Set terms.

Instance generators

- template<class Generator >
void [generate](#) (int n, int num_features, Generator generator)
Instance generator.
- void [random](#) (int n, int num_features)
Random instance.

Load and save instance

- void [load](#) (std::string path)
Load instance.
- void [save](#) (std::string path) const
Save instance.

Private Member Functions

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

Private Attributes

- `std::vector< function::WalshTerm > _terms`
Terms.

Friends

- class `boost::serialization::access`

5.165.1 Detailed Description

Walsh expansion.

Its expression is of the form

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

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

Definition at line 52 of file walsh-expansion.hh.

5.165.2 Member Function Documentation

5.165.2.1 `generate()`

```
void generate (
    int n,
    int num_features,
    Generator generator ) [inline]
```

Instance generator.

Parameters

| | |
|---------------------|---------------------------|
| <i>n</i> | Size of bit vectors |
| <i>num_features</i> | Number of feature vectors |
| <i>generator</i> | Coefficient generator |

Definition at line 85 of file walsh-expansion.hh.

5.165.2.2 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to load |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 130 of file walsh-expansion.hh.

5.165.2.3 random()

```
void random (
    int n,
    int num_features ) [inline]
```

Random instance.

The coefficients are sampled from the normal distribution.

Parameters

| | |
|---------------------|---------------------------|
| <i>n</i> | Size of bit vector |
| <i>num_features</i> | Number of feature vectors |

Definition at line 111 of file walsh-expansion.hh.

5.165.2.4 save()

```
void save (
    std::string path ) const [inline]
```

Save instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to save |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 137 of file walsh-expansion.hh.

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

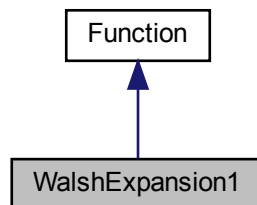
- lib/hnco/functions/collection/walsh/walsh-expansion.hh
- lib/hnco/functions/collection/walsh/walsh-expansion.cc

5.166 WalshExpansion1 Class Reference

Walsh expansion of degree 1.

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

Inheritance diagram for WalshExpansion1:



Public Member Functions

- [WalshExpansion1](#) ()
Constructor.

Instance generators

- `template<class Generator >`
void [generate](#) (int n, Generator generator)
Instance generator.
- void [random](#) (int n)
Random instance.

Load and save instance

- void [load](#) (std::string path)
Load instance.
- void [save](#) (std::string path) const
Save instance.

Evaluation

- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.
- double [evaluate_incrementally](#) (const [bit_vector_t](#) &x, double v, const [hnco::sparse_bit_vector_t](#) &flipped_bits) override
Incrementally evaluate a bit vector.

Information about the function

- int [get_bv_size](#) () const override
Get bit vector size.
- double [get_maximum](#) () const override
Get the global maximum.
- bool [has_known_maximum](#) () const override
Check for a known maximum.
- bool [provides_incremental_evaluation](#) () const override
Check whether the function provides incremental evaluation.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
Serialize.

Private Attributes

- std::vector< double > [_linear](#)
Linear part.

Friends

- class [boost::serialization::access](#)

5.166.1 Detailed Description

Walsh expansion of degree 1.

Its expression is of the form

$$f(x) = \sum_i a_i (1 - 2x_i)$$

or equivalently

$$f(x) = \sum_i a_i (-1)^{x_i}$$

Definition at line 49 of file walsh-expansion-1.hh.

5.166.2 Member Function Documentation

5.166.2.1 generate()

```
void generate (
    int n,
    Generator generator ) [inline]
```

Instance generator.

Parameters

| | |
|------------------|---------------------|
| <i>n</i> | Size of bit vectors |
| <i>generator</i> | Weight generator |

Definition at line 81 of file walsh-expansion-1.hh.

5.166.2.2 has_known_maximum()

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

Check for a known maximum.

Returns

true

Reimplemented from [Function](#).

Definition at line 149 of file walsh-expansion-1.hh.

5.166.2.3 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to load |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 113 of file walsh-expansion-1.hh.

5.166.2.4 provides_incremental_evaluation()

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

Check whether the function provides incremental evaluation.

Returns

true

Reimplemented from [Function](#).

Definition at line 154 of file walsh-expansion-1.hh.

5.166.2.5 random()

```
void random (
    int n ) [inline]
```

Random instance.

The weights are sampled from the normal distribution.

Parameters

| | |
|----------|---------------------|
| <i>n</i> | Size of bit vectors |
|----------|---------------------|

Definition at line 95 of file walsh-expansion-1.hh.

5.166.2.6 save()

```
void save (
    std::string path ) const [inline]
```

Save instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to save |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 120 of file walsh-expansion-1.hh.

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

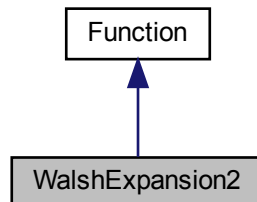
- lib/hnco/functions/collection/walsh/walsh-expansion-1.hh
- lib/hnco/functions/collection/walsh/walsh-expansion-1.cc

5.167 WalshExpansion2 Class Reference

Walsh expansion of degree 2.

```
#include <hnco/functions/collection/walsh/walsh-expansion-2.hh>
```

Inheritance diagram for WalshExpansion2:



Public Member Functions

- [WalshExpansion2](#) ()
Constructor.
- int [get_bv_size](#) () const override
Get bit vector size.
- double [evaluate](#) (const [bit_vector_t](#) &) override
Evaluate a bit vector.

Instance generators

- template<class LinearGen , class QuadraticGen >
void [generate](#) (int n, LinearGen linear_gen, QuadraticGen quadratic_gen)
Instance generators.
- void [random](#) (int n)
Instance generator.
- void [generate_ising1_long_range](#) (int n, double alpha)
Generate one dimensional Ising model with long range interactions.
- void [generate_ising1_long_range_periodic](#) (int n, double alpha)
Generate one dimensional Ising model with long range interactions and periodic boundary conditions.

Load and save instance

- void [load](#) (std::string path)
Load instance.
- void [save](#) (std::string path) const
Save instance.

Private Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
Serialize.
- void [resize](#) (int n)
Resize data structures.

Private Attributes

- std::vector< double > [_linear](#)
Linear part.
- std::vector< std::vector< double > > [_quadratic](#)
Quadratic part.

Friends

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

5.167.1 Detailed Description

Walsh expansion of degree 2.

Its expression is of the form

$$f(x) = \sum_i a_i (1 - 2x_i) + \sum_{i < j} a_{ij} (1 - 2x_i)(1 - 2x_j)$$

or equivalently

$$f(x) = \sum_i a_i (-1)^{x_i} + \sum_{i < j} a_{ij} (-1)^{x_i + x_j}$$

Definition at line 49 of file walsh-expansion-2.hh.

5.167.2 Member Function Documentation

5.167.2.1 generate()

```
void generate (
    int n,
    LinearGen linear_gen,
    QuadraticGen quadratic_gen ) [inline]
```

Instance generators.

Parameters

| | |
|----------------------|----------------------------------|
| <i>n</i> | Size of bit vectors |
| <i>linear_gen</i> | Generator for the linear part |
| <i>quadratic_gen</i> | Generator for the quadratic part |

Definition at line 93 of file walsh-expansion-2.hh.

5.167.2.2 generate_ising1_long_range()

```
void generate_ising1_long_range (
    int n,
    double alpha )
```

Generate one dimensional Ising model with long range interactions.

Similar to a Dyson-Ising model except for the finite, instead of infinite, linear chain of spins.

Its expression is of the form

$$f(x) = \sum_{i,j} J(d_{ij})(1 - 2x_i)(1 - 2x_j)$$

or equivalently

$$f(x) = \sum_{i,j} J(d_{ij})(-1)^{x_i+x_j}$$

where $J(d_{ij})$ is the interaction between sites i and j , $d_{ij} = |i - j|$, and $J(n) = n^{-\alpha}$.

Since we are maximizing f or minimizing $-f$, the expression of f is compatible with what can be found in physics textbooks.

Parameters

| | |
|--------------|-----------------------------|
| <i>n</i> | Size of bit vectors |
| <i>alpha</i> | Exponential decay parameter |

Definition at line 82 of file walsh-expansion-2.cc.

5.167.2.3 generate_ising1_long_range_periodic()

```
void generate_ising1_long_range_periodic (
    int n,
    double alpha )
```

Generate one dimensional Ising model with long range interactions and periodic boundary conditions.

Similar to a Dyson-Ising model except for the finite, instead of infinite, linear chain of spins.

Its expression is of the form

$$f(x) = \sum_{ij} J(d_{ij})(1 - 2x_i)(1 - 2x_j)$$

or equivalently

$$f(x) = \sum_{ij} J(d_{ij})(-1)^{x_i + x_j}$$

where $J(d_{ij})$ is the interaction between sites i and j , $d_{ij} = \min\{|i - j|, n - |i - j|\}$, and $J(n) = n^{-\alpha}$.

Since we are maximizing f or minimizing $-f$, the expression of f is compatible with what can be found in physics textbooks.

Parameters

| | |
|--------------|-----------------------------|
| <i>n</i> | Size of bit vectors |
| <i>alpha</i> | Exponential decay parameter |

Definition at line 103 of file walsh-expansion-2.cc.

5.167.2.4 load()

```
void load (
    std::string path ) [inline]
```

Load instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to load |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------------|--|
| <code>std::runtime_error</code> | |
|---------------------------------|--|

Definition at line 184 of file walsh-expansion-2.hh.

5.167.2.5 random()

```
void random (
    int n ) [inline]
```

Instance generator.

The weights are sampled from the normal distribution.

Parameters

| | |
|----------|--------------------|
| <i>n</i> | Size of bit vector |
|----------|--------------------|

Definition at line 115 of file walsh-expansion-2.hh.

5.167.2.6 save()

```
void save (
    std::string path ) const [inline]
```

Save instance.

Parameters

| | |
|-------------|------------------------------|
| <i>path</i> | Path of the instance to save |
|-------------|------------------------------|

Exceptions

| | |
|---------------------------|--|
| <i>std::runtime_error</i> | |
|---------------------------|--|

Definition at line 191 of file walsh-expansion-2.hh.

5.167.3 Member Data Documentation

5.167.3.1 _quadratic

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

Quadratic part.

Represented as a lower triangular matrix (without its diagonal).

Definition at line 71 of file walsh-expansion-2.hh.

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

- lib/hnco/functions/collection/walsh/walsh-expansion-2.hh
- lib/hnco/functions/collection/walsh/walsh-expansion-2.cc

5.168 WalshTerm Struct Reference

Walsh transform term.

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

Public Member Functions

- template<class Archive >
void [serialize](#) (Archive &ar, const unsigned int version)
Serialize.

Public Attributes

- std::vector< bool > [feature](#)
Feature.
- double [coefficient](#)
Coefficient.

5.168.1 Detailed Description

Walsh transform term.

Definition at line 35 of file walsh-term.hh.

5.168.2 Member Data Documentation

5.168.2.1 feature

```
std::vector<bool> feature
```

Feature.

Implemented with a vector bool instead of a `bit_vector_t` to reduce the memory consumption.

Definition at line 42 of file walsh-term.hh.

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

- lib/hnco/functions/walsh-term.hh

Index

- [_dominated](#)
 - [Nsga2ParetoFrontComputation, 273](#)
 - [_expression](#)
 - [AbstractMaxSat, 66](#)
 - [_functions](#)
 - [Algorithm, 73, 75](#)
 - [_index_of](#)
 - [ParsedMultivariateFunction< Parser >, 289](#)
 - [_lookup_tables](#)
 - [ParsedMultivariateFunction< Parser >, 289](#)
 - [_names](#)
 - [ParsedMultivariateFunction< Parser >, 290](#)
 - [_patience](#)
 - [RandomLocalSearch, 329](#)
 - [_pimpl](#)
 - [Hboa, 153](#)
 - [Ltga, 223](#)
 - [ParameterLessPopulationPyramid, 284](#)
 - [_q](#)
 - [Qubo, 326](#)
 - [_quadratic](#)
 - [WalshExpansion2, 410](#)
 - [_record_evaluation_time](#)
 - [ProgressTracker, 315](#)
 - [_variables](#)
 - [ParsedMultivariateFunction< Parser >, 290](#)
- [~Logger](#)
 - [Logger, 213](#)
- [AbstractMaxSat, 63](#)
 - [_expression, 66](#)
 - [load, 64](#)
 - [load_, 64](#)
 - [save, 65](#)
 - [save_, 65](#)
- [AdditiveGaussianNoise, 66](#)
- [AffineMap, 67](#)
 - [is_surjective, 69](#)
 - [load, 69](#)
 - [random, 69](#)
 - [save, 70](#)
- [Algorithm, 70, 74](#)
 - [_functions, 73, 75](#)
 - [Algorithm, 75](#)
 - [finalize, 72](#)
 - [set_solution, 72](#)
 - [update_solution, 73](#)
- [AlgorithmFactory, 76, 77](#)
 - [make, 76, 77](#)
- [average](#)
 - [SymmetricWalshMoment2, 363](#)
- [bernoulli_trials](#)
 - [MultiBitFlip, 240](#)
- [BiasedCrossover, 78](#)
 - [recombine, 79](#)
- [bm_add_columns](#)
 - [hnco, 20](#)
- [bm_add_rows](#)
 - [hnco, 20](#)
- [bm_identity](#)
 - [hnco, 21](#)
- [bm_invert](#)
 - [hnco, 22](#)
- [bm_multiply](#)
 - [hnco, 22](#)
- [bm_rank](#)
 - [hnco, 23](#)
- [bm_row_echelon_form](#)
 - [hnco, 23](#)
- [bm_set_column](#)
 - [hnco, 23](#)
- [bm_solve](#)
 - [hnco, 24](#)
- [bm_solve_upper_triangular](#)
 - [hnco, 24](#)
- [bm_transpose](#)
 - [hnco, 26](#)
- [BmPbil< GibbsSampler >, 79](#)
 - [RESET_BIT_VECTOR, 83](#)
 - [RESET_ITERATION, 83](#)
 - [RESET_NO_RESET, 83](#)
 - [SAMPLING_ASYNCHRONOUS, 82](#)
 - [SAMPLING_ASYNCHRONOUS_FULL_SCAN, 82](#)
 - [SAMPLING_SYNCHRONOUS, 82](#)
 - [set_selection_size, 83](#)
- [bound](#)
 - [LowerTriangularWalshMoment2, 217](#)
 - [SymmetricWalshMoment2, 363](#)
- [bv_add](#)
 - [hnco, 27](#)
- [bv_from_size_type](#)
 - [hnco, 28](#)
- [bv_from_stream](#)
 - [hnco, 28](#)
- [bv_from_string](#)
 - [hnco, 29](#)
- [bv_from_vector_bool](#)
 - [hnco, 29](#)
- [bv_to_size_type](#)

- hncv, 29, 30
- bv_to_vector_bool
 - hncv, 30
- Cache, 83
 - Cache, 85
 - provides_incremental_evaluation, 85
- CallCounter, 86
- CommandLineAlgorithmFactory, 87, 88
 - make, 88, 89
- CommandLineApplication, 89, 91
 - CommandLineApplication, 91, 92
- CommandLineFunctionFactory, 93, 94
- CommaSelection, 95
 - CommaSelection, 95
- CommutingTransvections
 - TsAffineMap, 383
- CompactGa, 96
- CompleteSearch, 97
- ComplexToDouble< T >, 98
- compute
 - Nsga2ParetoFrontComputation, 272
- compute_fast_walsh_transform
 - hncv::function, 45
- compute_lengths
 - DyadicFloatRepresentation< T >, 110
- compute_walsh_transform
 - hncv::function, 46
- Controller, 99
 - provides_incremental_evaluation, 100
- Crossover, 100
 - recombine, 101
- DeceptiveJump, 101
 - get_maximum, 103
 - has_known_maximum, 103
- DecoratedFunctionFactory, 103
 - make_function_controller, 104
- Decorator, 105
- describe
 - Function, 132, 136
- difference_is_safe
 - hncv::representation, 60
- DisjointTransvections
 - TsAffineMap, 383
- display
 - LowerTriangularWalshMoment2, 217
 - SymmetricWalshMoment2, 364
- dominates
 - hncv::multiobjective::function, 58
- DyadicComplexRepresentation
 - DyadicComplexRepresentation< T >, 107, 108
- DyadicComplexRepresentation< T >, 106
 - DyadicComplexRepresentation, 107, 108
- DyadicFloatRepresentation
 - DyadicFloatRepresentation< T >, 109, 110
- DyadicFloatRepresentation< T >, 108
 - compute_lengths, 110
 - DyadicFloatRepresentation, 109, 110
- DyadicIntegerRepresentation
 - DyadicIntegerRepresentation< T >, 112
- DyadicIntegerRepresentation< T >, 111
 - DyadicIntegerRepresentation, 112
- ensure
 - hncv, 31
- EqualProducts, 114
 - generate, 115
 - load, 115
 - random, 116
 - save, 116
- evaluate
 - Function, 133, 136
 - NearestNeighborIsingModel1, 249
 - NearestNeighborIsingModel2, 253
 - OnBudgetFunction, 275
 - StopOnTarget, 354
- evaluate_incrementally
 - Function, 133
 - OnBudgetFunction, 275
 - StopOnTarget, 355
- evaluate_safely
 - Function, 134
- ExtendedHypercubeIterator, 117
- Factorization, 118
 - Factorization, 120
 - load, 120
- feature
 - WalshTerm, 411
- FgenOptions, 121
- finalize
 - Algorithm, 72
- FirstAscentHillClimbing, 126
- FourPeaks, 128
 - get_maximum, 129
 - has_known_maximum, 129
- FrontDistancePair, 130
- Function, 131, 135
 - describe, 132, 136
 - evaluate, 133, 136
 - evaluate_incrementally, 133
 - evaluate_safely, 134
 - get_maximum, 134
 - provides_incremental_evaluation, 134
 - update, 135
- FunctionFactory, 137, 138
- FunctionMapComposition, 138
 - FunctionMapComposition, 140
 - get_maximum, 140
 - has_known_maximum, 141
- FunctionPlugin, 141
 - FunctionPlugin, 142
- generate
 - EqualProducts, 115
 - LinearFunction, 204
 - NearestNeighborIsingModel1, 249

- NearestNeighborIsingModel2, [254](#)
- NkLandscape, [265](#)
- Partition, [292](#)
- Tsp, [386](#)
- WalshExpansion, [400](#)
- WalshExpansion1, [404](#)
- WalshExpansion2, [408](#)
- generate_ising1_long_range
 - WalshExpansion2, [408](#)
- generate_ising1_long_range_periodic
 - WalshExpansion2, [409](#)
- Generator, [143](#)
 - reset, [143](#)
 - set_seed, [143](#)
- GeneticAlgorithm, [144](#)
 - GeneticAlgorithm, [146](#)
- get_best_bv
 - Population, [305](#)
- get_best_value
 - Population, [306](#)
- get_equivalent_bvs
 - Population, [306](#)
- get_last_improvement
 - ProgressTracker, [315](#)
- get_maximum
 - DeceptiveJump, [103](#)
 - FourPeaks, [129](#)
 - Function, [134](#)
 - FunctionMapComposition, [140](#)
 - Hiff, [158](#)
 - Jump, [197](#)
 - LeadingOnes, [200](#)
 - LongPath, [215](#)
 - Needle, [257](#)
 - OneMax, [277](#)
 - Plateau, [302](#)
 - PriorNoise, [312](#)
 - Ridge, [335](#)
 - SixPeaks, [346](#)
 - Trap, [380](#)
- get_worst_bv
 - Population, [307](#)
- HammingBall, [146](#)
 - HammingBall, [147](#)
- HammingSphere, [148](#)
 - HammingSphere, [149](#)
- HammingSphereIterator, [149](#)
 - HammingSphereIterator, [151](#)
- has_known_maximum
 - DeceptiveJump, [103](#)
 - FourPeaks, [129](#)
 - FunctionMapComposition, [141](#)
 - Hiff, [158](#)
 - Jump, [197](#)
 - LeadingOnes, [200](#)
 - LinearFunction, [204](#)
 - LongPath, [215](#)
 - Needle, [257](#)
 - OneMax, [278](#)
 - Plateau, [302](#)
 - PriorNoise, [312](#)
 - Ridge, [335](#)
 - SixPeaks, [346](#)
 - SummationCancellation, [361](#)
 - Trap, [380](#)
 - WalshExpansion1, [404](#)
- Hboa, [151](#)
 - _pimpl, [153](#)
- Hea
 - Hea< Herding >, [155](#)
- Hea< Herding >, [153](#)
 - Hea, [155](#)
 - set_reset_period, [156](#)
 - set_selection_size, [156](#)
- Hiff, [157](#)
 - get_maximum, [158](#)
 - has_known_maximum, [158](#)
- hnco, [15](#)
 - bm_add_columns, [20](#)
 - bm_add_rows, [20](#)
 - bm_identity, [21](#)
 - bm_invert, [22](#)
 - bm_multiply, [22](#)
 - bm_rank, [23](#)
 - bm_row_echelon_form, [23](#)
 - bm_set_column, [23](#)
 - bm_solve, [24](#)
 - bm_solve_upper_triangular, [24](#)
 - bm_transpose, [26](#)
 - bv_add, [27](#)
 - bv_from_size_type, [28](#)
 - bv_from_stream, [28](#)
 - bv_from_string, [29](#)
 - bv_from_vector_bool, [29](#)
 - bv_to_size_type, [29, 30](#)
 - bv_to_vector_bool, [30](#)
 - ensure, [31](#)
 - is_in_range, [31](#)
 - load_from_archive, [32](#)
 - perm_identity, [32](#)
 - perm_random, [33](#)
 - save_to_archive, [33](#)
 - sbv_flip, [33](#)
 - sbv_is_valid, [34](#)
 - sparse_bit_vector_t, [20](#)
- hnco::algorithm, [35](#)
 - pv_add, [37](#)
 - pv_average, [38](#)
 - pv_bound, [38](#)
 - pv_init, [38](#)
 - pv_sample, [39](#)
 - pv_uniform, [39](#)
 - pv_update, [39, 40](#)
- hnco::algorithm::fast_efficient_p3, [40](#)
- hnco::algorithm::walsh_moment, [41](#)
- hnco::app, [41](#)

- hnco::exception, 42
- hnco::function, 43
 - compute_fast_walsh_transform, 45
 - compute_walsh_transform, 46
- hnco::function::controller, 46
- hnco::function::modifier, 47
- hnco::logging, 48
- hnco::map, 48
 - transvection_sequence_t, 49
 - ts_is_valid, 50
 - ts_multiply, 51
 - ts_random, 52
 - ts_random_commuting, 52
 - ts_random_disjoint, 53
 - ts_random_non_commuting, 53
 - ts_random_unique_destination, 54
 - ts_random_unique_source, 54
- hnco::multiobjective, 55
- hnco::multiobjective::algorithm, 55
 - operator<, 56
- hnco::multiobjective::app, 56
- hnco::multiobjective::function, 57
 - dominates, 58
 - value_t, 58
- hnco::neighborhood, 58
- hnco::random, 59
- hnco::representation, 59
 - difference_is_safe, 60
- HncoEvaluator, 158
- HncoOptions, 159, 177
- Human, 183
- Hypercubeliterator, 184
- Implementation, 185
- init_beta
 - SimulatedAnnealing, 341
- Injection, 185
 - Injection, 186
- IntegerCategoricalRepresentation, 187
 - IntegerCategoricalRepresentation, 188
- is_in_range
 - hnco, 31
- is_non_dominated
 - Nsga2ParetoFrontComputation, 273
- is_surjective
 - AffineMap, 69
 - LinearMap, 207
 - Map, 224
 - MapComposition, 226
 - Permutation, 297
 - Projection, 318
 - Translation, 374
 - TsAffineMap, 383
- is_valid
 - Transvection, 376
- IterativeAlgorithm, 188, 191
 - IterativeAlgorithm, 190, 193
 - loop, 190, 193
 - maximize, 191
 - minimize, 194
 - set_num_iterations, 191, 194
- Iterator, 195
- Jump, 196
 - get_maximum, 197
 - has_known_maximum, 197
- Labs, 197
- LastEvaluation, 199
- LeadingOnes, 199
 - get_maximum, 200
 - has_known_maximum, 200
- LinearCategoricalRepresentation, 201
 - LinearCategoricalRepresentation, 202
- LinearFunction, 202
 - generate, 204
 - has_known_maximum, 204
 - load, 205
 - provides_incremental_evaluation, 205
 - random, 205
 - save, 206
- LinearMap, 206
 - is_surjective, 207
 - load, 208
 - random, 208
 - save, 209
- load
 - AbstractMaxSat, 64
 - AffineMap, 69
 - EqualProducts, 115
 - Factorization, 120
 - LinearFunction, 205
 - LinearMap, 208
 - MaxNae3Sat, 230
 - NearestNeighborIsingModel1, 250
 - NearestNeighborIsingModel2, 254
 - NkLandscape, 266
 - Partition, 292
 - Permutation, 297
 - Qubo, 325
 - Sudoku, 358
 - Translation, 374
 - TsAffineMap, 383
 - Tsp, 387
 - WalshExpansion, 401
 - WalshExpansion1, 404
 - WalshExpansion2, 409
- load_
 - AbstractMaxSat, 64
 - Sudoku, 358
 - Tsp, 387
- load_from_archive
 - hnco, 32
- LocalSearchAlgorithm< Neighborhood >, 209
- LogContext, 211
- Logger, 211
 - ~Logger, 213
 - Logger, 212

- LongPath, 213
 - get_maximum, 215
 - has_known_maximum, 215
- loop
 - IterativeAlgorithm, 190, 193
- LowerTriangularWalshMoment2, 215
 - bound, 217
 - display, 217
 - LowerTriangularWalshMoment2, 216
 - scaled_difference, 217
 - update, 218
- LowerTriangularWalshMoment2GibbsSampler, 219
- LowerTriangularWalshMoment2Herdning, 220
 - LowerTriangularWalshMoment2Herdning, 221
- Ltga, 222
 - _pimpl, 223
- make
 - AlgorithmFactory, 76, 77
 - CommandLineAlgorithmFactory, 88, 89
- make_function_controller
 - DecoratedFunctionFactory, 104
- Map, 223
 - is_surjective, 224
- map
 - Neighborhood, 262
- MapComposition, 225
 - is_surjective, 226
 - MapComposition, 226
- MapgenOptions, 227
- maximize
 - IterativeAlgorithm, 191
- MaxNae3Sat, 229
 - load, 230
- MaxSat, 230
 - random, 231, 232
- Mimic, 232
- minimize
 - IterativeAlgorithm, 194
- Mmas, 235
- Modifier, 236
- MuCommaLambdaEa, 237
 - MuCommaLambdaEa, 238
- MultiBitFlip, 239
 - bernoulli_trials, 240
 - MultiBitFlip, 240
 - rejection_sampling, 240
- multiply
 - Transvection, 376, 377
- MultivariateFunctionAdapter
 - MultivariateFunctionAdapter< Fn, Rep, Conv >, 242, 244
- MultivariateFunctionAdapter< Fn, Rep, Conv >, 241, 243
 - MultivariateFunctionAdapter, 242, 244
- MuPlusLambdaEa, 245
 - MuPlusLambdaEa, 246
- mutate
 - Neighborhood, 262
- NearestNeighborIsingModel1, 247
 - evaluate, 249
 - generate, 249
 - load, 250
 - provides_incremental_evaluation, 250
 - random, 250
 - save, 251
- NearestNeighborIsingModel2, 251
 - evaluate, 253
 - generate, 254
 - load, 254
 - provides_incremental_evaluation, 255
 - random, 255
 - save, 255
- Needle, 256
 - get_maximum, 257
 - has_known_maximum, 257
- Negation, 258
 - provides_incremental_evaluation, 259
- Neighborhood, 260
 - map, 262
 - mutate, 262
 - Neighborhood, 261
- NeighborhoodIterator, 263
 - NeighborhoodIterator, 263
- NkLandscape, 264
 - generate, 265
 - load, 266
 - random, 266
 - random_structure, 266
 - save, 267
- NonCommutingTransvections
 - TsAffineMap, 383
- NpsPbil, 267
- Nsga2, 269
 - Nsga2, 271
- Nsga2ParetoFrontComputation, 272
 - _dominated, 273
 - compute, 272
 - is_non_dominated, 273
- OnBudgetFunction, 274
 - evaluate, 275
 - evaluate_incrementally, 275
 - update, 275
- OneMax, 276
 - get_maximum, 277
 - has_known_maximum, 278
 - provides_incremental_evaluation, 278
- OnePlusLambdaCommaLambdaGa, 279
 - OnePlusLambdaCommaLambdaGa, 280
- OnePlusOneEa, 281
 - OnePlusOneEa, 282
 - set_num_iterations, 283
- operator<
 - hnco::multiobjective::algorithm, 56
- ParameterLessPopulationPyramid, 283
 - _pimpl, 284

- ParsedModifier, [285](#)
 - ParsedModifier, [286](#)
- ParsedMultivariateFunction
 - ParsedMultivariateFunction< Parser >, [287](#), [289](#)
- ParsedMultivariateFunction< Parser >, [286](#), [288](#)
 - _index_of, [289](#)
 - _lookup_tables, [289](#)
 - _names, [290](#)
 - _variables, [290](#)
 - ParsedMultivariateFunction, [287](#), [289](#)
- partial_sort
 - Population, [307](#)
- Partition, [290](#)
 - generate, [292](#)
 - load, [292](#)
 - random, [293](#)
 - save, [293](#)
- Pbil, [293](#)
- perm_identity
 - hnco, [32](#)
- perm_random
 - hnco, [33](#)
- Permutation, [295](#)
 - is_surjective, [297](#)
 - load, [297](#)
 - save, [297](#)
- PermutationFunctionAdapter
 - PermutationFunctionAdapter< Fn >, [299](#)
- PermutationFunctionAdapter< Fn >, [298](#)
 - PermutationFunctionAdapter, [299](#)
- PermutationRepresentation, [300](#)
 - PermutationRepresentation, [300](#)
- PermutationRepresentation::Element, [113](#)
- Plateau, [301](#)
 - get_maximum, [302](#)
 - has_known_maximum, [302](#)
- PlusSelection, [303](#)
 - PlusSelection, [303](#)
- Population, [304](#), [308](#)
 - get_best_bv, [305](#)
 - get_best_value, [306](#)
 - get_equivalent_bvs, [306](#)
 - get_worst_bv, [307](#)
 - partial_sort, [307](#)
 - Population, [305](#), [309](#)
 - resize, [310](#)
 - shrink, [310](#)
 - sort, [308](#)
- PriorNoise, [311](#)
 - get_maximum, [312](#)
 - has_known_maximum, [312](#)
 - provides_incremental_evaluation, [312](#)
- ProgressTracker, [313](#)
 - _record_evaluation_time, [315](#)
 - get_last_improvement, [315](#)
- ProgressTracker::Event, [117](#)
- ProgressTrackerContext, [316](#)
- Projection, [317](#)
 - is_surjective, [318](#)
 - Projection, [318](#)
- provides_incremental_evaluation
 - Cache, [85](#)
 - Controller, [100](#)
 - Function, [134](#)
 - LinearFunction, [205](#)
 - NearestNeighborIsingModel1, [250](#)
 - NearestNeighborIsingModel2, [255](#)
 - Negation, [259](#)
 - OneMax, [278](#)
 - PriorNoise, [312](#)
 - WalshExpansion1, [405](#)
- pv_add
 - hnco::algorithm, [37](#)
- pv_average
 - hnco::algorithm, [38](#)
- pv_bound
 - hnco::algorithm, [38](#)
- pv_init
 - hnco::algorithm, [38](#)
- pv_sample
 - hnco::algorithm, [39](#)
- pv_uniform
 - hnco::algorithm, [39](#)
- pv_update
 - hnco::algorithm, [39](#), [40](#)
- PvAlgorithm, [319](#)
- PythonFunction, [320](#), [322](#)
 - PythonFunction, [321](#), [323](#)
- Qubo, [323](#)
 - _q, [326](#)
 - load, [325](#)
- random
 - AffineMap, [69](#)
 - EqualProducts, [116](#)
 - LinearFunction, [205](#)
 - LinearMap, [208](#)
 - MaxSat, [231](#), [232](#)
 - NearestNeighborIsingModel1, [250](#)
 - NearestNeighborIsingModel2, [255](#)
 - NkLandscape, [266](#)
 - Partition, [293](#)
 - Sudoku, [358](#)
 - Transvection, [377](#)
 - TsAffineMap, [384](#)
 - Tsp, [387](#)
 - WalshExpansion, [401](#)
 - WalshExpansion1, [405](#)
 - WalshExpansion2, [410](#)
- random_non_commuting
 - Transvection, [378](#)
- random_structure
 - NkLandscape, [266](#)
- RandomLocalSearch, [326](#)
 - _patience, [329](#)
 - set_patience, [328](#)

- RandomSearch, [329](#)
- RandomSelection, [330](#)
 - RandomSelection, [331](#)
- RandomWalk, [331](#)
- recombine
 - BiasedCrossover, [79](#)
 - Crossover, [101](#)
 - UniformCrossover, [391](#)
- rejection_sampling
 - MultiBitFlip, [240](#)
- reset
 - Generator, [143](#)
- RESET_BIT_VECTOR
 - BmPbil< GibbsSampler >, [83](#)
- RESET_ITERATION
 - BmPbil< GibbsSampler >, [83](#)
- RESET_NO_RESET
 - BmPbil< GibbsSampler >, [83](#)
- resize
 - Population, [310](#)
- Restart, [333](#)
- Ridge, [334](#)
 - get_maximum, [335](#)
 - has_known_maximum, [335](#)
- SAMPLING_ASYNCHRONOUS
 - BmPbil< GibbsSampler >, [82](#)
- SAMPLING_ASYNCHRONOUS_FULL_SCAN
 - BmPbil< GibbsSampler >, [82](#)
- SAMPLING_SYNCHRONOUS
 - BmPbil< GibbsSampler >, [82](#)
- SamplingMode
 - TsAffineMap, [383](#)
- save
 - AbstractMaxSat, [65](#)
 - AffineMap, [70](#)
 - EqualProducts, [116](#)
 - LinearFunction, [206](#)
 - LinearMap, [209](#)
 - NearestNeighborIsingModel1, [251](#)
 - NearestNeighborIsingModel2, [255](#)
 - NkLandscape, [267](#)
 - Partition, [293](#)
 - Permutation, [297](#)
 - Sudoku, [359](#)
 - Translation, [375](#)
 - TsAffineMap, [384](#)
 - Tsp, [388](#)
 - WalshExpansion, [401](#)
 - WalshExpansion1, [405](#)
 - WalshExpansion2, [410](#)
- save_
 - AbstractMaxSat, [65](#)
- save_to_archive
 - hnco, [33](#)
- sbv_flip
 - hnco, [33](#)
- sbv_is_valid
 - hnco, [34](#)
- ScalarToDouble< T >, [336](#)
- scaled_difference
 - LowerTriangularWalshMoment2, [217](#)
 - SymmetricWalshMoment2, [364](#)
- select
 - TournamentSelection, [371](#)
- SelfAdjustingOnePlusOneEa, [337](#)
- set_mutation_rate
 - StandardBitMutation, [349](#)
- set_num_iterations
 - IterativeAlgorithm, [191](#), [194](#)
 - OnePlusOneEa, [283](#)
- set_patience
 - RandomLocalSearch, [328](#)
- set_reset_period
 - Hea< Herding >, [156](#)
- set_seed
 - Generator, [143](#)
- set_selection_size
 - BmPbil< GibbsSampler >, [83](#)
 - Hea< Herding >, [156](#)
- set_solution
 - Algorithm, [72](#)
- shrink
 - Population, [310](#)
- SimulatedAnnealing, [339](#)
 - init_beta, [341](#)
- SingleBitFlip, [341](#)
- SingleBitFlipIterator, [342](#)
 - SingleBitFlipIterator, [343](#)
- SinusSummationCancellation, [344](#)
- SixPeaks, [345](#)
 - get_maximum, [346](#)
 - has_known_maximum, [346](#)
- sort
 - Population, [308](#)
- sparse_bit_vector_t
 - hnco, [20](#)
- StandardBitMutation, [347](#)
 - set_mutation_rate, [349](#)
 - StandardBitMutation, [348](#), [349](#)
- SteepestAscentHillClimbing, [350](#)
- StopOnMaximum, [351](#)
 - StopOnMaximum, [352](#)
- StopOnTarget, [353](#)
 - evaluate, [354](#)
 - evaluate_incrementally, [355](#)
 - StopOnTarget, [354](#)
 - update, [355](#)
- StopWatch, [356](#)
- Sudoku, [356](#)
 - load, [358](#)
 - load_, [358](#)
 - random, [358](#)
 - save, [359](#)
- SummationCancellation, [359](#)
 - has_known_maximum, [361](#)
 - SummationCancellation, [361](#)

- SymmetricWalshMoment2, 362
 - average, 363
 - bound, 363
 - display, 364
 - scaled_difference, 364
 - SymmetricWalshMoment2, 363
 - update, 365
- SymmetricWalshMoment2GibbsSampler, 366
- SymmetricWalshMoment2Herding, 367
 - SymmetricWalshMoment2Herding, 368
- TargetReached, 369
- TournamentSelection, 369
 - select, 371
 - TournamentSelection, 370
- TournamentSelection < T, Compare >, 371
- Translation, 373
 - is_surjective, 374
 - load, 374
 - save, 375
- Transvection, 375
 - is_valid, 376
 - multiply, 376, 377
 - random, 377
 - random_non_commuting, 378
- transvection_sequence_t
 - hnco::map, 49
- Trap, 378
 - get_maximum, 380
 - has_known_maximum, 380
 - Trap, 380
- ts_is_valid
 - hnco::map, 50
- ts_multiply
 - hnco::map, 51
- ts_random
 - hnco::map, 52
- ts_random_commuting
 - hnco::map, 52
- ts_random_disjoint
 - hnco::map, 53
- ts_random_non_commuting
 - hnco::map, 53
- ts_random_unique_destination
 - hnco::map, 54
- ts_random_unique_source
 - hnco::map, 54
- TsAffineMap, 381
 - CommutingTransvections, 383
 - DisjointTransvections, 383
 - is_surjective, 383
 - load, 383
 - NonCommutingTransvections, 383
 - random, 384
 - SamplingMode, 383
 - save, 384
 - Unconstrained, 383
 - UniqueDestination, 383
 - UniqueSource, 383
- Tsp, 385
 - generate, 386
 - load, 387
 - load_, 387
 - random, 387
 - save, 388
- Umda, 388
- Unconstrained
 - TsAffineMap, 383
- UniformCrossover, 390
 - recombine, 391
- UniformSelection, 391
 - UniformSelection, 392
- UniqueDestination
 - TsAffineMap, 383
- UniqueSource
 - TsAffineMap, 383
- UniversalFunction, 392, 393
- UniversalFunctionAdapter, 394, 396
 - UniversalFunctionAdapter, 395, 398
- update
 - Function, 135
 - LowerTriangularWalshMoment2, 218
 - OnBudgetFunction, 275
 - StopOnTarget, 355
 - SymmetricWalshMoment2, 365
- update_solution
 - Algorithm, 73
- value_t
 - hnco::multiobjective::function, 58
- WalshExpansion, 399
 - generate, 400
 - load, 401
 - random, 401
 - save, 401
- WalshExpansion1, 402
 - generate, 404
 - has_known_maximum, 404
 - load, 404
 - provides_incremental_evaluation, 405
 - random, 405
 - save, 405
- WalshExpansion2, 406
 - _quadratic, 410
 - generate, 408
 - generate_ising1_long_range, 408
 - generate_ising1_long_range_periodic, 409
 - load, 409
 - random, 410
 - save, 410
- WalshTerm, 411
 - feature, 411