MPCOTool

Generated by Doxygen 1.9.4

1 MPCOTool	1
2 Data Structure Index	7
2.1 Data Structures	7
3 File Index	9
3.1 File List	9
4 Data Structure Documentation	11
4.1 Experiment Struct Reference	11
4.1.1 Detailed Description	11
4.1.2 Field Documentation	11
4.1.2.1 name	11
4.1.2.2 ninputs	12
4.1.2.3 stencil	12
4.1.2.4 weight	12
4.2 Input Struct Reference	12
4.2.1 Detailed Description	14
4.2.2 Field Documentation	14
4.2.2.1 adaptation_ratio	14
4.2.2.2 algorithm	14
4.2.2.3 climbing	14
4.2.2.4 directory	15
4.2.2.5 evaluator	15
4.2.2.6 experiment	15
4.2.2.7 mutation_ratio	15
4.2.2.8 name	15
4.2.2.9 nbest	16
4.2.2.10 nestimates	16
4.2.2.11 nexperiments	16
4.2.2.12 niterations	16
4.2.2.13 norm	16
4.2.2.14 nsimulations	17
4.2.2.15 nsteps	17
4.2.2.16 nvariables	17
4.2.2.17 p	17
4.2.2.18 relaxation	17
4.2.2.19 reproduction_ratio	18
4.2.2.20 result	18
4.2.2.21 seed	18
4.2.2.22 simulator	18
4.2.2.23 threshold	18
4.2.2.24 tolerance	19

4.2.2.25 type	. 19
4.2.2.26 variable	. 19
4.2.2.27 variables	. 19
4.3 Optimize Struct Reference	. 20
4.3.1 Detailed Description	. 22
4.3.2 Field Documentation	. 22
4.3.2.1 adaptation_ratio	. 22
4.3.2.2 algorithm	. 23
4.3.2.3 calculation_time	. 23
4.3.2.4 climbing	. 23
4.3.2.5 error_best	. 23
4.3.2.6 error_old	. 23
4.3.2.7 evaluator	. 24
4.3.2.8 experiment	. 24
4.3.2.9 file	. 24
4.3.2.10 file_result	. 24
4.3.2.11 file_variables	. 24
4.3.2.12 genetic_variable	. 25
4.3.2.13 label	. 25
4.3.2.14 mpi_rank	. 25
4.3.2.15 mutation_ratio	. 25
4.3.2.16 nbest	. 25
4.3.2.17 nbits	. 26
4.3.2.18 nend	. 26
4.3.2.19 nend_climbing	. 26
4.3.2.20 nestimates	. 26
4.3.2.21 nexperiments	. 26
4.3.2.22 ninputs	. 27
4.3.2.23 niterations	. 27
4.3.2.24 nsaveds	. 27
4.3.2.25 nsimulations	. 27
4.3.2.26 nstart	. 27
4.3.2.27 nstart_climbing	. 28
4.3.2.28 nsteps	. 28
4.3.2.29 nsweeps	. 28
4.3.2.30 nvariables	. 28
4.3.2.31 p	. 28
4.3.2.32 precision	. 29
4.3.2.33 rangemax	. 29
4.3.2.34 rangemaxabs	. 29
4.3.2.35 rangemin	. 29
4.3.2.36 rangeminabs	. 29

4.3.2.37 relaxation	 30
4.3.2.38 reproduction_ratio	 30
4.3.2.39 result	 30
4.3.2.40 rng	 30
4.3.2.41 seed	 30
4.3.2.42 simulation_best	 31
4.3.2.43 simulator	 31
4.3.2.44 step	 31
4.3.2.45 stop	 31
4.3.2.46 thread	 31
4.3.2.47 thread_climbing	 32
4.3.2.48 threshold	 32
4.3.2.49 tolerance	 32
4.3.2.50 value	 32
4.3.2.51 value_old	 32
4.3.2.52 variables	 33
4.3.2.53 weight	 33
4.4 Options Struct Reference	 33
4.4.1 Detailed Description	 34
4.4.2 Field Documentation	 34
4.4.2.1 dialog	 34
4.4.2.2 grid	 34
4.4.2.3 label_climbing	 34
4.4.2.4 label_seed	 34
4.4.2.5 label_threads	 35
4.4.2.6 spin_climbing	 35
4.4.2.7 spin_seed	 35
4.4.2.8 spin_threads	 35
4.5 ParallelData Struct Reference	 35
4.5.1 Detailed Description	 36
4.5.2 Field Documentation	 36
4.5.2.1 thread	 36
4.6 Running Struct Reference	 36
4.6.1 Detailed Description	 37
4.6.2 Field Documentation	 37
4.6.2.1 dialog	 37
4.6.2.2 grid	 37
4.6.2.3 label	 37
4.6.2.4 spinner	 37
4.7 Variable Struct Reference	 38
4.7.1 Detailed Description	 38
4.7.2 Field Documentation	 38

4.7.2.1 name	. 38
4.7.2.2 nbits	. 39
4.7.2.3 nsweeps	. 39
4.7.2.4 precision	. 39
4.7.2.5 rangemax	. 39
4.7.2.6 rangemaxabs	. 39
4.7.2.7 rangemin	. 40
4.7.2.8 rangeminabs	. 40
4.7.2.9 step	. 40
4.8 Window Struct Reference	. 40
4.8.1 Detailed Description	. 45
4.8.2 Field Documentation	. 45
4.8.2.1 application_directory	. 45
4.8.2.2 box_buttons	. 45
4.8.2.3 button_about	. 46
4.8.2.4 button_add_experiment	. 46
4.8.2.5 button_add_variable	. 46
4.8.2.6 button_algorithm	. 46
4.8.2.7 button_climbing	. 46
4.8.2.8 button_evaluator	. 47
4.8.2.9 button_exit	. 47
4.8.2.10 button_experiment	. 47
4.8.2.11 button_help	. 47
4.8.2.12 button_norm	. 47
4.8.2.13 button_open	. 48
4.8.2.14 button_options	. 48
4.8.2.15 button_remove_experiment	. 48
4.8.2.16 button_remove_variable	. 48
4.8.2.17 button_run	. 48
4.8.2.18 button_save	. 49
4.8.2.19 button_simulator	. 49
4.8.2.20 button_template	. 49
4.8.2.21 check_climbing	. 49
4.8.2.22 check_evaluator	. 49
4.8.2.23 check_maxabs	. 50
4.8.2.24 check_minabs	. 50
4.8.2.25 check_template	. 50
4.8.2.26 combo_experiment	. 50
4.8.2.27 combo_variable	. 50
4.8.2.28 entry_result	. 51
4.8.2.29 entry_variable	. 51
4.8.2.30 entry_variables	. 51

4.8.2.31 experiment
4.8.2.32 frame_algorithm
4.8.2.33 frame_experiment
4.8.2.34 frame_norm
4.8.2.35 frame_variable
4.8.2.36 grid
4.8.2.37 grid_algorithm
4.8.2.38 grid_climbing
4.8.2.39 grid_experiment
4.8.2.40 grid_files
4.8.2.41 grid_norm
4.8.2.42 grid_variable
4.8.2.43 id_experiment
4.8.2.44 id_experiment_name
4.8.2.45 id_input
4.8.2.46 id_template
4.8.2.47 id_variable
4.8.2.48 id_variable_label
4.8.2.49 label_adaptation
4.8.2.50 label_bests
4.8.2.51 label_bits
4.8.2.52 label_estimates
4.8.2.53 label_experiment
4.8.2.54 label_generations
4.8.2.55 label_iterations
4.8.2.56 label_max
4.8.2.57 label_min
4.8.2.58 label_mutation
4.8.2.59 label_p
4.8.2.60 label_population
4.8.2.61 label_precision
4.8.2.62 label_relaxation
4.8.2.63 label_reproduction
4.8.2.64 label_result
4.8.2.65 label_simulations
4.8.2.66 label_simulator
4.8.2.67 label_step
4.8.2.68 label_steps
4.8.2.69 label_sweeps
4.8.2.70 label_threshold
4.8.2.71 label_tolerance
4.8.2.72 label_variable

4.8.2.73 label_variables	60
4.8.2.74 label_weight	60
4.8.2.75 logo	60
4.8.2.76 nexperiments	60
4.8.2.77 nvariables	60
4.8.2.78 scrolled_max	61
4.8.2.79 scrolled_maxabs	61
4.8.2.80 scrolled_min	61
4.8.2.81 scrolled_minabs	61
4.8.2.82 scrolled_p	61
4.8.2.83 scrolled_step	62
4.8.2.84 scrolled_threshold	62
4.8.2.85 spin_adaptation	62
4.8.2.86 spin_bests	62
4.8.2.87 spin_bits	62
4.8.2.88 spin_estimates	63
4.8.2.89 spin_generations	63
4.8.2.90 spin_iterations	63
4.8.2.91 spin_max	63
4.8.2.92 spin_maxabs	63
4.8.2.93 spin_min	64
4.8.2.94 spin_minabs	64
4.8.2.95 spin_mutation	64
4.8.2.96 spin_p	64
4.8.2.97 spin_population	64
4.8.2.98 spin_precision	65
4.8.2.99 spin_relaxation	65
4.8.2.100 spin_reproduction	65
4.8.2.101 spin_simulations	65
4.8.2.102 spin_step	65
4.8.2.103 spin_steps	66
4.8.2.104 spin_sweeps	66
4.8.2.105 spin_threshold	66
4.8.2.106 spin_tolerance	66
4.8.2.107 spin_weight	66
4.8.2.108 variable	67
4.8.2.109 window	67
5 File Documentation	69
5.1 config.h File Reference	69
5.1.1 Detailed Description	72
5.1.2 Macro Definition Documentation	72

5.1.2.1 DEFAULT_PRECISION
5.1.2.2 DEFAULT_RANDOM_SEED
5.1.2.3 DEFAULT_RELAXATION
5.1.2.4 LABEL_ABSOLUTE_MAXIMUM
5.1.2.5 LABEL_ABSOLUTE_MINIMUM
5.1.2.6 LABEL_ADAPTATION
5.1.2.7 LABEL_ALGORITHM
5.1.2.8 LABEL_CLIMBING
5.1.2.9 LABEL_COORDINATES
5.1.2.10 LABEL_EUCLIDIAN
5.1.2.11 LABEL_EVALUATOR
5.1.2.12 LABEL_EXPERIMENT
5.1.2.13 LABEL_EXPERIMENTS
5.1.2.14 LABEL_GENETIC
5.1.2.15 LABEL_MAXIMUM
5.1.2.16 LABEL_MINIMUM
5.1.2.17 LABEL_MONTE_CARLO
5.1.2.18 LABEL_MUTATION
5.1.2.19 LABEL_NAME
5.1.2.20 LABEL_NBEST
5.1.2.21 LABEL_NBITS
5.1.2.22 LABEL_NESTIMATES
5.1.2.23 LABEL_NGENERATIONS
5.1.2.24 LABEL_NITERATIONS
5.1.2.25 LABEL_NORM
5.1.2.26 LABEL_NPOPULATION
5.1.2.27 LABEL_NSIMULATIONS
5.1.2.28 LABEL_NSTEPS
5.1.2.29 LABEL_NSWEEPS
5.1.2.30 LABEL_OPTIMIZE
5.1.2.31 LABEL_ORTHOGONAL
5.1.2.32 LABEL_P
5.1.2.33 LABEL_PRECISION
5.1.2.34 LABEL_RANDOM
5.1.2.35 LABEL_RELAXATION
5.1.2.36 LABEL_REPRODUCTION
5.1.2.37 LABEL_RESULT_FILE
5.1.2.38 LABEL_SEED
5.1.2.39 LABEL_SIMULATOR
5.1.2.40 LABEL_STEP
5.1.2.41 LABEL_SWEEP
5.1.2.42 LABEL TAXICAB

5.1.2.43 LABEL_TEMPLATE1		81
5.1.2.44 LABEL_TEMPLATE2		81
5.1.2.45 LABEL_TEMPLATE3		81
5.1.2.46 LABEL_TEMPLATE4		81
5.1.2.47 LABEL_TEMPLATE5		81
5.1.2.48 LABEL_TEMPLATE6		82
5.1.2.49 LABEL_TEMPLATE7		82
5.1.2.50 LABEL_TEMPLATE8		82
5.1.2.51 LABEL_THRESHOLD		82
5.1.2.52 LABEL_TOLERANCE		82
5.1.2.53 LABEL_VARIABLE		83
5.1.2.54 LABEL_VARIABLES		83
5.1.2.55 LABEL_VARIABLES_FILE		83
5.1.2.56 LABEL_WEIGHT		83
5.1.2.57 LOCALE_DIR		83
5.1.2.58 MAX_NINPUTS		84
5.1.2.59 NALGORITHMS		84
5.1.2.60 NCLIMBINGS		84
5.1.2.61 NNORMS		84
5.1.2.62 NPRECISIONS		84
5.1.2.63 PROGRAM_INTERFACE		85
5.1.3 Enumeration Type Documentation		85
5.1.3.1 INPUT_TYPE		85
5.2 config.h		85
5.3 experiment.c File Reference		87
5.3.1 Detailed Description		88
5.3.2 Macro Definition Documentation		88
5.3.2.1 DEBUG_EXPERIMENT		88
5.3.3 Function Documentation		88
5.3.3.1 experiment_error()		88
5.3.3.2 experiment_free()		89
5.3.3.3 experiment_new()		89
5.3.3.4 experiment_open_json()		90
5.3.3.5 experiment_open_xml()		92
5.3.4 Variable Documentation		94
5.3.4.1 stencil		94
5.4 experiment.c		94
5.5 experiment.h File Reference		98
5.5.1 Detailed Description		99
5.5.2 Function Documentation		99
5.5.2.1 experiment_error()		99
5.5.2.2 experiment_free()		99

5.5.2.3 experiment_open_json()	. 100
5.5.2.4 experiment_open_xml()	. 102
5.5.3 Variable Documentation	. 104
5.5.3.1 stencil	. 104
5.6 experiment.h	. 104
5.7 input.c File Reference	. 105
5.7.1 Detailed Description	. 106
5.7.2 Macro Definition Documentation	. 106
5.7.2.1 DEBUG_INPUT	. 106
5.7.3 Function Documentation	. 106
5.7.3.1 input_error()	. 106
5.7.3.2 input_free()	. 107
5.7.3.3 input_new()	. 108
5.7.3.4 input_open()	. 108
5.7.3.5 input_open_json()	. 109
5.7.3.6 input_open_xml()	. 115
5.7.4 Variable Documentation	. 121
5.7.4.1 input	. 121
5.7.4.2 result_name	. 121
5.7.4.3 variables_name	. 122
5.8 input.c	. 122
5.9 input.h File Reference	. 133
5.9.1 Detailed Description	. 134
5.9.2 Enumeration Type Documentation	. 134
5.9.2.1 ClimbingMethod	. 134
5.9.2.2 ErrorNorm	. 135
5.9.3 Function Documentation	. 135
5.9.3.1 input_free()	. 135
5.9.3.2 input_new()	. 136
5.9.3.3 input_open()	. 137
5.9.4 Variable Documentation	. 138
5.9.4.1 input	. 138
5.9.4.2 result_name	. 138
5.9.4.3 variables_name	. 139
5.10 input.h	. 139
5.11 interface.c File Reference	. 140
5.11.1 Detailed Description	. 142
5.11.2 Macro Definition Documentation	. 142
5.11.2.1 DEBUG_INTERFACE	. 142
5.11.2.2 INPUT_FILE	. 142
5.11.3 Function Documentation	. 143
5.11.3.1 dialog_evaluator()	. 143

5.11.3.2 dialog_evaluator_close()
5.11.3.3 dialog_name_experiment_close()
5.11.3.4 dialog_open_close()
5.11.3.5 dialog_options_close()
5.11.3.6 dialog_save_close()
5.11.3.7 dialog_simulator()
5.11.3.8 dialog_simulator_close()
5.11.3.9 input_save()
5.11.3.10 input_save_climbing_json()
5.11.3.11 input_save_climbing_xml()
5.11.3.12 input_save_json()
5.11.3.13 input_save_xml()
5.11.3.14 options_new()
5.11.3.15 running_new()
5.11.3.16 window_about()
5.11.3.17 window_add_experiment()
5.11.3.18 window_add_variable()
5.11.3.19 window_get_algorithm()
5.11.3.20 window_get_climbing()
5.11.3.21 window_get_norm()
5.11.3.22 window_help()
5.11.3.23 window_inputs_experiment()
5.11.3.24 window_label_variable()
5.11.3.25 window_name_experiment()
5.11.3.26 window_new()
5.11.3.27 window_open()
5.11.3.28 window_precision_variable()
5.11.3.29 window_rangemax_variable()
5.11.3.30 window_rangemaxabs_variable()
5.11.3.31 window_rangemin_variable()
5.11.3.32 window_rangeminabs_variable()
5.11.3.33 window_read()
5.11.3.34 window_remove_experiment()
5.11.3.35 window_remove_variable()
5.11.3.36 window_run()
5.11.3.37 window_save()
5.11.3.38 window_save_climbing()
5.11.3.39 window_set_algorithm()
5.11.3.40 window_set_experiment()
5.11.3.41 window_set_variable()
5.11.3.42 window_step_variable()
5.11.3.43 window_template_experiment()

5.11.3.44 window_template_experiment_close()
5.11.3.45 window_update()
5.11.3.46 window_update_climbing()
5.11.3.47 window_update_variable()
5.11.3.48 window_weight_experiment()
5.11.4 Variable Documentation
5.11.4.1 logo
5.11.4.2 options
5.11.4.3 running
5.11.4.4 window
5.12 interface.c
5.13 interface.h File Reference
5.13.1 Detailed Description
5.13.2 Macro Definition Documentation
5.13.2.1 MAX_LENGTH
5.13.3 Function Documentation
5.13.3.1 window_new()
5.13.4 Variable Documentation
5.13.4.1 window
5.14 interface.h
5.15 main.c File Reference
5.15.1 Detailed Description
5.15.2 Macro Definition Documentation
5.15.2.1 JBW
5.15.3 Function Documentation
5.15.3.1 main()
5.16 main.c
5.17 mpcotool.c File Reference
5.17.1 Detailed Description
5.17.2 Macro Definition Documentation
5.17.2.1 DEBUG_MPCOTOOL
5.17.3 Function Documentation
5.17.3.1 mpcotool()
5.18 mpcotool.c
5.19 mpcotool.h File Reference
5.19.1 Detailed Description
5.19.2 Function Documentation
5.19.2.1 mpcotool()
5.20 mpcotool.h
5.21 optimize.c File Reference
5.21.1 Detailed Description
5.21.2 Macro Definition Documentation

	5.21.2.1 DEBUG_OPTIMIZE	254
	5.21.2.2 RM	254
5.21.3	Function Documentation	254
	5.21.3.1 optimize_best()	254
	5.21.3.2 optimize_best_climbing()	255
	5.21.3.3 optimize_climbing()	255
	5.21.3.4 optimize_climbing_sequential()	256
	5.21.3.5 optimize_climbing_thread()	257
	5.21.3.6 optimize_estimate_climbing_coordinates()	258
	5.21.3.7 optimize_estimate_climbing_random()	259
	5.21.3.8 optimize_free()	259
	5.21.3.9 optimize_genetic()	260
	5.21.3.10 optimize_genetic_objective()	260
	5.21.3.11 optimize_input()	261
	5.21.3.12 optimize_iterate()	262
	5.21.3.13 optimize_merge()	263
	5.21.3.14 optimize_merge_old()	264
	5.21.3.15 optimize_MonteCarlo()	265
	5.21.3.16 optimize_norm_euclidian()	265
	5.21.3.17 optimize_norm_maximum()	266
	5.21.3.18 optimize_norm_p()	267
	5.21.3.19 optimize_norm_taxicab()	268
	5.21.3.20 optimize_open()	269
	5.21.3.21 optimize_orthogonal()	273
	5.21.3.22 optimize_parse()	273
	5.21.3.23 optimize_print()	275
	5.21.3.24 optimize_refine()	276
	5.21.3.25 optimize_save_old()	277
	5.21.3.26 optimize_save_variables()	277
	5.21.3.27 optimize_sequential()	278
	5.21.3.28 optimize_step()	279
	5.21.3.29 optimize_step_climbing()	279
	5.21.3.30 optimize_sweep()	280
	5.21.3.31 optimize_synchronise()	281
	5.21.3.32 optimize_thread()	282
5.21.4	Variable Documentation	282
	5.21.4.1 nthreads_climbing	283
	5.21.4.2 optimize	283
	5.21.4.3 optimize_algorithm	283
	5.21.4.4 optimize_estimate_climbing	283
	5.21.4.5 optimize_norm	283
5.22 optimize	9.C	284

5.23 optimize.h File Reference
5.23.1 Detailed Description
5.23.2 Function Documentation
5.23.2.1 optimize_free()
5.23.2.2 optimize_open()
5.23.3 Variable Documentation
5.23.3.1 nthreads_climbing
5.23.3.2 optimize
5.24 optimize.h
5.25 tools.c File Reference
5.25.1 Detailed Description
5.25.2 Variable Documentation
5.25.2.1 error_message
5.25.2.2 main_window
5.25.2.3 show_pending
5.26 tools.c
5.27 tools.h File Reference
5.27.1 Detailed Description
5.27.2 Macro Definition Documentation
5.27.2.1 ERROR_TYPE
5.27.2.2 INFO_TYPE
5.27.3 Variable Documentation
5.27.3.1 error_message
5.27.3.2 main_window
5.27.3.3 show_pending
5.28 tools.h
5.29 variable.c File Reference
5.29.1 Detailed Description
5.29.2 Macro Definition Documentation
5.29.2.1 DEBUG_VARIABLE
5.29.3 Function Documentation
5.29.3.1 variable_error()
5.29.3.2 variable_free()
5.29.3.3 variable_open_json()
5.29.3.4 variable_open_xml()
5.29.4 Variable Documentation
5.29.4.1 format
5.29.4.2 precision
5.30 variable.c
5.31 variable.h File Reference
5.31.1 Detailed Description
5.31.2 Enumeration Type Documentation

5.31.2.1 Algorithm	329
5.31.3 Function Documentation	329
5.31.3.1 variable_error()	329
5.31.3.2 variable_free()	330
5.31.3.3 variable_open_json()	330
5.31.3.4 variable_open_xml()	333
5.31.4 Variable Documentation	336
5.31.4.1 format	336
5.31.4.2 precision	336
5.32 variable.h	
ndex	339

Chapter 1

MPCOTool

:gb:english :es:español

The Multi-Purposes Calibration and Optimization Tool. A software to perform calibrations or optimizations of empirical parameters.

VERSIONS

• 4.4.6: Stable and recommended version.

AUTHORS

- Javier Burguete Tolosa (jburguete@eead.csic.es)
- Borja Latorre Garcés (borja.latorre@csic.es)

WINDOWS EXECUTABLE FILES

This repository contains source and example files with the latest version of MPCOTool. Stable versions with executable files and manuals for Microsoft Windows systems can be downloaded in digital.csic

TOOLS AND LIBRARIES REQUIRED TO BUILD THE EXECUTABLE

- gcc or clang (to compile the source code)
- make (to build the executable file)
- autoconf (to generate the Makefile in different operative systems)
- automake (to check the operative system)
- pkg-config (to find the libraries to compile)
- gsl (to generate random numbers)
- libxml (to deal with XML files)
- glib (extended utilities of C to work with data, lists, mapped files, regular expressions, using multicores in shared memory machines, ...)
- json-glib (to deal with JSON files)
- genetic (genetic algorithm)

2 MPCOTool

OPTIONAL TOOLS AND LIBRARIES

- gettext (to work with different locales)
- gtk+3 (to create the interactive GUI tool)
- openmpi or mpich (to run in parallelized tasks on multiple computers)
- doxygen (standard comments format to generate documentation)
- latex (to build the PDF manuals)

FILES

The source code has to have the following files:

- 4.4.6/configure.ac: configure generator.
- 4.4.6/Makefile.in: Makefile generator.
- 4.4.6/config.h.in: config header generator.
- 4.4.6/*.c: source code files.
- 4.4.6/*.h: header code files.
- 4.4.6/mpcotool.ico: icon file.
- 4.4.6/build.sh: shell script to build all.
- 4.4.6/logo.png: logo figure.
- 4.4.6/Doxyfile: configuration file to generate doxygen documentation.
- · TODO: tasks to do.
- · README.md: this file.
- · license.md: license file.
- tests/testX/*: several tests to check the program working.
- locales/*/LC_MESSAGES/mpcotool.po: translation files.
- manuals/*.eps: manual figures in EPS format.
- manuals/*.png: manual figures in PNG format.
- manuals/*.tex: documentation source files.
- applications/*/*: several practical application cases.
- check_errors/*.xml: several mistaken files to check error handling.

BUILDING INSTRUCTIONS

On Microsoft Windows systems you have to install MSYS2 and the required libraries and utilities. You can follow detailed instructions in install-unix. Optional Windows binary package can be built doing in the terminal:

\$ make windist

On NetBSD 9.3, to compile with last GCC version you have to do first on the building terminal:

\$ export PATH=/usr/pkg/gcc8/bin:\$PATH" </blockquote> On OpenBSD 7.2 you have to do first on the building terminal to select adequate versions and deactivate OpenMPI (does not link) building with CLang: <blockquote>\$ export AUTOCONF_VERSION=2.69 AUTOMAKE_VERSION=1.16 CC=clang </bl>
 On OpenIndiana Hipster, in order to enable OpenMPI compilation, do in a terminal: <blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge="line"><blookledge=" in order to enable OpenMPI compilation, in 64 bits version do in a terminal (OpenMPI configure export PATH=\$PATH:/usr/lib64/mpi/gcc/openmpi/bin </blockquote> This software has been built and tested in the following operative systems: * Arch Linux * Debian Linux 11 * Devuan Linux 4 * DragonFly BSD 6.4 * Fedora Linux 37 * FreeBSD 13.1 * Gentoo Linux * Linux Mint DE 5 * MacOS Catalina + Homebrew * Manjaro Linux * Microsoft Windows 10 * NetBSD 9.3 * OpenBSD 7.2 * OpenInidiana Hipster * OpenSUSE Linux Leap 15.4 * Ubuntu Linux 22.10 Probably, it can be built in other systems, distributions, or versions but it has not been tested. 1. Download the latest genetic doing on a terminal: <blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1"><blookledge="1 library: <blockquote>\$ cd genetic/3.0.1 \$ sh build.sh </blockquote> 3. Download this repository: <bloom> \$ cd ../.. \$ git clone https://github.com/jburguete/mpcotool.git </blockquote> 4. Link the latest genetic version to genetic: <blockquote>\$ cd mpcotool/4.4.6 \$ ln -s ../../genetic/3.0.1 genetic \$ In -s genetic/libgenetic.so (or .dll in Windows systems) </blockquote> 5. Build doing on a terminal: <bloom>
<bro>
<bro> clusters, where no-GUI with MPI parallelization is desirable, replace the 5th step of the previous section by: <blockguote>\$ sh build without gui.sh </blockguote>Linking as an external library <hr> MPCOTool can also be used as an external library: 1. First copy the dynamic libraries (libmpcotool.so and libgenetic.so on Unix systems or libmpcotool.dll and libgenetic.dll on Windows systems) to your program directory. 2. Include the function header in your source code:

 blockquote>extern int mpcotool (int argn, char **argc); </blockquote> 3. Build the executable file with the linker and compiler flags: <blockquote>\$ gcc -L. -WI,-rpath=. -Impcotool -Igenetic ... `pkg-config -cflags -libs gsl glib-2.0 json-glib-1.0 ...\' </blockquote> 4. Calling to this function is equivalent to command line order (see next chapter USER INSTRUCTIONS): * argn: number of arguments * argc[0]: "mpcotool" * argc[1]: first command line argument. ... * argc[argn-1]: last argument. FINAL VERSIONS <hr> Optionally, final compact versions without debug information can be built doing on the terminal: <blockquote>\$ make strip </blockquote> <h1>MAKING MANUALS INSTRUCTIONS </h1> On UNIX type systems you need texlive installed. On Windows systems you need MiKTeX. In order to compile the manuals you can type on a terminal: <blockquote>\$ make manuals </blockquote> <h1>MAKING TESTS INSTRUCTIONS </h1> In order to build the tests follow the next instructions: 1. Link some tests that needs genetic library doing in a terminal (assuming that you are in the directory mpcotool/4.4.6): <blookquote>\$ cd ../tests/test2 \$ In -s ../../.genetic/3.0.1 genetic \$ In -s genetic/libgenetic.so (or .dll on Windows systems) \$ cd ../test3 \$ ln -s ../../../genetic/3.0.1 genetic \$ ln -s genetic/libgenetic.so (or .dll on Windows systems) \$ cd ../test4 \$ In -s ../../genetic/3.0.1 genetic \$ In -s genetic/libgenetic.so (or .dll on Windows systems) </br/>blockquote> 2. Build all tests doing in the same terminal:
blockquote>\$ cd ../../4.4.6 $\$ make tests $<\!$ molecular linear linea are typed in square brackets. * Command line in sequential mode (where X is the number of threads to execute and S is a seed for the pseudo-random numbers generator): <blockquote>\$./mpcotoolbin [-nthreads X] [-seed S] input file.xml [result file] [variables file] </blockquote> * Command line in parallelized mode (where X is the number of threads to open for every node and S is a seed for the pseudo-random numbers generator): <blockquote>\$ mpirun [MPI options] ./mpcotoolbin [-nthreads

4 MPCOTool

X] [-seed S] input_file.xml [result_file] [variables_file] </blockquote> * The syntax of the simulator has to be: <blockguote>\$./simulator name input file 1 [input file 2] [input file 3] [input file ← 4] output_file </blockquote> * The syntax of the program to evaluate the objetive function has to ./evaluator_name simulated_file data_file results_file </blockquote> * On UNIX type systems the GUI application can be open doing on a terminal: <blockguote>\$./mpcotool </blockguote> <h1>INPUT FILE FORMAT </h1> The format of the main input file is as: @code{xml} <?xml version="1.← 0"?> <optimize simulator="simulator name" evaluator="evaluator name" algorithm="algorithm $_$ type" nsimulations="simulations $_$ number" niterations="iterations $_$ number" tolerance="tolerance \hookleftarrow _value" nbest="best number" npopulation="population number" ngenerations="generations ← number" mutation="mutation_ratio" reproduction="reproduction_ratio" adaptation="adaptation_← ratio" direction="direction_search_type" nsteps="steps_number" relaxation="relaxation_parameter" nestimates="estimates number" threshold="threshold_parameter" norm="norm type" parameter" seed="random seed" result file="result file" variables file="variables file" < experiment name="data_file_1" template1="template_1_1" template2="template_1_2" ... weight="weight_ ← 1"/> ... <experiment name="data_file_N" template1="template_N_1" template2="template_N_2" ... weight="weight N"/> <variable name="variable 1" minimum="min value" maximum="max value" precision="precision digits" sweeps="sweeps number" nbits="bits number" step="step size"/> ... <variable name="variable_M" minimum="min_value" maximum="max_value" precision="precision←"</p> digits" sweeps="sweeps number" nbits="bits number" step="step size"/> </optimize> @endcode with: * simulator: simulator executable file name. * evaluator↔ : optional. When needed is the evaluator executable file name. * seed: optional. Seed of the pseudo-random numbers generator (default value is 7007). * result_← file: optional. It is the name of the optime result file (default name is "result"). * variables file: optional. It is the name of all simulated variables file (default name is "variables"). * precision: optional, defined for each variable. Number of precision digits to evaluate the variable. 0 apply for integer numbers (default value is 14). * weight: optional, defined for each experiment. Multiplies the objective value obtained for each experiment in the final objective function value (default value is 1). * threshold: optional, to stop the simulations if objective function value less than the threshold is obtained (default value is 0). * algorithm: optimization algorithm type. * norm: error norm type. Implemented algorithms are: * sweep: Sweep brute force algorithm. It requires for each variable: * sweeps: number of sweeps to generate for each variable in every experiment. The total number of simulations to run is: <blockquote>(number of experiments) x (variable 1 number of sweeps) x ... x (variable n number of sweeps) x (number of iterations) </br/>/blockquote> * Monte-Carlo: Monte-Carlo brute force algorithm. It requires on calibrate: * nsimulations: number of simulations to run in every experiment. The total number of simulations to run is: <blockquote>(number of experiments) x (number of simulations) x (number of iterations) </blockguote> * orthogonal: Orthogonal sampling brute force algorithm. It requires for each variable: * sweeps: number of sweeps to generate for each variable in every experiment. The total number of simulations to run is: <blockquote>(number of experiments) x (variable 1 number of sweeps) x ... x (variable n number of sweeps) x (number of iterations) </bl>
</blockquote> * Three former brute force algorithms can be iterated to improve convergence by using the following parameters: * nbest: number of best simulations to calculate convergence interval on next iteration (default 1). * tolerance : tolerance parameter to increase convergence interval (default 0). * niterations: number of iterations (default 1). It multiplies the total number of simulations: <blockquote>x (number of iterations) </blockquote> * Moreover, brute force algorithms can be coupled with a direction search method by using: * direction: method to estimate the optimal direction. Two options are currently available: * coordinates: coordinates descent method. It increases the total number of simulations by: <blockquote>(number of experiments) x (number of iterations) x (number of steps) x 2 x (number of variables) * random: random method. It requires: * nestimates: number of random checks to estimate the optimal direction. </blockquote> It increases the total number of simulations by: <blockquote>(number of experiments) x (number of iterations) x (number of steps) x (number of estimates) </blockquote> Former methods require also: * nsteps: number of steps to perform the direction search method, * relaxation: relaxation parameter, and for each variable: * step: initial step size for the direction search method. * genetic: Genetic algorithm. It requires the following parameters: *

npopulation: number of population. * ngenerations: number of generations. * mutation: mutation ratio. * reproduction: reproduction ratio. * adaptation: adaptation ratio. and for each variable: * nbits: number of bits to encode each variable. The total number of simulations to run is: <blockquote>(number of experiments) x (npopulation) x [1 + (ngenerations - 1) x (mutation + reproduction + adaptation)] </bl></blockquote> Implemented error noms are: * euclidian: Euclidian norm. * maximum: maximum norm. * p: p-norm. It requires the parameter: * p: p exponent. * taxicab: Taxicab norm. Alternatively, the input file can be also written in JSON format as: @code{json} { "simulator": "simulator ← name", "evaluator": "evaluator_name", "algorithm": "algorithm_type", "nsimulations": "simulations_ number", "niterations": "iterations_number", "tolerance": "tolerance_value", "nbest": "best_number", "npopulation": "population_number", "ngenerations": "generations_number", "mutation": "mutation \leftarrow ratio", "reproduction": "reproduction_ratio", "adaptation": "adaptation_ratio", "direction": "direction ← search type", "nsteps": "steps number", "relaxation": "relaxation parameter", "nestimates" ← : "estimates_number", "threshold": "threshold_parameter", "norm": "norm_type", "p": "p_parameter", "seed": "random_seed", "result_file": "result_file", "variables_file": "variables_file", "experiments" -: [{ "name": "data file 1", "template1": "template 1 1", "template2": "template 1 2", ... "weight": "weight_1", }, ... { "name": "data_file_N", "template1": "template_N_1", "template2": "template_← N_2", ... "weight": "weight_N", }], "variables": [{ "name": "variable_1", "minimum": "min_value", "maximum": "max_value", "precision": "precision_digits", "sweeps": "sweeps_number", "nbits": "bits⊷ number", "step": "step size", }, ... { "name": "variable M", "minimum": "min value", "maximum": "max value", "precision": "precision digits", "sweeps": "sweeps number", "nbits": "bits number", "step": "step_size", }] } @endcode <h1>SOME EXAMPLES OF INPUT FILES </h1> Example 1 <hr> * The simulator program name is: pivot * The syntax is: <blockquote>\$./pivot input file output file </blockquote> * The program to evaluate the objective function is: compare * The syntax is: <blockquote>\$./compare simulated_file data_file result_file </bl>
</blockquote> * The calibration is performed with a sweep brute force algorithm. * The experimental data files are:

 27-48.txt 42.txt 52.txt 100.txt </br/> /blockquote> * Templates to get input files to simulator for each experiment are: <blockguote>template1.js template2.js template3.js template4.js </blockquote> * The variables to calibrate, ranges, precision and sweeps number to perform are: <blockquote>alpha1, [179.70, 180.20], 2, 5 alpha2, [179.30, 179.60], 2, 5 random, [0.00, 0.20], 2, 5 boot-time, [0.0, 3.0], 1, 5 </blockquote> * Then, the number of simulations to run is: 4x5x5x5x5=2500. * The input file is: @code{xml} <?xml version="1.0"?> <optimize simulator="pivot" evaluator="compare" algorithm="sweep"> <experiment name="27-48.←</pre> txt" template1="template1.js"/> <experiment name="42.txt" template1="template2.js"/> <experiment name="52.txt" template1="template3.js"/> <experiment name="100.txt" template1="template4.← js"/> <variable name="alpha1" minimum="179.70" maximum="180.20" precision="2" nsweeps="5"/> <variable name="alpha2" minimum="179.30" maximum="179.60" precision="2" nsweeps="5"/> <variable name="random" minimum="0.00" maximum="0.20" precision="2" nsweeps="5"/> <variable</pre> name="boot-time" minimum="0.0" maximum="3.0" precision="1" nsweeps="5"/> </optimize> @endcode * A template file as template1.js: @code{json} { "towers" : [{ "length" : 50.11, "velocity" : 0.02738, "@variable1" : @value1@, "@variable2@" : @value2@, "@variable3@" \leftarrow : @value3@, "@variable4@" : @value4@ }, { "length" : 50.11, "velocity" : 0.02824, "@variable1@" : @value1@, "@variable2@": @value2@, "@variable3@": @value3@, "@variable4@": @value4@ }, { "length" : 50.11, "velocity" : 0.03008, "@variable1@" : @value1@, "@variable2@" : @value2@, "@variable3@": @value3@, "@variable4@": @value4@ }, { "length": 50.11, "velocity": 0.03753, "@variable1@": @value1@, "@variable2@": @value2@, "@variable3@": @value3@, "@variable4@": @value4@ }], "cycle-time": 71.0, "plot-time": 1.0, "comp-time-step": 0.1, "active-percent" : 27.48 }

• produces simulator input files to reproduce the experimental data file 27-48.txt as:

6 MPCOTool

```
{
   "length" : 50.11,
   "velocity" : 0.02824,
   "alpha1" : 179.95,
   "alpha2" : 179.45,
   "random" : 0.10,
   "boot-time" : 1.5
},

{
   "length" : 50.11,
   "velocity" : 0.03008,
   "alpha1" : 179.95,
   "alpha2" : 179.45,
   "random" : 0.10,
   "boot-time" : 1.5
},

{
   "length" : 50.11,
   "velocity" : 0.03753,
   "alpha1" : 179.95,
   "alpha1" : 179.95,
   "alpha2" : 179.45,
   "random" : 0.10,
   "boot-time" : 1.5
},

"cycle-time" : 71.0,
   "plot-time" : 1.0,
   "comp-time-step": 0.1,
   "active-percent" : 27.48
```

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Experime	ant	
	Struct to define the experiment data	-11
Input		
	Struct to define the optimization input file	12
Optimize		
	Struct to define the optimization ation data	20
Options		
	Struct to define the options dialog	33
ParallelD	Pata Pata	
	Struct to pass to the GThreads parallelized function	35
Running		
	Struct to define the running dialog	36
Variable		
	Struct to define the variable data	38
Window		
	Struct to define the main window	40

8 Data Structure Index

Chapter 3

File Index

3.1 File List

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

config.h		
	Configuration header file	69
experim	ent.c	
	Source file to define the experiment data	87
experim	ent.h	
	Header file to define the experiment data	98
input.c		
	Source file to define the input functions	105
input.h		
	Header file to define the input functions	133
interface		
	Source file to define the graphical interface functions	140
interface		
	Header file to define the graphical interface functions	226
main.c		
	Main source file	239
mpcoto		
	Main function source file	243
mpcoto		
	Main function header file	248
optimize		
	Source file to define the optimization functions	251
optimize		
	Header file to define the optimization functions	302
tools.c		
	Source file to define some useful functions	309
tools.h		
	Header file to define some useful functions	312
variable		
	Source file to define the variable data	314
variable		
	Header file to define the variable data	327

10 File Index

Chapter 4

Data Structure Documentation

4.1 Experiment Struct Reference

Struct to define the experiment data.

```
#include <experiment.h>
```

Data Fields

• char * name

File name.

char * stencil [MAX_NINPUTS]

Array of template names of input files.

· double weight

Objective function weight.

· unsigned int ninputs

Number of input files to the simulator.

4.1.1 Detailed Description

Struct to define the experiment data.

Definition at line 45 of file experiment.h.

4.1.2 Field Documentation

4.1.2.1 name

char* Experiment::name

File name.

Definition at line 47 of file experiment.h.

4.1.2.2 ninputs

unsigned int Experiment::ninputs

Number of input files to the simulator.

Definition at line 50 of file experiment.h.

4.1.2.3 stencil

char* Experiment::stencil[MAX_NINPUTS]

Array of template names of input files.

Definition at line 48 of file experiment.h.

4.1.2.4 weight

double Experiment::weight

Objective function weight.

Definition at line 49 of file experiment.h.

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

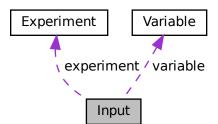
• experiment.h

4.2 Input Struct Reference

Struct to define the optimization input file.

#include <input.h>

Collaboration diagram for Input:



Data Fields

Experiment * experiment

Array or experiments.

• Variable * variable

Array of variables.

• char * result

Name of the result file.

· char * variables

Name of the variables file.

• char * simulator

Name of the simulator program.

· char * evaluator

Name of the program to evaluate the objective function.

· char * directory

Working directory.

• char * name

Input data file name.

· double tolerance

Algorithm tolerance.

double mutation_ratio

Mutation probability.

• double reproduction_ratio

Reproduction probability.

double adaptation_ratio

Adaptation probability.

double relaxation

Relaxation parameter.

double p

Exponent of the P error norm.

· double threshold

Threshold to finish the optimization.

unsigned long int seed

Seed of the pseudo-random numbers generator.

• unsigned int nvariables

Variables number.

• unsigned int nexperiments

Experiments number.

• unsigned int nsimulations

Simulations number per experiment.

• unsigned int algorithm

Algorithm type.

· unsigned int nsteps

Number of steps to do the hill climbing method.

· unsigned int climbing

Method to estimate the hill climbing.

· unsigned int nestimates

Number of simulations to estimate the hill climbing.

· unsigned int niterations

Number of algorithm iterations.

unsigned int nbest

Number of best simulations.

• unsigned int norm

Error norm type.

· unsigned int type

Type of input file.

4.2.1 Detailed Description

Struct to define the optimization input file.

Definition at line 65 of file input.h.

4.2.2 Field Documentation

4.2.2.1 adaptation_ratio

double Input::adaptation_ratio

Adaptation probability.

Definition at line 79 of file input.h.

4.2.2.2 algorithm

unsigned int Input::algorithm

Algorithm type.

Definition at line 88 of file input.h.

4.2.2.3 climbing

unsigned int Input::climbing

Method to estimate the hill climbing.

Definition at line 91 of file input.h.

4.2.2.4 directory

char* Input::directory

Working directory.

Definition at line 74 of file input.h.

4.2.2.5 evaluator

```
char* Input::evaluator
```

Name of the program to evaluate the objective function.

Definition at line 72 of file input.h.

4.2.2.6 experiment

Experiment* Input::experiment

Array or experiments.

Definition at line 67 of file input.h.

4.2.2.7 mutation_ratio

double Input::mutation_ratio

Mutation probability.

Definition at line 77 of file input.h.

4.2.2.8 name

char* Input::name

Input data file name.

Definition at line 75 of file input.h.

4.2.2.9 nbest

unsigned int Input::nbest

Number of best simulations.

Definition at line 95 of file input.h.

4.2.2.10 nestimates

unsigned int Input::nestimates

Number of simulations to estimate the hill climbing.

Definition at line 92 of file input.h.

4.2.2.11 nexperiments

unsigned int Input::nexperiments

Experiments number.

Definition at line 86 of file input.h.

4.2.2.12 niterations

unsigned int Input::niterations

Number of algorithm iterations.

Definition at line 94 of file input.h.

4.2.2.13 norm

unsigned int Input::norm

Error norm type.

Definition at line 96 of file input.h.

4.2.2.14 nsimulations

unsigned int Input::nsimulations

Simulations number per experiment.

Definition at line 87 of file input.h.

4.2.2.15 nsteps

unsigned int Input::nsteps

Number of steps to do the hill climbing method.

Definition at line 89 of file input.h.

4.2.2.16 nvariables

unsigned int Input::nvariables

Variables number.

Definition at line 85 of file input.h.

4.2.2.17 p

double Input::p

Exponent of the P error norm.

Definition at line 81 of file input.h.

4.2.2.18 relaxation

double Input::relaxation

Relaxation parameter.

Definition at line 80 of file input.h.

4.2.2.19 reproduction_ratio

double Input::reproduction_ratio

Reproduction probability.

Definition at line 78 of file input.h.

4.2.2.20 result

char* Input::result

Name of the result file.

Definition at line 69 of file input.h.

4.2.2.21 seed

unsigned long int Input::seed

Seed of the pseudo-random numbers generator.

Definition at line 83 of file input.h.

4.2.2.22 simulator

char* Input::simulator

Name of the simulator program.

Definition at line 71 of file input.h.

4.2.2.23 threshold

double Input::threshold

Threshold to finish the optimization.

Definition at line 82 of file input.h.

4.2.2.24 tolerance

double Input::tolerance

Algorithm tolerance.

Definition at line 76 of file input.h.

4.2.2.25 type

unsigned int Input::type

Type of input file.

Definition at line 97 of file input.h.

4.2.2.26 variable

Variable* Input::variable

Array of variables.

Definition at line 68 of file input.h.

4.2.2.27 variables

char* Input::variables

Name of the variables file.

Definition at line 70 of file input.h.

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

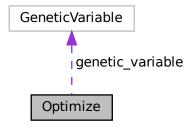
• input.h

4.3 Optimize Struct Reference

Struct to define the optimization ation data.

```
#include <optimize.h>
```

Collaboration diagram for Optimize:



Data Fields

GMappedFile ** file [MAX_NINPUTS]

Matrix of input template files.

char ** experiment

Array of experimental data file names.

• char ** label

Array of variable names.

• gsl_rng * rng

GSL random number generator.

GeneticVariable * genetic_variable

Array of variables for the genetic algorithm.

• FILE * file_result

Result file.

• FILE * file_variables

Variables file.

• char * result

Name of the result file.

char * variables

Name of the variables file.

· char * simulator

Name of the simulator program.

char * evaluator

Name of the program to evaluate the objective function.

double * value

Array of variable values.

• double * rangemin

Array of minimum variable values.

double * rangemax

Array of maximum variable values.

double * rangeminabs

Array of absolute minimum variable values.

double * rangemaxabs

Array of absolute maximum variable values.

double * error best

Array of the best minimum errors.

double * weight

Array of the experiment weights.

double * step

Array of hill climbing method step sizes.

double * climbing

Vector of hill climbing estimation.

double * value old

Array of the best variable values on the previous step.

double * error_old

Array of the best minimum errors on the previous step.

• unsigned int * precision

Array of variable precisions.

• unsigned int * nsweeps

Array of sweeps of the sweep algorithm.

unsigned int * nbits

Array of bits number of the genetic algorithm.

• unsigned int * thread

Array of simulation numbers to calculate on the thread.

- unsigned int * thread_climbing
- unsigned int * simulation_best

Array of best simulation numbers.

· double tolerance

Algorithm tolerance.

double mutation_ratio

Mutation probability.

• double reproduction_ratio

Reproduction probability.

• double adaptation_ratio

Adaptation probability.

• double relaxation

Relaxation parameter.

double calculation_time

Calculation time.

• double p

Exponent of the P error norm.

· double threshold

Threshold to finish the optimization.

• unsigned long int seed

Seed of the pseudo-random numbers generator.

• unsigned int nvariables

Variables number.

· unsigned int nexperiments

Experiments number.

· unsigned int ninputs

Number of input files to the simulator.

• unsigned int nsimulations

Simulations number per experiment.

• unsigned int nsteps

Number of steps for the hill climbing method.

• unsigned int nestimates

Number of simulations to estimate the climbing.

· unsigned int algorithm

Algorithm type.

· unsigned int nstart

Beginning simulation number of the task.

· unsigned int nend

Ending simulation number of the task.

• unsigned int nstart_climbing

Beginning simulation number of the task for the hill climbing method.

• unsigned int nend_climbing

Ending simulation number of the task for the hill climbing method.

· unsigned int niterations

Number of algorithm iterations.

· unsigned int nbest

Number of best simulations.

unsigned int nsaveds

Number of saved simulations.

· unsigned int stop

To stop the simulations.

• int mpi_rank

Number of MPI task.

4.3.1 Detailed Description

Struct to define the optimization ation data.

Definition at line 45 of file optimize.h.

4.3.2 Field Documentation

4.3.2.1 adaptation_ratio

double Optimize::adaptation_ratio

Adaptation probability.

Definition at line 86 of file optimize.h.

4.3.2.2 algorithm

unsigned int Optimize::algorithm

Algorithm type.

Definition at line 101 of file optimize.h.

4.3.2.3 calculation_time

double Optimize::calculation_time

Calculation time.

Definition at line 88 of file optimize.h.

4.3.2.4 climbing

double* Optimize::climbing

Vector of hill climbing estimation.

Definition at line 68 of file optimize.h.

4.3.2.5 error_best

double* Optimize::error_best

Array of the best minimum errors.

Definition at line 65 of file optimize.h.

4.3.2.6 error_old

double* Optimize::error_old

Array of the best minimum errors on the previous step.

Definition at line 71 of file optimize.h.

4.3.2.7 evaluator

```
char* Optimize::evaluator
```

Name of the program to evaluate the objective function.

Definition at line 58 of file optimize.h.

4.3.2.8 experiment

```
char** Optimize::experiment
```

Array of experimental data file names.

Definition at line 48 of file optimize.h.

4.3.2.9 file

```
GMappedFile** Optimize::file[MAX_NINPUTS]
```

Matrix of input template files.

Definition at line 47 of file optimize.h.

4.3.2.10 file_result

```
FILE* Optimize::file_result
```

Result file.

Definition at line 53 of file optimize.h.

4.3.2.11 file_variables

```
FILE* Optimize::file_variables
```

Variables file.

Definition at line 54 of file optimize.h.

4.3.2.12 genetic_variable

```
GeneticVariable* Optimize::genetic_variable
```

Array of variables for the genetic algorithm.

Definition at line 51 of file optimize.h.

4.3.2.13 label

```
char** Optimize::label
```

Array of variable names.

Definition at line 49 of file optimize.h.

4.3.2.14 mpi_rank

```
int Optimize::mpi_rank
```

Number of MPI task.

Definition at line 113 of file optimize.h.

4.3.2.15 mutation_ratio

```
double Optimize::mutation_ratio
```

Mutation probability.

Definition at line 84 of file optimize.h.

4.3.2.16 nbest

unsigned int Optimize::nbest

Number of best simulations.

Definition at line 109 of file optimize.h.

4.3.2.17 nbits

```
unsigned int* Optimize::nbits
```

Array of bits number of the genetic algorithm.

Definition at line 75 of file optimize.h.

4.3.2.18 nend

```
unsigned int Optimize::nend
```

Ending simulation number of the task.

Definition at line 103 of file optimize.h.

4.3.2.19 nend_climbing

```
unsigned int Optimize::nend_climbing
```

Ending simulation number of the task for the hill climbing method.

Definition at line 106 of file optimize.h.

4.3.2.20 nestimates

```
unsigned int Optimize::nestimates
```

Number of simulations to estimate the climbing.

Definition at line 99 of file optimize.h.

4.3.2.21 nexperiments

unsigned int Optimize::nexperiments

Experiments number.

Definition at line 94 of file optimize.h.

4.3.2.22 ninputs

unsigned int Optimize::ninputs

Number of input files to the simulator.

Definition at line 95 of file optimize.h.

4.3.2.23 niterations

unsigned int Optimize::niterations

Number of algorithm iterations.

Definition at line 108 of file optimize.h.

4.3.2.24 nsaveds

unsigned int Optimize::nsaveds

Number of saved simulations.

Definition at line 110 of file optimize.h.

4.3.2.25 nsimulations

unsigned int Optimize::nsimulations

Simulations number per experiment.

Definition at line 96 of file optimize.h.

4.3.2.26 nstart

unsigned int Optimize::nstart

Beginning simulation number of the task.

Definition at line 102 of file optimize.h.

4.3.2.27 nstart_climbing

```
unsigned int Optimize::nstart_climbing
```

Beginning simulation number of the task for the hill climbing method.

Definition at line 104 of file optimize.h.

4.3.2.28 nsteps

```
unsigned int Optimize::nsteps
```

Number of steps for the hill climbing method.

Definition at line 97 of file optimize.h.

4.3.2.29 nsweeps

```
unsigned int* Optimize::nsweeps
```

Array of sweeps of the sweep algorithm.

Definition at line 74 of file optimize.h.

4.3.2.30 nvariables

```
unsigned int Optimize::nvariables
```

Variables number.

Definition at line 93 of file optimize.h.

4.3.2.31 p

```
double Optimize::p
```

Exponent of the P error norm.

Definition at line 89 of file optimize.h.

4.3.2.32 precision

unsigned int* Optimize::precision

Array of variable precisions.

Definition at line 73 of file optimize.h.

4.3.2.33 rangemax

double* Optimize::rangemax

Array of maximum variable values.

Definition at line 62 of file optimize.h.

4.3.2.34 rangemaxabs

double* Optimize::rangemaxabs

Array of absolute maximum variable values.

Definition at line 64 of file optimize.h.

4.3.2.35 rangemin

double* Optimize::rangemin

Array of minimum variable values.

Definition at line 61 of file optimize.h.

4.3.2.36 rangeminabs

double* Optimize::rangeminabs

Array of absolute minimum variable values.

Definition at line 63 of file optimize.h.

4.3.2.37 relaxation

double Optimize::relaxation

Relaxation parameter.

Definition at line 87 of file optimize.h.

4.3.2.38 reproduction_ratio

double Optimize::reproduction_ratio

Reproduction probability.

Definition at line 85 of file optimize.h.

4.3.2.39 result

char* Optimize::result

Name of the result file.

Definition at line 55 of file optimize.h.

4.3.2.40 rng

gsl_rng* Optimize::rng

GSL random number generator.

Definition at line 50 of file optimize.h.

4.3.2.41 seed

unsigned long int Optimize::seed

Seed of the pseudo-random numbers generator.

Definition at line 91 of file optimize.h.

4.3.2.42 simulation_best

unsigned int* Optimize::simulation_best

Array of best simulation numbers.

Definition at line 82 of file optimize.h.

4.3.2.43 simulator

char* Optimize::simulator

Name of the simulator program.

Definition at line 57 of file optimize.h.

4.3.2.44 step

double* Optimize::step

Array of hill climbing method step sizes.

Definition at line 67 of file optimize.h.

4.3.2.45 stop

unsigned int Optimize::stop

To stop the simulations.

Definition at line 111 of file optimize.h.

4.3.2.46 thread

unsigned int* Optimize::thread

Array of simulation numbers to calculate on the thread.

Definition at line 77 of file optimize.h.

4.3.2.47 thread_climbing

```
unsigned int* Optimize::thread_climbing
```

Array of simulation numbers to calculate on the thread for the hill climbing method.

Definition at line 79 of file optimize.h.

4.3.2.48 threshold

```
double Optimize::threshold
```

Threshold to finish the optimization.

Definition at line 90 of file optimize.h.

4.3.2.49 tolerance

double Optimize::tolerance

Algorithm tolerance.

Definition at line 83 of file optimize.h.

4.3.2.50 value

double* Optimize::value

Array of variable values.

Definition at line 60 of file optimize.h.

4.3.2.51 value_old

double* Optimize::value_old

Array of the best variable values on the previous step.

Definition at line 69 of file optimize.h.

4.3.2.52 variables

```
char* Optimize::variables
```

Name of the variables file.

Definition at line 56 of file optimize.h.

4.3.2.53 weight

```
double* Optimize::weight
```

Array of the experiment weights.

Definition at line 66 of file optimize.h.

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

· optimize.h

4.4 Options Struct Reference

Struct to define the options dialog.

```
#include <interface.h>
```

Data Fields

• GtkDialog * dialog

Main GtkDialog.

• GtkGrid * grid

Main GtkGrid.

GtkLabel * label_seed

Pseudo-random numbers generator seed GtkLabel.

• GtkSpinButton * spin_seed

 ${\it Pseudo-random\ numbers\ generator\ seed\ GtkSpinButton}.$

• GtkLabel * label_threads

Threads number GtkLabel.

• GtkSpinButton * spin_threads

Threads number GtkSpinButton.

• GtkLabel * label_climbing

Climbing threads number GtkLabel.

• GtkSpinButton * spin_climbing

 ${\it Climbing threads number GtkSpinButton}.$

4.4.1 Detailed Description

Struct to define the options dialog.

Definition at line 48 of file interface.h.

4.4.2 Field Documentation

4.4.2.1 dialog

GtkDialog* Options::dialog

Main GtkDialog.

Definition at line 50 of file interface.h.

4.4.2.2 grid

GtkGrid* Options::grid

Main GtkGrid.

Definition at line 51 of file interface.h.

4.4.2.3 label_climbing

GtkLabel* Options::label_climbing

Climbing threads number GtkLabel.

Definition at line 58 of file interface.h.

4.4.2.4 label_seed

GtkLabel* Options::label_seed

Pseudo-random numbers generator seed GtkLabel.

Definition at line 52 of file interface.h.

4.4.2.5 label_threads

GtkLabel* Options::label_threads

Threads number GtkLabel.

Definition at line 56 of file interface.h.

4.4.2.6 spin_climbing

GtkSpinButton* Options::spin_climbing

Climbing threads number GtkSpinButton.

Definition at line 59 of file interface.h.

4.4.2.7 spin_seed

GtkSpinButton* Options::spin_seed

Pseudo-random numbers generator seed GtkSpinButton.

Definition at line 54 of file interface.h.

4.4.2.8 spin_threads

GtkSpinButton* Options::spin_threads

 $Threads\ number\ GtkSpinButton.$

Definition at line 57 of file interface.h.

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

· interface.h

4.5 ParallelData Struct Reference

Struct to pass to the GThreads parallelized function.

#include <optimize.h>

Data Fields

· unsigned int thread

Thread number.

4.5.1 Detailed Description

Struct to pass to the GThreads parallelized function.

Definition at line 121 of file optimize.h.

4.5.2 Field Documentation

4.5.2.1 thread

```
unsigned int ParallelData::thread
```

Thread number.

Definition at line 123 of file optimize.h.

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

• optimize.h

4.6 Running Struct Reference

Struct to define the running dialog.

```
#include <interface.h>
```

Data Fields

• GtkDialog * dialog

Main GtkDialog.

• GtkLabel * label

Label GtkLabel.

• GtkSpinner * spinner

Animation GtkSpinner.

• GtkGrid * grid

Grid GtkGrid.

4.6.1 Detailed Description

Struct to define the running dialog.

Definition at line 66 of file interface.h.

4.6.2 Field Documentation

4.6.2.1 dialog

GtkDialog* Running::dialog

Main GtkDialog.

Definition at line 68 of file interface.h.

4.6.2.2 grid

GtkGrid* Running::grid

Grid GtkGrid.

Definition at line 71 of file interface.h.

4.6.2.3 label

GtkLabel* Running::label

Label GtkLabel.

Definition at line 69 of file interface.h.

4.6.2.4 spinner

GtkSpinner* Running::spinner

Animation GtkSpinner.

Definition at line 70 of file interface.h.

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

• interface.h

4.7 Variable Struct Reference

Struct to define the variable data.

#include <variable.h>

Data Fields

• char * name

Variable name.

• double rangemin

Minimum variable value.

double rangemax

Maximum variable value.

· double rangeminabs

Absolute minimum variable value.

· double rangemaxabs

Absolute maximum variable value.

· double step

Hill climbing method step size.

• unsigned int precision

Variable precision.

• unsigned int nsweeps

Sweeps of the sweep algorithm.

· unsigned int nbits

Bits number of the genetic algorithm.

4.7.1 Detailed Description

Struct to define the variable data.

Definition at line 54 of file variable.h.

4.7.2 Field Documentation

4.7.2.1 name

char* Variable::name

Variable name.

Definition at line 56 of file variable.h.

4.7.2.2 nbits

unsigned int Variable::nbits

Bits number of the genetic algorithm.

Definition at line 64 of file variable.h.

4.7.2.3 nsweeps

unsigned int Variable::nsweeps

Sweeps of the sweep algorithm.

Definition at line 63 of file variable.h.

4.7.2.4 precision

unsigned int Variable::precision

Variable precision.

Definition at line 62 of file variable.h.

4.7.2.5 rangemax

double Variable::rangemax

Maximum variable value.

Definition at line 58 of file variable.h.

4.7.2.6 rangemaxabs

double Variable::rangemaxabs

Absolute maximum variable value.

Definition at line 60 of file variable.h.

4.7.2.7 rangemin

double Variable::rangemin

Minimum variable value.

Definition at line 57 of file variable.h.

4.7.2.8 rangeminabs

double Variable::rangeminabs

Absolute minimum variable value.

Definition at line 59 of file variable.h.

4.7.2.9 step

double Variable::step

Hill climbing method step size.

Definition at line 61 of file variable.h.

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

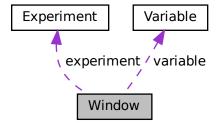
· variable.h

4.8 Window Struct Reference

Struct to define the main window.

#include <interface.h>

Collaboration diagram for Window:



Data Fields

GtkWindow * window

Main GtkWindow.

GtkGrid * grid

Main GtkGrid.

• GtkBox * box_buttons

GtkBox to store the main buttons.

• GtkButton * button_open

Open GtkButton.

• GtkButton * button_save

Save GtkButton.

• GtkButton * button_run

Run GtkButton.

• GtkButton * button_options

Options GtkButton.

• GtkButton * button_help

Help GtkButton.

• GtkButton * button_about

Help GtkButton.

• GtkButton * button_exit

Exit GtkButton.

GtkGrid * grid_files

Files GtkGrid.

• GtkLabel * label_simulator

Simulator program GtkLabel.

• GtkButton * button_simulator

Simulator program GtkButton.

GtkCheckButton * check_evaluator

Evaluator program GtkCheckButton.

• GtkButton * button evaluator

Evaluator program GtkButton.

• GtkLabel * label_result

Result file GtkLabel.

GtkEntry * entry_result

Result file GtkEntry.

• GtkLabel * label_variables

Variables file GtkLabel.

• GtkEntry * entry_variables

Variables file GtkEntry.

• GtkFrame * frame norm

GtkFrame to set the error norm.

GtkGrid * grid_norm

GtkGrid to set the error norm.

GtkRadioButton * button norm [NNORMS]

Array of GtkRadioButtons to set the error norm.

GtkLabel * label_p

GtkLabel to set the p parameter.

GtkSpinButton * spin p

GtkSpinButton to set the p parameter.

GtkScrolledWindow * scrolled_p

GtkScrolledWindow to set the p parameter.

• GtkFrame * frame_algorithm

GtkFrame to set the algorithm.

• GtkGrid * grid algorithm

GtkGrid to set the algorithm.

GtkRadioButton * button algorithm [NALGORITHMS]

Array of GtkRadioButtons to set the algorithm.

GtkLabel * label_simulations

GtkLabel to set the simulations number.

• GtkSpinButton * spin_simulations

GtkSpinButton to set the simulations number.

• GtkLabel * label iterations

GtkLabel to set the iterations number.

• GtkSpinButton * spin_iterations

GtkSpinButton to set the iterations number.

• GtkLabel * label_tolerance

GtkLabel to set the tolerance.

• GtkSpinButton * spin tolerance

GtkSpinButton to set the tolerance.

GtkLabel * label bests

GtkLabel to set the best number.

• GtkSpinButton * spin_bests

GtkSpinButton to set the best number.

• GtkLabel * label_population

GtkLabel to set the population number.

 $\bullet \ \, \mathsf{GtkSpinButton} * \mathsf{spin_population} \\$

GtkSpinButton to set the population number.

GtkLabel * label_generations

GtkLabel to set the generations number.

• GtkSpinButton * spin generations

GtkSpinButton to set the generations number.

GtkLabel * label_mutation

GtkLabel to set the mutation ratio.

• GtkSpinButton * spin_mutation

GtkSpinButton to set the mutation ratio.

• GtkLabel * label_reproduction

GtkLabel to set the reproduction ratio.

• GtkSpinButton * spin_reproduction

GtkSpinButton to set the reproduction ratio.

GtkLabel * label adaptation

GtkLabel to set the adaptation ratio.

• GtkSpinButton * spin adaptation

GtkSpinButton to set the adaptation ratio.

GtkCheckButton * check_climbing

GtkCheckButton to check running the hill climbing method.

• GtkGrid * grid_climbing

GtkGrid to pack the hill climbing method widgets.

GtkRadioButton * button_climbing [NCLIMBINGS]

Array of GtkRadioButtons array to set the hill climbing method.

• GtkLabel * label_steps

GtkLabel to set the steps number.

• GtkSpinButton * spin_steps

GtkSpinButton to set the steps number.

• GtkLabel * label estimates

GtkLabel to set the estimates number.

GtkSpinButton * spin_estimates

GtkSpinButton to set the estimates number.

• GtkLabel * label relaxation

GtkLabel to set the relaxation parameter.

• GtkSpinButton * spin_relaxation

GtkSpinButton to set the relaxation parameter.

• GtkLabel * label threshold

GtkLabel to set the threshold.

GtkSpinButton * spin_threshold

GtkSpinButton to set the threshold.

• GtkScrolledWindow * scrolled threshold

GtkScrolledWindow to set the threshold.

• GtkFrame * frame_variable

Variable GtkFrame.

• GtkGrid * grid_variable

Variable GtkGrid.

GtkComboBoxText * combo variable

GtkComboBoxEntry to select a variable.

• GtkButton * button add variable

GtkButton to add a variable.

• GtkButton * button_remove_variable

GtkButton to remove a variable.

GtkLabel * label_variable

Variable GtkLabel.

• GtkEntry * entry_variable

GtkEntry to set the variable name.

• GtkLabel * label min

Minimum GtkLabel.

• GtkSpinButton * spin_min

Minimum GtkSpinButton.

• GtkScrolledWindow * scrolled min

Minimum GtkScrolledWindow.

GtkLabel * label_max

Maximum GtkLabel.

• GtkSpinButton * spin_max

Maximum GtkSpinButton.

• GtkScrolledWindow * scrolled max

Maximum GtkScrolledWindow.

GtkCheckButton * check_minabs

Absolute minimum GtkCheckButton.

• GtkSpinButton * spin minabs

Absolute minimum GtkSpinButton.

GtkScrolledWindow * scrolled_minabs

Absolute minimum GtkScrolledWindow.

GtkCheckButton * check maxabs

Absolute maximum GtkCheckButton.

GtkSpinButton * spin_maxabs

Absolute maximum GtkSpinButton.

GtkScrolledWindow * scrolled_maxabs

Absolute maximum GtkScrolledWindow.

• GtkLabel * label precision

Precision GtkLabel.

GtkSpinButton * spin precision

Precision digits GtkSpinButton.

GtkLabel * label_sweeps

Sweeps number GtkLabel.

• GtkSpinButton * spin_sweeps

Sweeps number GtkSpinButton.

• GtkLabel * label_bits

Bits number GtkLabel.

• GtkSpinButton * spin_bits

Bits number GtkSpinButton.

• GtkLabel * label_step

GtkLabel to set the step.

• GtkSpinButton * spin_step

GtkSpinButton to set the step.

GtkScrolledWindow * scrolled step

step GtkScrolledWindow.

• GtkFrame * frame_experiment

Experiment GtkFrame.

• GtkGrid * grid_experiment

Experiment GtkGrid.

 $\bullet \ \, \mathsf{GtkComboBoxText} * \mathbf{combo_experiment}$

Experiment GtkComboBoxEntry.

GtkButton * button_add_experiment

GtkButton to add a experiment.

• GtkButton * button remove experiment

GtkButton to remove a experiment.

• GtkLabel * label experiment

Experiment GtkLabel.

• GtkButton * button_experiment

GtkButton to set the experimental data file.

• GtkLabel * label_weight

Weight GtkLabel.

• GtkSpinButton * spin_weight

Weight GtkSpinButton.

GtkCheckButton * check_template [MAX_NINPUTS]

Array of GtkCheckButtons to set the input templates.

• GtkButton * button_template [MAX_NINPUTS]

Array of GtkButtons to set the input templates.

GdkPixbuf * logo

Logo GdkPixbuf.

• Experiment * experiment

Array of experiments data.

• Variable * variable

Array of variables data.

· char * application_directory

Application directory.

• gulong id_experiment

Identifier of the combo_experiment signal.

• gulong id_experiment_name

Identifier of the button_experiment signal.

• gulong id_variable

Identifier of the combo_variable signal.

• gulong id_variable_label

Identifier of the entry_variable signal.

• gulong id_template [MAX_NINPUTS]

Array of identifiers of the check_template signal.

• gulong id_input [MAX_NINPUTS]

Array of identifiers of the button_template signal.

• unsigned int nexperiments

Number of experiments.

• unsigned int nvariables

Number of variables.

4.8.1 Detailed Description

Struct to define the main window.

Definition at line 78 of file interface.h.

4.8.2 Field Documentation

4.8.2.1 application_directory

char* Window::application_directory

Application directory.

Definition at line 215 of file interface.h.

4.8.2.2 box_buttons

GtkBox* Window::box_buttons

GtkBox to store the main buttons.

Definition at line 82 of file interface.h.

4.8.2.3 button_about

GtkButton* Window::button_about

Help GtkButton.

Definition at line 88 of file interface.h.

4.8.2.4 button_add_experiment

GtkButton* Window::button_add_experiment

GtkButton to add a experiment.

Definition at line 201 of file interface.h.

4.8.2.5 button_add_variable

GtkButton* Window::button_add_variable

GtkButton to add a variable.

Definition at line 173 of file interface.h.

4.8.2.6 button_algorithm

 ${\tt GtkRadioButton*\ Window::button_algorithm[NALGORITHMS]}$

Array of GtkRadioButtons to set the algorithm.

Definition at line 115 of file interface.h.

4.8.2.7 button_climbing

GtkRadioButton* Window::button_climbing[NCLIMBINGS]

Array of GtkRadioButtons array to set the hill climbing method.

Definition at line 150 of file interface.h.

4.8.2.8 button_evaluator

GtkButton* Window::button_evaluator

Evaluator program GtkButton.

Definition at line 94 of file interface.h.

4.8.2.9 button_exit

GtkButton* Window::button_exit

Exit GtkButton.

Definition at line 89 of file interface.h.

4.8.2.10 button_experiment

GtkButton* Window::button_experiment

GtkButton to set the experimental data file.

Definition at line 204 of file interface.h.

4.8.2.11 button_help

GtkButton* Window::button_help

Help GtkButton.

Definition at line 87 of file interface.h.

4.8.2.12 button_norm

GtkRadioButton* Window::button_norm[NNORMS]

Array of GtkRadioButtons to set the error norm.

Definition at line 102 of file interface.h.

4.8.2.13 button_open

GtkButton* Window::button_open

Open GtkButton.

Definition at line 83 of file interface.h.

4.8.2.14 button_options

GtkButton* Window::button_options

Options GtkButton.

Definition at line 86 of file interface.h.

4.8.2.15 button_remove_experiment

GtkButton* Window::button_remove_experiment

GtkButton to remove a experiment.

Definition at line 202 of file interface.h.

4.8.2.16 button_remove_variable

GtkButton* Window::button_remove_variable

GtkButton to remove a variable.

Definition at line 174 of file interface.h.

4.8.2.17 button_run

GtkButton* Window::button_run

Run GtkButton.

Definition at line 85 of file interface.h.

4.8.2.18 button_save

GtkButton* Window::button_save

Save GtkButton.

Definition at line 84 of file interface.h.

4.8.2.19 button_simulator

GtkButton* Window::button_simulator

Simulator program GtkButton.

Definition at line 92 of file interface.h.

4.8.2.20 button_template

GtkButton* Window::button_template[MAX_NINPUTS]

Array of GtkButtons to set the input templates.

Definition at line 210 of file interface.h.

4.8.2.21 check_climbing

GtkCheckButton* Window::check_climbing

GtkCheckButton to check running the hill climbing method.

Definition at line 145 of file interface.h.

4.8.2.22 check_evaluator

GtkCheckButton* Window::check_evaluator

Evaluator program GtkCheckButton.

Definition at line 93 of file interface.h.

4.8.2.23 check_maxabs

GtkCheckButton* Window::check_maxabs

Absolute maximum GtkCheckButton.

Definition at line 186 of file interface.h.

4.8.2.24 check_minabs

GtkCheckButton* Window::check_minabs

Absolute minimum GtkCheckButton.

Definition at line 183 of file interface.h.

4.8.2.25 check_template

GtkCheckButton* Window::check_template[MAX_NINPUTS]

Array of GtkCheckButtons to set the input templates.

Definition at line 208 of file interface.h.

4.8.2.26 combo_experiment

GtkComboBoxText* Window::combo_experiment

Experiment GtkComboBoxEntry.

Definition at line 200 of file interface.h.

4.8.2.27 combo_variable

GtkComboBoxText* Window::combo_variable

GtkComboBoxEntry to select a variable.

Definition at line 171 of file interface.h.

4.8.2.28 entry_result

GtkEntry* Window::entry_result

Result file GtkEntry.

Definition at line 96 of file interface.h.

4.8.2.29 entry_variable

GtkEntry* Window::entry_variable

GtkEntry to set the variable name.

Definition at line 176 of file interface.h.

4.8.2.30 entry_variables

GtkEntry* Window::entry_variables

Variables file GtkEntry.

Definition at line 98 of file interface.h.

4.8.2.31 experiment

Experiment* Window::experiment

Array of experiments data.

Definition at line 213 of file interface.h.

4.8.2.32 frame_algorithm

GtkFrame* Window::frame_algorithm

GtkFrame to set the algorithm.

Definition at line 112 of file interface.h.

4.8.2.33 frame_experiment

GtkFrame* Window::frame_experiment

Experiment GtkFrame.

Definition at line 198 of file interface.h.

4.8.2.34 frame_norm

GtkFrame* Window::frame_norm

GtkFrame to set the error norm.

Definition at line 99 of file interface.h.

4.8.2.35 frame_variable

GtkFrame* Window::frame_variable

Variable GtkFrame.

Definition at line 169 of file interface.h.

4.8.2.36 grid

GtkGrid* Window::grid

Main GtkGrid.

Definition at line 81 of file interface.h.

4.8.2.37 grid_algorithm

GtkGrid* Window::grid_algorithm

GtkGrid to set the algorithm.

Definition at line 113 of file interface.h.

4.8.2.38 grid_climbing

GtkGrid* Window::grid_climbing

GtkGrid to pack the hill climbing method widgets.

Definition at line 147 of file interface.h.

4.8.2.39 grid_experiment

GtkGrid* Window::grid_experiment

Experiment GtkGrid.

Definition at line 199 of file interface.h.

4.8.2.40 grid_files

GtkGrid* Window::grid_files

Files GtkGrid.

Definition at line 90 of file interface.h.

4.8.2.41 grid_norm

GtkGrid* Window::grid_norm

GtkGrid to set the error norm.

Definition at line 100 of file interface.h.

4.8.2.42 grid_variable

GtkGrid* Window::grid_variable

Variable GtkGrid.

Definition at line 170 of file interface.h.

4.8.2.43 id_experiment

```
gulong Window::id_experiment
```

Identifier of the combo_experiment signal.

Definition at line 216 of file interface.h.

4.8.2.44 id_experiment_name

```
gulong Window::id_experiment_name
```

Identifier of the button_experiment signal.

Definition at line 217 of file interface.h.

4.8.2.45 id_input

```
gulong Window::id_input[MAX_NINPUTS]
```

Array of identifiers of the button_template signal.

Definition at line 222 of file interface.h.

4.8.2.46 id_template

```
gulong Window::id_template[MAX_NINPUTS]
```

Array of identifiers of the check_template signal.

Definition at line 220 of file interface.h.

4.8.2.47 id_variable

```
gulong Window::id_variable
```

Identifier of the combo variable signal.

Definition at line 218 of file interface.h.

4.8.2.48 id_variable_label

gulong Window::id_variable_label

Identifier of the entry_variable signal.

Definition at line 219 of file interface.h.

4.8.2.49 label_adaptation

GtkLabel* Window::label_adaptation

GtkLabel to set the adaptation ratio.

Definition at line 142 of file interface.h.

4.8.2.50 label_bests

GtkLabel* Window::label_bests

GtkLabel to set the best number.

Definition at line 129 of file interface.h.

4.8.2.51 label_bits

GtkLabel* Window::label_bits

Bits number GtkLabel.

Definition at line 193 of file interface.h.

4.8.2.52 label_estimates

GtkLabel* Window::label_estimates

GtkLabel to set the estimates number.

Definition at line 158 of file interface.h.

4.8.2.53 label_experiment

GtkLabel* Window::label_experiment

Experiment GtkLabel.

Definition at line 203 of file interface.h.

4.8.2.54 label_generations

GtkLabel* Window::label_generations

GtkLabel to set the generations number.

Definition at line 134 of file interface.h.

4.8.2.55 label_iterations

GtkLabel* Window::label_iterations

GtkLabel to set the iterations number.

Definition at line 124 of file interface.h.

4.8.2.56 label_max

GtkLabel* Window::label_max

Maximum GtkLabel.

Definition at line 180 of file interface.h.

4.8.2.57 label_min

GtkLabel* Window::label_min

Minimum GtkLabel.

Definition at line 177 of file interface.h.

4.8.2.58 label_mutation

GtkLabel* Window::label_mutation

GtkLabel to set the mutation ratio.

Definition at line 137 of file interface.h.

4.8.2.59 label_p

GtkLabel* Window::label_p

GtkLabel to set the p parameter.

Definition at line 108 of file interface.h.

4.8.2.60 label_population

GtkLabel* Window::label_population

GtkLabel to set the population number.

Definition at line 131 of file interface.h.

4.8.2.61 label_precision

GtkLabel* Window::label_precision

Precision GtkLabel.

Definition at line 189 of file interface.h.

4.8.2.62 label_relaxation

GtkLabel* Window::label_relaxation

GtkLabel to set the relaxation parameter.

Definition at line 161 of file interface.h.

4.8.2.63 label_reproduction

GtkLabel* Window::label_reproduction

GtkLabel to set the reproduction ratio.

Definition at line 139 of file interface.h.

4.8.2.64 label_result

GtkLabel* Window::label_result

Result file GtkLabel.

Definition at line 95 of file interface.h.

4.8.2.65 label_simulations

GtkLabel* Window::label_simulations

GtkLabel to set the simulations number.

Definition at line 121 of file interface.h.

4.8.2.66 label_simulator

GtkLabel* Window::label_simulator

Simulator program GtkLabel.

Definition at line 91 of file interface.h.

4.8.2.67 label_step

GtkLabel* Window::label_step

GtkLabel to set the step.

Definition at line 195 of file interface.h.

4.8.2.68 label_steps

GtkLabel* Window::label_steps

GtkLabel to set the steps number.

Definition at line 156 of file interface.h.

4.8.2.69 label_sweeps

GtkLabel* Window::label_sweeps

Sweeps number GtkLabel.

Definition at line 191 of file interface.h.

4.8.2.70 label_threshold

GtkLabel* Window::label_threshold

GtkLabel to set the threshold.

Definition at line 165 of file interface.h.

4.8.2.71 label_tolerance

GtkLabel* Window::label_tolerance

GtkLabel to set the tolerance.

Definition at line 127 of file interface.h.

4.8.2.72 label_variable

GtkLabel* Window::label_variable

Variable GtkLabel.

Definition at line 175 of file interface.h.

4.8.2.73 label_variables

GtkLabel* Window::label_variables

Variables file GtkLabel.

Definition at line 97 of file interface.h.

4.8.2.74 label_weight

GtkLabel* Window::label_weight

Weight GtkLabel.

Definition at line 206 of file interface.h.

4.8.2.75 logo

GdkPixbuf* Window::logo

Logo GdkPixbuf.

Definition at line 212 of file interface.h.

4.8.2.76 nexperiments

unsigned int Window::nexperiments

Number of experiments.

Definition at line 224 of file interface.h.

4.8.2.77 nvariables

unsigned int Window::nvariables

Number of variables.

Definition at line 225 of file interface.h.

4.8.2.78 scrolled_max

GtkScrolledWindow* Window::scrolled_max

Maximum GtkScrolledWindow.

Definition at line 182 of file interface.h.

4.8.2.79 scrolled_maxabs

GtkScrolledWindow* Window::scrolled_maxabs

Absolute maximum GtkScrolledWindow.

Definition at line 188 of file interface.h.

4.8.2.80 scrolled_min

GtkScrolledWindow* Window::scrolled_min

Minimum GtkScrolledWindow.

Definition at line 179 of file interface.h.

4.8.2.81 scrolled_minabs

GtkScrolledWindow* Window::scrolled_minabs

Absolute minimum GtkScrolledWindow.

Definition at line 185 of file interface.h.

4.8.2.82 scrolled_p

GtkScrolledWindow* Window::scrolled_p

GtkScrolledWindow to set the p parameter.

Definition at line 110 of file interface.h.

4.8.2.83 scrolled_step

GtkScrolledWindow* Window::scrolled_step

step GtkScrolledWindow.

Definition at line 197 of file interface.h.

4.8.2.84 scrolled_threshold

GtkScrolledWindow* Window::scrolled_threshold

GtkScrolledWindow to set the threshold.

Definition at line 167 of file interface.h.

4.8.2.85 spin_adaptation

GtkSpinButton* Window::spin_adaptation

GtkSpinButton to set the adaptation ratio.

Definition at line 143 of file interface.h.

4.8.2.86 spin_bests

GtkSpinButton* Window::spin_bests

GtkSpinButton to set the best number.

Definition at line 130 of file interface.h.

4.8.2.87 spin_bits

GtkSpinButton* Window::spin_bits

Bits number GtkSpinButton.

Definition at line 194 of file interface.h.

4.8.2.88 spin_estimates

GtkSpinButton* Window::spin_estimates

GtkSpinButton to set the estimates number.

Definition at line 159 of file interface.h.

4.8.2.89 spin_generations

GtkSpinButton* Window::spin_generations

GtkSpinButton to set the generations number.

Definition at line 135 of file interface.h.

4.8.2.90 spin_iterations

GtkSpinButton* Window::spin_iterations

GtkSpinButton to set the iterations number.

Definition at line 125 of file interface.h.

4.8.2.91 spin_max

GtkSpinButton* Window::spin_max

Maximum GtkSpinButton.

Definition at line 181 of file interface.h.

4.8.2.92 spin_maxabs

GtkSpinButton* Window::spin_maxabs

Absolute maximum GtkSpinButton.

Definition at line 187 of file interface.h.

4.8.2.93 spin_min

GtkSpinButton* Window::spin_min

Minimum GtkSpinButton.

Definition at line 178 of file interface.h.

4.8.2.94 spin_minabs

GtkSpinButton* Window::spin_minabs

Absolute minimum GtkSpinButton.

Definition at line 184 of file interface.h.

4.8.2.95 spin_mutation

GtkSpinButton* Window::spin_mutation

GtkSpinButton to set the mutation ratio.

Definition at line 138 of file interface.h.

4.8.2.96 spin_p

GtkSpinButton* Window::spin_p

GtkSpinButton to set the p parameter.

Definition at line 109 of file interface.h.

4.8.2.97 spin_population

GtkSpinButton* Window::spin_population

GtkSpinButton to set the population number.

Definition at line 132 of file interface.h.

4.8.2.98 spin_precision

GtkSpinButton* Window::spin_precision

Precision digits GtkSpinButton.

Definition at line 190 of file interface.h.

4.8.2.99 spin_relaxation

GtkSpinButton* Window::spin_relaxation

GtkSpinButton to set the relaxation parameter.

Definition at line 163 of file interface.h.

4.8.2.100 spin_reproduction

GtkSpinButton* Window::spin_reproduction

GtkSpinButton to set the reproduction ratio.

Definition at line 140 of file interface.h.

4.8.2.101 spin_simulations

GtkSpinButton* Window::spin_simulations

GtkSpinButton to set the simulations number.

Definition at line 122 of file interface.h.

4.8.2.102 spin_step

GtkSpinButton* Window::spin_step

GtkSpinButton to set the step.

Definition at line 196 of file interface.h.

4.8.2.103 spin_steps

GtkSpinButton* Window::spin_steps

GtkSpinButton to set the steps number.

Definition at line 157 of file interface.h.

4.8.2.104 spin_sweeps

GtkSpinButton* Window::spin_sweeps

Sweeps number GtkSpinButton.

Definition at line 192 of file interface.h.

4.8.2.105 spin_threshold

GtkSpinButton* Window::spin_threshold

GtkSpinButton to set the threshold.

Definition at line 166 of file interface.h.

4.8.2.106 spin_tolerance

GtkSpinButton* Window::spin_tolerance

GtkSpinButton to set the tolerance.

Definition at line 128 of file interface.h.

4.8.2.107 spin_weight

GtkSpinButton* Window::spin_weight

Weight GtkSpinButton.

Definition at line 207 of file interface.h.

4.8.2.108 variable

Variable* Window::variable

Array of variables data.

Definition at line 214 of file interface.h.

4.8.2.109 window

GtkWindow* Window::window

Main GtkWindow.

Definition at line 80 of file interface.h.

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

• interface.h

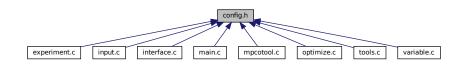
Chapter 5

File Documentation

5.1 config.h File Reference

Configuration header file.

This graph shows which files directly or indirectly include this file:



Macros

• #define MAX_NINPUTS 8

Maximum number of input files in the simulator program.

• #define NALGORITHMS 4

Number of stochastic algorithms.

• #define NCLIMBINGS 2

Number of hill climbing estimate methods.

• #define NNORMS 4

Number of error norms.

• #define NPRECISIONS 15

Number of precisions.

• #define DEFAULT_PRECISION (NPRECISIONS - 1)

Default precision digits.

• #define DEFAULT_RANDOM_SEED 7007

Default pseudo-random numbers seed.

#define DEFAULT_RELAXATION 1.

Default relaxation parameter.

• #define LOCALE_DIR "locales"

Locales directory.

#define PROGRAM_INTERFACE "mpcotool"

Name of the interface program.

#define LABEL_ABSOLUTE_MINIMUM "absolute_minimum"

absolute minimum label.

#define LABEL_ABSOLUTE_MAXIMUM "absolute_maximum"

absolute maximum label.

#define LABEL ADAPTATION "adaptation"

adaption label.

• #define LABEL ALGORITHM "algorithm"

algoritm label.

#define LABEL_CLIMBING "climbing"

climbing label.

• #define LABEL_COORDINATES "coordinates"

coordinates label.

• #define LABEL_EUCLIDIAN "euclidian"

euclidian label.

• #define LABEL EVALUATOR "evaluator"

evaluator label.

• #define LABEL_EXPERIMENT "experiment"

experiment label.

#define LABEL EXPERIMENTS "experiments"

experiment label.

• #define LABEL_GENETIC "genetic"

genetic label.

• #define LABEL_MINIMUM "minimum"

minimum label.

• #define LABEL MAXIMUM "maximum"

maximum label.

• #define LABEL_MONTE_CARLO "Monte-Carlo"

Monte-Carlo label.

• #define LABEL MUTATION "mutation"

mutation label.

• #define LABEL NAME "name"

name label.

• #define LABEL_NBEST "nbest"

nbest label.

• #define LABEL_NBITS "nbits"

nbits label.

• #define LABEL_NESTIMATES "nestimates"

nestimates label.

#define LABEL_NGENERATIONS "ngenerations"

ngenerations label.

• #define LABEL NITERATIONS "niterations"

niterations label.

#define LABEL_NORM "norm"

norm label.

• #define LABEL_NPOPULATION "npopulation"

npopulation label.

#define LABEL_NSIMULATIONS "nsimulations"

nsimulations label.

• #define LABEL_NSTEPS "nsteps"

nsteps label.

```
    #define LABEL_NSWEEPS "nsweeps"

     nsweeps label.
• #define LABEL OPTIMIZE "optimize"
     optimize label.

    #define LABEL_ORTHOGONAL "orthogonal"

     orthogonal label.
• #define LABEL_P "p"
     p label.

    #define LABEL_PRECISION "precision"

     precision label.

    #define LABEL RANDOM "random"

     random label.

    #define LABEL_RELAXATION "relaxation"

     relaxation label.
• #define LABEL REPRODUCTION "reproduction"
     reproduction label.

    #define LABEL_RESULT_FILE "result_file"

     result file label.
• #define LABEL_SIMULATOR "simulator"
     simulator label.

    #define LABEL_SEED "seed"

     seed label.

    #define LABEL STEP "step"

     step label.
• #define LABEL_SWEEP "sweep"
     sweep label.
• #define LABEL_TAXICAB "taxicab"
     taxicab label.
• #define LABEL_TEMPLATE1 "template1"
     template1 label.

    #define LABEL_TEMPLATE2 "template2"

     template2 label.
• #define LABEL_TEMPLATE3 "template3"
     template3 label.

    #define LABEL_TEMPLATE4 "template4"

     template4 label.
• #define LABEL TEMPLATE5 "template5"
     template5 label.

    #define LABEL_TEMPLATE6 "template6"

     template6 label.
• #define LABEL TEMPLATE7 "template7"
     template7 label.

    #define LABEL_TEMPLATE8 "template8"

     template8 label.
• #define LABEL_THRESHOLD "threshold"
     threshold label.

    #define LABEL TOLERANCE "tolerance"

     tolerance label.

    #define LABEL VARIABLE "variable"
```

variable label.

#define LABEL_VARIABLES "variables"

```
variables label.
```

• #define LABEL_VARIABLES_FILE "variables_file"

variables label.

#define LABEL_WEIGHT "weight"

weight label.

Enumerations

```
• enum INPUT_TYPE { INPUT_TYPE_XML = 0 , INPUT_TYPE_JSON = 1 } 
Enum to define the input file types.
```

5.1.1 Detailed Description

Configuration header file.

Authors

Javier Burguete and Borja Latorre.

Copyright

Copyright 2012-2018, all rights reserved.

Definition in file config.h.

5.1.2 Macro Definition Documentation

5.1.2.1 DEFAULT_PRECISION

```
#define DEFAULT_PRECISION (NPRECISIONS - 1)
```

Default precision digits.

Definition at line 55 of file config.h.

5.1.2.2 DEFAULT_RANDOM_SEED

```
#define DEFAULT_RANDOM_SEED 7007
```

Default pseudo-random numbers seed.

Definition at line 56 of file config.h.

5.1.2.3 DEFAULT_RELAXATION

```
#define DEFAULT_RELAXATION 1.
```

Default relaxation parameter.

Definition at line 57 of file config.h.

5.1.2.4 LABEL_ABSOLUTE_MAXIMUM

```
#define LABEL_ABSOLUTE_MAXIMUM "absolute_maximum"
```

absolute maximum label.

Definition at line 69 of file config.h.

5.1.2.5 LABEL_ABSOLUTE_MINIMUM

```
#define LABEL_ABSOLUTE_MINIMUM "absolute_minimum"
```

absolute minimum label.

Definition at line 67 of file config.h.

5.1.2.6 LABEL_ADAPTATION

```
#define LABEL_ADAPTATION "adaptation"
```

adaption label.

Definition at line 70 of file config.h.

5.1.2.7 LABEL_ALGORITHM

```
#define LABEL_ALGORITHM "algorithm"
```

algoritm label.

Definition at line 71 of file config.h.

5.1.2.8 LABEL_CLIMBING

```
#define LABEL_CLIMBING "climbing"
```

climbing label.

Definition at line 72 of file config.h.

5.1.2.9 LABEL_COORDINATES

```
#define LABEL_COORDINATES "coordinates"
```

coordinates label.

Definition at line 73 of file config.h.

5.1.2.10 LABEL_EUCLIDIAN

```
#define LABEL_EUCLIDIAN "euclidian"
```

euclidian label.

Definition at line 74 of file config.h.

5.1.2.11 LABEL_EVALUATOR

```
#define LABEL_EVALUATOR "evaluator"
```

evaluator label.

Definition at line 75 of file config.h.

5.1.2.12 LABEL_EXPERIMENT

```
#define LABEL_EXPERIMENT "experiment"
```

experiment label.

Definition at line 76 of file config.h.

5.1.2.13 LABEL_EXPERIMENTS

#define LABEL_EXPERIMENTS "experiments"

experiment label.

Definition at line 77 of file config.h.

5.1.2.14 LABEL_GENETIC

#define LABEL_GENETIC "genetic"

genetic label.

Definition at line 78 of file config.h.

5.1.2.15 LABEL_MAXIMUM

#define LABEL_MAXIMUM "maximum"

maximum label.

Definition at line 80 of file config.h.

5.1.2.16 LABEL_MINIMUM

#define LABEL_MINIMUM "minimum"

minimum label.

Definition at line 79 of file config.h.

5.1.2.17 LABEL_MONTE_CARLO

#define LABEL_MONTE_CARLO "Monte-Carlo"

Monte-Carlo label.

Definition at line 81 of file config.h.

5.1.2.18 LABEL_MUTATION

```
#define LABEL_MUTATION "mutation"
```

mutation label.

Definition at line 82 of file config.h.

5.1.2.19 **LABEL_NAME**

```
#define LABEL_NAME "name"
```

name label.

Definition at line 83 of file config.h.

5.1.2.20 LABEL_NBEST

```
#define LABEL_NBEST "nbest"
```

nbest label.

Definition at line 84 of file config.h.

5.1.2.21 LABEL_NBITS

```
#define LABEL_NBITS "nbits"
```

nbits label.

Definition at line 85 of file config.h.

5.1.2.22 LABEL_NESTIMATES

```
#define LABEL_NESTIMATES "nestimates"
```

nestimates label.

Definition at line 86 of file config.h.

5.1.2.23 LABEL_NGENERATIONS

#define LABEL_NGENERATIONS "ngenerations"

ngenerations label.

Definition at line 87 of file config.h.

5.1.2.24 LABEL_NITERATIONS

#define LABEL_NITERATIONS "niterations"

niterations label.

Definition at line 88 of file config.h.

5.1.2.25 LABEL_NORM

#define LABEL_NORM "norm"

norm label.

Definition at line 89 of file config.h.

5.1.2.26 LABEL_NPOPULATION

#define LABEL_NPOPULATION "npopulation"

npopulation label.

Definition at line 90 of file config.h.

5.1.2.27 LABEL_NSIMULATIONS

#define LABEL_NSIMULATIONS "nsimulations"

nsimulations label.

Definition at line 91 of file config.h.

5.1.2.28 LABEL_NSTEPS

```
#define LABEL_NSTEPS "nsteps"
```

nsteps label.

Definition at line 92 of file config.h.

5.1.2.29 LABEL_NSWEEPS

```
#define LABEL_NSWEEPS "nsweeps"
```

nsweeps label.

Definition at line 93 of file config.h.

5.1.2.30 LABEL_OPTIMIZE

```
#define LABEL_OPTIMIZE "optimize"
```

optimize label.

Definition at line 94 of file config.h.

5.1.2.31 LABEL_ORTHOGONAL

```
#define LABEL_ORTHOGONAL "orthogonal"
```

orthogonal label.

Definition at line 95 of file config.h.

5.1.2.32 LABEL_P

```
#define LABEL_P "p"
```

p label.

Definition at line 96 of file config.h.

5.1.2.33 LABEL_PRECISION

```
#define LABEL_PRECISION "precision"
```

precision label.

Definition at line 97 of file config.h.

5.1.2.34 LABEL_RANDOM

```
#define LABEL_RANDOM "random"
```

random label.

Definition at line 98 of file config.h.

5.1.2.35 LABEL_RELAXATION

```
#define LABEL_RELAXATION "relaxation"
```

relaxation label.

Definition at line 99 of file config.h.

5.1.2.36 LABEL_REPRODUCTION

```
#define LABEL_REPRODUCTION "reproduction"
```

reproduction label.

Definition at line 100 of file config.h.

5.1.2.37 LABEL_RESULT_FILE

```
#define LABEL_RESULT_FILE "result_file"
```

result_file label.

Definition at line 101 of file config.h.

5.1.2.38 LABEL_SEED

```
#define LABEL_SEED "seed"
```

seed label.

Definition at line 103 of file config.h.

5.1.2.39 LABEL_SIMULATOR

```
#define LABEL_SIMULATOR "simulator"
```

simulator label.

Definition at line 102 of file config.h.

5.1.2.40 LABEL_STEP

```
#define LABEL_STEP "step"
```

step label.

Definition at line 104 of file config.h.

5.1.2.41 LABEL_SWEEP

```
#define LABEL_SWEEP "sweep"
```

sweep label.

Definition at line 105 of file config.h.

5.1.2.42 LABEL_TAXICAB

#define LABEL_TAXICAB "taxicab"

taxicab label.

Definition at line 106 of file config.h.

5.1.2.43 LABEL_TEMPLATE1

```
#define LABEL_TEMPLATE1 "template1"
```

template1 label.

Definition at line 107 of file config.h.

5.1.2.44 LABEL_TEMPLATE2

```
#define LABEL_TEMPLATE2 "template2"
```

template2 label.

Definition at line 108 of file config.h.

5.1.2.45 LABEL_TEMPLATE3

```
#define LABEL_TEMPLATE3 "template3"
```

template3 label.

Definition at line 109 of file config.h.

5.1.2.46 LABEL_TEMPLATE4

```
#define LABEL_TEMPLATE4 "template4"
```

template4 label.

Definition at line 110 of file config.h.

5.1.2.47 LABEL_TEMPLATE5

```
#define LABEL_TEMPLATE5 "template5"
```

template5 label.

Definition at line 111 of file config.h.

5.1.2.48 LABEL_TEMPLATE6

```
#define LABEL_TEMPLATE6 "template6"
```

template6 label.

Definition at line 112 of file config.h.

5.1.2.49 LABEL_TEMPLATE7

```
#define LABEL_TEMPLATE7 "template7"
```

template7 label.

Definition at line 113 of file config.h.

5.1.2.50 LABEL_TEMPLATE8

```
#define LABEL_TEMPLATE8 "template8"
```

template8 label.

Definition at line 114 of file config.h.

5.1.2.51 LABEL_THRESHOLD

```
#define LABEL_THRESHOLD "threshold"
```

threshold label.

Definition at line 115 of file config.h.

5.1.2.52 LABEL_TOLERANCE

#define LABEL_TOLERANCE "tolerance"

tolerance label.

Definition at line 116 of file config.h.

5.1.2.53 LABEL_VARIABLE

```
#define LABEL_VARIABLE "variable"
```

variable label.

Definition at line 117 of file config.h.

5.1.2.54 LABEL_VARIABLES

```
#define LABEL_VARIABLES "variables"
```

variables label.

Definition at line 118 of file config.h.

5.1.2.55 LABEL_VARIABLES_FILE

```
#define LABEL_VARIABLES_FILE "variables_file"
```

variables label.

Definition at line 119 of file config.h.

5.1.2.56 LABEL_WEIGHT

```
#define LABEL_WEIGHT "weight"
```

weight label.

Definition at line 120 of file config.h.

5.1.2.57 LOCALE_DIR

```
#define LOCALE_DIR "locales"
```

Locales directory.

Definition at line 61 of file config.h.

5.1.2.58 MAX_NINPUTS

```
#define MAX_NINPUTS 8
```

Maximum number of input files in the simulator program.

Definition at line 47 of file config.h.

5.1.2.59 NALGORITHMS

```
#define NALGORITHMS 4
```

Number of stochastic algorithms.

Definition at line 48 of file config.h.

5.1.2.60 NCLIMBINGS

```
#define NCLIMBINGS 2
```

Number of hill climbing estimate methods.

Definition at line 49 of file config.h.

5.1.2.61 NNORMS

#define NNORMS 4

Number of error norms.

Definition at line 50 of file config.h.

5.1.2.62 NPRECISIONS

#define NPRECISIONS 15

Number of precisions.

Definition at line 51 of file config.h.

5.2 config.h 85

5.1.2.63 PROGRAM_INTERFACE

```
#define PROGRAM_INTERFACE "mpcotool"
```

Name of the interface program.

Definition at line 62 of file config.h.

5.1.3 Enumeration Type Documentation

5.1.3.1 INPUT_TYPE

```
enum INPUT_TYPE
```

Enum to define the input file types.

Enumerator

INPUT_TYPE_XML	XML input file.
INPUT_TYPE_JSON	JSON input file.

Definition at line 125 of file config.h.

5.2 config.h

Go to the documentation of this file.

```
00001 /* config.h.
00002 /*
                      Generated from config.h.in by configure.
00003 MPCOTool:
00004 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00005 calibrations or optimizations of empirical parameters.
00006
00007 AUTHORS: Javier Burguete and Borja Latorre.
00008
00009 Copyright 2012-2018, AUTHORS.
00010
00011 Redistribution and use in source and binary forms, with or without modification,
00012 are permitted provided that the following conditions are met:
00013
00014 1. Redistributions of source code must retain the above copyright notice,
00015 this list of conditions and the following disclaimer.
00016
00017 2.
         Redistributions in binary form must reproduce the above copyright notice,
00018 this list of conditions and the following disclaimer in the
00019 documentation and/or other materials provided with the distribution.
00020
00021 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00022 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
00023 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00024 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00025 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
```

```
00026 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00027 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00028 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING 00029 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00030 OF SUCH DAMAGE.
00031 */
00032
00039 #ifndef CONFIG__H
00040 #define CONFIG__H 1
00041
00042 /* #undef HAVE MPI */
00043
00044 // Array sizes
00045
00046 #define MAX_NINPUTS 8
00048 #define NALGORITHMS 4
00049 #define NCLIMBINGS 2
00050 #define NNORMS 4
00051 #define NPRECISIONS 15
00052
00053 // Default choices
00054
00055 #define DEFAULT_PRECISION (NPRECISIONS - 1)
00056 #define DEFAULT_RANDOM_SEED 7007
00057 #define DEFAULT_RELAXATION 1.
00058
00059 // Interface labels
00060
00061 #define LOCALE_DIR "locales"
00062 #define PROGRAM_INTERFACE "mpcotool"
00063
00064 // Labels
00065
00066 #define LABEL_ABSOLUTE_MINIMUM "absolute_minimum"
00068 #define LABEL_ABSOLUTE_MAXIMUM "absolute_maximum" 00070 #define LABEL_ADAPTATION "adaptation"
00071 #define LABEL_ALGORITHM "algorithm"
00072 #define LABEL_CLIMBING "climbing"
00073 #define LABEL_COORDINATES "coordinates"
00074 #define LABEL_EUCLIDIAN "euclidian"
00075 #define LABEL_EVALUATOR "evaluator"
00076 #define LABEL_EXPERIMENT "experiment"
00077 #define LABEL_EXPERIMENTS "experiments"
00078 #define LABEL_GENETIC "genetic"
00079 #define LABEL_MINIMUM "minimum"
00080 #define LABEL_MAXIMUM "maximum"
00081 #define LABEL_MONTE_CARLO "Monte-Carlo"
00082 #define LABEL_MUTATION "mutation"
00082 #define LABEL_NAME "name"
00084 #define LABEL_NBEST "nbest"
00085 #define LABEL_NBITS "nbits"
00086 #define LABEL_NESTIMATES "nestimates"
00087 #define LABEL_NGENERATIONS "ngenerations"
00088 #define LABEL_NITERATIONS "niterations"
00089 #define LABEL_NORM "norm"
00090 #define LABEL_NPOPULATION "npopulation"
00091 #define LABEL_NSIMULATIONS "nsimulations"
00092 #define LABEL_NSTEPS "nsteps"
00093 #define LABEL_NSWEEPS "nsweeps"
00094 #define LABEL_OPTIMIZE "optimize"
00095 #define LABEL_ORTHOGONAL "orthogonal"
00096 #define LABEL_P "p"
00097 #define LABEL_PRECISION "precision"
00098 #define LABEL_RANDOM "random"
00099 #define LABEL_RELAXATION "relaxation"
00100 #define LABEL_REPRODUCTION "reproduction"
00101 #define LABEL_RESULT_FILE "result_file"
00102 #define LABEL_SIMULATOR "simulator"
00103 #define LABEL_SEED "seed"
00104 #define LABEL_STEP "step"
00105 #define LABEL_SWEEP "sweep"
00106 #define LABEL_TAXICAB "taxicab"
00107 #define LABEL_TEMPLATE1 "template1"
00108 #define LABEL_TEMPLATE2 "template2"
00109 #define LABEL_TEMPLATE3 "template3"
00110 #define LABEL_TEMPLATE4 "template4"
00111 #define LABEL_TEMPLATE5 "template5"
00112 #define LABEL_TEMPLATE6 "template6"
00113 #define LABEL_TEMPLATE7 "template7"
00114 #define LABEL_TEMPLATE8 "template8"
00115 #define LABEL_THRESHOLD "threshold"
00116 #define LABEL_TOLERANCE "tolerance"
00117 #define LABEL_VARIABLE "variable"
00118 #define LABEL_VARIABLES "variables"
00119 #define LABEL_VARIABLES_FILE "variables_file"
00120 #define LABEL_WEIGHT "weight"
00121
```

5.3 experiment.c File Reference

Source file to define the experiment data.

```
#include "config.h"
#include <stdio.h>
#include <libxml/parser.h>
#include <libintl.h>
#include <glib.h>
#include <json-glib/json-glib.h>
#include "jb/src/jb_xml.h"
#include "jb/src/jb_json.h"
#include "jb/src/jb_win.h"
#include "tools.h"
#include "experiment.h"
Include dependency graph for experiment.c:
```

experiment.c



Macros

• #define DEBUG EXPERIMENT 0

Macro to debug experiment functions.

Functions

- static void experiment_new (Experiment *experiment)
- void experiment_free (Experiment *experiment, unsigned int type)
- void experiment error (Experiment *experiment, char *message)
- int experiment_open_xml (Experiment *experiment, xmlNode *node, unsigned int ninputs)
- int experiment_open_json (Experiment *experiment, JsonNode *node, unsigned int ninputs)

Variables

const char * stencil [MAX_NINPUTS]

Array of xmlChar strings with stencil labels.

5.3.1 Detailed Description

Source file to define the experiment data.

Authors

Javier Burguete and Borja Latorre.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file experiment.c.

5.3.2 Macro Definition Documentation

5.3.2.1 DEBUG_EXPERIMENT

```
#define DEBUG_EXPERIMENT 0
```

Macro to debug experiment functions.

Definition at line 51 of file experiment.c.

5.3.3 Function Documentation

5.3.3.1 experiment error()

Function to print a message error opening an Experiment struct.

Parameters

experiment	Experiment struct.
message	Error message.

Definition at line 112 of file experiment.c.

```
00114 {
00115    if (!experiment->name)
```

5.3.3.2 experiment_free()

Function to free the memory of an Experiment struct.

Parameters

experiment	Experiment struct.
type	Type of input file.

Definition at line 83 of file experiment.c.

```
00085 {
00086
        unsigned int i;
00087 #if DEBUG_EXPERIMENT
00088 fprintf (stderr, "experiment_free: start\n");
00089 #endif
00090 if (type == INPUT_TYPE_XML)
       {
00091
00092
           for (i = 0; i < experiment->ninputs; ++i)
00093
             xmlFree (experiment->stencil[i]);
00094
           xmlFree (experiment->name);
00095
00096
       else
       {
00097
00098
           for (i = 0; i < experiment->ninputs; ++i)
00099
             g_free (experiment->stencil[i]);
00100
           g_free (experiment->name);
         }
00101     }
00102     experiment->ninputs = 0;
00103 #if DEBUG_EXPERIMENT
00104 fprintf (stderr, "experiment_free: end\n");
00105 #endif
00106 }
```

5.3.3.3 experiment_new()

Function to create a new Experiment struct.

Parameters

experiment | Experiment struct.

Definition at line 64 of file experiment.c.

```
00065 {
00066 unsigned int i;
```

5.3.3.4 experiment_open_json()

Function to open the Experiment struct on a XML node.

Returns

1 on success, 0 on error.

Parameters

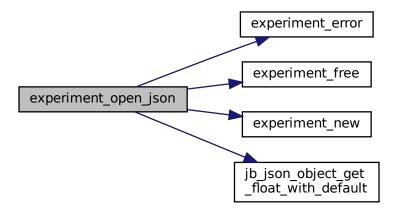
experiment	Experiment struct.
node	JSON node.
ninputs	Number of the simulator input files.

Definition at line 231 of file experiment.c.

```
00235 {
00236
        char buffer[64];
        JsonObject *object;
const char *name;
00237
00238
00239
        int error_code;
00240
       unsigned int i;
00241
00242 #if DEBUG_EXPERIMENT
00243
       fprintf (stderr, "experiment_open_json: start\n");
00244 #endif
00245
00246
        // Resetting experiment data
00247
       experiment_new (experiment);
00248
00249
        // Getting JSON object
00250
        object = json_node_get_object (node);
00251
00252
        // Reading the experimental data
00253
        name = json_object_get_string_member (object, LABEL_NAME);
        if (!name)
00254
00255
         {
00256
            experiment_error (experiment, _("no data file name"));
00257
            goto exit_on_error;
00258
00259 experiment->name = g_strdup (name); 00260 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_json: name=%s\n", experiment->name);
00261
00262 #endif
00263
00264
           = jb_json_object_get_float_with_default (object, LABEL_WEIGHT, &error_code,
00265
00266
        if (!error_code)
00267
00268
            experiment_error (experiment, _("bad weight"));
00269
            goto exit_on_error;
```

```
00271 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_json: weight=%lg\n", experiment->weight);
00272
00273 #endif
00274 name = json_object_get_string_member (object, stencil[0]);
00275
       if (name)
00276
00277 #if DEBUG_EXPERIMENT
00278 fprintf (stderr, "experiment_open_json: experiment=%s template1=%s\n",
00279
                     name, stencil[0]);
00280 #endif
           ++experiment->ninputs;
00281
00282
00283
       else
00284
        {
00285
            experiment_error (experiment, _("no template"));
00286
           goto exit_on_error;
00287
       experiment->stencil[0] = g_strdup (name);
00288
00289
       for (i = 1; i < MAX_NINPUTS; ++i)</pre>
00290
00291 #if DEBUG_EXPERIMENT
            fprintf (stderr, "experiment_open_json: stencil%u\n", i + 1);
00292
00293 #endif
00294
            if (json_object_get_member (object, stencil[i]))
00295
00296
                if (ninputs && ninputs <= i)</pre>
00297
00298
                    experiment_error (experiment, _("bad templates number"));
00299
                    goto exit_on_error;
00300
00301
                name = json_object_get_string_member (object, stencil[i]);
00302 #if DEBUG_EXPERIMENT
00303
            fprintf (stderr,
00304
                         "experiment_open_json: experiment=%s stencil%u=%sn",
                         experiment->nexperiments, name, stencil[i]);
00305
00306 #endif
                experiment->stencil[i] = g_strdup (name);
00308
                ++experiment->ninputs;
00309
00310
            else if (ninputs && ninputs > i)
            {
00311
               snprintf (buffer, 64, "%s%u", _("no template"), i + 1);
experiment_error (experiment, buffer);
00312
00313
00314
                goto exit_on_error;
00315
             }
00316
           else
00317
             break;
         }
00318
00319
00320 #if DEBUG_EXPERIMENT
00321 fprintf (stderr, "experiment_open_json: end\n");
00322 #endif
00323
       return 1;
00324
00325 exit_on_error:
00326 experiment_free (experiment, INPUT_TYPE_JSON);
00327 #if DEBUG_EXPERIMENT
00328
       fprintf (stderr, "experiment_open_json: end\n");
00329 #endif
       return 0:
00330
00331 }
```

Here is the call graph for this function:



5.3.3.5 experiment_open_xml()

Function to open the Experiment struct on a XML node.

Returns

1 on success, 0 on error.

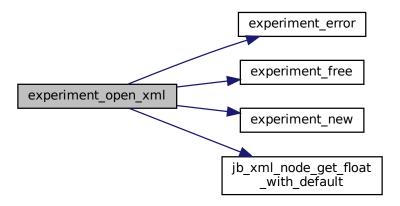
Parameters

experiment	Experiment struct.
node	XML node.
ninputs	Number of the simulator input files.

Definition at line 128 of file experiment.c.

```
// Reading the experimental data
00145
        experiment->name = (char *) xmlGetProp (node, (const xmlChar *) LABEL_NAME);
00146
           (!experiment->name)
00147
        {
00148
            experiment_error (experiment, _("no data file name"));
00149
            goto exit on error:
00150
00151 #if DEBUG_EXPERIMENT
00152
       fprintf (stderr, "experiment_open_xml: name=%s\n", experiment->name);
00153 #endif
00154
       experiment->weight
         = jb_xml_node_get_float_with_default (node, (const xmlChar *) LABEL_WEIGHT,
00155
00156
                                                 &error code, 1.);
00157
       if (!error_code)
00158
        {
00159
            experiment_error (experiment, _("bad weight"));
00160
            goto exit_on_error;
00161
00162 #if DEBUG_EXPERIMENT
00163
       fprintf (stderr, "experiment_open_xml: weight=%lg\n", experiment->weight);
00164 #endif
00165 experiment->stencil[0]
00166
         = (char *) xmlGetProp (node, (const xmlChar *) stencil[0]);
00167
       if (experiment->stencil[0])
00168
00169 #if DEBUG_EXPERIMENT
00170
            fprintf (stderr, "experiment_open_xml: experiment=%s stencil1=%s\n",
00171
                     experiment->name, stencil[0]);
00172 #endif
00173
           ++experiment->ninputs;
         }
00174
00175
       else
00176
00177
            experiment_error (experiment, _("no template"));
00178
           goto exit_on_error;
         }
00179
00180
       for (i = 1; i < MAX NINPUTS; ++i)</pre>
00181
00182 #if DEBUG_EXPERIMENT
00183
            fprintf (stderr, "experiment_open_xml: stencil%u\n", i + 1);
00184 #endif
00185
            if (xmlHasProp (node, (const xmlChar *) stencil[i]))
00186
00187
                if (ninputs && ninputs <= i)</pre>
00188
                 {
00189
                    experiment_error (experiment, _("bad templates number"));
00190
                    goto exit_on_error;
00191
00192
               experiment->stencil[i]
00193
                  = (char *) xmlGetProp (node, (const xmlChar *) stencil[i]);
00194 #if DEBUG_EXPERIMENT
00195
               fprintf (stderr,
00196
                         "experiment_open_xml: experiment=%s stencil%u=%sn",
00197
                         experiment->nexperiments, experiment->name,
00198
                         experiment->stencil[i]);
00199 #endif
                ++experiment->ninputs;
00201
00202
            else if (ninputs && ninputs > i)
00203
             {
               snprintf (buffer, 64, "%s%u", _("no template"), i + 1);
experiment_error (experiment, buffer);
00204
00205
00206
                goto exit_on_error;
00207
00208
           else
00209
              break;
00210
         }
00211
00212 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_xml: end\n");
00214 #endif
00215
       return 1;
00216
00217 exit on error:
       experiment_free (experiment, INPUT_TYPE_XML);
00218
00219 #if DEBUG_EXPERIMENT
00220
       fprintf (stderr, "experiment_open_xml: end\n");
00221 #endif
00222
       return 0;
00223 }
```

Here is the call graph for this function:



5.3.4 Variable Documentation

5.3.4.1 stencil

```
const char* stencil[MAX_NINPUTS]
```

Initial value:

```
= {
    LABEL_TEMPLATE1, LABEL_TEMPLATE2, LABEL_TEMPLATE3, LABEL_TEMPLATE4,
    LABEL_TEMPLATE5, LABEL_TEMPLATE6, LABEL_TEMPLATE7, LABEL_TEMPLATE8
```

Array of xmlChar strings with stencil labels.

Definition at line 53 of file experiment.c.

5.4 experiment.c

Go to the documentation of this file.

```
00001 /*
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00005
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00009
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are met:
00012
00013 1. Redistributions of source code must retain the above copyright notice,
00014 this list of conditions and the following disclaimer.
00015
00016 2. Redistributions in binary form must reproduce the above copyright notice,
```

5.4 experiment.c 95

```
00017 this list of conditions and the following disclaimer in the
00018 documentation and/or other materials provided with the distribution.
00019
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF 00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING 00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
00031
00038 #define _GNU_SOURCE
00039 #include "config.h"
00040 #include <stdio.h>
00041 #include <libxml/parser.h>
00042 #include <libintl.h>
00043 #include <glib.h>
00044 #include <json-glib/json-glib.h>
00045 #include 'jso/src/jb_xml.h"
00046 #include "jb/src/jb_json.h"
00047 #include "jb/src/jb_win.h"
00048 #include "tools.h"
00049 #include "experiment.h"
00050
00051 #define DEBUG_EXPERIMENT 0
00052
00053 const char *stencil[MAX_NINPUTS] = {
00054    LABEL_TEMPLATE1, LABEL_TEMPLATE2, LABEL_TEMPLATE3, LABEL_TEMPLATE4,
00055    LABEL_TEMPLATE5, LABEL_TEMPLATE6, LABEL_TEMPLATE7, LABEL_TEMPLATE8
00056 };
00057
00059
00063 static void
00064 experiment_new (Experiment * experiment)
00065 {
00066
         unsigned int i;
00067 #if DEBUG_EXPERIMENT
        fprintf (stderr, "experiment_new: start\n");
00068
00069 #endif
00070 experiment->name = NULL;
00071 experiment->ninputs = 0;
        for (i = 0; i < MAX_NINPUTS; ++i)
00072
00073 experiment->stencil[i] = NULL;
00074 #if DEBUG_EXPERIMENT
00075 fprintf (stderr, "input_new: end\n");
00076 #endif
00077 }
00078
00082 void
00083 experiment_free (Experiment * experiment,
00084
                           unsigned int type)
00085 {
         unsigned int i;
00087 #if DEBUG_EXPERIMENT
00088
         fprintf (stderr, "experiment_free: start\n");
00089 #endif
00090
         if (type == INPUT_TYPE_XML)
00091
          {
00092
              for (i = 0; i < experiment->ninputs; ++i)
00093
                 xmlFree (experiment->stencil[i]);
00094
              xmlFree (experiment->name);
00095
           }
00096
         else
00097
         {
00098
             for (i = 0; i < experiment->ninputs; ++i)
00099
                g_free (experiment->stencil[i]);
00100
              g_free (experiment->name);
00101
00102
         experiment->ninputs = 0;
00103 #if DEBUG_EXPERIMENT
         fprintf (stderr, "experiment_free: end\n");
00104
00105 #endif
00106 }
00107
00111 void
00112 experiment_error (Experiment * experiment,
00113
                            char *message)
00114 {
00115
         if (!experiment->name)
00116
           error_message = g_strconcat (_("Experiment"), ": ", message, NULL);
00117
         else
           error_message = g_strconcat (_("Experiment"), " ", experiment->name, ": ",
00118
00119
                                               message, NULL);
```

```
00120 }
00121
00127 int
00128 experiment_open_xml (Experiment * experiment,
00129
                           xmlNode * node.
                           unsigned int ninputs)
00130
00132 {
00133
       char buffer[64];
       int error_code;
00134
00135
       unsigned int i;
00136
00137 #if DEBUG_EXPERIMENT
00138
       fprintf (stderr, "experiment_open_xml: start\n");
00139 #endif
00140
00141
       // Resetting experiment data
00142
       experiment_new (experiment);
00143
       // Reading the experimental data
00145
       experiment->name = (char *) xmlGetProp (node, (const xmlChar *) LABEL_NAME);
00146
       if (!experiment->name)
00147
00148
            experiment_error (experiment, _("no data file name"));
00149
           goto exit_on_error;
00150
00151 #if DEBUG_EXPERIMENT
00152
       fprintf (stderr, "experiment_open_xml: name=%s\n", experiment->name);
00153 #endif
00154
       experiment->weight
          = jb_xml_node_get_float_with_default (node, (const xmlChar *) LABEL_WEIGHT,
00155
00156
                                                &error code, 1.);
00157
        if (!error code)
00158
00159
            experiment_error (experiment, _("bad weight"));
00160
            goto exit_on_error;
00161
00162 #if DEBUG_EXPERIMENT
00163 fprintf (stderr, "experiment_open_xml: weight=%lg\n", experiment->weight);
00164 #endif
00165 experiment->stencil[0]
00166
          = (char *) xmlGetProp (node, (const xmlChar *) stencil[0]);
       if (experiment->stencil[0])
00167
00168
00169 #if DEBUG_EXPERIMENT
00170 fprintf (stderr, "experiment_open_xml: experiment=%s stencil1=%s\n",
00171
                     experiment->name, stencil[0]);
00172 #endif
00173
           ++experiment->ninputs;
         }
00174
00175
       else
00176
       {
         experiment_error (experiment, _("no template"));
00177
00178
           goto exit_on_error;
00179
       for (i = 1; i < MAX_NINPUTS; ++i)</pre>
00180
00181
00182 #if DEBUG_EXPERIMENT
00183
            fprintf (stderr, "experiment_open_xml: stencil%u\n", i + 1);
00184 #endif
00185
            if (xmlHasProp (node, (const xmlChar *) stencil[i]))
00186
00187
               if (ninputs && ninputs <= i)
00188
                 {
                  experiment_error (experiment, _("bad templates number"));
00189
00190
                   goto exit_on_error;
00191
00192
                experiment->stencil[i]
                  = (char *) xmlGetProp (node, (const xmlChar *) stencil[i]);
00193
00194 #if DEBUG_EXPERIMENT
00195
             fprintf (stderr,
00196
                         "experiment_open_xml: experiment=%s stencil%u=%sn",
00197
                         experiment->nexperiments, experiment->name,
                         experiment->stencil[i]);
00198
00199 #endif
00200
               ++experiment->ninputs;
00201
00202
            else if (ninputs && ninputs > i)
00203
               snprintf (buffer, 64, "%s%u", _("no template"), i + 1);
experiment_error (experiment, buffer);
00204
00205
00206
                goto exit_on_error;
00207
              }
00208
            else
              break;
00209
00210
         }
00211
00212 #if DEBUG_EXPERIMENT
```

5.4 experiment.c 97

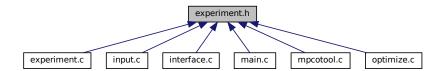
```
fprintf (stderr, "experiment_open_xml: end\n");
00214 #endif
00215
        return 1;
00216
00217 exit_on_error:
00218
       experiment_free (experiment, INPUT_TYPE_XML);
00219 #if DEBUG_EXPERIMENT
00220
       fprintf (stderr, "experiment_open_xml: end\n");
00221 #endif
ouz22 return 0;
00224
00230 int
00231 experiment_open_json (Experiment * experiment,
00232
                            JsonNode * node,
00233
                            unsigned int ninputs)
00235 {
00236
        char buffer[64];
       JsonObject *object;
00237
00238
        const char *name;
00239
       int error_code;
00240
       unsigned int i;
00241
00242 #if DEBUG_EXPERIMENT
00243
       fprintf (stderr, "experiment_open_json: start\n");
00244 #endif
00245
00246
        // Resetting experiment data
00247
       experiment_new (experiment);
00248
00249
       // Getting JSON object
00250
       object = json_node_get_object (node);
00251
00252
        // Reading the experimental data
00253
        name = json_object_get_string_member (object, LABEL_NAME);
00254
        if (!name)
00255
         {
            experiment_error (experiment, _("no data file name"));
00257
            goto exit_on_error;
00258
00259
        experiment->name = g_strdup (name);
00260 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_json: name=%s\n", experiment->name);
00261
00262 #endif
00263
       experiment->weight
00264
           = jb_json_object_get_float_with_default (object, LABEL_WEIGHT, &error_code,
00265
00266
       if (!error_code)
00267
        {
00268
           experiment error (experiment, ("bad weight"));
00269
            goto exit_on_error;
00270
00271 #if DEBUG_EXPERIMENT
00272
       fprintf (stderr, "experiment_open_json: weight=%lg\n", experiment->weight);
00273 #endif
00274
       name = json_object_get_string_member (object, stencil[0]);
00275
        if (name)
00276
00277 #if DEBUG_EXPERIMENT
           fprintf (stderr, "experiment_open_json: experiment=%s template1=%s\n",
00278
                     name, stencil[0]);
00279
00280 #endif
00281
            ++experiment->ninputs;
00282
00283
       else
        {
00284
00285
           experiment_error (experiment, _("no template"));
00286
           goto exit_on_error;
00287
00288
        experiment->stencil[0] = g_strdup (name);
       for (i = 1; i < MAX_NINPUTS; ++i)
00289
00290
00291 #if DEBUG EXPERIMENT
            fprintf (stderr, "experiment_open_json: stencil%u\n", i + 1);
00292
00293 #endif
00294
               (json_object_get_member (object, stencil[i]))
00295
00296
                if (ninputs && ninputs <= i)</pre>
00297
00298
                    experiment_error (experiment, _("bad templates number"));
00299
                    goto exit_on_error;
00300
00301
                name = json_object_get_string_member (object, stencil[i]);
00302 #if DEBUG_EXPERIMENT
00303
                fprintf (stderr,
                          "experiment_open_json: experiment=%s stencil%u=%s\n",
00304
00305
                         experiment->nexperiments, name, stencil[i]);
```

```
00306 #endif
                 experiment->stencil[i] = g_strdup (name);
00308
                 ++experiment->ninputs;
00309
00310
            else if (ninputs && ninputs > i)
00311
                snprintf (buffer, 64, "%s%u", _("no template"), i + 1);
experiment_error (experiment, buffer);
00312
00313
00314
                 goto exit_on_error;
00315
00316
            else
00317
              break:
00318
          }
00319
00320 #if DEBUG_EXPERIMENT
00321
        fprintf (stderr, "experiment_open_json: end\n");
00322 #endif
00323
        return 1;
00324
00325 exit_on_error:
00326
        experiment_free (experiment, INPUT_TYPE_JSON);
00327 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_json: end\n");
00328
00329 #endif
00330
        return 0;
00331 }
```

5.5 experiment.h File Reference

Header file to define the experiment data.

This graph shows which files directly or indirectly include this file:



Data Structures

struct Experiment

Struct to define the experiment data.

Functions

- void experiment_free (Experiment *experiment, unsigned int type)
- void experiment error (Experiment *experiment, char *message)
- int experiment_open_xml (Experiment *experiment, xmlNode *node, unsigned int ninputs)
- int experiment open json (Experiment *experiment, JsonNode *node, unsigned int ninputs)

Variables

const char * stencil [MAX_NINPUTS]

Array of xmlChar strings with stencil labels.

5.5.1 Detailed Description

Header file to define the experiment data.

Authors

Javier Burguete.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file experiment.h.

5.5.2 Function Documentation

5.5.2.1 experiment_error()

Function to print a message error opening an Experiment struct.

Parameters

experiment	Experiment struct.
message	Error message.

Definition at line 112 of file experiment.c.

5.5.2.2 experiment_free()

Function to free the memory of an Experiment struct.

Parameters

experiment	Experiment struct.
type	Type of input file.

Definition at line 83 of file experiment.c.

```
00085
00086
        unsigned int i;
00087 #if DEBUG_EXPERIMENT
88000
       fprintf (stderr, "experiment_free: start\n");
00089 #endif
       if (type == INPUT_TYPE_XML)
00090
00091
            for (i = 0; i < experiment->ninputs; ++i)
00093
              xmlFree (experiment->stencil[i]);
00094
            xmlFree (experiment->name);
00095
00096
        else
00097
        {
00098
            for (i = 0; i < experiment->ninputs; ++i)
00099
             g_free (experiment->stencil[i]);
00100
            g_free (experiment->name);
00101
00102
       experiment->ninputs = 0;
00103 #if DEBUG_EXPERIMENT
00104 fprintf (stderr, "experiment_free: end\n");
00105 #endif
00106 }
```

5.5.2.3 experiment_open_json()

Function to open the Experiment struct on a XML node.

Returns

1 on success, 0 on error.

Parameters

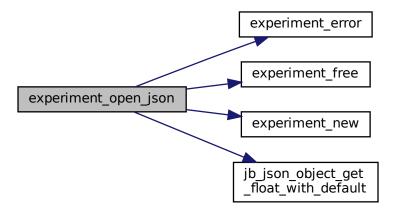
experiment	Experiment struct.
node	JSON node.
ninputs	Number of the simulator input files.

Definition at line 231 of file experiment.c.

```
00235 {
00236
        char buffer[64];
00237
        JsonObject *object;
00238
        const char *name:
00239
        int error_code;
00240
        unsigned int i;
00241
00242 #if DEBUG_EXPERIMENT
00243 fprintf (stderr, "experiment_open_json: start\n");
00244 #endif
00245
00246
        // Resetting experiment data
00247
       experiment_new (experiment);
```

```
00248
00249
        // Getting JSON object
00250
       object = json_node_get_object (node);
00251
00252
       // Reading the experimental data
       name = json_object_get_string_member (object, LABEL_NAME);
00253
00254
       if (!name)
00255
00256
           experiment_error (experiment, _("no data file name"));
00257
           goto exit_on_error;
         }
00258
       experiment->name = g_strdup (name);
00259
00260 #if DEBUG_EXPERIMENT
00261
       fprintf (stderr, "experiment_open_json: name=%s\n", experiment->name);
00262 #endif
00263
       experiment->weight
          = jb_json_object_get_float_with_default (object, LABEL_WEIGHT, &error_code,
00264
00265
00266
       if (!error_code)
       {
00267
00268
           experiment_error (experiment, _("bad weight"));
00269
           goto exit_on_error;
00270
.
00271 #if DEBUG_EXPERIMENT
00272
       fprintf (stderr, "experiment_open_json: weight=%lq\n", experiment->weight);
00273 #endif
00274   name = json_object_get_string_member (object, stencil[0]);
00275 if (name)
00276
00277 #if DEBUG_EXPERIMENT
fprintf (stderr, "experiment_open_json: experiment=%s template1=%s\n",
00279
                    name, stencil[0]);
00280 #endif
00281
           ++experiment->ninputs;
00282
00283
       else
00284
       {
           experiment_error (experiment, _("no template"));
00286
           goto exit_on_error;
00287
00288
       experiment->stencil[0] = g_strdup (name);
       for (i = 1; i < MAX_NINPUTS; ++i)
00289
00290
00291 #if DEBUG_EXPERIMENT
00292
           fprintf (stderr, "experiment_open_json: stencil%u\n", i + 1);
00293 #endif
00294
         if (json_object_get_member (object, stencil[i]))
00295
               if (ninputs && ninputs <= i)</pre>
00296
00297
                 {
00298
                   experiment_error (experiment, _("bad templates number"));
00299
                   goto exit_on_error;
00300
00301
               name = json_object_get_string_member (object, stencil[i]);
00302 #if DEBUG_EXPERIMENT
            fprintf (stderr,
00303
00304
                         "experiment_open_json: experiment=%s stencil%u=%s\n",
00305
                        experiment->nexperiments, name, stencil[i]);
00306 #endif
00307
               experiment->stencil[i] = g_strdup (name);
00308
               ++experiment->ninputs;
00309
             }
00310
           else if (ninputs && ninputs > i)
00311
            {
               snprintf (buffer, 64, "%s%u", _("no template"), i + 1);
00312
00313
                experiment_error (experiment, buffer);
00314
               goto exit_on_error;
00315
             }
00316
           else
00317
             break;
00318
        }
00319
00320 #if DEBUG_EXPERIMENT 00321 fprintf (stderr, "experiment_open_json: end\n");
00322 #endif
00323
      return 1;
00324
00325 exit_on_error:
00326
       experiment_free (experiment, INPUT_TYPE_JSON);
00327 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_json: end\n");
00328
00329 #endif
00330 return 0;
00331 }
```

Here is the call graph for this function:



5.5.2.4 experiment_open_xml()

Function to open the Experiment struct on a XML node.

Returns

1 on success, 0 on error.

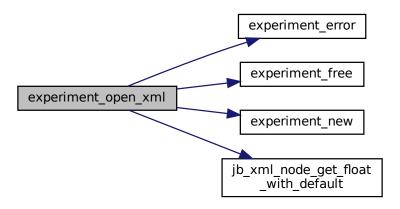
Parameters

experiment	Experiment struct.
node	XML node.
ninputs	Number of the simulator input files.

Definition at line 128 of file experiment.c.

```
// Reading the experimental data
00145
        experiment->name = (char *) xmlGetProp (node, (const xmlChar *) LABEL_NAME);
00146
           (!experiment->name)
00147
        {
00148
            experiment_error (experiment, _("no data file name"));
00149
            goto exit on error:
00150
00151 #if DEBUG_EXPERIMENT
00152
       fprintf (stderr, "experiment_open_xml: name=%s\n", experiment->name);
00153 #endif
00154
       experiment->weight
         = jb_xml_node_get_float_with_default (node, (const xmlChar *) LABEL_WEIGHT,
00155
00156
                                                 &error code, 1.);
00157
       if (!error_code)
00158
        {
00159
            experiment_error (experiment, _("bad weight"));
00160
            goto exit_on_error;
00161
00162 #if DEBUG_EXPERIMENT
00163
       fprintf (stderr, "experiment_open_xml: weight=%lg\n", experiment->weight);
00164 #endif
00165 experiment->stencil[0]
00166
         = (char *) xmlGetProp (node, (const xmlChar *) stencil[0]);
00167
       if (experiment->stencil[0])
00168
00169 #if DEBUG_EXPERIMENT
00170
            fprintf (stderr, "experiment_open_xml: experiment=%s stencil1=%s\n",
00171
                     experiment->name, stencil[0]);
00172 #endif
00173
           ++experiment->ninputs;
         }
00174
00175
       else
00176
00177
            experiment_error (experiment, _("no template"));
00178
           goto exit_on_error;
         }
00179
00180
       for (i = 1; i < MAX NINPUTS; ++i)</pre>
00181
00182 #if DEBUG_EXPERIMENT
00183
            fprintf (stderr, "experiment_open_xml: stencil%u\n", i + 1);
00184 #endif
00185
            if (xmlHasProp (node, (const xmlChar *) stencil[i]))
00186
00187
                if (ninputs && ninputs <= i)</pre>
00188
                 {
00189
                    experiment_error (experiment, _("bad templates number"));
00190
                    goto exit_on_error;
00191
00192
               experiment->stencil[i]
00193
                  = (char *) xmlGetProp (node, (const xmlChar *) stencil[i]);
00194 #if DEBUG_EXPERIMENT
00195
               fprintf (stderr,
00196
                          "experiment_open_xml: experiment=%s stencil%u=%sn",
00197
                         experiment->nexperiments, experiment->name,
00198
                         experiment->stencil[i]);
00199 #endif
                ++experiment->ninputs;
00201
00202
            else if (ninputs && ninputs > i)
00203
             {
               snprintf (buffer, 64, "%s%u", _("no template"), i + 1);
experiment_error (experiment, buffer);
00204
00205
00206
                goto exit_on_error;
00207
00208
           else
00209
              break;
00210
         }
00211
00212 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_xml: end\n");
00214 #endif
00215
       return 1;
00216
00217 exit on error:
       experiment_free (experiment, INPUT_TYPE_XML);
00218
00219 #if DEBUG_EXPERIMENT
00220
       fprintf (stderr, "experiment_open_xml: end\n");
00221 #endif
00222
       return 0;
00223 }
```

Here is the call graph for this function:



5.5.3 Variable Documentation

5.5.3.1 stencil

```
const char* stencil[MAX_NINPUTS] [extern]
```

Array of xmlChar strings with stencil labels.

Definition at line 53 of file experiment.c.

5.6 experiment.h

Go to the documentation of this file.

```
00001 /
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00005
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00009
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are met:
00012
00013 1. Redistributions of source code must retain the above copyright notice,
{\tt 00014} this list of conditions and the following disclaimer.
00015
           Redistributions in binary form must reproduce the above copyright notice,
00017 this list of conditions and the following disclaimer in the
00018 documentation and/or other materials provided with the distribution.
00019
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED 00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF 00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
```

```
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING 00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00031
00038 #ifndef EXPERIMENT__H
00039 #define EXPERIMENT__H 1
00040
00045 typedef struct
00046 {
00047
         char *name;
00048
          char *stencil[MAX_NINPUTS];
00049
         double weight;
00050
         unsigned int ninputs;
00051 } Experiment;
00052
00053 extern const char *stencil[MAX_NINPUTS];
00054
00055 // Public functions
00056 void experiment_free (Experiment * experiment, unsigned int type);
00057 void experiment_error (Experiment * experiment, char *message);
00058 int experiment_open_xml (Experiment * experiment, xmlNode * node,
                                      unsigned int ninputs);
00060 int experiment_open_json (Experiment * experiment, JsonNode * node,
00061
                                        unsigned int ninputs);
00062
00063 #endif
```

5.7 input.c File Reference

Source file to define the input functions.

```
#include "config.h"
#include <stdio.h>
#include <string.h>
#include <libxml/parser.h>
#include <libintl.h>
#include <glib.h>
#include <glib/gstdio.h>
#include <json-glib/json-glib.h>
#include "jb/src/jb_xml.h"
#include "jb/src/jb_json.h"
#include "jb/src/jb_win.h"
#include "tools.h"
#include "experiment.h"
#include "variable.h"
#include "input.h"
```



Macros

• #define DEBUG INPUT 0

Include dependency graph for input.c:

Macro to debug input functions.

Functions

- void input_new ()
- void input_free ()
- static void input_error (char *message)
- static int input_open_xml (xmlDoc *doc)
- static int input_open_json (JsonParser *parser)
- int input_open (char *filename)

Variables

• Input input [1]

Global Input struct to set the input data.

• const char * result name = "result"

Name of the result file.

• const char * variables name = "variables"

Name of the variables file.

5.7.1 Detailed Description

Source file to define the input functions.

Authors

Javier Burguete and Borja Latorre.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file input.c.

5.7.2 Macro Definition Documentation

5.7.2.1 DEBUG_INPUT

```
#define DEBUG_INPUT 0
```

Macro to debug input functions.

Definition at line 55 of file input.c.

5.7.3 Function Documentation

5.7.3.1 input_error()

Function to print an error message opening an Input struct.

Parameters

```
message Error message.
```

Definition at line 122 of file input.c.

5.7.3.2 input_free()

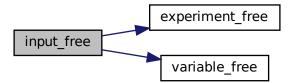
```
void input_free ( )
```

Function to free the memory of the input file data.

Definition at line 84 of file input.c.

```
00085
00086
        unsigned int i;
00087 #if DEBUG_INPUT
       fprintf (stderr, "input_free: start\n");
00089 #endi
00090 g_free (input->name);
00091
        g_free (input->directory);
       for (i = 0; i < input->nexperiments; ++i)
  experiment_free (input->experiment + i, input->type);
00092
00093
00094
        for (i = 0; i < input->nvariables; ++i)
00095
         variable_free (input->variable + i, input->type);
00096
        g_free (input->experiment);
        g_free (input->variable);
00097
        if (input->type == INPUT_TYPE_XML)
00098
00099
         {
00100
            xmlFree (input->evaluator);
00101
            xmlFree (input->simulator);
00102
            xmlFree (input->result);
00103
            xmlFree (input->variables);
00104
00105
        else
00106
         {
00107
           g_free (input->evaluator);
00108
            g_free (input->simulator);
00109
            g_free (input->result);
00110
            g_free (input->variables);
00111
00112
        input->nexperiments = input->nvariables = input->nsteps = 0;
00113 #if DEBUG_INPU
00114
       fprintf (stderr, "input_free: end\n");
00115 #endif
00116 }
```

Here is the call graph for this function:



5.7.3.3 input_new()

```
void input_new ( )
```

Function to create a new Input struct.

Definition at line 66 of file input.c.

```
00067 {
00068 #if DEBUG_INPUT
00069 fprintf (stderr, "input_new: start\n");
00070 #endif
00071 input->nvariables = input->nexperiments = input->nsteps = 0;
00072 input->simulator = input->evaluator = input->directory = input->name = NULL;
00073 input->experiment = NULL;
00074 input->variable = NULL;
00075 #if DEBUG_INPUT
00076 fprintf (stderr, "input_new: end\n");
00077 #endif
00078 }
```

5.7.3.4 input open()

Function to open the input file.

Returns

1_on_success, 0_on_error.

Parameters

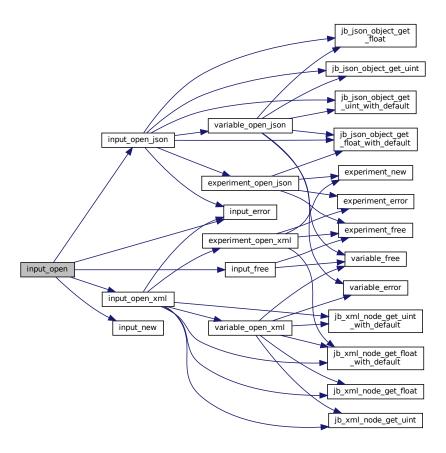
filename Input data file name.

Definition at line 961 of file input.c.

```
00962 {
00963
        xmlDoc *doc;
00964
        JsonParser *parser;
00965
00966 #if DEBUG_INPUT
        fprintf (stderr, "input_open: start\n");
00967
00968 #endif
00969
00970
        // Resetting input data
00971
        input_new ();
00972
        // Opening input file
00973
00974 #if DEBUG_INPUT
00975 fprintf (stderr, "input_open: opening the input file %s\n", filename);
00976 fprintf (stderr, "input_open: trying XML format\n");
00977 #endif
00978 doc = xmlParseFile (filename);
00979
        if (!doc)
00980
00981 #if DEBUG_INPUT
            fprintf (stderr, "input_open: trying JSON format\n");
00982
00983 #endif
00984
            parser = json_parser_new ();
00985
             if (!json_parser_load_from_file (parser, filename, NULL))
00986
                 input_error (_("Unable to parse the input file"));
00987
00988
                 goto exit_on_error;
00989
00990
             if (!input_open_json (parser))
```

```
goto exit_on_error;
00992
         else if (!input_open_xml (doc))
00993
00994
          goto exit_on_error;
00995
        // Getting the working directory
input->directory = g_path_get_dirname (filename);
00996
00998
         input->name = g_path_get_basename (filename);
00999
01000 #if DEBUG_INPUT
        fprintf (stderr, "input_open: end\n");
01001
01002 #endif
01003
         return 1;
01004
01005 exit_on_error:
01006 jbw_show_error (error_message);
01007 g free (error message);
01007    g_free (error_message);
01008    input_free ();
01009 #if DEBUG_INPUT
        fprintf (stderr, "input_open: end\n");
01011 #endif
01012
        return 0;
01013 }
```

Here is the call graph for this function:



5.7.3.5 input_open_json()

Function to open the input file in JSON format.

Returns

1_on_success, 0_on_error.

Parameters

00642

```
parser | JsonParser struct.
```

Definition at line 570 of file input.c.

```
00572
        JsonNode *node, *child;
00573
        JsonObject *object;
        JsonArray *array;
const char *buffer;
00574
00575
00576
       int error code;
       unsigned int i, n;
00578
00579 #if DEBUG_INPUT
00580
       fprintf (stderr, "input_open_json: start\n");
00581 #endif
00582
00583
        // Resetting input data
00584
       input->type = INPUT_TYPE_JSON;
00585
       // Getting the root node
00586
00587 #if DEBUG_INPUT
       fprintf (stderr, "input_open_json: getting the root node\n");
00588
00589 #endif
00590 node = json_parser_get_root (parser);
00591
        object = json_node_get_object (node);
00592
        \ensuremath{//} Getting result and variables file names
00593
00594
        if (!input->result)
00595
        {
00596
            buffer = json_object_get_string_member (object, LABEL_RESULT_FILE);
00597
            if (!buffer)
00598
              buffer = result_name;
00599
            input->result = g_strdup (buffer);
00600
00601
        else
00602
          input->result = g_strdup (result_name);
00603
        if (!input->variables)
00604
            buffer = json_object_get_string_member (object, LABEL_VARIABLES_FILE);
00605
            if (!buffer)
buffer = variables_name;
00606
00607
00608
            input->variables = g_strdup (buffer);
00609
00610
00611
          input->variables = g_strdup (variables_name);
00612
00613
        // Opening simulator program name
00614
        buffer = json_object_get_string_member (object, LABEL_SIMULATOR);
00615
        if (!buffer)
00616
            input_error (_("Bad simulator program"));
00617
00618
            goto exit_on_error;
00619
00620
        input->simulator = g_strdup (buffer);
00621
00622
        // Opening evaluator program name
00623
        buffer = json_object_get_string_member (object, LABEL_EVALUATOR);
        if (buffer)
00624
00625
          input->evaluator = g_strdup (buffer);
00626
00627
        // Obtaining pseudo-random numbers generator seed
00628
00629
          = jb_json_object_get_uint_with_default (object, LABEL_SEED,
00630
                                                      &error_code, DEFAULT_RANDOM_SEED);
00631
        if (!error_code)
00632
          {
00633
            input_error (_("Bad pseudo-random numbers generator seed"));
00634
            goto exit_on_error;
00635
00636
00637
        \label{eq:continuous} \ensuremath{\text{// Opening algorithm}}
        buffer = json_object_get_string_member (object, LABEL_ALGORITHM);
if (!strcmp (buffer, LABEL_MONTE_CARLO))
00638
00639
00640
00641
             input->algorithm = ALGORITHM_MONTE_CARLO;
```

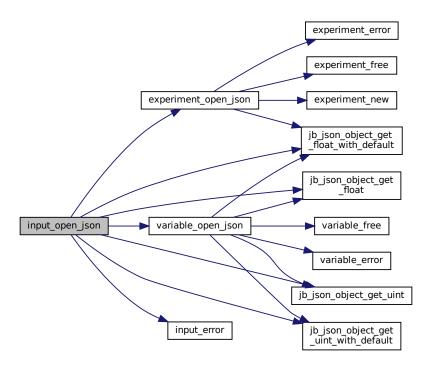
```
// Obtaining simulations number
00644
            input->nsimulations
00645
              = jb_json_object_get_uint (object, LABEL_NSIMULATIONS, &error_code);
00646
            if (!error_code)
00647
              {
                input_error (_("Bad simulations number"));
00648
00649
                goto exit_on_error;
00650
00651
00652
        else if (!strcmp (buffer, LABEL_SWEEP))
          input->algorithm = ALGORITHM_SWEEP;
00653
        else if (!strcmp (buffer, LABEL_ORTHOGONAL))
00654
          input->algorithm = ALGORITHM_ORTHOGONAL;
00655
00656
        else if (!strcmp (buffer, LABEL_GENETIC))
00657
00658
            input->algorithm = ALGORITHM_GENETIC;
00659
00660
            // Obtaining population
00661
            if (json_object_get_member (object, LABEL_NPOPULATION))
00662
              {
00663
                input->nsimulations
00664
                  = jb_json_object_get_uint (object, LABEL_NPOPULATION, &error_code);
00665
                if (!error_code || input->nsimulations < 3)</pre>
00666
00667
                    input_error (_("Invalid population number"));
00668
                    goto exit_on_error;
00669
00670
00671
            else
00672
                input_error (_("No population number"));
00673
00674
                goto exit_on_error;
00675
00676
00677
            // Obtaining generations
            if (json_object_get_member (object, LABEL_NGENERATIONS))
00678
00679
              {
00680
                input->niterations
00681
                  = jb_json_object_get_uint_with_default (object, LABEL_NGENERATIONS,
00682
                                                            &error_code, 1);
00683
                if (!error_code || !input->niterations)
00684
                  {
                    input_error (_("Invalid generations number"));
00685
00686
                    goto exit_on_error;
00687
00688
00689
            else
00690
              {
                input_error (_("No generations number"));
00691
00692
                goto exit_on_error;
00693
00694
00695
            // Obtaining mutation probability
00696
            if (json_object_get_member (object, LABEL_MUTATION))
00697
00698
                input->mutation ratio
                    jb_json_object_get_float (object, LABEL_MUTATION, &error_code);
00699
00700
                if (!error_code || input->mutation_ratio < 0.</pre>
00701
                    || input->mutation_ratio >= 1.)
00702
                    input_error (_("Invalid mutation probability"));
00703
00704
                    goto exit_on_error;
00705
                  }
00706
            else
00707
00708
             {
00709
                input_error (_("No mutation probability"));
00710
                goto exit_on_error;
00711
00712
00713
            // Obtaining reproduction probability
00714
            if (json_object_get_member (object, LABEL_REPRODUCTION))
00715
00716
                input->reproduction ratio
00717
                  = jb_json_object_get_float (object, LABEL_REPRODUCTION,
00718
                                               &error_code);
00719
                if (!error_code || input->reproduction_ratio < 0.</pre>
00720
                    || input->reproduction_ratio >= 1.0)
00721
00722
                    input error ( ("Invalid reproduction probability"));
00723
                    goto exit_on_error;
00724
                  }
00725
00726
            else
00727
                input_error (_("No reproduction probability"));
00728
00729
                goto exit on error;
```

```
00730
              }
00731
00732
             // Obtaining adaptation probability
            if (json_object_get_member (object, LABEL_ADAPTATION))
00733
00734
00735
                 input->adaptation ratio
00736
                     jb_json_object_get_float (object, LABEL_ADAPTATION, &error_code);
00737
                 if (!error_code || input->adaptation_ratio < 0.</pre>
00738
                     || input->adaptation_ratio >= 1.)
00739
00740
                     input_error (_("Invalid adaptation probability"));
00741
                     goto exit_on_error;
00742
00743
00744
            else
00745
                input_error (_("No adaptation probability"));
00746
00747
                goto exit_on_error;
00749
00750
             // Checking survivals
00751
            i = input->mutation_ratio * input->nsimulations;
            i += input->reproduction_ratio * input->nsimulations;
i += input->adaptation_ratio * input->nsimulations;
00752
00753
00754
             if (i > input->nsimulations - 2)
00755
              {
00756
                 input_error
00757
                   (_("No enough survival entities to reproduce the population"));
00758
                goto exit_on_error;
00759
               }
00760
          }
00761
        else
00762
00763
             input_error (_("Unknown algorithm"));
00764
            goto exit_on_error;
00765
00766
00767
        if (input->algorithm == ALGORITHM_MONTE_CARLO
00768
             || input->algorithm == ALGORITHM_SWEEP
00769
             || input->algorithm == ALGORITHM_ORTHOGONAL)
00770
00771
00772
             // Obtaining iterations number
00773
             input->niterations
00774
                jb_json_object_get_uint (object, LABEL_NITERATIONS, &error_code);
00775
             if (!error_code || !input->niterations)
00776
              {
00777
                input_error (_("Bad iterations number"));
00778
                goto exit_on_error;
00779
00780
00781
             // Obtaining best number
             input->nbest
00782
00783
               = jb_json_object_get_uint_with_default (object, LABEL_NBEST,
00784
                                                         &error_code, 1);
00785
            if (!error code || !input->nbest)
00786
00787
                 input_error (_("Invalid best number"));
00788
                 goto exit_on_error;
00789
00790
00791
             // Obtaining tolerance
00792
             input->tolerance
00793
               = jb_json_object_get_float_with_default (object, LABEL_TOLERANCE,
00794
                                                           &error_code, 0.);
00795
             if (!error_code || input->tolerance < 0.)</pre>
00796
00797
                 input_error (_("Invalid tolerance"));
00798
                 goto exit on error;
00799
00800
00801
             // Getting hill climbing method parameters
00802
             if (json_object_get_member (object, LABEL_NSTEPS))
00803
00804
                 input->nsteps
                   = jb_json_object_get_uint (object, LABEL_NSTEPS, &error_code);
00805
00806
                 if (!error_code)
00807
                     input_error (_("Invalid steps number"));
00808
00809
                     goto exit_on_error;
00810
00811
                 buffer = json_object_get_string_member (object, LABEL_CLIMBING);
                 if (!strcmp (buffer, LABEL_COORDINATES))
  input->climbing = CLIMBING_METHOD_COORDINATES;
00812
00813
00814
                 else if (!strcmp (buffer, LABEL_RANDOM))
00815
                   {
00816
                     input->climbing = CLIMBING_METHOD_RANDOM;
```

```
input->nestimates
00817
00818
                      = jb_json_object_get_uint (object, LABEL_NESTIMATES,
00819
                                                  &error_code);
                    if (!error_code || !input->nestimates)
00820
00821
00822
                        input_error (_("Invalid estimates number"));
                        goto exit_on_error;
00824
00825
00826
                else
00827
                 {
00828
                    input error ( ("Unknown method to estimate the hill climbing"));
00829
                    goto exit on error;
00830
00831
                input->relaxation
00832
                  = jb_json_object_get_float_with_default (object, LABEL_RELAXATION,
00833
                                                            &error_code,
                                                           DEFAULT_RELAXATION);
00834
00835
                if (!error_code || input->relaxation < 0. || input->relaxation > 2.)
00836
                 {
00837
                    input_error (_("Invalid relaxation parameter"));
00838
                    goto exit_on_error;
                  }
00839
00840
00841
            else
00842
             input->nsteps = 0;
00843
00844
        // Obtaining the threshold
00845
        input->threshold
          = jb_json_object_get_float_with_default (object, LABEL_THRESHOLD,
00846
00847
                                                    &error code, 0.);
00848
00849
        if (!error_code)
00850
        {
00851
            input_error (_("Invalid threshold"));
00852
            goto exit_on_error;
00853
00855
        // Reading the experimental data
00856
        array = json_object_get_array_member (object, LABEL_EXPERIMENTS);
00857
        n = json_array_get_length (array);
00858
        input->experiment = (Experiment \star) g_malloc (n \star sizeof (Experiment));
        for (i = 0; i < n; ++i)
00859
00860
00861 #if DEBUG_INPUT
00862
            fprintf (stderr, "input_open_json: nexperiments=u\n",
00863
                     input->nexperiments);
00864 #endif
            child = json_array_get_element (array, i);
00865
            if (!input->nexperiments)
00866
00867
             {
00868
                if (!experiment_open_json (input->experiment, child, 0))
00869
                  goto exit_on_error;
00870
00871
            else
00872
            {
               if (!experiment_open_json (input->experiment + input->nexperiments,
00874
                                           child, input->experiment->ninputs))
00875
                 goto exit_on_error;
00876
            ++input->nexperiments;
00877
00878 #if DEBUG_INPUT
00879
           fprintf (stderr, "input_open_json: nexperiments=%u\n",
00880
                    input->nexperiments);
00881 #endif
00882
00883
        if (!input->nexperiments)
        {
00884
00885
           input_error (_("No optimization experiments"));
00886
            goto exit_on_error;
00887
00888
00889
        // Reading the variables data
        array = json_object_get_array_member (object, LABEL_VARIABLES);
00890
00891
        n = json_array_get_length (array);
00892
        input->variable = (Variable *) g_malloc (n * sizeof (Variable));
00893
        for (i = 0; i < n; ++i)
00894
00895 #if DEBUG_INPUT
00896
           fprintf (stderr, "input_open_json: nvariables=%u\n", input->nvariables);
00897 #endif
00898
            child = json_array_get_element (array, i);
00899
            if (!variable_open_json (input->variable + input->nvariables, child,
00900
                                     input->algorithm, input->nsteps))
00901
              goto exit_on_error;
00902
            ++input->nvariables;
00903
```

```
00904
        if (!input->nvariables)
00905
            input_error (_("No optimization variables"));
00906
00907
            goto exit_on_error;
00908
00909
00910
        // Obtaining the error norm
00911
        if (json_object_get_member (object, LABEL_NORM))
00912
            buffer = json_object_get_string_member (object, LABEL_NORM);
00913
            if (!strcmp (buffer, LABEL_EUCLIDIAN))
  input->norm = ERROR_NORM_EUCLIDIAN;
00914
00915
            else if (!strcmp (buffer, LABEL_MAXIMUM))
00916
00917
              input->norm = ERROR_NORM_MAXIMUM;
00918
            else if (!strcmp (buffer, LABEL_P))
00919
                input->norm = ERROR_NORM_P;
00920
                input >n = jb_json_object_get_float (object, LABEL_P, &error_code);
if (!error_code)
00921
00923
                  {
00924
                   input_error (_("Bad P parameter"));
00925
                    goto exit_on_error;
00926
                  }
00927
00928
            else if (!strcmp (buffer, LABEL_TAXICAB))
00929
              input->norm = ERROR_NORM_TAXICAB;
00930
00931
                input_error (_("Unknown error norm"));
00932
00933
                goto exit_on_error;
00934
00935
00936
00937
          input->norm = ERROR_NORM_EUCLIDIAN;
00938
        // Closing the JSON document
00939
00940
       g_object_unref (parser);
00942 #if DEBUG_INPUT
00943 fprintf (stderr, "input_open_json: end\n");
00944 #endif
00945
       return 1;
00946
00947 exit_on_error:
00948 g_object_unref (parser);
00949 #if DEBUG_INPUT
00950 fprintf (stderr, "input_open_json: end\n");
00951 #endif
00952
       return 0;
00953 }
```

Here is the call graph for this function:



5.7.3.6 input_open_xml()

Function to open the input file in XML format.

Returns

1_on_success, 0_on_error.

Parameters

```
doc xmlDoc struct.
```

Definition at line 133 of file input.c.

```
00134 {
00135     char buffer2[64];
00136     xmlNode *node, *child;
00137     xmlChar *buffer;
00138     int error_code;
00139     unsigned int i;
00140
00141 #if DEBUG_INPUT
00142     fprintf (stderr, "input_open_xml: start\n");
```

```
00143 #endif
00144
00145
        // Resetting input data
       buffer = NULL;
00146
       input->type = INPUT_TYPE_XML;
00147
00148
        // Getting the root node
00150 #if DEBUG_INPUT
00151
       fprintf (stderr, "input_open_xml: getting the root node\n");
00152 #endif
00153
       node = xmlDocGetRootElement (doc);
       if (xmlStrcmp (node->name, (const xmlChar *) LABEL_OPTIMIZE))
00154
00155
         {
00156
            input_error (_("Bad root XML node"));
00157
           goto exit_on_error;
00158
00159
00160
       // Getting result and variables file names
       if (!input->result)
00161
00162
        {
00163
            input->result =
00164
              (char *) xmlGetProp (node, (const xmlChar *) LABEL_RESULT_FILE);
            if (!input->result)
00165
00166
             input->result = (char *) xmlStrdup ((const xmlChar *) result name);
00167
00168 #if DEBUG_INPUT
00169
       fprintf (stderr, "input_open_xml: result file=%s\n", input->result);
00170 #endif
00171
       if (!input->variables)
00172
         {
00173
           input->variables =
00174
              (char *) xmlGetProp (node, (const xmlChar *) LABEL_VARIABLES_FILE);
00175
            if (!input->variables)
             input->variables =
00176
00177
                (char *) xmlStrdup ((const xmlChar *) variables_name);
00178
00179 #if DEBUG_INPUT
00180
       fprintf (stderr, "input_open_xml: variables file=%s\n", input->variables);
00181 #endif
00182
00183
        // Opening simulator program name
00184
       input->simulator =
         (char *) xmlGetProp (node. (const xmlChar *) LABEL SIMULATOR):
00185
00186
        if (!input->simulator)
00187
00188
           input_error (_("Bad simulator program"));
00189
           goto exit_on_error;
00190
00191
00192
       // Opening evaluator program name
00193
       input->evaluator
00194
          (char *) xmlGetProp (node, (const xmlChar *) LABEL_EVALUATOR);
00195
00196
        // Obtaining pseudo-random numbers generator seed
00197
       input->seed
00198
         = jb xml node get uint with default (node, (const xmlChar *) LABEL SEED,
                                               &error_code, DEFAULT_RANDOM_SEED);
00199
00200
        if (!error code)
00201
00202
           input_error (_("Bad pseudo-random numbers generator seed"));
00203
           goto exit_on_error;
00204
00205
00206
        // Opening algorithm
00207
       buffer = xmlGetProp (node, (const xmlChar *) LABEL_ALGORITHM);
00208
       if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_MONTE_CARLO))
00209
00210
            input->algorithm = ALGORITHM MONTE CARLO;
00211
00212
            // Obtaining simulations number
00213
            input->nsimulations
00214
              = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSIMULATIONS,
00215
                                     &error_code);
00216
            if (!error_code)
00217
             {
00218
               input_error (_("Bad simulations number"));
00219
               goto exit_on_error;
00220
00221
         }
       else if (!xmlStrcmp (buffer. (const xmlChar *) LABEL SWEEP))
00222
         input->algorithm = ALGORITHM_SWEEP;
00223
       else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_ORTHOGONAL))
00224
00225
         input->algorithm = ALGORITHM_ORTHOGONAL;
00226
       else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_GENETIC))
00227
           input->algorithm = ALGORITHM GENETIC;
00228
00229
```

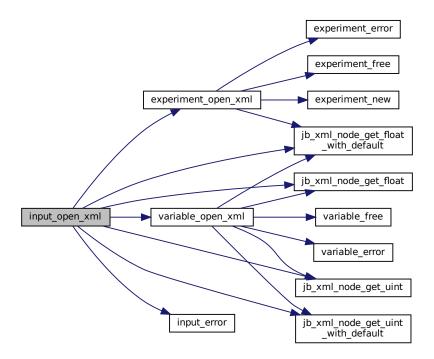
```
00230
            // Obtaining population
00231
            if (xmlHasProp (node, (const xmlChar *) LABEL_NPOPULATION))
00232
00233
                input->nsimulations
00234
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NPOPULATION,
00235
                                           &error code):
00236
                if (!error_code || input->nsimulations < 3)</pre>
00237
00238
                    input_error (_("Invalid population number"));
00239
                    goto exit_on_error;
                  }
00240
00241
              }
00242
            else
00243
             {
00244
                input_error (_("No population number"));
00245
                goto exit_on_error;
00246
00247
00248
            // Obtaining generations
00249
            if (xmlHasProp (node, (const xmlChar *) LABEL_NGENERATIONS))
00250
00251
                input->niterations
00252
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NGENERATIONS,
00253
                                           %error code):
00254
                if (!error_code || !input->niterations)
00255
00256
                    input_error (_("Invalid generations number"));
                    goto exit_on_error;
00257
00258
00259
              }
00260
            else
00261
              {
00262
                input_error (_("No generations number"));
00263
                goto exit_on_error;
00264
00265
00266
            // Obtaining mutation probability
            if (xmlHasProp (node, (const xmlChar *) LABEL_MUTATION))
00267
00268
              {
00269
                input->mutation_ratio
00270
                   = jb_xml_node_get_float (node, (const xmlChar *) LABEL_MUTATION,
00271
                                            &error_code);
                if (!error code || input->mutation ratio < 0.</pre>
00272
00273
                    || input->mutation_ratio >= 1.)
00274
00275
                    input_error (_("Invalid mutation probability"));
00276
                    goto exit_on_error;
00277
00278
              }
00279
            else
00280
              {
00281
                input_error (_("No mutation probability"));
00282
                goto exit_on_error;
00283
              }
00284
00285
            // Obtaining reproduction probability
            if (xmlHasProp (node, (const xmlChar *) LABEL_REPRODUCTION))
00287
              {
00288
                input->reproduction_ratio
00289
                   = jb_xml_node_get_float (node, (const xmlChar *) LABEL_REPRODUCTION,
00290
                                            &error_code);
                if (!error_code || input->reproduction_ratio < 0.</pre>
00291
00292
                    || input->reproduction_ratio >= 1.0)
00293
00294
                    input_error (_("Invalid reproduction probability"));
00295
                    goto exit_on_error;
00296
00297
              }
00298
            else
00299
              {
00300
                input_error (_("No reproduction probability"));
00301
                goto exit_on_error;
              }
00302
00303
00304
            // Obtaining adaptation probability
            if (xmlHasProp (node, (const xmlChar *) LABEL_ADAPTATION))
00305
00306
              {
00307
                input->adaptation_ratio
00308
                   = jb_xml_node_get_float (node, (const xmlChar *) LABEL_ADAPTATION,
00309
                                            &error code);
                if (!error_code || input->adaptation_ratio < 0.</pre>
00310
00311
                    || input->adaptation_ratio >= 1.)
00312
00313
                    input_error (_("Invalid adaptation probability"));
00314
                    goto exit_on_error;
00315
00316
              }
```

```
00317
           else
00318
             {
                input_error (_("No adaptation probability"));
00319
00320
                goto exit_on_error;
00321
00322
             // Checking survivals
00324
             i = input->mutation_ratio * input->nsimulations;
            i += input->reproduction_ratio * input->nsimulations;
i += input->adaptation_ratio * input->nsimulations;
00325
00326
            if (i > input->nsimulations - 2)
00327
00328
              {
00329
                input_error
00330
                  (_("No enough survival entities to reproduce the population"));
00331
                goto exit_on_error;
00332
00333
          1
00334
        else
00335
00336
            input_error (_("Unknown algorithm"));
00337
            goto exit_on_error;
00338
        xmlFree (buffer);
00339
00340
        buffer = NULL:
00341
        if (input->algorithm == ALGORITHM_MONTE_CARLO
00342
00343
             || input->algorithm == ALGORITHM_SWEEP
00344
             || input->algorithm == ALGORITHM_ORTHOGONAL)
00345
00346
00347
             // Obtaining iterations number
00348
            input->niterations = jb_xml_node_get_uint_with_default
00349
              (node, (const xmlChar *) LABEL_NITERATIONS, &error_code, 1);
00350
             if (!error_code || !input->niterations)
00351
                input_error (_("Bad iterations number"));
00352
                goto exit_on_error;
00353
00355
00356
             // Obtaining best number
             input->nbest
00357
00358
              = jb_xml_node_get_uint_with_default (node,
                                                      (const_xmlChar *) LABEL NBEST.
00359
00360
                                                      &error_code, 1);
00361
             if (!error_code || !input->nbest)
00362
00363
                input_error (_("Invalid best number"));
00364
                goto exit_on_error;
00365
00366
00367
             // Obtaining tolerance
00368
             input->tolerance
00369
               = jb_xml_node_get_float_with_default (node,
00370
                                                      (const xmlChar *) LABEL TOLERANCE,
00371
                                                       &error_code, 0.);
00372
            if (!error code || input->tolerance < 0.)</pre>
00374
                 input_error (_("Invalid tolerance"));
00375
                goto exit_on_error;
00376
00377
            // Getting hill climbing method parameters
00378
00379
            if (xmlHasProp (node, (const xmlChar *) LABEL_NSTEPS))
00380
00381
                 input->nsteps =
00382
                   jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSTEPS,
00383
                                          &error_code);
00384
                 if (!error code)
00385
00386
                     input_error (_("Invalid steps number"));
00387
                     goto exit_on_error;
00388
00389 #if DEBUG_INPUT
                 fprintf (stderr, "input_open_xml: nsteps=%u\n", input->nsteps);
00390
00391 #endif
00392
                 buffer = xmlGetProp (node, (const xmlChar *) LABEL_CLIMBING);
                if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_COORDINATES))
  input->climbing = CLIMBING_METHOD_COORDINATES;
00393
00394
00395
                 else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_RANDOM))
00396
                  -{
00397
                     input->climbing = CLIMBING METHOD RANDOM;
00398
                     input->nestimates
00399
                       = jb_xml_node_get_uint (node,
00400
                                                 (const xmlChar *) LABEL_NESTIMATES,
00401
                                                 &error_code);
                     if (!error_code || !input->nestimates)
00402
00403
```

```
00404
                        input_error (_("Invalid estimates number"));
00405
                       goto exit_on_error;
00406
00407
00408
               else
00409
                 {
00410
                   input_error (_("Unknown method to estimate the hill climbing"));
00411
                   goto exit_on_error;
00412
00413
               xmlFree (buffer);
00414
               buffer = NULL:
               input->relaxation
00415
00416
                  = jb_xml_node_get_float_with_default (node,
00417
                                                        (const xmlChar *)
00418
                                                        LABEL_RELAXATION,
00419
                                                        &error_code,
                                                       DEFAULT RELAXATION):
00420
               if (!error_code || input->relaxation < 0. || input->relaxation > 2.)
00421
00422
00423
                    input_error (_("Invalid relaxation parameter"));
00424
                   goto exit_on_error;
00425
00426
             }
00427
            else
00428
             input->nsteps = 0;
00429
00430
        // Obtaining the threshold
00431
       input->threshold =
00432
          jb_xml_node_get_float_with_default (node, (const xmlChar *) LABEL_THRESHOLD,
00433
                                             &error_code, 0.);
00434
       if (!error code)
00435
         {
00436
            input_error (_("Invalid threshold"));
00437
           goto exit_on_error;
00438
00439
00440
       // Reading the experimental data
       for (child = node->children; child; child = child->next)
00441
00442
00443
            if (xmlStrcmp (child->name, (const xmlChar *) LABEL_EXPERIMENT))
00444
             break;
00445 #if DEBUG_INPUT
           fprintf (stderr, "input_open_xml: nexperiments=%u\n",
00446
00447
                    input->nexperiments);
00449
           input->experiment = (Experiment *)
00450
             g_realloc (input->experiment,
00451
                        (1 + input->nexperiments) * sizeof (Experiment));
            if (!input->nexperiments)
00452
00453
00454
               if (!experiment_open_xml (input->experiment, child, 0))
00455
                 goto exit_on_error;
00456
00457
           else
00458
             {
00459
               if (!experiment_open_xml (input->experiment + input->nexperiments,
                                         child, input->experiment->ninputs))
00460
00461
                 goto exit_on_error;
00462
00463 ++input->nexperiments;
00464 #if DEBUG_INPUT
           fprintf (stderr, "input_open_xml: nexperiments=%u\n",
00465
00466
                    input->nexperiments);
00467 #endif
00468
00469
       if (!input->nexperiments)
00470
00471
           input_error (_("No optimization experiments"));
00472
           goto exit_on_error;
00473
00474
       buffer = NULL;
00475
00476
        // Reading the variables data
       if (input->algorithm == ALGORITHM_SWEEP
00477
00478
            || input->algorithm == ALGORITHM_ORTHOGONAL)
00479
          input->nsimulations = 1;
00480
       for (; child; child = child->next)
00481
00482 #if DEBUG_INPUT
           fprintf (stderr, "input_open_xml: nvariables=%u\n", input->nvariables);
00483
00484 #endif
00485
           if (xmlStrcmp (child->name, (const xmlChar *) LABEL_VARIABLE))
00486
               00487
00488
00489
00490
               goto exit_on_error;
```

```
00491
00492
            input->variable = (Variable *)
00493
             g_realloc (input->variable,
                         (1 + input->nvariables) * sizeof (Variable));
00494
            00495
00496
              goto exit_on_error;
00498
            if (input->algorithm == ALGORITHM_SWEEP
              || input->algorithm == ALGORITHM_ORTHOGONAL)
input->nsimulations *= input->variable[input->nvariables].nsweeps;
00499
00500
            ++input->nvariables;
00501
00502
00503
        if (!input->nvariables)
00504
00505
            input_error (_("No optimization variables"));
00506
            goto exit_on_error;
00507
00508
        if (input->nbest > input->nsimulations)
00509
00510
            input_error (_("Best number higher than simulations number"));
00511
           goto exit_on_error;
00512
        buffer = NULL:
00513
00514
00515
        // Obtaining the error norm
00516
        if (xmlHasProp (node, (const xmlChar *) LABEL_NORM))
00517
00518
            buffer = xmlGetProp (node, (const xmlChar *) LABEL_NORM);
            if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_EUCLIDIAN))
  input->norm = ERROR_NORM_EUCLIDIAN;
00519
00520
            else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_MAXIMUM))
00521
00522
              input->norm = ERROR_NORM_MAXIMUM;
00523
            else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_P))
00524
              {
00525
                input->norm = ERROR_NORM_P;
00526
                input->p
00527
                  = jb_xml_node_get_float (node, (const xmlChar *) LABEL_P,
                                            &error_code);
00529
                if (!error_code)
00530
00531
                    input_error (_("Bad P parameter"));
00532
                    goto exit_on_error;
00533
00534
00535
            else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_TAXICAB))
00536
              input->norm = ERROR_NORM_TAXICAB;
00537
            else
00538
             {
                input_error (_("Unknown error norm"));
00539
00540
                goto exit_on_error;
00541
00542
            xmlFree (buffer);
00543
00544
       else
         input->norm = ERROR_NORM_EUCLIDIAN;
00545
00546
00547
       // Closing the XML document
00548
       xmlFreeDoc (doc);
00549
00550 #if DEBUG_INPUT
       fprintf (stderr, "input_open_xml: end\n");
00551
00552 #endif
00553
       return 1;
00554
00555 exit_on_error:
00556 xmlFree (buffer);
00557
       xmlFreeDoc (doc);
00558 #if DEBUG_INPUT
00559 fprintf (stderr, "input_open_xml: end\n");
00560 #endif
00561
      return 0;
00562 }
```

Here is the call graph for this function:



5.7.4 Variable Documentation

5.7.4.1 input

Input input[1]

Global Input struct to set the input data.

Definition at line 57 of file input.c.

5.7.4.2 result_name

const char* result_name = "result"

Name of the result file.

Definition at line 59 of file input.c.

5.7.4.3 variables_name

```
const char* variables_name = "variables"
```

Name of the variables file.

Definition at line 60 of file input.c.

5.8 input.c

Go to the documentation of this file.

```
00001 /*
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are met:
00012
00013 1.
          Redistributions of source code must retain the above copyright notice,
00014 this list of conditions and the following disclaimer.
          Redistributions in binary form must reproduce the above copyright notice,
00017 this list of conditions and the following disclaimer in the
00018 documentation and/or other materials provided with the distribution.
00019
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF 00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR 00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
00031
00038 #define _GNU_SOURCE
00039 #include "config.h"
00040 #include <stdio.h>
00041 #include <string.h>
00042 #include <libxml/parser.h>
00043 #include <libintl.h>
00044 #include <glib.h>
00045 #include <glib/gstdio.h>
00046 #include <json-glib/json-glib.h>
00047 #include "jb/src/jb_xml.h"
00048 #include "jb/src/jb_json.h"
00049 #include "jb/src/jb_win.h"
00050 #include "tools.h"
00051 #include "experiment.h"
00052 #include "variable.h"
00053 #include "input.h'
00054
00055 #define DEBUG_INPUT 0
00056
00057 Input input[1];
00058
00059 const char *result_name = "result";
00060 const char *variables_name = "variables";
00061
00065 void
00066 input_new ()
00067 {
00068 #if DEBUG_INPUT
       fprintf (stderr, "input_new: start\n");
00069
00070 #endif
00071 input->nvariables = input->nexperiments = input->nsteps = 0;
00072
        input->simulator = input->evaluator = input->directory = input->name = NULL;
        input->experiment = NULL;
00073
      input->variable = NULL;
00075 #if DEBUG_INPUT
```

5.8 input.c 123

```
fprintf (stderr, "input_new: end\n");
00077 #endif
00078 }
00079
00083 void
00084 input_free ()
00085 {
00086
        unsigned int i;
00087 #if DEBUG_INPUT
       fprintf (stderr, "input_free: start\n");
00088
00089 #endif
00090
       q_free (input->name);
00091
       g free (input->directory);
00092
       for (i = 0; i < input->nexperiments; ++i)
00093
         experiment_free (input->experiment + i, input->type);
00094
       for (i = 0; i < input->nvariables; ++i)
00095
         variable_free (input->variable + i, input->type);
00096
       g_free (input->experiment);
       g_free (input->variable);
00097
00098
       if (input->type == INPUT_TYPE_XML)
00099
00100
            xmlFree (input->evaluator);
00101
            xmlFree (input->simulator);
            xmlFree (input->result);
00102
00103
           xmlFree (input->variables);
00104
00105
       else
00106
        {
00107
           g_free (input->evaluator);
00108
           g_free (input->simulator);
00109
           g_free (input->result);
00110
           g_free (input->variables);
00111
00112
       input->nexperiments = input->nvariables = input->nsteps = 0;
00113 #if DEBUG_INPUT
       fprintf (stderr, "input_free: end\n");
00114
00115 #endif
00116 }
00117
00121 static void
00122 input_error (char *message)
00123 {
       error_message = g_strconcat (_("Input"), ": ", message, "\n", NULL);
00124
00125 }
00126
00132 static inline int
00133 input_open_xml (xmlDoc * doc)
00134 {
00135
       char buffer2[64];
00136
       xmlNode *node, *child;
       xmlChar *buffer;
00137
00138
       int error_code;
00139
       unsigned int i;
00140
00141 #if DEBUG_INPUT
       fprintf (stderr, "input_open_xml: start\n");
00142
00143 #endif
00144
00145
        // Resetting input data
00146
       buffer = NULL;
       input->type = INPUT_TYPE_XML;
00147
00148
00149
        // Getting the root node
00150 #if DEBUG_INPUT
00151
       fprintf (stderr, "input_open_xml: getting the root node\n");
00152 #endif
00153
       node = xmlDocGetRootElement (doc);
        if (xmlStrcmp (node->name, (const xmlChar *) LABEL_OPTIMIZE))
00154
00155
00156
            input_error (_("Bad root XML node"));
00157
           goto exit_on_error;
00158
00159
       // Getting result and variables file names
00160
00161
       if (!input->result)
00162
00163
00164
             (char *) xmlGetProp (node, (const xmlChar *) LABEL_RESULT_FILE);
00165
            if (!input->result)
00166
             input->result = (char *) xmlStrdup ((const xmlChar *) result name);
00167
00168 #if DEBUG_INPUT
00169
       fprintf (stderr, "input_open_xml: result file=%s\n", input->result);
00170 #endif
00171
       if (!input->variables)
00172
00173
            input->variables =
```

```
(char *) xmlGetProp (node, (const xmlChar *) LABEL_VARIABLES_FILE);
00175
            if (!input->variables)
              input->variables =
00176
00177
                (char *) xmlStrdup ((const xmlChar *) variables_name);
00178
00179 #if DEBUG_INPUT
00180
       fprintf (stderr, "input_open_xml: variables file=%s\n", input->variables);
00181 #endif
00182
00183
        // Opening simulator program name
00184
        input->simulator =
00185
         (char *) xmlGetProp (node, (const xmlChar *) LABEL SIMULATOR);
00186
           (!input->simulator)
00187
00188
            input_error (_("Bad simulator program"));
00189
           goto exit_on_error;
00190
00191
00192
        // Opening evaluator program name
00193
        input->evaluator
00194
          (char *) xmlGetProp (node, (const xmlChar *) LABEL_EVALUATOR);
00195
00196
        // Obtaining pseudo-random numbers generator seed
00197
       input->seed
00198
          = jb_xml_node_get_uint_with_default (node, (const xmlChar *) LABEL_SEED,
                                                &error_code, DEFAULT_RANDOM_SEED);
00199
00200
        if (!error_code)
00201
00202
            input_error (_("Bad pseudo-random numbers generator seed"));
00203
           goto exit_on_error;
00204
00205
00206
        // Opening algorithm
00207
        buffer = xmlGetProp (node, (const xmlChar *) LABEL_ALGORITHM);
00208
        if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_MONTE_CARLO))
00209
00210
            input->algorithm = ALGORITHM MONTE CARLO;
00212
            // Obtaining simulations number
00213
            input->nsimulations
00214
              = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSIMULATIONS,
00215
                                      &error_code);
00216
            if (!error_code)
00217
             {
                input_error (_("Bad simulations number"));
00218
00219
                goto exit_on_error;
00220
00221
          }
00222
       else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL SWEEP))
         input->algorithm = ALGORITHM_SWEEP;
00223
        else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_ORTHOGONAL))
00224
00225
         input->algorithm = ALGORITHM_ORTHOGONAL;
00226
        else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_GENETIC))
00227
            input->algorithm = ALGORITHM_GENETIC;
00228
00229
            // Obtaining population
00231
            if (xmlHasProp (node, (const xmlChar *) LABEL_NPOPULATION))
00232
00233
                input->nsimulations
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NPOPULATION, &error_code);
00234
00235
00236
                if (!error_code || input->nsimulations < 3)</pre>
00237
00238
                    input_error (_("Invalid population number"));
00239
                    goto exit_on_error;
00240
00241
              }
00242
            else
00243
             {
00244
                input_error (_("No population number"));
00245
                goto exit_on_error;
              }
00246
00247
00248
            // Obtaining generations
00249
            if (xmlHasProp (node, (const xmlChar *) LABEL_NGENERATIONS))
00250
              {
00251
                input->niterations
00252
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NGENERATIONS,
00253
                                           &error code);
                if (!error_code || !input->niterations)
00254
00255
                 {
00256
                    input_error (_("Invalid generations number"));
00257
                    goto exit_on_error;
00258
                  }
00259
00260
            else
```

5.8 input.c 125

```
{
00262
                input_error (_("No generations number"));
00263
                goto exit_on_error;
00264
00265
00266
            // Obtaining mutation probability
            if (xmlHasProp (node, (const xmlChar *) LABEL_MUTATION))
00267
00268
00269
                input->mutation_ratio
00270
                   = jb_xml_node_get_float (node, (const xmlChar *) LABEL_MUTATION,
00271
                                            &error_code);
                if (!error_code || input->mutation_ratio < 0.</pre>
00272
00273
                    || input->mutation_ratio >= 1.)
00274
00275
                    input_error (_("Invalid mutation probability"));
00276
                    goto exit_on_error;
00277
00278
            else
00280
              {
00281
                input_error (_("No mutation probability"));
00282
                goto exit_on_error;
              }
00283
00284
00285
            // Obtaining reproduction probability
            if (xmlHasProp (node, (const xmlChar *) LABEL_REPRODUCTION))
00287
00288
                input->reproduction_ratio
00289
                   = jb_xml_node_get_float (node, (const xmlChar *) LABEL_REPRODUCTION,
00290
                                            &error_code);
00291
                if (!error_code || input->reproduction_ratio < 0.</pre>
00292
                    || input->reproduction_ratio >= 1.0)
00293
00294
                    input_error (_("Invalid reproduction probability"));
00295
                    goto exit_on_error;
00296
00297
              }
00298
            else
00299
              {
00300
                input_error (_("No reproduction probability"));
00301
                goto exit_on_error;
              }
00302
00303
00304
            // Obtaining adaptation probability
            if (xmlHasProp (node, (const xmlChar *) LABEL_ADAPTATION))
00305
00306
00307
                input->adaptation_ratio
00308
                  = jb_xml_node_get_float (node, (const xmlChar *) LABEL_ADAPTATION,
00309
                                            &error_code);
00310
                if (!error_code || input->adaptation_ratio < 0.</pre>
00311
                    || input->adaptation_ratio >= 1.)
00312
00313
                    input_error (_("Invalid adaptation probability"));
00314
                    goto exit_on_error;
00315
00316
            else
00318
              {
00319
                input_error (_("No adaptation probability"));
00320
                goto exit_on_error;
00321
00322
00323
            // Checking survivals
00324
            i = input->mutation_ratio * input->nsimulations;
00325
            i += input->reproduction_ratio * input->nsimulations;
00326
            i += input->adaptation_ratio * input->nsimulations;
            if (i > input->nsimulations - 2)
00327
00328
              {
00329
                input_error
00330
                  (_("No enough survival entities to reproduce the population"));
00331
                goto exit_on_error;
00332
00333
        else
00334
00335
00336
            input_error (_("Unknown algorithm"));
00337
            goto exit_on_error;
00338
00339
        xmlFree (buffer);
00340
        buffer = NULL:
00341
00342
        if (input->algorithm == ALGORITHM_MONTE_CARLO
00343
            || input->algorithm == ALGORITHM_SWEEP
00344
            || input->algorithm == ALGORITHM_ORTHOGONAL)
00345
00346
00347
            // Obtaining iterations number
```

```
input->niterations = jb_xml_node_get_uint_with_default
00349
              (node, (const xmlChar *) LABEL_NITERATIONS, &error_code, 1);
00350
            if (!error_code || !input->niterations)
00351
            {
00352
                input_error (_("Bad iterations number"));
00353
                goto exit_on_error;
00354
00355
00356
            // Obtaining best number
            input->nbest
00357
00358
              = jb_xml_node_get_uint_with_default (node,
00359
                                                    (const xmlChar *) LABEL NBEST.
00360
                                                    &error code, 1);
00361
            if (!error_code || !input->nbest)
00362
             {
00363
                input_error (_("Invalid best number"));
00364
                goto exit_on_error;
00365
00366
00367
            // Obtaining tolerance
00368
            input->tolerance
00369
              = jb_xml_node_get_float_with_default (node,
                                                     (const xmlChar *) LABEL TOLERANCE,
00370
00371
                                                     &error_code, 0.);
00372
            if (!error_code || input->tolerance < 0.)</pre>
00373
00374
                input_error (_("Invalid tolerance"));
00375
                goto exit_on_error;
00376
00377
00378
            // Getting hill climbing method parameters
00379
            if (xmlHasProp (node, (const xmlChar *) LABEL_NSTEPS))
00380
00381
                input->nsteps =
00382
                  jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSTEPS,
00383
                                        &error_code);
00384
                if (!error code)
00385
00386
                    input_error (_("Invalid steps number"));
00387
                    goto exit_on_error;
00388
00389 #if DEBUG INPUT
                fprintf (stderr, "input_open_xml: nsteps=%u\n", input->nsteps);
00390
00391 #endif
                buffer = xmlGetProp (node, (const xmlChar *) LABEL_CLIMBING);
00392
00393
                if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_COORDINATES))
                  input->climbing = CLIMBING_METHOD_COORDINATES;
00394
00395
                else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_RANDOM))
00396
                 {
00397
                    input->climbing = CLIMBING_METHOD_RANDOM;
00398
                    input->nestimates
00399
                      = jb_xml_node_get_uint (node,
00400
                                               (const xmlChar *) LABEL_NESTIMATES,
00401
                                               &error_code);
00402
                    if (!error_code || !input->nestimates)
00403
                      {
00404
                        input_error (_("Invalid estimates number"));
00405
                        goto exit_on_error;
00406
00407
                  }
00408
                else
00409
                 {
00410
                    input_error (_("Unknown method to estimate the hill climbing"));
00411
                    goto exit_on_error;
00412
00413
                xmlFree (buffer);
00414
                buffer = NULL;
                input->relaxation
00415
00416
                 = jb_xml_node_get_float_with_default (node,
00417
                                                         (const xmlChar *)
00418
                                                         LABEL_RELAXATION,
00419
                                                         &error_code,
00420
                                                         DEFAULT_RELAXATION);
                if (!error_code || input->relaxation < 0. || input->relaxation > 2.)
00421
00422
                  {
                   input_error (_("Invalid relaxation parameter"));
00423
00424
                    goto exit_on_error;
00425
00426
00427
            else
00428
             input->nsteps = 0;
00429
00430
        // Obtaining the threshold
        input->threshold =
00431
00432
          jb_xml_node_get_float_with_default (node, (const xmlChar *) LABEL_THRESHOLD,
00433
                                               &error_code, 0.);
00434
       if (!error code)
```

5.8 input.c 127

```
{
            input_error (_("Invalid threshold"));
00436
00437
            goto exit_on_error;
00438
00439
00440
        // Reading the experimental data
        for (child = node->children; child; child = child->next)
00441
00442
00443
            if (xmlStrcmp (child->name, (const xmlChar *) LABEL_EXPERIMENT))
00444
             break;
00445 #if DEBUG_INPUT
           fprintf (stderr, "input_open_xml: nexperiments=%u\n",
00446
00447
                     input->nexperiments);
00448 #endif
00449
           input->experiment = (Experiment *)
00450
             g_realloc (input->experiment,
00451
                         (1 + input->nexperiments) * sizeof (Experiment));
00452
            if (!input->nexperiments)
00453
00454
                if (!experiment_open_xml (input->experiment, child, 0))
                  goto exit_on_error;
00455
00456
00457
           else
00458
             {
00459
                if (!experiment_open_xml (input->experiment + input->nexperiments,
00460
                                          child, input->experiment->ninputs))
00461
00462
00463 ++input->nexperiments;
00464 #if DEBUG_INPUT
00465
            fprintf (stderr, "input_open_xml: nexperiments=%u\n",
00466
                     input->nexperiments);
00467 #endif
00468
00469
        if (!input->nexperiments)
00470
00471
            input_error (_("No optimization experiments"));
           goto exit_on_error;
00473
00474
        buffer = NULL;
00475
00476
        // Reading the variables data
        if (input->algorithm == ALGORITHM_SWEEP
00477
00478
            || input->algorithm == ALGORITHM_ORTHOGONAL)
00479
          input->nsimulations = 1;
00480
        for (; child; child = child->next)
00481
00482 #if DEBUG INPUT
            fprintf (stderr, "input_open_xml: nvariables=%u\n", input->nvariables);
00483
00484 #endif
00485
            if (xmlStrcmp (child->name, (const xmlChar *) LABEL_VARIABLE))
00486
               00487
00488
00489
00490
               goto exit_on_error;
00491
            input->variable = (Variable *)
00492
00493
             g_realloc (input->variable,
00494
                         (1 + input->nvariables) * sizeof (Variable));
            if (!variable_open_xml (input->variable + input->nvariables, child,
00495
00496
                                    input->algorithm, input->nsteps))
00497
              goto exit_on_error;
00498
            if (input->algorithm == ALGORITHM_SWEEP
00499
                || input->algorithm == ALGORITHM_ORTHOGONAL)
00500
              input->nsimulations *= input->variable[input->nvariables].nsweeps;
00501
            ++input->nvariables;
00502
00503
        if (!input->nvariables)
00504
         {
00505
            input_error (_("No optimization variables"));
00506
            goto exit_on_error;
00507
00508
        if (input->nbest > input->nsimulations)
00509
         {
00510
            input_error (_("Best number higher than simulations number"));
00511
            goto exit_on_error;
00512
00513
        buffer = NULL:
00514
00515
        // Obtaining the error norm
        if (xmlHasProp (node, (const xmlChar *) LABEL_NORM))
00517
00518
            buffer = xmlGetProp (node, (const xmlChar *) LABEL_NORM);
            if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_EUCLIDIAN))
  input->norm = ERROR_NORM_EUCLIDIAN;
00519
00520
            else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_MAXIMUM))
00521
```

```
input->norm = ERROR_NORM_MAXIMUM;
00523
            else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_P))
00524
              {
                input->norm = ERROR_NORM_P;
00525
00526
                input->p
00527
                  = jb_xml_node_get_float (node, (const xmlChar *) LABEL_P,
                                           &error_code);
00529
                if (!error_code)
00530
                    input_error (_("Bad P parameter"));
00531
00532
                    goto exit_on_error;
00533
00534
00535
            else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_TAXICAB))
00536
              input->norm = ERROR_NORM_TAXICAB;
00537
00538
              {
                input_error (_("Unknown error norm"));
00539
00540
                goto exit_on_error;
00541
00542
            xmlFree (buffer);
00543
00544
       else
         input->norm = ERROR_NORM_EUCLIDIAN;
00545
00546
00547
       // Closing the XML document
00548
        xmlFreeDoc (doc);
00549
00550 #if DEBUG_INPUT
00551
       fprintf (stderr, "input_open_xml: end\n");
00552 #endif
00553
       return 1;
00554
00555 exit_on_error:
00556 xmlFree (buffer);
00557
        xmlFreeDoc (doc);
00558 #if DEBUG_INPUT
       fprintf (stderr, "input_open_xml: end\n");
00560 #endif
00561
       return 0;
00562 }
00563
00569 static inline int
00570 input_open_json (JsonParser * parser)
00571 {
00572
       JsonNode *node, *child;
00573
       JsonObject *object;
00574
       JsonArrav *arrav;
00575
       const char *buffer:
00576
       int error code;
00577
       unsigned int i, n;
00578
00579 #if DEBUG_INPUT
00580
       fprintf (stderr, "input_open_json: start\n");
00581 #endif
00582
00583
        // Resetting input data
00584
       input->type = INPUT_TYPE_JSON;
00585
00586
       // Getting the root node
00587 #if DEBUG_INPUT
       fprintf (stderr, "input_open_json: getting the root node\n");
00588
00589 #endif
00590
       node = json_parser_get_root (parser);
00591
        object = json_node_get_object (node);
00592
00593
        // Getting result and variables file names
00594
        if (!input->result)
00595
00596
            buffer = json_object_get_string_member (object, LABEL_RESULT_FILE);
00597
            if (!buffer)
00598
             buffer = result_name;
00599
            input->result = g_strdup (buffer);
00600
00601
        else
00602
         input->result = g_strdup (result_name);
        if (!input->variables)
00603
00604
00605
           buffer = json_object_get_string_member (object, LABEL_VARIABLES_FILE);
            if (!buffer)
00606
00607
              buffer = variables name;
            input->variables = g_strdup (buffer);
00608
00609
00610
        else
00611
         input->variables = g_strdup (variables_name);
00612
00613
       // Opening simulator program name
```

5.8 input.c 129

```
buffer = json_object_get_string_member (object, LABEL_SIMULATOR);
        if (!buffer)
00615
00616
00617
            input_error (_("Bad simulator program"));
00618
            goto exit_on_error;
00619
00620
        input->simulator = g_strdup (buffer);
00621
        // Opening evaluator program name
00622
00623
        buffer = json_object_get_string_member (object, LABEL_EVALUATOR);
        if (buffer)
00624
00625
         input->evaluator = g_strdup (buffer);
00626
00627
        // Obtaining pseudo-random numbers generator seed
00628
00629
          = jb_json_object_get_uint_with_default (object, LABEL_SEED,
                                                    &error_code, DEFAULT_RANDOM_SEED);
00630
00631
        if (!error code)
00632
00633
            input_error (_("Bad pseudo-random numbers generator seed"));
00634
            goto exit_on_error;
00635
00636
        // Opening algorithm
00637
        buffer = json_object_get_string_member (object, LABEL_ALGORITHM);
if (!strcmp (buffer, LABEL_MONTE_CARLO))
00638
00639
00640
00641
            input->algorithm = ALGORITHM_MONTE_CARLO;
00642
00643
            // Obtaining simulations number
00644
            input->nsimulations
00645
                jb_json_object_get_uint (object, LABEL_NSIMULATIONS, &error_code);
00646
            if (!error_code)
00647
              {
00648
                input_error (_("Bad simulations number"));
00649
                goto exit_on_error;
              }
00650
00651
          }
00652
        else if (!strcmp (buffer, LABEL_SWEEP))
00653
          input->algorithm = ALGORITHM_SWEEP;
        else if (!strcmp (buffer, LABEL_ORTHOGONAL))
00654
          input->algorithm = ALGORITHM_ORTHOGONAL;
00655
00656
        else if (!strcmp (buffer, LABEL_GENETIC))
00657
          {
00658
            input->algorithm = ALGORITHM_GENETIC;
00659
00660
            // Obtaining population
00661
            if (json_object_get_member (object, LABEL_NPOPULATION))
00662
              {
00663
                input->nsimulations
00664
                    jb_json_object_get_uint (object, LABEL_NPOPULATION, &error_code);
00665
                if (!error_code || input->nsimulations < 3)</pre>
00666
00667
                    input_error (_("Invalid population number"));
00668
                    goto exit_on_error;
                  }
00669
00670
00671
            else
00672
             {
00673
                input_error (_("No population number"));
00674
                goto exit_on_error;
00675
00676
00677
            // Obtaining generations
00678
            if (json_object_get_member (object, LABEL_NGENERATIONS))
00679
00680
                input->niterations
                   = jb_json_object_get_uint_with_default (object, LABEL_NGENERATIONS,
00681
00682
                                                            &error code, 1);
00683
                if (!error_code || !input->niterations)
00684
00685
                    input_error (_("Invalid generations number"));
00686
                    goto exit_on_error;
00687
00688
            else
00689
00690
              {
00691
                input_error (_("No generations number"));
00692
                goto exit_on_error;
00693
00694
00695
            // Obtaining mutation probability
00696
            if (json_object_get_member (object, LABEL_MUTATION))
00697
00698
                input->mutation_ratio
                   = jb_json_object_get_float (object, LABEL_MUTATION, &error_code);
00699
00700
                if (!error_code || input->mutation_ratio < 0.</pre>
```

```
|| input->mutation_ratio >= 1.)
00702
                    input_error (_("Invalid mutation probability"));
00703
00704
                    goto exit_on_error;
00705
00706
              }
00707
            else
00708
              {
00709
                input_error (_("No mutation probability"));
00710
                goto exit_on_error;
              }
00711
00712
00713
            // Obtaining reproduction probability
00714
            if (json_object_get_member (object, LABEL_REPRODUCTION))
00715
00716
                input->reproduction_ratio
                  = jb_json_object_get_float (object, LABEL_REPRODUCTION,
00717
00718
                                               &error_code);
                if (!error_code || input->reproduction_ratio < 0.</pre>
00720
                    || input->reproduction_ratio >= 1.0)
00721
00722
                    input_error (_("Invalid reproduction probability"));
00723
                    goto exit_on_error;
00724
00725
00726
            else
00727
00728
                input_error (_("No reproduction probability"));
00729
                goto exit_on_error;
00730
00731
00732
            // Obtaining adaptation probability
00733
            if (json_object_get_member (object, LABEL_ADAPTATION))
00734
00735
                input->adaptation_ratio
                  = jb_json_object_get_float (object, LABEL_ADAPTATION, &error_code);
00736
00737
                if (!error code || input->adaptation ratio < 0.
00738
                    || input->adaptation_ratio >= 1.)
00739
00740
                     input_error (_("Invalid adaptation probability"));
00741
                    goto exit_on_error;
00742
                  }
00743
              }
00744
            else
00745
             {
00746
                input_error (_("No adaptation probability"));
00747
                goto exit_on_error;
00748
00749
00750
            // Checking survivals
00751
            i = input->mutation_ratio * input->nsimulations;
00752
            i += input->reproduction_ratio * input->nsimulations;
00753
            i += input->adaptation_ratio * input->nsimulations;
00754
            if (i > input->nsimulations - 2)
00755
00756
                input error
00757
                  (_("No enough survival entities to reproduce the population"));
00758
                goto exit_on_error;
00759
00760
          }
00761
        else
00762
         {
00763
            input_error (_("Unknown algorithm"));
00764
            goto exit_on_error;
00765
00766
        if (input->algorithm == ALGORITHM_MONTE_CARLO
00767
00768
            || input->algorithm == ALGORITHM_SWEEP
00769
            || input->algorithm == ALGORITHM_ORTHOGONAL)
00770
00771
00772
            // Obtaining iterations number
00773
            input->niterations
            i jb_json_object_get_uint (object, LABEL_NITERATIONS, &error_code);
if (!error_code || !input->niterations)
00774
00775
00776
                input_error (_("Bad iterations number"));
00777
00778
                goto exit_on_error;
00779
00780
00781
            // Obtaining best number
00782
            input->nbest
00783
               = jb_json_object_get_uint_with_default (object, LABEL_NBEST,
00784
                                                        &error_code, 1);
00785
            if (!error_code || !input->nbest)
00786
00787
                input_error (_("Invalid best number"));
```

5.8 input.c 131

```
goto exit_on_error;
00789
00790
00791
             // Obtaining tolerance
00792
             input->tolerance
00793
               = jb_json_object_get_float_with_default (object, LABEL_TOLERANCE,
00794
                                                          &error_code, 0.);
00795
             if (!error_code || input->tolerance < 0.)</pre>
00796
                input_error (_("Invalid tolerance"));
00797
00798
                 goto exit_on_error;
00799
00800
00801
             // Getting hill climbing method parameters
00802
             if (json_object_get_member (object, LABEL_NSTEPS))
00803
                 input->nsteps
00804
00805
                   = jb_json_object_get_uint (object, LABEL_NSTEPS, &error_code);
00806
                 if (!error_code)
00807
                  {
00808
                     input_error (_("Invalid steps number"));
00809
                     goto exit_on_error;
00810
                buffer = json_object_get_string_member (object, LABEL_CLIMBING);
if (!strcmp (buffer, LABEL_COORDINATES))
  input->climbing = CLIMBING_METHOD_COORDINATES;
00811
00812
                 else if (!strcmp (buffer, LABEL_RANDOM))
00814
00815
00816
                     input->climbing = CLIMBING_METHOD_RANDOM;
00817
                     input->nestimates
00818
                       = jb_json_object_get_uint (object, LABEL_NESTIMATES,
00819
                                                    &error code);
00820
                     if (!error_code || !input->nestimates)
00821
00822
                         input_error (_("Invalid estimates number"));
00823
                         goto exit_on_error;
00824
00826
                 else
00827
00828
                     input_error (_("Unknown method to estimate the hill climbing"));
00829
                    goto exit_on_error;
00830
00831
                 input->relaxation
                  = jb_json_object_get_float_with_default (object, LABEL_RELAXATION,
00832
00833
                                                               &error_code,
00834
                                                              DEFAULT RELAXATION):
00835
                 if (!error_code || input->relaxation < 0. || input->relaxation > 2.)
00836
                  {
                    input_error (_("Invalid relaxation parameter"));
00837
00838
                     goto exit_on_error;
00839
00840
00841
            else
00842
              input->nsteps = 0;
00843
00844
        // Obtaining the threshold
00845
        input->threshold
00846
           = jb_json_object_get_float_with_default (object, LABEL_THRESHOLD,
00847
                                                      &error_code, 0.);
00848
00849
        if (!error_code)
00850
          {
            input_error (_("Invalid threshold"));
00851
00852
            goto exit_on_error;
00853
00854
00855
        // Reading the experimental data
00856
        array = json_object_get_array_member (object, LABEL_EXPERIMENTS);
        n = json_array_get_length (array);
00858
        input->experiment = (Experiment *) g_malloc (n * sizeof (Experiment));
00859
        for (i = 0; i < n; ++i)
00860
00861 #if DEBUG_INPUT
            fprintf (stderr, "input_open_json: nexperiments=%u\n",
00862
                     input->nexperiments);
00863
00864 #endif
00865
            child = json_array_get_element (array, i);
00866
             if (!input->nexperiments)
00867
              {
00868
                 if (!experiment_open_json (input->experiment, child, 0))
00869
                  goto exit_on_error;
00870
00871
             else
00872
              {
                if (!experiment_open_json (input->experiment + input->nexperiments,
00873
00874
                                             child, input->experiment->ninputs))
```

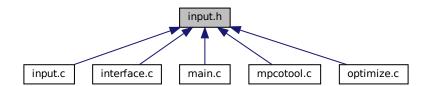
```
goto exit_on_error;
00876
00877
            ++input->nexperiments;
00878 #if DEBUG_INPUT
           fprintf (stderr, "input_open_json: nexperiments=%u\n",
00879
00880
                      input->nexperiments):
00881 #endif
00882
00883
        if (!input->nexperiments)
00884
            input_error (_("No optimization experiments"));
00885
00886
            goto exit_on_error;
00887
00888
00889
        // Reading the variables data
00890
        array = json_object_get_array_member (object, LABEL_VARIABLES);
00891
        n = json\_array\_get\_length (array);
00892
        input->variable = (Variable *) g_malloc (n * sizeof (Variable));
        for (i = 0; i < n; ++i)
00893
00894
00895 #if DEBUG_INPUT
00896
            fprintf (stderr, "input_open_json: nvariables=%u\n", input->nvariables);
00897 #endif
            child = json_array_get_element (array, i);
if (!variable_open_json (input->variable + input->nvariables, child,
00898
00899
00900
                                      input->algorithm, input->nsteps))
00901
              goto exit_on_error;
00902
            ++input->nvariables;
00903
00904
        if (!input->nvariables)
00905
00906
            input_error (_("No optimization variables"));
00907
            goto exit_on_error;
00908
00909
        // Obtaining the error norm
00910
00911
        if (json_object_get_member (object, LABEL_NORM))
00912
00913
            buffer = json_object_get_string_member (object, LABEL_NORM);
00914
            if (!strcmp (buffer, LABEL_EUCLIDIAN))
00915
              input->norm = ERROR_NORM_EUCLIDIAN;
            else if (!strcmp (buffer, LABEL_MAXIMUM))
  input->norm = ERROR_NORM_MAXIMUM;
00916
00917
00918
            else if (!strcmp (buffer, LABEL_P))
00919
              {
00920
                input->norm = ERROR_NORM_P;
00921
                input->p = jb_json_object_get_float (object, LABEL_P, &error_code);
00922
                if (!error_code)
00923
                 {
00924
                    input_error (_("Bad P parameter"));
00925
                    goto exit_on_error;
00926
00927
00928
            else if (!strcmp (buffer, LABEL_TAXICAB))
00929
              input->norm = ERROR_NORM_TAXICAB;
00930
            else
00931
              {
00932
                input_error (_("Unknown error norm"));
00933
                goto exit_on_error;
00934
00935
          }
00936
       else
00937
          input->norm = ERROR_NORM_EUCLIDIAN;
00938
00939
       // Closing the JSON document
00940 g_object_unref (parser);
00941
00942 #if DEBUG_INPUT
00943 fprintf (stderr, "input_open_json: end\n");
00944 #endif
00945
       return 1;
00946
00947 exit on error:
00948 g_object_unref (parser);
00949 #if DEBUG_INPUT
00950
       fprintf (stderr, "input_open_json: end\n");
00951 #endif
00952
       return 0;
00953 }
00954
00960 int
00961 input_open (char *filename)
00962 {
        xmlDoc *doc;
00963
00964
       JsonParser *parser;
00965
00966 #if DEBUG_INPUT
```

```
00967
        fprintf (stderr, "input_open: start\n");
00968 #endif
00969
00970
        // Resetting input data
00971
        input_new ();
00972
         // Opening input file
00974 #if DEBUG_INPUT
      fprintf (stderr, "input_open: opening the input file %s\n", filename);
fprintf (stderr, "input_open: trying XML format\n");
00975
00976
00977 #endif
00978 doc = xmlParseFile (filename);
00979
        if (!doc)
00980
00982 fprintf (stderr, "input_open: trying JSON format\n"); 00983 #endif
00981 #if DEBUG_INPUT
00984
            parser = json_parser_new ();
             if (!json_parser_load_from_file (parser, filename, NULL))
00986
00987
                 input_error (_("Unable to parse the input file"));
00988
                 goto exit_on_error;
00989
00990
            if (!input_open_json (parser))
00991
              goto exit_on_error;
00992
00993
        else if (!input_open_xml (doc))
00994
          goto exit_on_error;
00995
00996
        // Getting the working directory
        input->directory = g_path_get_dirname (filename);
00997
00998
        input->name = g_path_get_basename (filename);
00999
01000 #if DEBUG_INPUT
01001
        fprintf (stderr, "input_open: end\n");
01002 #endif
01003
        return 1;
01005 exit_on_error:
01006
      jbw_show_error (error_message);
01007
        g_free (error_message);
01008 input_free ();
01009 #if DEBUG_INPUT
01010
        fprintf (stderr, "input_open: end\n");
01011 #endif
        return 0;
01012
01013 }
```

5.9 input.h File Reference

Header file to define the input functions.

This graph shows which files directly or indirectly include this file:



Data Structures

struct Input

Struct to define the optimization input file.

Enumerations

• enum ClimbingMethod { CLIMBING_METHOD_COORDINATES = 0 , CLIMBING_METHOD_RANDOM = 1 } Enum to define the methods to estimate the hill climbing.

enum ErrorNorm { ERROR_NORM_EUCLIDIAN = 0 , ERROR_NORM_MAXIMUM = 1 , ERROR_NORM_P = 2 , ERROR_NORM_TAXICAB = 3 }

Enum to define the error norm.

Functions

- void input_new ()
- void input_free ()
- int input_open (char *filename)

Variables

• Input input [1]

Global Input struct to set the input data.

• const char * result name

Name of the result file.

· const char * variables name

Name of the variables file.

5.9.1 Detailed Description

Header file to define the input functions.

Authors

Javier Burguete.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file input.h.

5.9.2 Enumeration Type Documentation

5.9.2.1 ClimbingMethod

enum ClimbingMethod

Enum to define the methods to estimate the hill climbing.

Enumerator

CLIMBING_METHOD_COORDINATES	Coordinates hill climbing method.
CLIMBING_METHOD_RANDOM	Random hill climbing method.

Definition at line 42 of file input.h.

```
00043 {
00044 CLIMBING_METHOD_COORDINATES = 0,
00045 CLIMBING_METHOD_RANDOM = 1,
00046 };
```

5.9.2.2 ErrorNorm

```
enum ErrorNorm
```

Enum to define the error norm.

Enumerator

ERROR_NORM_EUCLIDIAN	Euclidian norm: $\sqrt{\sum_{i} (w_i x_i)^2}$.
ERROR_NORM_MAXIMUM	Maximum norm: $\max_i w_i x_i$.
ERROR_NORM_P	P-norm $\sqrt[p]{\sum_i w_i x_i ^p}$.
ERROR_NORM_TAXICAB	Taxicab norm $\sum_i w_i x_i $.

Definition at line 49 of file input.h.

5.9.3 Function Documentation

5.9.3.1 input_free()

```
void input_free ( )
```

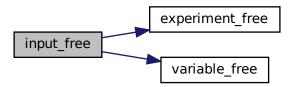
Function to free the memory of the input file data.

Definition at line 84 of file input.c.

```
00085 {
00086 unsigned int i;
00087 #if DEBUG_INPUT
00088 fprintf (stderr, "input_free: start\n");
00089 #endif
00090 g_free (input->name);
00091 g_free (input->directory);
```

```
for (i = 0; i < input->nexperiments; ++i)
00093
          experiment_free (input->experiment + i, input->type);
00094
        for (i = 0; i < input->nvariables; ++i)
00095
         variable_free (input->variable + i, input->type);
00096
        g_free (input->experiment);
        g_free (input->variable);
if (input->type == INPUT_TYPE_XML)
00097
00099
00100
             xmlFree (input->evaluator);
            xmlFree (input->simulator);
xmlFree (input->result);
00101
00102
00103
             xmlFree (input->variables);
00104
00105
        else
00106
00107
             g_free (input->evaluator);
             g_free (input->simulator);
g_free (input->result);
00108
00109
             g_free (input->variables);
00110
00111
00112
        input->nexperiments = input->nvariables = input->nsteps = 0;
00113 #if DEBUG_INPUT
00114 fprintf (stderr, "input_free: end\n");
00115 #endif
00116 }
```

Here is the call graph for this function:



5.9.3.2 input_new()

```
void input_new ( )
```

Function to create a new Input struct.

Definition at line 66 of file input.c.

```
00067 {
00068 #if DEBUG_INPUT
00069 fprintf (stderr, "input_new: start\n");
00070 #endif
00071 input->nvariables = input->nexperiments = input->nsteps = 0;
00072 input->simulator = input->evaluator = input->directory = input->name = NULL;
00073 input->experiment = NULL;
00074 input->variable = NULL;
00075 #if DEBUG_INPUT
00076 fprintf (stderr, "input_new: end\n");
00077 #endif
00078 }
```

5.9.3.3 input_open()

Function to open the input file.

Returns

1 on success, 0 on error.

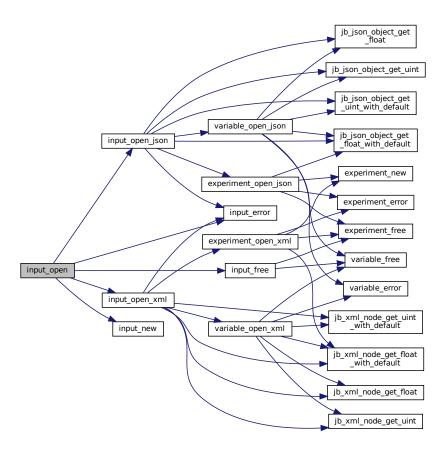
Parameters

filename Input data file name.

Definition at line 961 of file input.c.

```
00962 {
        xmlDoc *doc;
00964
        JsonParser *parser;
00965
00966 #if DEBUG_INPUT
       fprintf (stderr, "input_open: start\n");
00967
00968 #endif
00969
00970
        // Resetting input data
00971
        input_new ();
00972
        // Opening input file
00973
00974 #if DEBUG_INPUT
00975 fprintf (stderr, "input_open: opening the input file %s\n", filename);
00976 fprintf (stderr, "input_open: trying XML format\n");
00977 #endif
00978 doc = xmlParseFile (filename);
00979
       if (!doc)
00980
00981 #if DEBUG_INPUT
            fprintf (stderr, "input_open: trying JSON format\n");
00983 #endif
00984
            parser = json_parser_new ();
00985
             if (!json_parser_load_from_file (parser, filename, NULL))
00986
              {
                 input_error (_("Unable to parse the input file"));
00987
00988
                 goto exit_on_error;
00989
00990
            if (!input_open_json (parser))
00991
              goto exit_on_error;
00992
00993
        else if (!input_open_xml (doc))
00994
          goto exit on error;
00995
00996
        // Getting the working directory
00997
        input->directory = g_path_get_dirname (filename);
00998
       input->name = g_path_get_basename (filename);
00999
01000 #if DEBUG_INPUT
01001 fprintf (stderr, "input_open: end\n");
01002 #endif
01003 return 1;
01004
01005 exit on error:
01006 jbw_show_error (error_message);
01007 g_free (error_message);
01008 input_free ();
01009 #if DEBUG_INPUT
01010
       fprintf (stderr, "input_open: end\n");
01011 #endif
       return 0;
01012
01013 }
```

Here is the call graph for this function:



5.9.4 Variable Documentation

5.9.4.1 input

Input input[1] [extern]

Global Input struct to set the input data.

Definition at line 57 of file input.c.

5.9.4.2 result_name

const char* result_name [extern]

Name of the result file.

Definition at line 59 of file input.c.

5.10 input.h 139

5.9.4.3 variables_name

```
const char* variables_name [extern]
```

Name of the variables file.

Definition at line 60 of file input.c.

5.10 input.h

Go to the documentation of this file.

```
00001 /
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00005
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00009
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are met:
00012
00013 1.
         Redistributions of source code must retain the above copyright notice,
00014 this list of conditions and the following disclaimer.
00015
         Redistributions in binary form must reproduce the above copyright notice,
00017 this list of conditions and the following disclaimer in the
00018 documentation and/or other materials provided with the distribution.
00019
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF 00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
00031
00038 #ifndef INPUT__H
00039 #define INPUT__H 1
00040
00042 enum ClimbingMethod
00043 {
00044
        CLIMBING METHOD COORDINATES = 0.
00045
       CLIMBING_METHOD_RANDOM = 1,
00046 };
00047
00049 enum ErrorNorm
00050 {
00051
        ERROR_NORM_EUCLIDIAN = 0,
        ERROR_NORM_MAXIMUM = 1,
ERROR_NORM_P = 2,
00053
00055
00057
        ERROR_NORM_TAXICAB = 3
00059 };
00060
00065 typedef struct
00066 {
        Experiment *experiment:
00067
00068
        Variable *variable;
00069
       char *result;
00070
        char *variables;
00071
        char *simulator;
00072
        char *evaluator:
00074
        char *directory;
00075
        char *name;
00076
        double tolerance;
00077
        double mutation_ratio;
00078
        double reproduction_ratio;
00079
        double adaptation_ratio;
08000
        double relaxation;
00081
       double p;
double threshold;
00082
00083
       unsigned long int seed;
```

```
unsigned int nvariables;
00086
       unsigned int nexperiments;
00087
       unsigned int nsimulations;
00088
       unsigned int algorithm;
00089
       unsigned int nsteps;
00091
       unsigned int climbing;
00092
       unsigned int nestimates;
00094
       unsigned int niterations;
00095
       unsigned int nbest;
00096
       unsigned int norm;
00097
       unsigned int type;
00098 } Input;
00099
00100 extern Input input[1];
00101 extern const char *result_name;
00102 extern const char *variables_name;
00103
00104 // Public functions
00105 void input_new ();
00106 void input_free ();
00107 int input_open (char *filename);
00108
00109 #endif
```

5.11 interface.c File Reference

Source file to define the graphical interface functions.

```
#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <gsl/gsl_rng.h>
#include <libxml/parser.h>
#include <libintl.h>
#include <qlib.h>
#include <glib/gstdio.h>
#include <json-glib/json-glib.h>
#include <mpi.h>
#include <gio/gio.h>
#include <gtk/gtk.h>
#include "jb/src/jb_xml.h"
#include "jb/src/jb_json.h"
#include "jb/src/jb_win.h"
#include "genetic/genetic.h"
#include "tools.h"
#include "experiment.h"
#include "variable.h"
#include "input.h"
#include "optimize.h"
#include "interface.h"
Include dependency graph for interface.c:
```



Macros

• #define DEBUG_INTERFACE 1

Macro to debug interface functions.

#define INPUT_FILE "test-ga.xml"

Macro to define the initial input file.

Functions

```
    static void input_save_climbing_xml (xmlNode *node)

    static void input_save_climbing_json (JsonNode *node)

    static void input save xml (xmlDoc *doc)

    static void input_save_json (JsonGenerator *generator)

• static void input save (char *filename)

    static void dialog_options_close (GtkDialog *dlg, int response_id)

• static void options_new ()
• static void running new ()

    static unsigned int window_get_algorithm ()

    static unsigned int window get climbing ()

    static unsigned int window_get_norm ()

• static void window_save_climbing ()

    static void dialog_save_close (GtkFileChooserDialog *dlg, int response_id)

    static void window save ()

• static void window run ()

    static void window_help ()

    static void window_about ()

· static void window update climbing ()
• static void window update ()
• static void window set algorithm ()

    static void window set experiment ()

    static void window_remove_experiment ()

    static void window_add_experiment ()

    static void dialog name experiment close (GtkFileChooserDialog *dlg, int response id, void *data)

    static void window name experiment ()

    static void window weight experiment ()

    static void window_inputs_experiment ()

    static void window template experiment close (GtkFileChooserDialog *dlg, int response id, void *data)

    static void window_template_experiment (void *data)

• static void window_set_variable ()
• static void window remove variable ()

    static void window add variable ()

• static void window_label_variable ()
• static void window_precision_variable ()
• static void window rangemin variable ()

    static void window rangemax variable ()

    static void window_rangeminabs_variable ()

    static void window_rangemaxabs_variable ()

• static void window step variable ()

    static void window update variable ()

    static int window read (char *filename)

    static void dialog_open_close (GtkFileChooserDialog *dlg, int response_id)

• static void window open ()

    static void dialog_simulator_close (GtkFileChooserDialog *dlg, int response_id)

• static void dialog_simulator ()
• static void dialog evaluator close (GtkFileChooserDialog *dlg, int response id)

    static void dialog evaluator ()

    void window_new (GtkApplication *application)
```

Variables

• Window window [1]

Window struct to define the main interface window.

• static const char * logo []

Logo pixmap.

• static Options options [1]

Options struct to define the options dialog.

• static Running running [1]

Running struct to define the running dialog.

5.11.1 Detailed Description

Source file to define the graphical interface functions.

Authors

Javier Burguete and Borja Latorre.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file interface.c.

5.11.2 Macro Definition Documentation

5.11.2.1 DEBUG_INTERFACE

```
#define DEBUG_INTERFACE 1
```

Macro to debug interface functions.

Definition at line 69 of file interface.c.

5.11.2.2 **INPUT_FILE**

```
#define INPUT_FILE "test-ga.xml"
```

Macro to define the initial input file.

Definition at line 78 of file interface.c.

5.11.3 Function Documentation

5.11.3.1 dialog_evaluator()

```
static void dialog_evaluator ( ) [static]
```

Function to open a dialog to save the evaluator file.

Definition at line 2251 of file interface.c.

```
02252 {
02253
         GtkFileChooserDialog *dlg;
02254 #if DEBUG_INTERFACE
02255
         fprintf (stderr, "dialog_evaluator: start\n");
02256 #endif
02257 dlg = (GtkFileChooserDialog *)
02258
         gtk_file_chooser_dialog_new (_("Open evaluator file"),
02259
                                               window->window
02260
                                              GTK_FILE_CHOOSER_ACTION_OPEN,
        __("_Cancel"), GTK_RESPONSE_CANCEL,
__("_Open"), GTK_RESPONSE_ACCEPT, NULL);
g_signal_connect (dlg, "response", G_CALLBACK (dialog_evaluator_close), NULL);
02261
02263
02264
         gtk_window_present (GTK_WINDOW (dlg));
02265 #if DEBUG_INTERFACE
02266 fprintf (stderr, "dialog_evaluator: end\n");
02267 #endif
02268 }
```

Here is the call graph for this function:



5.11.3.2 dialog_evaluator_close()

Function to save the close the evaluator file dialog.

Parameters

dlg	GtkFileChooserDialog dialog.
response←	Response identifier.
_id	

Definition at line 2220 of file interface.c.

```
GFile *file1, *file2;
02224
02225
        char *buffer1, *buffer2;
02226 #if DEBUG INTERFACE
        fprintf (stderr, "dialog_evaluator_close: start\n");
02227
02228 #endif
02229
        if (response_id == GTK_RESPONSE_OK)
02230
02231
            buffer1 = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
             file1 = g_file_new_for_path (buffer1);
file2 = g_file_new_for_path (input->directory);
02232
02233
             buffer2 = g_file_get_relative_path (file2, file1);
02234
             input->evaluator = g_strdup (buffer2);
02235
02236
             g_free (buffer2);
             g_object_unref (file2);
g_object_unref (file1);
02237
02238
02239
            g_free (buffer1);
02241
        gtk_window_destroy (GTK_WINDOW (dlg));
02242 #if DEBUG_INTERFACE
02243 fprintf (stderr, "dialog_evaluator_close: end\n");
02244 #endif
02245 }
```

5.11.3.3 dialog_name_experiment_close()

Function to close the experiment name dialog.

Parameters

dlg	GtkFileChooserDialog struct.
response <i>←</i> _id	Response identifier.
data	Function data.

Definition at line 1459 of file interface.c.

```
01463 {
01464
        char *buffer;
01465 unsigned int i;
01466 #if DEBUG_INTERFACE
01467 fprintf (stderr, "window_name_experiment_close: start\n");
01468 #endif
      i = (size_t) data;
01470
        if (response_id == GTK_RESPONSE_OK)
01471
            buffer = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
01472
01473
            g_signal_handler_block (window->combo_experiment, window->id_experiment);
01474
            gtk_combo_box_text_remove (window->combo_experiment, i);
            gtk_combo_box_text_insert_text (window->combo_experiment, i, buffer);
01476
            gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), i);
01477
            g_signal_handler_unblock (window->combo_experiment,
01478
                                       window->id_experiment);
            g_free (buffer);
01479
01480
01481 #if DEBUG_INTERFACE
01482
       fprintf (stderr, "window_name_experiment_close: end\n");
01483 #endif
01484 }
```

5.11.3.4 dialog_open_close()

Function to close the input data dialog.

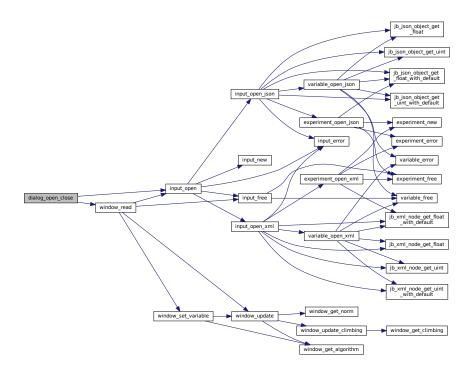
Parameters

dlg	GtkFileChooserDialog dialog.
response <i>←</i> _id	Response identifier.

Definition at line 2054 of file interface.c.

```
02057
       char *buffer, *directory, *name;
02058
02059
       GFile *file;
02060
02061 #if DEBUG_INTERFACE
02062
       fprintf (stderr, "dialog_open_close: start\n");
02063 #endif
02064
       // Saving a backup of the current input file
02065
02066
       directory = g_strdup (input->directory);
02067
       name = g_strdup (input->name);
02068
02069
       // If OK saving
       if (response_id == GTK_RESPONSE_OK)
02070
02071
         {
02072
02073
            // Traying to open the input file
02074
            file = gtk_file_chooser_get_file (GTK_FILE_CHOOSER (dlg));
02075
            buffer = g_file_get_path (file);
02076 #if DEBUG_INTERFACE
02077
           fprintf (stderr, "dialog_open_close: file name=%s\n", buffer);
02078 #endif
02079
          if (!window_read (buffer))
02080
02081 #if DEBUG_INTERFACE
                fprintf (stderr, "dialog_open_close: error reading input file\n");
02082
02083 #endif
02084
               g free (buffer);
02085
02086
                // Reading backup file on error
02087
               buffer = g_build_filename (directory, name, NULL);
02088
                input->result = input->variables = NULL;
02089
                if (!input_open (buffer))
02090
                 {
02092
                   // Closing on backup file reading error
02093 #if DEBUG_INTERFACE
02094
            fprintf (stderr,
02095
                              "dialog_open_close: error reading backup file\n");
02096 #endif
02097
                 }
02098
02099
           g_free (buffer);
02100
           g_object_unref (file);
02101
02102
       // Freeing and closing
02103
02104 g_free (name);
02105
       g_free (directory);
02106
       gtk_window_destroy (GTK_WINDOW (dlg));
02107
02108 #if DEBUG_INTERFACE
02109 fprintf (stderr, "dialog_open_close: end\n");
02110 #endif
02112 }
```

Here is the call graph for this function:



5.11.3.5 dialog_options_close()

Function to close the options dialog.

Parameters

dlg	GtkDialog options dialog.
response⇔	Response identifier.
id	

Definition at line 638 of file interface.c.

```
00640
00641 #if DEBUG_INTERFACE
         fprintf (stderr, "dialog_options_close: start\n");
00642
00643 #endif
00644
        if (response_id == GTK_RESPONSE_OK)
00645
00646
             = (unsigned long int) gtk_spin_button_get_value (options->spin_seed);
nthreads = gtk_spin_button_get_value_as_int (options->spin_threads);
00647
00648
00649
             nthreads_climbing
00650
                = gtk_spin_button_get_value_as_int (options->spin_climbing);
00651
        gtk_window_destroy (GTK_WINDOW (dlg));
00652
00653 #if DEBUG_INTERFACE
```

```
00654 fprintf (stderr, "dialog_options_close: end\n"); 00655 #endif 00656 }
```

5.11.3.6 dialog_save_close()

Function to close the save dialog.

Parameters

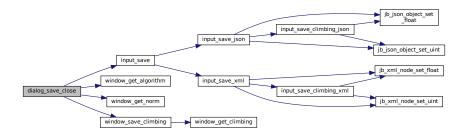
dlg	GtkFileChooserDialog dialog.
response←	Response identifier.
_id	

Definition at line 852 of file interface.c.

```
00855 {
00856
        GtkFileFilter *filter1;
00857
        char *buffer;
00858 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_save_close: start\n");
00859
00860 #endif
00861
       // If OK response then saving
        if (response_id == GTK_RESPONSE_OK)
00863
         {
00864
            // Setting input file type
            filter1 = gtk_file_chooser_get_filter (GTK_FILE_CHOOSER (dlg));
00865
            buffer = (char *) gtk_file_filter_get_name (filter1);
if (!strcmp (buffer, "XML"))
00866
00867
              input->type = INPUT_TYPE_XML;
00868
00869
00870
              input->type = INPUT_TYPE_JSON;
00871
00872
            // Adding properties to the root XML node
00873
            input->simulator
00874
              = g_strdup (gtk_button_get_label (window->button_simulator));
00875
            if (gtk_check_button_get_active (window->check_evaluator))
00876
             input->evaluator
00877
                 = g_strdup (gtk_button_get_label (window->button_evaluator));
00878
00879
              input->evaluator = NULL;
08800
            if (input->type == INPUT_TYPE_XML)
00881
00882
                input->result
00883
                  = (char *) xmlStrdup ((const xmlChar *)
00884
                                         gtk_entry_get_text (window->entry_result));
00885
                input->variables
00886
                  = (char *) xmlStrdup ((const xmlChar *)
00887
                                         gtk_entry_get_text (window->entry_variables));
00888
00889
            else
00890
              {
00891
                input->result = g_strdup (gtk_entry_get_text (window->entry_result));
00892
                input->variables =
00893
                  g_strdup (gtk_entry_get_text (window->entry_variables));
00894
00895
            \ensuremath{//} Setting the algorithm
00896
00897
            switch (window_get_algorithm ())
00898
              {
00899
              case ALGORITHM_MONTE_CARLO:
00900
                input->algorithm = ALGORITHM_MONTE_CARLO;
00901
                input->nsimulations
00902
                  = gtk_spin_button_get_value_as_int (window->spin_simulations);
00903
                input->niterations
00904
                  = gtk spin button get value as int (window->spin iterations);
00905
                input->tolerance = gtk_spin_button_get_value (window->spin_tolerance);
                input->nbest = gtk_spin_button_get_value_as_int (window->spin_bests);
```

```
00907
                window_save_climbing ();
00908
                break;
00909
              case ALGORITHM_SWEEP:
00910
                input->algorithm = ALGORITHM_SWEEP;
00911
                input->niterations
00912
                  = qtk_spin_button_get_value_as_int (window->spin_iterations);
00913
                input->tolerance = gtk_spin_button_get_value (window->spin_tolerance);
00914
                input->nbest = gtk_spin_button_get_value_as_int (window->spin_bests);
00915
                window_save_climbing ();
              break;
case ALGORITHM_ORTHOGONAL:
00916
00917
00918
                input->algorithm = ALGORITHM ORTHOGONAL:
00919
                input->niterations
00920
                   = gtk_spin_button_get_value_as_int (window->spin_iterations);
00921
                input->tolerance = gtk_spin_button_get_value (window->spin_tolerance);
00922
                input->nbest = gtk_spin_button_get_value_as_int (window->spin_bests);
                window_save_climbing ();
00923
00924
                break;
00925
              default:
00926
                input->algorithm = ALGORITHM_GENETIC;
                input->nsimulations
00927
00928
                   = gtk_spin_button_get_value_as_int (window->spin_population);
00929
                input->niterations
00930
                  = gtk_spin_button_get_value_as_int (window->spin_generations);
00931
                input->mutation_ratio
00932
                   gtk_spin_button_get_value (window->spin_mutation);
00933
                input->reproduction_ratio
00934
                  = gtk_spin_button_get_value (window->spin_reproduction);
00935
                input->adaptation ratio
00936
                   = gtk_spin_button_get_value (window->spin_adaptation);
00937
00938
            input->norm = window_get_norm ();
00939
            input->p = gtk_spin_button_get_value (window->spin_p);
00940
            input->threshold = gtk_spin_button_get_value (window->spin_threshold);
00941
00942
            // Saving the XML file
00943
            buffer = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
00944
            input_save (buffer);
00945
00946
            // Closing and freeing memory
00947
            g_free (buffer);
00948
        gtk_window_destroy (GTK_WINDOW (dlg));
00949
00950 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_save_close: end\n");
00952 #endif
00953 }
```

Here is the call graph for this function:



5.11.3.7 dialog_simulator()

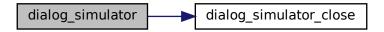
```
static void dialog_simulator ( ) [static]
```

Function to open a dialog to save the simulator file.

Definition at line 2197 of file interface.c.

```
02198 {
02199
         GtkFileChooserDialog *dlg;
02200 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_simulator: start\n");
02201
02202 #endif
        dlg = (GtkFileChooserDialog *)
02203
          gtk_file_chooser_dialog_new (_("Open simulator file"),
02205
02206
                                             GTK_FILE_CHOOSER_ACTION_OPEN,
        _("_Cancel"), GTK_RESPONSE_CANCEL,
_("_Open"), GTK_RESPONSE_ACCEPT, NULL);
g_signal_connect (dlg, "response", G_CALLBACK (dialog_simulator_close), NULL);
02207
02208
02209
02210
         gtk_window_present (GTK_WINDOW (dlg));
02211 #if DEBUG_INTERFACE
02212
        fprintf (stderr, "dialog_simulator: end\n");
02213 #endif
02214 }
```

Here is the call graph for this function:



5.11.3.8 dialog_simulator_close()

Function to save the close the simulator file dialog.

Parameters

dlg	GtkFileChooserDialog dialog.
response <i>←</i> _id	Response identifier.

Definition at line 2166 of file interface.c.

```
02169 {
          GFile *file1, *file2;
char *buffer1, *buffer2;
02170
02171
02172 #if DEBUG_INTERFACE
02173
          fprintf (stderr, "dialog_simulator_close: start\n");
02174 #endif
02175
         if (response_id == GTK_RESPONSE_OK)
02176
02177
               buffer1 = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
02178
               file1 = g_file_new_for_path (buffer1);
file2 = g_file_new_for_path (input->directory);
buffer2 = g_file_get_relative_path (file2, file1);
02179
02180
02181
               input->simulator = g_strdup (buffer2);
02182
               g_free (buffer2);
               g_object_unref (file2);
g_object_unref (file1);
02183
02184
02185
               g_free (buffer1);
02186
```

```
02187    gtk_window_destroy (GTK_WINDOW (dlg));
02188  #if DEBUG_INTERFACE
02189    fprintf (stderr, "dialog_simulator_close: end\n");
02190  #endif
02191 }
```

5.11.3.9 input_save()

Function to save the input file.

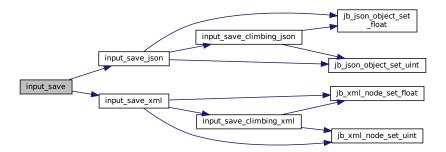
Parameters

filename Input file name.

Definition at line 590 of file interface.c.

```
00591 {
00592
        xmlDoc *doc:
00593
        JsonGenerator *generator;
00594
00595 #if DEBUG_INTERFACE
00596 fprintf (stderr, "input_save: start\n");
00597 #endif
00598
00599
        // Getting the input file directory
00600
        input->name = g_path_get_basename (filename);
00601
        input->directory = g_path_get_dirname (filename);
00602
00603
        if (input->type == INPUT_TYPE_XML)
00604
         {
            // Opening the input file
00605
            doc = xmlNewDoc ((const xmlChar *) "1.0");
input_save_xml (doc);
00606
00607
00608
00609
            \ensuremath{//} Saving the XML file
00610
            xmlSaveFormatFile (filename, doc, 1);
00611
00612
             // Freeing memory
00613
            xmlFreeDoc (doc);
00614
00615
        else
00616
         {
            // Opening the input file
00617
            generator = json_generator_new ();
00618
             json_generator_set_pretty (generator, TRUE);
00619
00620
            input_save_json (generator);
00621
            // Saving the JSON file
json_generator_to_file (generator, filename, NULL);
00622
00623
00624
00625
             // Freeing memory
00626
            g_object_unref (generator);
00627
00628
00629 #if DEBUG_INTERFACE
00630 fprintf (stderr, "input_save: end\n");
00631 #endif
00632 }
```

Here is the call graph for this function:



5.11.3.10 input save climbing json()

Function to save the hill climbing method data in a JSON node.

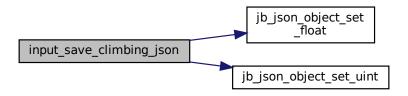
Parameters

```
node JSON node.
```

Definition at line 206 of file interface.c.

```
00208
         JsonObject *object;
00219 #if DEBUG_INTERFACE
00210 fprintf (stderr, "input_save_climbing_json: start\n");
00211 #endif
00212
        object = json_node_get_object (node);
00213
         if (input->nsteps)
00214
00215
              jb_json_object_set_uint (object, LABEL_NSTEPS, input->nsteps);
00216
             if (input->relaxation != DEFAULT_RELAXATION)
             jb_json_object_set_float (object, LABEL_RELAXATION, input->relaxation);
switch (input->climbing)
00217
00218
00219
               {
00220
               case CLIMBING_METHOD_COORDINATES:
00221
                 json_object_set_string_member (object, LABEL_CLIMBING,
00222
                                                      LABEL_COORDINATES);
00223
00224
                default:
                  json_object_set_string_member (object, LABEL_CLIMBING, LABEL_RANDOM);
jb_json_object_set_uint (object, LABEL_NESTIMATES, input->nestimates);
00225
00226
00227
00228
00229 #if DEBUG_INTERFACE
00230 fprintf (stderr, "input_save_climbing_json: end\n");
00231 #endif
00232 }
```

Here is the call graph for this function:



5.11.3.11 input save climbing xml()

Function to save the hill climbing method data in a XML node.

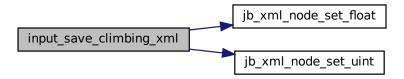
Parameters

```
node XML node.
```

Definition at line 172 of file interface.c.

```
00173 {
00174 #if DEBUG_INTERFACE
00175 fprintf (stderr, "input_save_climbing_xml: start\n");
00176 #endif
00177
      if (input->nsteps)
00178
00179
          jb_xml_node_set_uint (node, (const xmlChar *) LABEL_NSTEPS,
         input->nsteps);
if (input->relaxation != DEFAULT_RELAXATION)
00180
00181
           00182
00184
         switch (input->climbing)
00185
           case CLIMBING_METHOD_COORDINATES:
00186
            00187
00188
00189
             break;
00190
00191
             xmlSetProp (node, (const xmlChar *) LABEL_CLIMBING,
             00192
00193
00194
00195
00196
00197 #if DEBUG_INTERFACE
00198 fprintf (stderr, "input_save_climbing_xml: end\n");
00190 #endif
00200 }
```

Here is the call graph for this function:



5.11.3.12 input_save_json()

Function to save the input file in JSON format.

Parameters

```
generator JsonGenerator struct.
```

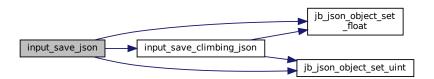
Definition at line 418 of file interface.c.

```
00419 {
00420
        unsigned int i, j;
00421
        char *buffer;
00422
        JsonNode *node, *child;
00423
        JsonObject *object;
00424
        JsonArray *array;
        GFile *file, *file2;
00425
00426
00427 #if DEBUG_INTERFACE
00428 fprintf (stderr, "input_save_json: start\n");
00429 #endif
00430
00431
        // Setting root JSON node
00432
        object = json_object_new ();
00433
        node = json_node_new (JSON_NODE_OBJECT);
00434
        json_node_set_object (node, object);
00435
        json_generator_set_root (generator, node);
00436
00437
        // Adding properties to the root JSON node
00438
        if (strcmp (input->result, result_name))
00439
          json_object_set_string_member (object, LABEL_RESULT_FILE, input->result);
00440
           (strcmp (input->variables, variables_name))
          json_object_set_string_member (object, LABEL_VARIABLES_FILE,
00441
00442
                                           input->variables);
        file = g_file_new_for_path (input->directory);
00443
        file2 = g_file_new_for_path (input->simulator);
buffer = g_file_get_relative_path (file, file2);
00444
00445
        g_object_unref (file2);
00446
00447
         json_object_set_string_member (object, LABEL_SIMULATOR, buffer);
00448
        g_free (buffer);
00449
        if (input->evaluator)
00450
          {
00451
            file2 = g_file_new_for_path (input->evaluator);
00452
            buffer = g_file_get_relative_path (file, file2);
             g_object_unref (file2);
00453
00454
             if (strlen (buffer))
            json_object_set_string_member (object, LABEL_EVALUATOR, buffer);
g_free (buffer);
00455
00456
```

```
00458
           if (input->seed != DEFAULT_RANDOM_SEED)
             jb_json_object_set_uint (object, LABEL_SEED, input->seed);
00459
00460
00461
           // Setting the algorithm
00462
          buffer = (char *) q_slice_alloc (64);
          switch (input->algorithm)
00463
00464
00465
             case ALGORITHM_MONTE_CARLO:
                00466
00467
                snprintf (buffer, 64, "%u", input->nsimulations);
00468
                json_object_set_string_member (object, LABEL_NSIMULATIONS, buffer);
snprintf (buffer, 64, "%u", input->niterations);
00469
00470
00471
                json_object_set_string_member (object, LABEL_NITERATIONS, buffer);
                sprintf (buffer, 64, "%.31g", input->tolerance);
json_object_set_string_member (object, LABEL_TOLERANCE, buffer);
snprintf (buffer, 64, "%u", input->nbest);
json_object_set_string_member (object, LABEL_NBEST, buffer);
00472
00473
00474
00476
                input_save_climbing_json (node);
00477
00478
             case ALGORITHM SWEEP:
                json_object_set_string_member (object, LABEL_ALGORITHM, LABEL_SWEEP);
snprintf (buffer, 64, "%u", input->niterations);
00479
00480
                json_object_set_string_member (object, LABEL_NITERATIONS, buffer);
snprintf (buffer, 64, "%.31g", input->tolerance);
json_object_set_string_member (object, LABEL_TOLERANCE, buffer);
00481
00483
00484
                snprintf (buffer, 64, "%u", input->nbest);
00485
                json_object_set_string_member (object, LABEL_NBEST, buffer);
00486
                input_save_climbing_json (node);
00487
                break:
00488
             case ALGORITHM_ORTHOGONAL:
00489
                json_object_set_string_member (object, LABEL_ALGORITHM, LABEL_ORTHOGONAL);
00490
                snprintf (buffer, 64, "%u", input->niterations);
                json_object_set_string_member (object, LABEL_NITERATIONS, buffer);
snprintf (buffer, 64, "%.31g", input->tolerance);
json_object_set_string_member (object, LABEL_TOLERANCE, buffer);
snprintf (buffer, 64, "%u", input->nbest);
00491
00492
00493
00495
                json_object_set_string_member (object, LABEL_NBEST, buffer);
00496
                input_save_climbing_json (node);
00497
                break:
00498
             default:
                json_object_set_string_member (object, LABEL_ALGORITHM, LABEL_GENETIC);
snprintf (buffer, 64, "%u", input->nsimulations);
00499
00500
                json_object_set_string_member (object, LABEL_NPOPULATION, buffer);
00501
00502
                snprintf (buffer, 64, "%u", input->niterations);
                snprintr (buffer, 64, "%1", input->niterations);
json_object_set_string_member (object, LABEL_NGENERATIONS, buffer);
snprintf (buffer, 64, "%.31g", input->mutation_ratio);
json_object_set_string_member (object, LABEL_MUTATION, buffer);
snprintf (buffer, 64, "%.31g", input->reproduction_ratio);
json_object_set_string_member (object, LABEL_REPRODUCTION, buffer);
snprintf (buffer, 64, "%.31g", input->adaptation_ratio);
json_object_set_string_member (object, LABEL_ADAPTATION, buffer);
00503
00504
00505
00506
00508
00509
00510
                break;
00511
00512
          g slice freel (64, buffer);
              (input->threshold != 0.)
00513
00514
             jb_json_object_set_float (object, LABEL_THRESHOLD, input->threshold);
00515
00516
           // Setting the experimental data
00517
          array = json_array_new ();
for (i = 0; i < input->nexperiments; ++i)
00518
00519
00520
                child = json_node_new (JSON_NODE_OBJECT);
                object = json_node_get_object (child);
00521
                json_object_set_string_member (object, LABEL_NAME,
00522
00523
                input->experiment[i].name);
if (input->experiment[i].weight != 1.)
00524
00525
                  jb_json_object_set_float (object, LABEL_WEIGHT,
                                                      input->experiment[i].weight);
00526
00527
                for (j = 0; j < input->experiment->ninputs; ++j)
00528
                   json_object_set_string_member (object, stencil[j],
00529
                                                             input->experiment[i].stencil[j]);
00530
                json_array_add_element (array, child);
00531
00532
           json_object_set_array_member (object, LABEL_EXPERIMENTS, array);
00533
00534
          // Setting the variables data
          array = json_array_new ();
for (i = 0; i < input->nvariables; ++i)
00535
00536
00537
                child = json_node_new (JSON_NODE_OBJECT);
object = json_node_get_object (child);
00539
00540
                json_object_set_string_member (object, LABEL_NAME,
00541
                                                           input->variable[i].name);
                00542
00543
```

```
if (input->variable[i].rangeminabs != -G_MAXDOUBLE)
00545
              jb_json_object_set_float (object, LABEL_ABSOLUTE_MINIMUM,
                                         input->variable[i].rangeminabs);
00546
            jb_json_object_set_float (object, LABEL_MAXIMUM,
00547
00548
                                       input->variable[i].rangemax);
00549
            if (input->variable[i].rangemaxabs != G_MAXDOUBLE)
00550
              jb_json_object_set_float (object, LABEL_ABSOLUTE_MAXIMUM,
00551
                                         input->variable[i].rangemaxabs);
00552
            if (input->variable[i].precision != DEFAULT_PRECISION)
00553
              jb_json_object_set_uint (object, LABEL_PRECISION,
                                        input->variable[i].precision);
00554
            if (input->algorithm == ALGORITHM_SWEEP
00555
                || input->algorithm == ALGORITHM_ORTHOGONAL)
00556
00557
              jb_json_object_set_uint (object, LABEL_NSWEEPS,
00558
                                       input->variable[i].nsweeps);
00559
            else if (input->algorithm == ALGORITHM_GENETIC)
            jb_json_object_set_uint (object, LABEL_NBITS, input->variable[i].nbits);
if (input->nsteps)
00560
00561
              jb_json_object_set_float (object, LABEL_STEP, input->variable[i].step);
00562
00563
            json_array_add_element (array, child);
00564
00565
        json_object_set_array_member (object, LABEL_VARIABLES, array);
00566
00567
        // Saving the error norm
00568
        switch (input->norm)
00569
00570
          case ERROR_NORM_MAXIMUM:
00571
            json_object_set_string_member (object, LABEL_NORM, LABEL_MAXIMUM);
00572
00573
          case ERROR NORM P:
            json_object_set_string_member (object, LABEL_NORM, LABEL_P);
00574
00575
            jb_json_object_set_float (object, LABEL_P, input->p);
00576
00577
          case ERROR_NORM_TAXICAB:
00578
            json_object_set_string_member (object, LABEL_NORM, LABEL_TAXICAB);
00579
00580
00581 #if DEBUG_INTERFACE
       fprintf (stderr, "input_save_json: end\n");
00583 #endif
00584 }
```

Here is the call graph for this function:



5.11.3.13 input_save_xml()

Function to save the input file in XML format.

Parameters

doc xmlDoc struct.

Definition at line 238 of file interface.c.

```
00240
        unsigned int i, j;
00241
        char *buffer;
00242
        xmlNode *node, *child;
        GFile *file, *file2;
00243
00244
00245 #if DEBUG_INTERFACE
       fprintf (stderr, "input_save_xml: start\n");
00246
00247 #endif
00248
00249
        // Setting root XML node
00250
       node = xmlNewDocNode (doc, 0, (const xmlChar *) LABEL_OPTIMIZE, 0);
00251
        xmlDocSetRootElement (doc, node);
00252
00253
        // Adding properties to the root XML node
00254
       if (xmlStrcmp
          ((const xmlChar *) input->result, (const xmlChar *) result_name))
xmlSetProp (node, (const xmlChar *) LABEL_RESULT_FILE,
00255
00257
                      (xmlChar *) input->result);
00258
00259
            ((const xmlChar *) input->variables, (const xmlChar *) variables_name))
          00260
00261
00262
        file = q_file_new_for_path (input->directory);
        file2 = g_file_new_for_path (input->simulator);
00263
00264
                 g_file_get_relative_path (file, file2);
00265
        g_object_unref (file2);
        xmlSetProp (node, (const xmlChar *) LABEL_SIMULATOR, (xmlChar *) buffer);
00266
        g_free (buffer);
00267
00268
        if (input->evaluator)
00269
         {
00270
            file2 = g_file_new_for_path (input->evaluator);
00271
            buffer = g_file_get_relative_path (file, file2);
00272
            g_object_unref (file2);
00273
            if (xmlStrlen ((xmlChar *) buffer))
00274
             xmlSetProp (node, (const xmlChar *) LABEL_EVALUATOR,
                           (xmlChar *) buffer);
00276
            g_free (buffer);
00277
00278
        if (input->seed != DEFAULT_RANDOM_SEED)
          jb_xml_node_set_uint (node, (const xmlChar *) LABEL_SEED, input->seed);
00279
00280
00281
        // Setting the algorithm
        buffer = (char *) g_slice_alloc (64);
00282
00283
        switch (input->algorithm)
00284
          case ALGORITHM_MONTE_CARLO:
00285
            00286
00287
00289
            xmlSetProp (node, (const xmlChar *) LABEL_NSIMULATIONS,
00290
                         (xmlChar *) buffer);
            snprintf (buffer, 64, "%u", input->niterations);
xmlSetProp (node, (const xmlChar *) LABEL_NITERATIONS,
00291
00292
            (xmlChar *) buffer);
snprintf (buffer, 64, "%.31g", input->tolerance);
00293
00295
            xmlSetProp (node, (const xmlChar *) LABEL_TOLERANCE, (xmlChar *) buffer);
00296
            snprintf (buffer, 64, "%u", input->nbest);
            xmlSetProp (node, (const xmlChar *) LABEL_NBEST, (xmlChar *) buffer);
00297
            input_save_climbing_xml (node);
00298
00299
            break;
00300
          case ALGORITHM_SWEEP:
            xmlSetProp (node, (const xmlChar *) LABEL_ALGORITHM,
00301
00302
                         (const xmlChar *) LABEL_SWEEP);
00303
            snprintf (buffer, 64, "%u", input->niterations);
            xmlSetProp (node, (const xmlChar *) LABEL_NITERATIONS,
00304
            (xmlChar *) buffer);
snprintf (buffer, 64, "%.31g", input->tolerance);
00305
00306
            xmlSetProp (node, (const xmlChar *) LABEL_TOLERANCE, (xmlChar *) buffer);
00307
00308
            snprintf (buffer, 64, "%u", input->nbest);
00309
            xmlSetProp (node, (const xmlChar *) LABEL_NBEST, (xmlChar *) buffer);
00310
            input_save_climbing_xml (node);
00311
            break;
          case ALGORITHM_ORTHOGONAL:
00312
            xmlSetProp (node, (const xmlChar *) LABEL_ALGORITHM,
00313
00314
                         (const xmlChar *) LABEL_ORTHOGONAL);
00315
            snprintf (buffer, 64, "%u", input->niterations);
00316
            xmlSetProp (node, (const xmlChar *) LABEL_NITERATIONS,
                        (xmlChar *) buffer);
00317
            snprintf (buffer, 64, "%.31g", input->tolerance);
00318
            xmlSetProp (node, (const xmlChar *) LABEL_TOLERANCE, (xmlChar *) buffer);
snprintf (buffer, 64, "%u", input->nbest);
00320
00321
            xmlSetProp (node, (const xmlChar *) LABEL_NBEST, (xmlChar *) buffer);
00322
            input_save_climbing_xml (node);
00323
            break;
00324
         default:
```

```
xmlSetProp (node, (const xmlChar *) LABEL_ALGORITHM,
            (const xmlChar *) LABEL_GENETIC);
snprintf (buffer, 64, "%u", input->nsimulations);
00326
00327
00328
            xmlSetProp (node, (const xmlChar *) LABEL_NPOPULATION,
            (xmlChar *) buffer);
snprintf (buffer, 64, "%u", input->niterations);
xmlSetProp (node, (const xmlChar *) LABEL_NGENERATIONS,
00329
00330
00332
                         (xmlChar *) buffer);
00333
            snprintf (buffer, 64, "%.31g", input->mutation_ratio);
            xmlSetProp (node, (const xmlChar *) LABEL_MUTATION, (xmlChar *) buffer);
snprintf (buffer, 64, "%.31g", input->reproduction_ratio);
00334
00335
            xmlSetProp (node, (const xmlChar *) LABEL_REPRODUCTION,
00336
            (xmlChar *) buffer);
snprintf (buffer, 64, "%.31g", input->adaptation_ratio);
00337
00338
00339
            xmlSetProp (node, (const xmlChar *) LABEL_ADAPTATION, (xmlChar *) buffer);
00340
00341
00342
        g slice freel (64, buffer);
        if (input->threshold != 0.)
00343
00344
          jb_xml_node_set_float (node, (const xmlChar *) LABEL_THRESHOLD,
00345
                                  input->threshold);
00346
00347
        \ensuremath{//} Setting the experimental data
00348
        for (i = 0; i < input->nexperiments; ++i)
00349
00350
            child = xmlNewChild (node, 0, (const xmlChar *) LABEL_EXPERIMENT, 0);
            xmlSetProp (child, (const xmlChar *) LABEL_NAME,
00351
00352
                         (xmlChar *) input->experiment[i].name);
00353
            if (input->experiment[i].weight != 1.)
              jb_xml_node_set_float (child, (const xmlChar *) LABEL_WEIGHT,
00354
00355
                                      input->experiment[i].weight);
00356
            for (j = 0; j < input->experiment->ninputs; ++j)
00357
             xmlSetProp (child, (const xmlChar *) stencil[j],
00358
                          (xmlChar *) input->experiment[i].stencil[j]);
00359
00360
00361
        // Setting the variables data
        for (i = 0; i < input->nvariables; ++i)
00362
00363
00364
            child = xmlNewChild (node, 0, (const xmlChar *) LABEL_VARIABLE, 0);
            00365
00366
            jb_xml_node_set_float (child, (const xmlChar *) LABEL_MINIMUM,
00367
                                    input->variable[i].rangemin);
00368
            if (input->variable[i].rangeminabs != -G_MAXDOUBLE)
00369
00370
              jb_xml_node_set_float (child, (const xmlChar *) LABEL_ABSOLUTE_MINIMUM,
00371
                                      input->variable[i].rangeminabs);
00372
            jb_xml_node_set_float (child, (const xmlChar *) LABEL_MAXIMUM,
                                    input->variable[i].rangemax);
00373
00374
            if (input->variable[i].rangemaxabs != G_MAXDOUBLE)
00375
              jb_xml_node_set_float (child, (const xmlChar *) LABEL_ABSOLUTE_MAXIMUM,
00376
                                      input->variable[i].rangemaxabs);
00377
            if (input->variable[i].precision != DEFAULT_PRECISION)
              00378
00379
00380
            if (input->algorithm == ALGORITHM_SWEEP
                || input->algorithm == ALGORITHM_ORTHOGONAL)
00382
              jb_xml_node_set_uint (child, (const xmlChar *) LABEL_NSWEEPS,
00383
                                     input->variable[i].nsweeps);
            else if (input->algorithm == ALGORITHM_GENETIC)
00384
              jb_xml_node_set_uint (child, (const xmlChar *) LABEL_NBITS,
00385
00386
                                     input->variable[i].nbits);
00387
            if (input->nsteps)
00388
              jb_xml_node_set_float (child, (const xmlChar *) LABEL_STEP,
00389
                                      input->variable[i].step);
00390
00391
00392
        // Saving the error norm
00393
        switch (input->norm)
00394
00395
          case ERROR_NORM_MAXIMUM:
00396
            xmlSetProp (node, (const xmlChar *) LABEL_NORM,
00397
                         (const xmlChar *) LABEL_MAXIMUM);
00398
            break:
00399
          case ERROR_NORM_P:
           xmlSetProp (node, (const xmlChar *) LABEL_NORM,
00400
00401
                         (const xmlChar *) LABEL_P);
00402
            jb_xml_node_set_float (node, (const xmlChar *) LABEL_P, input->p);
00403
            break;
          case ERROR NORM TAXICAB:
00404
           00405
00407
00408
00409 #if DEBUG_INTERFACE 00410 fprintf (stderr, "input_save: end\n");
00411 #endif
```

```
00412 }
```

Here is the call graph for this function:



5.11.3.14 options_new()

```
static void options_new ( ) [static]
```

Function to open the options dialog.

Definition at line 662 of file interface.c.

```
00663
00664 #if DEBUG_INTERFACE
        fprintf (stderr, "options_new: start\n");
00665
00666 #endif
        options->label_seed = (GtkLabel *)
00668
           gtk_label_new (_("Pseudo-random numbers generator seed"));
00669
        options->spin_seed = (GtkSpinButton *)
00670
           gtk_spin_button_new_with_range (0., (gdouble) G_MAXULONG, 1.);
00671
        gtk_widget_set_tooltip_text
  (GTK_WIDGET (options->spin_seed),
00672
00673
            _("Seed to init the pseudo-random numbers generator"));
00674
        gtk_spin_button_set_value (options->spin_seed, (gdouble) input->seed);
00675
        options->label_threads = (GtkLabel *)
        gtk_label_new (_("Threads number for the stochastic algorithm"));
options->spin_threads
00676
00677
00678
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 64., 1.);
        gtk_widget_set_tooltip_text
  (GTK_WIDGET (options->spin_threads),
00679
00680
00681
            \_("Number of threads to perform the calibration/optimization for "
00682
              "the stochastic algorithm"));
00683
        gtk_spin_button_set_value (options->spin_threads, (gdouble) nthreads);
options->label_climbing = (GtkLabel *)
00684
        gtk_label_new (_("Threads number for the hill climbing method"));
options->spin_climbing =
00685
00686
00687
           (GtkSpinButton *) gtk_spin_button_new_with_range (1., 64., 1.);
00688
        gtk_widget_set_tooltip_text
00689
           (GTK_WIDGET (options->spin_climbing),
            ("Number of threads to perform the calibration/optimization for the "hill climbing method"));
00690
00691
00692
        gtk_spin_button_set_value (options->spin_climbing,
00693
                                       (gdouble) nthreads_climbing);
00694
        options->grid = (GtkGrid *) gtk_grid_new ();
        gtk_grid_attach (options->grid, GTK_WIDGET (options->label_seed), 0, 0, 1, 1);
gtk_grid_attach (options->grid, GTK_WIDGET (options->spin_seed), 1, 0, 1, 1);
00695
00696
        gtk_grid_attach (options->grid, GTK_WIDGET (options->label_threads),
00697
00698
                            0, 1, 1, 1);
00699
        gtk_grid_attach (options->grid, GTK_WIDGET (options->spin_threads),
00700
                            1, 1, 1, 1);
00701
        gtk_grid_attach (options->grid, GTK_WIDGET (options->label_climbing), 0, 2, 1,
00702
                            1);
00703
        gtk_grid_attach (options->grid, GTK_WIDGET (options->spin_climbing), 1, 2, 1,
00704
                            1);
00705 #if !GTK4
00706
        gtk_widget_show_all (GTK_WIDGET (options->grid));
00707 #else
00708
        gtk_widget_show (GTK_WIDGET (options->grid));
00709 #endif
00710
        options->dialog = (GtkDialog *)
00711
           gtk_dialog_new_with_buttons (_("Options"),
```

```
00712
                                          window->window,
00713
                                          GTK_DIALOG_MODAL,
                                         _("_OK"), GTK_RESPONSE_OK,
_("_Cancel"), GTK_RESPONSE_CANCEL, NULL);
00714
00715
00716
        gtk_box_append (GTK_BOX (gtk_dialog_get_content_area (options->dialog)),
00717
                         GTK_WIDGET (options->grid));
00718
        g_signal_connect (options->dialog, "response",
00719
                            G_CALLBACK (dialog_options_close), NULL);
00720
        gtk_window_present (GTK_WINDOW (options->dialog));
00721 #if DEBUG INTERFACE
00722 fprintf (stderr, "options_new: end\n");
00723 #endif
00724 }
```

5.11.3.15 running new()

```
static void running_new ( ) [inline], [static]
```

Function to open the running dialog.

Definition at line 730 of file interface.c.

```
00731
00732 #if DEBUG_INTERFACE
        fprintf (stderr, "running_new: start\n");
00733
00734 #endif
00735
        running->label = (GtkLabel *) gtk_label_new (_("Calculating ..."));
00736
         running->spinner = (GtkSpinner *) gtk_spinner_new ();
        running->grid = (GtkGrid *) gtk_grid_new ();
gtk_grid_attach (running->grid, GTK_WIDGET (running->label), 0, 0, 1, 1);
00737
00738
        gtk_grid_attach (running->grid, GTK_WIDGET (running->spinner), 0, 1, 1, 1);
running->dialog = (GtkDialog *)
00739
00740
00741
           gtk_dialog_new_with_buttons (_("Calculating"),
00742
                                             window->window, GTK_DIALOG_MODAL, NULL, NULL);
00743
        gtk_window_set_child (GTK_WINDOW
00744
                                  (gtk_dialog_get_content_area (running->dialog)),
GTK_WIDGET (running->grid));
00745
00746
        gtk_spinner_start (running->spinner);
00747 #if !GTK4
00748
        gtk_widget_show_all (GTK_WIDGET (running->dialog));
00749 #else
00750 gtk_widget_show (GTK_WIDGET (running->dialog));
00751 #endif
00752 #if DEBUG_INTERFACE
00753 fprintf (stderr,
        fprintf (stderr, "running_new: end\n");
00754 #endif
00755 }
```

5.11.3.16 window about()

```
static void window_about ( ) [static]
```

Function to show an about dialog.

Definition at line 1095 of file interface.c.

```
01096 {
01097
        static const gchar *authors[] = {
01098
          "Javier Burguete Tolosa <jburguete@eead.csic.es>",
          "Borja Latorre Garcés <borja.latorre@csic.es>",
01099
01100
         NULL
01101
01102 #if DEBUG_INTERFACE
01103
       fprintf (stderr, "window_about: start\n");
01104 #endif
01105
       gtk_show_about_dialog
          (window->window,
01106
01107
           "program_name", "MPCOTool",
01108
           "comments",
01109
           _("The Multi-Purposes Calibration and Optimization Tool.\n"
```

```
"A software to perform calibrations or optimizations of empirical "
               "parameters"),
01111
             "authors", authors,
01112
             "translator-credits",
01113
             "Javier Burguete Tolosa <jburguete@eead.csic.es> "
01114
             "(english, french and spanish)\n"
01115
             "Uğur Çayoğlu (german)",
01116
             "version", "4.4.6",
"copyright", "Copyright 2012-2023 Javier Burguete Tolosa",
01117
01118
             "logo", window->logo,
01119
01120 "website", "https://github.com/jburguete/mpcotool",
01121 "license-type", GTK_LICENSE_BSD, NULL);
01122 #if DEBUG_INTERFACE
01123 fprintf (stderr, "window_about: end\n");
01124 #endif
01125 }
```

5.11.3.17 window_add_experiment()

```
static void window_add_experiment ( ) [static]
```

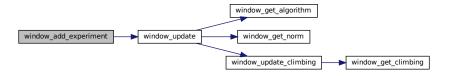
Function to add an experiment in the main window.

```
Definition at line 1410 of file interface.c.
```

01411

```
unsigned int i, j;
01412
01413 #if DEBUG INTERFACE
01414
         fprintf (stderr, "window_add_experiment: start\n");
01415 #endif
        i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01417
         g_signal_handler_block (window->combo_experiment, window->id_experiment);
         gtk_combo_box_text_insert_text
01418
         (window->combo_experiment, i, input->experiment[i].name);
g_signal_handler_unblock (window->combo_experiment, window->id_experiment);
01419
01420
         input->experiment = (Experiment *) g_realloc
01422
           (input->experiment, (input->nexperiments + 1) * sizeof (Experiment));
         for (j = input->nexperiments - 1; j > i; --j)
  memcpy (input->experiment + j + 1, input->experiment + j,
01423
01424
                    sizeof (Experiment));
01425
         input->experiment[j + 1].weight = input->experiment[j].weight;
input->experiment[j + 1].ninputs = input->experiment[j].ninputs;
01426
01427
01428
            (input->type == INPUT_TYPE_XML)
01429
01430
              input->experiment[j + 1].name
              = (char *) xmlStrdup ((xmlChar *) input->experiment[j].name);
for (j = 0; j < input->experiment->ninputs; ++j)
  input->experiment[i + 1].stencil[j]
01431
01432
01433
01434
                   = (char *) xmlStrdup ((xmlChar *) input->experiment[i].stencil[j]);
01435
01436
         else
01437
              input->experiment[j + 1].name = g_strdup (input->experiment[j].name);
01438
              for (j = 0; j < input->experiment->ninputs; ++j)
01439
                input->experiment[i + 1].stencil[j]
01441
                   = g_strdup (input->experiment[i].stencil[j]);
01442
01443
         ++input->nexperiments;
         for (j = 0; j < input->experiment->ninputs; ++j)
01444
01445
           g_signal_handler_block (window->button_template[j], window->id_input[j]);
         gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), i + 1);
01446
01447
         for (j = 0; j < input->experiment->ninputs; ++j)
01448
           g_signal_handler_unblock (window->button_template[j], window->id_input[j]);
         window_update ();
01449
01450 #if DEBUG_INTERFACE
        fprintf (stderr, "window_add_experiment: end\n");
01451
01452 #endif
01453 }
```

Here is the call graph for this function:



5.11.3.18 window add variable()

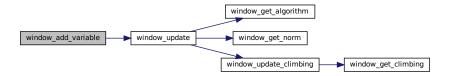
```
static void window_add_variable ( ) [static]
```

Function to add a variable in the main window.

Definition at line 1745 of file interface.c.

```
01746 {
01747
         unsigned int i, j;
01748 #if DEBUG_INTERFACE
01749
         fprintf (stderr, "window_add_variable: start\n");
01750 #endif
        i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01751
01752
         g_signal_handler_block (window->combo_variable, window->id_variable);
01753
         gtk_combo_box_text_insert_text (window->combo_variable, i,
01754
                                               input->variable[i].name);
01755
         g_signal_handler_unblock (window->combo_variable, window->id_variable);
01756
         input->variable = (Variable *) g_realloc
        (input->variable (input->nvariables + 1) * sizeof (Variable));
for (j = input->nvariables - 1; j > i; --j)
  memcpy (input->variable + j + 1, input->variable + j, sizeof (Variable));
memcpy (input->variable + j + 1, input->variable + j, sizeof (Variable));
01757
01758
01759
01760
01761
         if (input->type == INPUT_TYPE_XML)
01762
           input->variable[j + 1].name
01763
              = (char *) xmlStrdup ((xmlChar *) input->variable[j].name);
01764
01765
           input->variable[j + 1].name = g_strdup (input->variable[j].name);
01766
        ++input->nvariables;
01767
         g_signal_handler_block (window->entry_variable, window->id_variable_label);
01768
         gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), i + 1);
01769
         g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
01770 window_update ();
01771 #if DEBUG_INTERFACE
        fprintf (stderr, "window_add_variable: end\n");
01773 #endif
01774 }
```

Here is the call graph for this function:



5.11.3.19 window_get_algorithm()

```
static unsigned int window_get_algorithm ( ) [static]
```

Function to get the stochastic algorithm number.

Returns

Stochastic algorithm number.

Definition at line 763 of file interface.c.

```
00764 {
00765 unsigned int i;
00766 #if DEBUG_INTERFACE
00767 fprintf (stderr, "window_get_algorithm: start\n");
00768 #endif
00769 i = jbw_array_buttons_get_active (window->button_algorithm, NALGORITHMS);
00770 #if DEBUG_INTERFACE
00771 fprintf (stderr, "window_get_algorithm: %u\n", i);
00772 fprintf (stderr, "window_get_algorithm: end\n");
00773 #endif
00774 return i;
```

5.11.3.20 window_get_climbing()

```
static unsigned int window_get_climbing ( ) [static]
```

Function to get the hill climbing method number.

Returns

Hill climbing method number.

Definition at line 783 of file interface.c.

```
00784 {
00785    unsigned int i;
00786 #if DEBUG_INTERFACE
00787    fprintf (stderr, "window_get_climbing: start\n");
00788 #endif
00789    i = jbw_array_buttons_get_active (window->button_climbing, NCLIMBINGS);
00790 #if DEBUG_INTERFACE
00791    fprintf (stderr, "window_get_climbing: %u\n", i);
00792    fprintf (stderr, "window_get_climbing: end\n");
00793 #endif
00794    return i;
00795 }
```

5.11.3.21 window_get_norm()

```
static unsigned int window_get_norm ( ) [static]
```

Function to get the norm method number.

Returns

Norm method number.

Definition at line 803 of file interface.c.

```
00804 {
00805    unsigned int i;
00806 #if DEBUG_INTERFACE
00807    fprintf (stderr, "window_get_norm: start\n");
00808 #endif
00809    i = jbw_array_buttons_get_active (window->button_norm, NNORMS);
00810 #if DEBUG_INTERFACE
00811    fprintf (stderr, "window_get_norm: %u\n", i);
00812    fprintf (stderr, "window_get_norm: end\n");
00813 #endif
00814    return i;
00815 }
```

5.11.3.22 window_help()

```
static void window_help ( ) [static]
```

Function to show a help dialog.

Definition at line 1067 of file interface.c.

```
01068
01069
       char *buffer, *buffer2;
01070 #if DEBUG_INTERFACE
       fprintf (stderr, "window_help: start\n");
01071
01072 #endif
      01073
01074
01075
      buffer = g_filename_to_uri (buffer2, NULL, NULL);
01076
       g_free (buffer2);
01077 #if
01078
      gtk_show_uri (window->window, buffer, GDK_CURRENT_TIME);
01079 #else
01080
      gtk_show_uri_on_window (window->window, buffer, GDK_CURRENT_TIME, NULL);
01081 #endif
01082 #if DEBUG_INTERFACE
01083
      fprintf (stderr, "window_help: uri=%s\n", buffer);
01084 #endif
01085
      g_free (buffer);
01086 #if DEBUG_INTERFACE
      fprintf (stderr, "window_help: end\n");
01087
01088 #endif
01089 }
```

5.11.3.23 window_inputs_experiment()

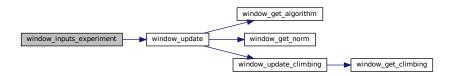
```
static void window_inputs_experiment ( ) [static]
```

Function to update the experiment input templates number in the main window.

Definition at line 1543 of file interface.c.

```
01544 {
01545
       unsigned int j;
01546 #if DEBUG_INTERFACE
01547
       fprintf (stderr, "window_inputs_experiment: start\n");
01548 #endif
01549
       j = input->experiment->ninputs - 1;
       if (j && !gtk_check_button_get_active (window->check_template[j]))
01550
01551
          --input->experiment->ninputs:
       if (input->experiment->ninputs < MAX_NINPUTS</pre>
01552
01553
            && gtk_check_button_get_active (window->check_template[j]))
01554
          ++input->experiment->ninputs;
01555
       window_update ();
01556 #if DEBUG_INTERFACE
       fprintf (stderr, "window_inputs_experiment: end\n");
01557
01558 #endif
01559 }
```

Here is the call graph for this function:



5.11.3.24 window_label_variable()

```
static void window_label_variable ( ) [static]
```

Function to set the variable label in the main window.

Definition at line 1780 of file interface.c.

```
01781
01782
         unsigned int i;
01783
         const char *buffer;
01784 #if DEBUG_INTERFACE
         fprintf (stderr, "window_label_variable: start\n");
01785
01786 #endif
         i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
buffer = gtk_entry_get_text (window->entry_variable);
01787
01788
         g_signal_handler_block (window->combo_variable, window->id_variable);
01790
         gtk_combo_box_text_remove (window->combo_variable, i);
01791
         gtk_combo_box_text_insert_text (window->combo_variable, i, buffer);
         gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), i);
g_signal_handler_unblock (window->combo_variable, window->id_variable);
01792
01793
01794 #if DEBUG INTERFACE
01795 fprintf (stderr, "window_label_variable: end\n");
01796 #endif
01797 }
```

5.11.3.25 window name experiment()

```
static void window_name_experiment ( ) [static]
```

Function to set the experiment name in the main window.

Definition at line 1490 of file interface.c.

```
01491 {
01492
       GtkFileChooserDialog *dlg;
       GMainLoop *loop;
const char *buffer;
01493
01494
01495
       unsigned int i;
01496 #if DEBUG_INTERFACE
01497
       fprintf (stderr, "window_name_experiment: start\n");
01498 #endif
01499
      i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
       buffer = gtk_button_get_label (window->button_experiment);
01500
01501
       dlg = (GtkFileChooserDialog *)
01502
         gtk_file_chooser_dialog_new (_("Open experiment file"),
01503
                                     window->window,
01504
                                     GTK_FILE_CHOOSER_ACTION_OPEN,
                                      (" Cancel").
01505
                                     GTK_RESPONSE_CANCEL,
01506
01507
                                     _("_Open"), GTK_RESPONSE_OK, NULL);
01508
       gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (dlg), buffer);
       01509
01510
       {\tt gtk\_window\_present~(GTK\_WINDOW~(dlg));}
01511
       loop = q_main_loop_new (NULL, 0);
01512
       g_signal_connect_swapped (dlg, "destroy", G_CALLBACK (g_main_loop_quit),
01513
                                loop);
01515
       g_main_loop_run (loop);
01516
       g_main_loop_unref (loop);
01517 #if DEBUG_INTERFACE
       fprintf (stderr, "window_name_experiment: end\n");
01518
01519 #endif
01520 }
```

Here is the call graph for this function:

```
window_name_experiment dialog_name_experiment __close
```

5.11.3.26 window new()

Function to open the main window.

Parameters

application | GtkApplication struct.

Definition at line 2274 of file interface.c.

```
02276
        unsigned int i;
        char *buffer, *buffer2, buffer3[64];
const char *label_algorithm[NALGORITHMS] = {
02277
02278
02279
          "_Monte-Carlo", _("_Sweep"), _("_Genetic"), _("_Orthogonal")
02280
02281
        const char *tip_algorithm[NALGORITHMS] = {
02282
        _("Monte-Carlo brute force algorithm"),
         _("Sweep brute force algorithm"),
02283
         _("Genetic algorithm"),
02284
02285
          \_("Orthogonal sampling brute force algorithm"),
02286
02287
        const char *label_climbing[NCLIMBINGS] = {
02288
          _("_Coordinates climbing"), _("_Random climbing")
02289
        const char *tip_climbing[NCLIMBINGS] = {
02290
        _("Coordinates climbing estimate method"),
02291
02292
          _("Random climbing estimate method")
02293
02294
        const char *label_norm[NNORMS] = { "L2", "L", "Lp", "L1" };
02295
        const char *tip_norm[NNORMS] =
        _("Euclidean error norm (L2)"),
02296
         _("Maximum error norm (L)"),
02297
         _("P error norm (Lp)"),
02298
02299
          _("Taxicab error norm (L1)")
02300
       };
02301 #if !GTK4
       const char *close = "delete-event";
02302
02303 #else
02304
       const char *close = "close-request";
02305 #endif
02306
02307 #if DEBUG_INTERFACE
       fprintf (stderr, "window_new: start\n");
02308
02309 #endif
02310
02311
        // Creating the window
02312
       window->window = window_parent = main_window
02313
          = (GtkWindow *) gtk_application_window_new (application);
02314
       // Finish when closing the window
02315
       g_signal_connect_swapped (window->window, close,
02316
                                   G_CALLBACK (g_application_quit),
02318
                                   G_APPLICATION (application));
02319
02320
       // Setting the window title
        gtk_window_set_title (window->window, "MPCOTool");
02321
02322
02323
        // Creating the open button
02324
        window->button_open = (GtkButton *)
02325 #if !GTK4
02326
          gtk_button_new_from_icon_name ("document-open", GTK_ICON_SIZE_BUTTON);
02327 #else
         gtk_button_new_from_icon_name ("document-open");
02328
02329 #endif
02330
      gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_open),
       __("Open a case"));
g_signal_connect (window->button_open, "clicked", window_open, NULL);
02331
02332
02333
02334
       // Creating the save button
02335
       window->button_save = (GtkButton *)
02336 #if !GTK4
```

```
02337
          gtk_button_new_from_icon_name ("document-save", GTK_ICON_SIZE_BUTTON);
02338 #else
02339
          gtk_button_new_from_icon_name ("document-save");
02340 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_save),
02341
02342
                                        ("Save the case"));
        g_signal_connect (window->button_save, "clicked", (GCallback) window_save,
02343
02344
                            NULL);
02345
02346
        // Creating the run button
02347
        window->button_run = (GtkButton *)
02348 #if !GTK4
02349
          gtk button new from icon name ("system-run", GTK ICON SIZE BUTTON);
02350 #else
02351
          gtk_button_new_from_icon_name ("system-run");
02352 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_run),
02353
        _("Run the optimization"));
g_signal_connect (window->button_run, "clicked", window_run, NULL);
02354
02355
02356
02357
        // Creating the options button
02358
        window->button_options = (GtkButton *)
02359 #if !GTK4
          gtk_button_new_from_icon_name ("preferences-system", GTK_ICON_SIZE_BUTTON);
02360
02361 #else
02362
          gtk_button_new_from_icon_name ("preferences-system");
02363 #endif
02364
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_options),
        _("Edit the case"));
g_signal_connect (window->button_options, "clicked", options_new, NULL);
02365
02366
02367
02368
        // Creating the help button
02369
        window->button_help = (GtkButton *)
02370 #if !GTK4
02371
          gtk_button_new_from_icon_name ("help-browser", GTK_ICON_SIZE_BUTTON);
02372 #else
02373
          gtk button new from icon name ("help-browser");
02374 #endif
02375
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_help),
                                                                              ("Help"));
02376
        g_signal_connect (window->button_help, "clicked", window_help, NULL);
02377
02378
        // Creating the about button
        window->button_about = (GtkButton *)
02379
02380 #if !GTK4
02381
          gtk_button_new_from_icon_name ("help-about", GTK_ICON_SIZE_BUTTON);
02382 #else
02383
          gtk_button_new_from_icon_name ("help-about");
02384 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_about), _("About"));
g_signal_connect (window->button_about, "clicked", window_about, NULL);
02385
02386
02387
02388
        // Creating the exit button
02389
        window->button_exit = (GtkButton *)
02390 #if !GTK4
          gtk_button_new_from_icon_name ("application-exit", GTK_ICON_SIZE_BUTTON);
02391
02392 #else
02393
          gtk_button_new_from_icon_name ("application-exit");
02394 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_exit), _("Exit"));
g_signal_connect_swapped (window->button_exit, "clicked",
02395
02396
                                     G_CALLBACK (g_application_quit),
02397
02398
                                     G_APPLICATION (application));
02399
02400
        // Creating the buttons bar
02401
        window->box_buttons = (GtkBox *) gtk_box_new (GTK_ORIENTATION_HORIZONTAL, 0);
02402
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_open));
02403
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_save));
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_run));
02404
02405
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_options));
02406
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_help));
02407
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_about));
02408
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_exit));
02409
02410
        \ensuremath{//} Creating the simulator program label and entry
        window->label_simulator = (GtkLabel *) gtk_label_new (_("Simulator program"));
window->button_simulator = (GtkButton *)
02411
02412
02413
          gtk_button_new_with_mnemonic (_("Simulator program"));
02414
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_simulator)
02415
                                         _("Simulator program executable file"));
        gtk_widget_set_hexpand (GTK_WIDGET (window->button_simulator), TRUE);
g_signal_connect (window->button_simulator, "clicked",
02416
02417
02418
                            G_CALLBACK (dialog_simulator), NULL);
02419
02420
        // Creating the evaluator program label and entry
02421
        window->check_evaluator = (GtkCheckButton *)
          gtk_check_button_new_with_mnemonic (_("_Evaluator program"));
02422
02423
        q_signal_connect (window->check_evaluator, "toggled", window_update, NULL);
```

```
02424
        window->button_evaluator = (GtkButton *)
           gtk_button_new_with_mnemonic (_("Evaluator program"));
02425
02426
        gtk_widget_set_tooltip_text
02427
           (GTK_WIDGET (window->button_evaluator),
02428
        _("Optional evaluator program executable file"));
g_signal_connect (window->button_evaluator, "clicked"
02429
02430
                            G_CALLBACK (dialog_evaluator), NULL);
02431
02432
         // Creating the results files labels and entries
        window->label_result = (GtkLabel *) gtk_label_new (_("Result file"));
window->entry_result = (GtkEntry *) gtk_entry_new ();
02433
02434
02435
        gtk_widget_set_tooltip_text
        (GTT_WIDGET (window->entry_result), _("Best results file"));
window->label_variables = (GtkLabel *) gtk_label_new (_("Variables file"));
window->entry_variables = (GtkEntry *) gtk_entry_new ();
02436
02437
02438
02439
        gtk_widget_set_tooltip_text
02440
           (GTK_WIDGET (window->entry_variables), _("All simulated results file"));
02441
02442
        // Creating the files grid and attaching widgets
02443
        window->grid_files = (GtkGrid *) gtk_grid_new ();
02444
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_simulator),
02445
                           0, 0, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_simulator),
02446
02447
                           1, 0, 1, 1);
02448
        qtk_grid_attach (window->grid_files, GTK_WIDGET (window->check_evaluator),
02449
                           0, 1, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_evaluator),
02450
02451
                           1, 1, 1, 1);
02452
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_result),
02453
                           0, 2, 1, 1);
02454
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->entry_result),
02455
                            1, 2, 1, 1);
02456
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_variables),
02457
                           0, 3, 1, 1);
02458
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->entry_variables),
02459
                           1, 3, 1, 1);
02460
02461
         // Creating the algorithm properties
02462
        window->label_simulations = (GtkLabel *) gtk_label_new
02463
           (_("Simulations number"));
02464
        window->spin_simulations
02465
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02466
        {\tt gtk\_widget\_set\_tooltip\_text}
02467
           (GTK_WIDGET (window->spin_simulations),
            _("Number of simulations to perform for each iteration"));
02468
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_simulations), TRUE);
02469
02470
        window->label_iterations = (GtkLabel *)
02471
          gtk_label_new (_("Iterations number"));
        window->spin iterations
02472
02473
           = (GtkSpinButton *) gtk spin button new with range (1., 1.e6, 1.);
02474
        gtk_widget_set_tooltip_text
02475
           (GTK_WIDGET (window->spin_iterations), _("Number of iterations"));
02476
        g_signal_connect
        (window->spin_iterations, "value-changed", window_update, NULL);
gtk_widget_set_hexpand (GTK_WIDGET (window->spin_iterations), TRUE);
02477
02478
02479
         window->label_tolerance = (GtkLabel *) gtk_label_new (_("Tolerance"));
        window->spin_tolerance =
02480
02481
           (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02482
         gtk_widget_set_tooltip_text
02483
           (GTK_WIDGET (window->spin_tolerance),
            _("Tolerance to set the variable interval on the next iteration"));
02484
02485
        window->label_bests = (GtkLabel *) gtk_label_new (_("Bests number"));
02486
        window->spin_bests
02487
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e6, 1.);
        gtk_widget_set_tooltip_text
02488
02489
           (GTK_WIDGET (window->spin_bests),
            _("Number of best simulations used to set the variable interval " \,
02490
              "on the next iteration"));
02491
02492
        window->label_population
02493
            = (GtkLabel *) gtk_label_new (_("Population number"));
02494
        window->spin_population
02495
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02496
        {\tt gtk\_widget\_set\_tooltip\_text}
02497
           (GTK_WIDGET (window->spin_population),
        __("Number of population for the genetic algorithm"));
gtk_widget_set_hexpand (GTK_WIDGET (window->spin_population), TRUE);
02498
02499
02500
        window->label_generations
02501
           = (GtkLabel *) gtk_label_new (_("Generations number"));
02502
        window->spin_generations
02503
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e6, 1.);
        gtk_widget_set_tooltip_text
02504
02505
           (GTK_WIDGET (window->spin_generations),
02506
            _("Number of generations for the genetic algorithm"));
02507
        window->label_mutation = (GtkLabel *) gtk_label_new (_("Mutation ratio"));
02508
        window->spin_mutation
02509
           = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02510
        gtk_widget_set_tooltip_text
```

```
(GTK_WIDGET (window->spin_mutation),
            _("Ratio of mutation for the genetic algorithm"));
02512
02513
        window->label_reproduction
          = (GtkLabel *) gtk_label_new (_("Reproduction ratio"));
02514
02515
         window->spin reproduction
02516
           = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
        gtk_widget_set_tooltip_text
           (GTK_WIDGET (window->spin_reproduction),
02518
02519
            _("Ratio of reproduction for the genetic algorithm"));
02520
        window->label_adaptation = (GtkLabel *) gtk_label_new (_("Adaptation ratio"));
02521
        window->spin_adaptation
02522
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
        gtk_widget_set_tooltip_text
  (GTK_WIDGET (window->spin_adaptation),
02523
02524
02525
            _("Ratio of adaptation for the genetic algorithm"));
        window->label_threshold = (GtkLabel *) gtk_label_new (_("Threshold"));
window->spin_threshold = (GtkSpinButton *)
02526
02527
          gtk_spin_button_new_with_range (-G_MAXDOUBLE, G_MAXDOUBLE,
02528
                                             precision[DEFAULT_PRECISION]);
02529
02530
        gtk_widget_set_tooltip_text
02531
           (GTK_WIDGET (window->spin_threshold),
02532
            _("Threshold in the objective function to finish the simulations"));
        window->scrolled_threshold = (GtkScrolledWindow *)
02533
02534 #if !GTK4
02535
          gtk_scrolled_window_new (NULL, NULL);
02536 #else
02537
          gtk_scrolled_window_new ();
02538 #endif
02539
        gtk_scrolled_window_set_child (window->scrolled_threshold,
02540
                                          GTK_WIDGET (window->spin_threshold));
          gtk_widget_set_hexpand (GTK_WIDGET (window->scrolled_threshold), TRUE);
02541 //
02542 //
          gtk_widget_set_halign (GTK_WIDGET (window->scrolled_threshold),
02543 //
                                          GTK_ALIGN_FILL);
02544
02545
         // Creating the hill climbing method properties
        window->check_climbing = (GtkCheckButton *)
02546
        gtk_check_button_new_with_mnemonic (_("_Hill climbing method"));
g_signal_connect (window->check_climbing, "toggled", window_update, NULL);
02547
02549
         window->grid_climbing = (GtkGrid *) gtk_grid_new ();
02550 #if !GTK4
02551
        window->button_climbing[0] = (GtkRadioButton *)
02552
          gtk_radio_button_new_with_mnemonic (NULL, label_climbing[0]);
02553 #else
02554
        window->button_climbing[0] = (GtkCheckButton *)
02555
          gtk_check_button_new_with_mnemonic (label_climbing[0]);
02556 #endif
02557
        gtk_grid_attach (window->grid_climbing,
02558
                           GTK_WIDGET (window->button_climbing[0]), 0, 0, 1, 1);
        g_signal_connect (window->button_climbing[0], "toggled", window_update, NULL);
for (i = 0; ++i < NCLIMBINGS;)</pre>
02559
02560
02561
02562 #if !GTK4
02563
             window->button_climbing[i] = (GtkRadioButton *)
02564
               {\tt gtk\_radio\_button\_new\_with\_mnemonic}
02565
               ({\tt gtk\_radio\_button\_get\_group} \ ({\tt window->button\_climbing[0]}) \, ,
02566
                label climbing[i]);
02567 #else
02568
             window->button_climbing[i] = (GtkCheckButton *)
02569
               gtk_check_button_new_with_mnemonic (label_climbing[i]);
02570
             gtk_check_button_set_group (window->button_climbing[i]
                                           window->button_climbing[0]);
02571
02572 #endif
             gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_climbing[i]),
02574
                                            tip_climbing[i]);
02575
             gtk_grid_attach (window->grid_climbing,
             GTK_WIDGET (window->button_climbing[i]), 0, i, 1, 1); g_signal_connect (window->button_climbing[i], "toggled", window_update,
02576
02577
                                NULL);
02578
02579
         window->label_steps = (GtkLabel *) gtk_label_new (_("Steps number"));
        window>spin_steps = (GtkSpinButton *)
  gtk_spin_button_new_with_range (1., 1.e12, 1.);
02581
02582
02583
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_steps), TRUE);
02584
        window->label_estimates
        = (GtkLabel *) gtk_label_new (_("Climbing estimates number"));
window->spin_estimates = (GtkSpinButton *)
02585
02586
02587
          gtk_spin_button_new_with_range (1., 1.e3, 1.);
02588
        window->label_relaxation
02589
          = (GtkLabel *) gtk_label_new (_("Relaxation parameter"));
        window->spin_relaxation = (GtkSpinButton *)
02590
          gtk_spin_button_new_with_range (0., 2., 0.001);
02591
02592
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->label_steps),
02593
                           0, NCLIMBINGS, 1, 1);
02594
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_steps),
02595
                           1, NCLIMBINGS, 1, 1);
        02596
02597
```

```
02598
       gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_estimates),
                         1, NCLIMBINGS + 1, 1, 1);
02599
02600
       gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->label_relaxation),
                         0, NCLIMBINGS + 2, 1, 1);
02601
       \tt gtk\_grid\_attach \ (window->grid\_climbing, \ GTK\_WIDGET \ (window->spin\_relaxation) \ ,
02602
                         1, NCLIMBINGS + 2, 1, 1);
02603
02604
02605
       \ensuremath{//} Creating the array of algorithms
02606
       window->grid_algorithm = (GtkGrid *) gtk_grid_new ();
02607 #if !GTK4
02608
       window->button_algorithm[0] = (GtkRadioButton *)
02609
         gtk_radio_button_new_with_mnemonic (NULL, label_algorithm[0]);
02610 #else
02611
       window->button_algorithm[0] = (GtkCheckButton *)
02612
         gtk_check_button_new_with_mnemonic (label_algorithm[0]);
02613 #endif
       gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_algorithm[0]),
02614
02615
                                    tip_algorithm[0]);
       gtk_grid_attach (window->grid_algorithm,
02616
02617
                        GTK_WIDGET (window->button_algorithm[0]), 0, 0, 1, 1);
02618
       g_signal_connect (window->button_algorithm[0], "toggled",
02619
                          window_set_algorithm, NULL);
       for (i = 0; ++i < NALGORITHMS;)</pre>
02620
02621
02622 #if !GTK4
02623
           window->button_algorithm[i] = (GtkRadioButton *)
02624
             gtk_radio_button_new_with_mnemonic
              (gtk_radio_button_get_group (window->button_algorithm[0]),
02625
02626
              label_algorithm[i]);
02627 #else
           window->button algorithm[i] = (GtkCheckButton *)
02628
02629
             gtk_check_button_new_with_mnemonic (label_algorithm[i]);
02630
            gtk_check_button_set_group (window->button_algorithm[i],
02631
                                        window->button_algorithm[0]),
02632 #endif
             gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_algorithm[i]),
02633
02634
                                           tip_algorithm[i]);
02635
           gtk_grid_attach (window->grid_algorithm,
02636
                            GTK_WIDGET (window->button_algorithm[i]), 0, i, 1, 1);
            g_signal_connect (window->button_algorithm[i], "toggled",
02637
02638
                              window_set_algorithm, NULL);
02639
       gtk_grid_attach (window->grid_algorithm,
02640
02641
                        GTK_WIDGET (window->label_simulations),
                         0, NALGORITHMS, 1, 1);
02642
02643
       gtk_grid_attach (window->grid_algorithm,
02644
                         GTK_WIDGET (window->spin_simulations), 1, NALGORITHMS, 1, 1);
02645
       {\tt gtk\_grid\_attach~(window->grid\_algorithm,}
                         GTK WIDGET (window->label iterations).
02646
                         0, NALGORITHMS + 1, 1, 1);
02647
02648
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_iterations),
                         1, NALGORITHMS + 1, 1, 1);
02649
02650
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_tolerance),
02651
                         0, NALGORITHMS + 2, 1, 1);
       02652
02653
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_bests),
02654
                         0, NALGORITHMS + 3, 1, 1);
02655
02656
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_bests),
02657
                         1, NALGORITHMS + 3, 1, 1);
02658
       gtk grid attach (window->grid algorithm,
02659
                        GTK_WIDGET (window->label_population),
                         0, NALGORITHMS + 4, 1, 1);
02660
02661
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_population),
02662
                         1, NALGORITHMS + 4, 1, 1);
02663
       gtk_grid_attach (window->grid_algorithm,
02664
                        GTK_WIDGET (window->label_generations),
                         0, NALGORITHMS + 5, 1, 1);
02665
02666
       gtk_grid_attach (window->grid_algorithm,
02667
                        GTK_WIDGET (window->spin_generations),
02668
                         1, NALGORITHMS + 5, 1, 1);
02669
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_mutation),
       02670
02671
02672
       gtk_grid_attach (window->grid_algorithm,
02673
                         GTK_WIDGET (window->label_reproduction),
02674
02675
                         0, NALGORITHMS + 7, 1, 1);
02676
       gtk_grid_attach (window->grid_algorithm,
02677
                        GTK_WIDGET (window->spin_reproduction),
                         1, NALGORITHMS + 7, 1, 1);
02678
02679
       gtk_grid_attach (window->grid_algorithm,
                         GTK_WIDGET (window->label_adaptation),
02680
02681
                         0, NALGORITHMS + 8, 1, 1);
02682
       \verb|gtk_grid_attach| (\verb|window->grid_algorithm|, GTK_WIDGET| (\verb|window->spin_adaptation)|, \\
02683
                         1, NALGORITHMS + 8, 1, 1);
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->check_climbing),
02684
```

```
02685
                            0, NALGORITHMS + 9, 2, 1);
         gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->grid_climbing),
0, NALGORITHMS + 10, 2, 1);
02686
02687
         \verb|gtk_grid_attach| (\verb|window->grid_algorithm|, GTK_WIDGET| (\verb|window->label_threshold|)|,
02688
                            0, NALGORITHMS + 11, 1, 1);
02689
         gtk_grid_attach (window->grid_algorithm,
02690
02691
                            GTK_WIDGET (window->scrolled_threshold),
                            1, NALGORITHMS + 11, 1, 1);
02692
02693
         window->frame_algorithm = (GtkFrame *) gtk_frame_new (_("Algorithm"));
02694
         gtk_frame_set_child (window->frame_algorithm,
                                 GTK_WIDGET (window->grid_algorithm));
02695
02696
02697
         // Creating the variable widgets
         window->combo_variable = (GtkComboBoxText *) gtk_combo_box_text_new ();
02698
02699
         gtk_widget_set_tooltip_text
         (GTK_WIDGET (window->combo_variable), _("Variables selector"));
window->id_variable = g_signal_connect
  (window->combo_variable, "changed", window_set_variable, NULL);
02700
02701
02702
02703 #if !GTK4
02704
        window->button_add_variable = (GtkButton *)
           gtk_button_new_from_icon_name ("list-add", GTK_ICON_SIZE_BUTTON);
02705
02706 #else
02707
         window->button_add_variable = (GtkButton *)
02708
           gtk_button_new_from_icon_name ("list-add");
02709 #endif
02710
         g_signal_connect (window->button_add_variable, "clicked", window_add_variable,
                             NULL);
02711
02712
         gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_add_variable),
02713
                                          _("Add variable"));
02714 #if !GTK4
02715
         window->button remove variable = (GtkButton *)
02716
           gtk_button_new_from_icon_name ("list-remove", GTK_ICON_SIZE_BUTTON);
02717 #else
02718
         window->button_remove_variable = (GtkButton *)
02719
           gtk_button_new_from_icon_name ("list-remove");
02720 #endif
02721
         g_signal_connect (window->button_remove_variable, "clicked",
02722
                             window_remove_variable, NULL);
02723
         gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_remove_variable),
02724
                                          _("Remove variable"));
02725
         window->label_variable = (GtkLabel *) gtk_label_new (_("Name"));
         window->entry_variable = (GtkEntry *) gtk_entry_new ();
02726
02727
         gtk_widget_set_tooltip_text
02728
           (GTK_WIDGET (window->entry_variable), _("Variable name"));
02729
         gtk_widget_set_hexpand (GTK_WIDGET (window->entry_variable), TRUE);
         window->id_variable_label = g_signal_connect
  (window->entry_variable, "changed", window_label_variable, NULL);
02730
02731
02732
         window->label_min = (GtkLabel *) gtk_label_new (_("Minimum"));
         window->spin_min = (GtkSpinButton *) gtk_spin_button_new_with_range
  (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02733
02734
02735
         gtk_widget_set_tooltip_text
02736
           (GTK_WIDGET (window->spin_min), _("Minimum initial value of the variable"));
         window->scrolled_min = (GtkScrolledWindow *)
02737
02738 #if !GTK4
02739
           gtk_scrolled_window_new (NULL, NULL);
02740 #else
02741
           gtk_scrolled_window_new ();
02742 #endif
02743
         gtk_scrolled_window_set_child (window->scrolled_min,
02744
                                            GTK_WIDGET (window->spin_min));
02745
         g_signal_connect (window->spin_min, "value-changed",
02746
                             window rangemin variable, NULL);
02747
         window->label_max = (GtkLabel *) gtk_label_new (_("Maximum"));
02748
         window->spin_max = (GtkSpinButton *) gtk_spin_button_new_with_range
02749
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02750
         gtk_widget_set_tooltip_text
02751
         (GTK_WIDGET (window->spin_max), _("Maximum window->scrolled_max = (GtkScrolledWindow *)
                                                 ("Maximum initial value of the variable"));
02752
02753 #if !GTK4
02754
           gtk_scrolled_window_new (NULL, NULL);
02755 #else
02756
           gtk_scrolled_window_new ();
02757 #endif
         gtk_scrolled_window_set_child (window->scrolled_max,
02758
02759
                                            GTK WIDGET (window->spin max));
02760
         g_signal_connect (window->spin_max, "value-changed",
02761
                             window_rangemax_variable, NULL);
        window->check_minabs = (GtkCheckButton *)
  gtk_check_button_new_with_mnemonic (_("_Absolute minimum"));
g_signal_connect (window->check_minabs, "toggled", window_update, NULL);
window->spin_minabs = (GtkSpinButton *) gtk_spin_button_new_with_range
02762
02763
02764
02765
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02767
         gtk_widget_set_tooltip_text
02768
           (GTK_WIDGET (window->spin_minabs),
02769
            _("Minimum allowed value of the variable"));
        window->scrolled_minabs = (GtkScrolledWindow *)
02770
02771 #if !GTK4
```

```
gtk_scrolled_window_new (NULL, NULL);
02773 #else
02774
          gtk_scrolled_window_new ();
02775 #endif
        gtk_scrolled_window_set_child (window->scrolled_minabs,
02776
02777
                                          GTK_WIDGET (window->spin_minabs));
        g_signal_connect (window->spin_minabs, "value-changed",
02778
02779
                            window_rangeminabs_variable, NULL);
02780
        window->check_maxabs = (GtkCheckButton *)
        gtk_check_button_new_with_mnemonic (_("_Absolute maximum"));
g_signal_connect (window->check_maxabs, "toggled", window_update, NULL);
window->spin_maxabs = (GtkSpinButton *) gtk_spin_button_new_with_range
02781
02782
02783
02784
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02785
        gtk_widget_set_tooltip_text
02786
           (GTK_WIDGET (window->spin_maxabs),
02787
            _("Maximum allowed value of the variable"));
02788
        window->scrolled_maxabs = (GtkScrolledWindow *)
02789 #if !GTK4
02790
          gtk_scrolled_window_new (NULL, NULL);
02791 #else
02792
          gtk scrolled window new ();
02793 #endif
02794
        gtk_scrolled_window_set_child (window->scrolled_maxabs,
                                          GTK_WIDGET (window->spin_maxabs));
02795
02796
        g_signal_connect (window->spin_maxabs, "value-changed",
02797
                           window_rangemaxabs_variable, NULL);
02798
        window->label_precision = (GtkLabel *) gtk_label_new (_("Precision digits"));
        window->spin_precision = (GtkSpinButton *)
02799
02800
          gtk_spin_button_new_with_range (0., (gdouble) DEFAULT_PRECISION, 1.);
        gtk_widget_set_tooltip_text
  (GTK_WIDGET (window->spin_precision),
02801
02802
            _("Number of precision floating point digits\n"
02803
02804
              "0 is for integer numbers"));
02805
        g_signal_connect (window->spin_precision, "value-changed",
02806
                           window_precision_variable, NULL);
        window->label_sweeps = (GtkLabel *) gtk_label_new (_("Sweeps number"));
02807
        window->spin_sweeps =
02808
02809
           (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02810
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->spin_sweeps),
        __("Number of steps sweeping the variable"));
g_signal_connect (window->spin_sweeps, "value-changed",
02811
02812
                            window_update_variable, NULL);
02813
02814
        window->label_bits = (GtkLabel *) gtk_label_new (_("Bits number"));
02815
        window->spin_bits
02816
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 64., 1.);
        gtk_widget_set_tooltip_text
02817
02818
           (GTK_WIDGET (window->spin_bits),
02819
            _("Number of bits to encode the variable"));
02820
        g_signal_connect
          (window->spin_bits, "value-changed", window_update_variable, NULL);
02821
        window->label_step = (GtkLabel *) gtk_label_new (_("Step size"));
window->spin_step = (GtkSpinButton *) gtk_spin_button_new_with_range
02822
02823
02824
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
        gtk_widget_set_tooltip_text
02825
02826
          (GTK_WIDGET (window->spin_step),
    ("Initial step size for the hill climbing method"));
02827
         window->scrolled_step = (GtkScrolledWindow *)
02828
02829 #if !GTK4
02830
          gtk_scrolled_window_new (NULL, NULL);
02831 #else
02832
          gtk_scrolled_window_new ();
02833 #endif
02834
        gtk_scrolled_window_set_child (window->scrolled_step,
02835
                                          GTK_WIDGET (window->spin_step));
02836
02837
           (window->spin_step, "value-changed", window_step_variable, NULL);
        window->grid_variable = (GtkGrid *) gtk_grid_new ();
02838
        gtk_grid_attach (window->grid_variable,
02839
02840
                           GTK_WIDGET (window->combo_variable), 0, 0, 2, 1);
02841
        gtk_grid_attach (window->grid_variable,
02842
                           GTK_WIDGET (window->button_add_variable), 2, 0, 1, 1);
02843
        gtk_grid_attach (window->grid_variable,
02844
                           GTK_WIDGET (window->button_remove_variable), 3, 0, 1, 1);
02845
        gtk_grid_attach (window->grid_variable,
                           GTK_WIDGET (window->label_variable), 0, 1, 1, 1);
02846
        gtk_grid_attach (window->grid_variable,
02847
                           GTK_WIDGET (window->entry_variable), 1, 1, 3, 1);
02848
02849
        gtk_grid_attach (window->grid_variable,
02850
                           GTK_WIDGET (window->label_min), 0, 2, 1, 1);
02851
        gtk grid attach (window->grid variable,
                           GTK WIDGET (window->scrolled min), 1, 2, 3, 1);
02852
02853
        gtk_grid_attach (window->grid_variable,
                           GTK_WIDGET (window->label_max), 0, 3, 1, 1);
02854
02855
        gtk_grid_attach (window->grid_variable,
02856
                           GTK_WIDGET (window->scrolled_max), 1, 3, 3, 1);
02857
        gtk_grid_attach (window->grid_variable,
02858
                           GTK WIDGET (window->check minabs), 0, 4, 1, 1);
```

```
gtk_grid_attach (window->grid_variable,
                          GTK_WIDGET (window->scrolled_minabs), 1, 4, 3, 1);
02860
02861
        gtk_grid_attach (window->grid_variable,
                         GTK_WIDGET (window->check_maxabs), 0, 5, 1, 1);
02862
02863
        gtk_grid_attach (window->grid_variable,
                          GTK_WIDGET (window->scrolled_maxabs), 1, 5, 3, 1);
02864
        gtk_grid_attach (window->grid_variable,
                          GTK_WIDGET (window->label_precision), 0, 6, 1, 1);
02866
02867
        gtk_grid_attach (window->grid_variable,
02868
                          GTK_WIDGET (window->spin_precision), 1, 6, 3, 1);
        gtk_grid_attach (window->grid_variable,
02869
02870
                         GTK WIDGET (window->label sweeps), 0, 7, 1, 1);
02871
        gtk_grid_attach (window->grid_variable,
02872
                          GTK_WIDGET (window->spin_sweeps), 1, 7, 3, 1);
02873
        gtk_grid_attach (window->grid_variable,
02874
                          GTK_WIDGET (window->label_bits), 0, 8, 1, 1);
02875
        gtk_grid_attach (window->grid_variable,
02876
                          GTK WIDGET (window->spin bits), 1, 8, 3, 1);
02877
        gtk_grid_attach (window->grid_variable,
02878
                          GTK_WIDGET (window->label_step), 0, 9, 1, 1);
02879
        gtk_grid_attach (window->grid_variable,
02880
                         GTK_WIDGET (window->scrolled_step), 1, 9, 3, 1);
        window->frame_variable = (GtkFrame *) gtk_frame_new (_("Variable"));
gtk_frame_set_child (window->frame_variable,
02881
02882
02883
                             GTK_WIDGET (window->grid_variable));
02884
02885
        // Creating the experiment widgets
02886
        window->combo_experiment = (GtkComboBoxText *) gtk_combo_box_text_new ();
02887
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->combo_experiment),
02888
                                      _("Experiment selector"));
        window->id_experiment = g_signal_connect
02889
02890
          (window->combo_experiment, "changed", window_set_experiment, NULL);
02891 #if !GTK4
02892
        window->button_add_experiment = (GtkButton *)
02893
          gtk_button_new_from_icon_name ("list-add", GTK_ICON_SIZE_BUTTON);
02894 #else
02895
        window->button add experiment = (GtkButton *)
02896
          gtk_button_new_from_icon_name ("list-add");
02897 #endif
02898
        g_signal_connect
02899
          (window->button_add_experiment, "clicked", window_add_experiment, NULL);
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_add_experiment),
02900
                                       _("Add experiment"));
02901
02902 #if !GTK4
02903
        window->button_remove_experiment = (GtkButton *)
          gtk_button_new_from_icon_name ("list-remove", GTK_ICON_SIZE_BUTTON);
02904
02905 #else
02906
        \label{lem:window-button_remove_experiment} = (\texttt{GtkButton} \ \star)
          gtk_button_new_from_icon_name ("list-remove");
02907
02908 #endif
02909
        g_signal_connect (window->button_remove_experiment, "clicked",
02910
                           window_remove_experiment, NULL);
02911
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_remove_experiment),
02912
                                      _("Remove experiment"));
02913
        window->label_experiment
02914
          = (GtkLabel *) gtk label new ( ("Experimental data file"));
02915
        window->button_experiment = (GtkButton *)
02916
          gtk_button_new_with_mnemonic (_("Experimental data file"));
02917
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_experiment),
        __("Experimental data file"));
g_signal_connect (window->button_experiment, "clicked",
02918
02919
02920
                          window_name_experiment, NULL);
02921
        gtk_widget_set_hexpand (GTK_WIDGET (window->button_experiment), TRUE);
02922
        window->label_weight = (GtkLabel *) gtk_label_new (_("Weight"));
02923
        window->spin_weight
02924
         = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02925
        {\tt gtk\_widget\_set\_tooltip\_text}
          (GTK WIDGET (window->spin weight).
02926
02927
           _("Weight factor to build the objective function"));
02928
        g_signal_connect
02929
          (window->spin_weight, "value-changed", window_weight_experiment, NULL);
02930
        window->grid_experiment = (GtkGrid *) gtk_grid_new ();
        gtk_grid_attach (window->grid_experiment,
02931
02932
                         GTK_WIDGET (window->combo_experiment), 0, 0, 2, 1);
02933
        gtk_grid_attach (window->grid_experiment,
02934
                         GTK_WIDGET (window->button_add_experiment), 2, 0, 1, 1);
        gtk_grid_attach (window->grid_experiment,
02935
02936
                          GTK_WIDGET (window->button_remove_experiment), 3, 0, 1, 1);
02937
        gtk_grid_attach (window->grid_experiment,
                          GTK WIDGET (window->label_experiment), 0, 1, 1, 1);
02938
02939
        gtk_grid_attach (window->grid_experiment,
02940
                          GTK_WIDGET (window->button_experiment), 1, 1, 3, 1);
02941
        gtk_grid_attach (window->grid_experiment,
02942
                          GTK_WIDGET (window->label_weight), 0, 2, 1, 1);
02943
        gtk_grid_attach (window->grid_experiment,
                          GTK_WIDGET (window->spin_weight), 1, 2, 3, 1);
02944
        for (i = 0; i < MAX_NINPUTS; ++i)</pre>
02945
```

```
02946
02947
             snprintf (buffer3, 64, "%s %u", _("Input template"), i + 1);
             window->check_template[i] = (GtkCheckButton *)
02948
02949
              gtk_check_button_new_with_label (buffer3);
02950
             window->id template[i]
              = g_signal_connect (window->check_template[i], "toggled",
02951
02952
                                    window_inputs_experiment, NULL);
            gtk_grid_attach (window->grid_experiment,
02953
            GTK_WIDGET (window->check_template[i]), 0, 3 + i, 1, 1);
window->button_template[i] = (GtkButton *)
02954
02955
              gtk_button_new_with_mnemonic (_("Input template"));
02956
02957
02958
            gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_template[i]),
02959
                                           _("Experimental input template file"));
02960
             window->id_input[i] =
              g_signal_connect_swapped (window->button_template[i], "clicked",
02961
02962
                                          (GCallback) window_template_experiment,
            (void *) (size_t) i);
gtk_grid_attach (window->grid_experiment,
02963
02964
02965
                              GTK_WIDGET (window->button_template[i]), 1, 3 + i, 3, 1);
02966
02967
        window->frame_experiment = (GtkFrame *) gtk_frame_new (_("Experiment"));
        gtk_frame_set_child (window->frame_experiment,
02968
02969
                               GTK_WIDGET (window->grid_experiment));
02970
02971
        // Creating the error norm widgets
02972
        window->frame_norm = (GtkFrame *) gtk_frame_new (_("Error norm"));
02973
        window->grid_norm = (GtkGrid *) gtk_grid_new ();
02974
        gtk_frame_set_child (window->frame_norm, GTK_WIDGET (window->grid_norm));
02975 #if !GTK4
02976
        window->button_norm[0] = (GtkRadioButton *)
02977
          gtk_radio_button_new_with_mnemonic (NULL, label_norm[0]);
02978 #else
02979
        window->button_norm[0] = (GtkCheckButton *)
02980
          gtk_check_button_new_with_mnemonic (label_norm[0]);
02981 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_norm[0]),
02982
                                       tip_norm[0]);
02984
       gtk_grid_attach (window->grid_norm,
02985
                          GTK_WIDGET (window->button_norm[0]), 0, 0, 1, 1);
       g_signal_connect (window->button_norm[0], "toggled", window_update, NULL);
for (i = 0; ++i < NNORMS;)</pre>
02986
02987
02988
02989 #if !GTK4
02990
             window->button_norm[i] = (GtkRadioButton *)
02991
              gtk_radio_button_new_with_mnemonic
02992
               (gtk_radio_button_get_group (window->button_norm[0]), label_norm[i]);
02993 #else
02994
            window->button norm[i] = (GtkCheckButton *)
02995
              qtk_check_button_new_with_mnemonic (label_norm[i]);
             gtk_check_button_set_group (window->button_norm[i]
02997
                                          window->button_norm[0]);
02998 #endif
02999
             gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_norm[i]),
03000
                                           tip_norm[i]);
03001
             gtk_grid_attach (window->grid_norm,
                              GTK_WIDGET (window->button_norm[i]), 0, i, 1, 1);
03002
03003
            g_signal_connect (window->button_norm[i], "toggled", window_update, NULL);
03004
03005
        window->label_p = (GtkLabel *) gtk_label_new (_("P parameter"));
        gtk_grid_attach (window->grid_norm, GTK_WIDGET (window->label_p), 1, 1, 1, 1);
window->spin_p = (GtkSpinButton *)
03006
03007
03008
          gtk_spin_button_new_with_range (-G_MAXDOUBLE, G_MAXDOUBLE, 0.01);
03009
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->spin_p),
03010
                                        ("P parameter for the P error norm"));
03011
        window->scrolled_p = (GtkScrolledWindow *)
03012 #if !GTK4
03013
          gtk scrolled window new (NULL, NULL);
03014 #else
03015
          gtk_scrolled_window_new ();
03016 #endif
03017
        gtk_scrolled_window_set_child (window->scrolled_p,
                                         GTK_WIDGET (window->spin_p));
03018
        gtk_widget_set_hexpand (GTK_WIDGET (window->scrolled_p), TRUE);
gtk_widget_set_halign (GTK_WIDGET (window->scrolled_p), GTK_ALIGN_FILL);
03019
03020
        gtk_grid_attach (window->grid_norm, GTK_WIDGET (window->scrolled_p),
03021
03022
                          1, 2, 1, 2);
03023
03024
        \ensuremath{//} Creating the grid and attaching the widgets to the grid
03025
        window->grid = (GtkGrid *) gtk_grid_new ();
        gtk_grid_attach (window->grid, GTK_WIDGET (window->box_buttons), 0, 0, 3, 1);
03026
        gtk_grid_attach (window->grid, GTK_WIDGET (window->grid_files), 0, 1, 1, 1);
03027
03028
        gtk_grid_attach (window->grid,
03029
                          GTK_WIDGET (window->frame_algorithm), 0, 2, 1, 1);
03030
        gtk_grid_attach (window->grid,
                          GTK_WIDGET (window->frame_variable), 1, 2, 1, 1);
03031
03032
        gtk grid attach (window->grid,
```

```
GTK_WIDGET (window->frame_experiment), 2, 2, 1, 1);
03034
         gtk_grid_attach (window->grid, GTK_WIDGET (window->frame_norm), 1, 1, 2, 1);
03035
         gtk_window_set_child (window->window, GTK_WIDGET (window->grid));
03036
03037
         // Setting the window logo
03038
         window->logo = gdk_pixbuf_new_from_xpm_data (logo);
03039 #if !GTK4
03040
         gtk_window_set_icon (window->window, window->logo);
03041 #endif
03042
03043
         // Showing the window
03044 #if !GTK4
03045
         gtk_widget_show_all (GTK_WIDGET (window->window));
03046 #else
03047
         gtk_widget_show (GTK_WIDGET (window->window));
03048 #endif
03049
03050
          // In GTK+ 3.16 and 3.18 the default scrolled size is wrong
03051 #if GTK_MINOR_VERSION >= 16
03052 gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_min), -1, 40);
03053
         gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_max), -1, 40);
03054
         gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_minabs), -1, 40);
         gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_maxabs), -1, 40);
gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_step), -1, 40);
gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_step), -1, 40);
gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_p), -1, 40);
gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_threshold), -1, 40);
03055
03056
03057
03058
03059 #endif
03060
03061
         // Reading initial example
03062
        input_new ();
buffer2 = g_get_current_dir ();
03063
03064
         buffer = g_build_filename (buffer2, "..", "tests", "test1", INPUT_FILE, NULL);
03065
         g_free (buffer2);
03066
        window_read (buffer);
03067
        g_free (buffer);
03068
03069 #if DEBUG_INTERFACE
03070 fprintf (stderr, "window_new: start\n");
03071 #endif
03072 }
```

5.11.3.27 window_open()

```
static void window_open ( ) [static]
```

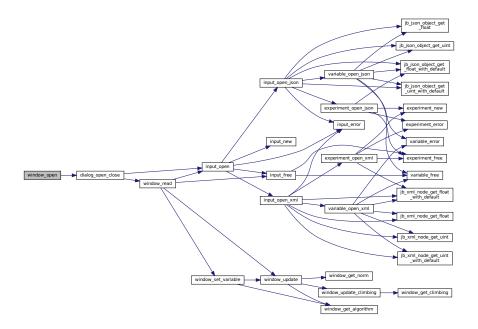
Function to open the input data.

Definition at line 2118 of file interface.c.

```
02119 {
          GtkFileChooserDialog *dlg;
02120
02121
          GtkFileFilter *filter;
02123 #if DEBUG_INTERFACE
         fprintf (stderr, "window_open: start\n");
02124
02125 #endif
02126
02127
           // Opening dialog
02128
         dlg = (GtkFileChooserDialog *)
02129
            gtk_file_chooser_dialog_new (_("Open input file"),
02130
                                                    window->window,
02131
                                                    GTK FILE CHOOSER ACTION OPEN,
                                                   _("_Cancel"), GTK_RESPONSE_CANCEL,
_("_OK"), GTK_RESPONSE_OK, NULL);
02132
02133
02134
02135
           // Adding XML filter
          filter = (GtkFileFilter *) gtk_file_filter_new ();
02136
          gtk_file_filter_set_name (filter, "XML");
gtk_file_filter_add_pattern (filter, "*.xml");
gtk_file_filter_add_pattern (filter, "*.XML");
gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter);
02137
02138
02139
02140
02141
02142
           // Adding JSON filter
02143
          filter = (GtkFileFilter *) gtk_file_filter_new ();
02144
          gtk_file_filter_set_name (filter, "JSON");
          gtk_file_filter_add_pattern (filter, "*.json");
gtk_file_filter_add_pattern (filter, "*.json");
gtk_file_filter_add_pattern (filter, "*.jso");
02145
02146
02147
          gtk_file_filter_add_pattern (filter, "*.JS");
02148
```

```
gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter);
02150
        // Connecting the close function
g_signal_connect (dlg, "response", G_CALLBACK (dialog_open_close), NULL);
02151
02152
02153
02154
        // Showing modal dialog
02155
        gtk_window_present (GTK_WINDOW (dlg));
02156
02157 #if DEBUG_INTERFACE
        fprintf (stderr, "window_open: end\n");
02158
02159 #endif
02160 }
```

Here is the call graph for this function:



5.11.3.28 window_precision_variable()

```
static void window_precision_variable ( ) [static]
```

Function to update the variable precision in the main window.

Definition at line 1803 of file interface.c.

```
01804 {
01805
        unsigned int i;
01806 #if DEBUG INTERFACE
       fprintf (stderr, "window_precision_variable: start\n");
01807
01808 #endif
        i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01809
01810
       01811
       gtk_spin_button_set_digits (window->spin_min, input->variable[i].precision);
gtk_spin_button_set_digits (window->spin_max, input->variable[i].precision);
01812
01813
        gtk_spin_button_set_digits (window->spin_minabs,
01814
                                     input->variable[i].precision);
01815
01816
       gtk_spin_button_set_digits (window->spin_maxabs,
01817
                                     input->variable[i].precision);
01818 #if DEBUG_INTERFACE
01819 fprintf (stderr, "window_precision_variable: end\n");
01820 #endif
01821 }
```

5.11.3.29 window_rangemax_variable()

```
static void window_rangemax_variable ( ) [static]
```

Function to update the variable rangemax in the main window.

Definition at line 1844 of file interface.c.

```
01845 {
01846    unsigned int i;
01847    #if DEBUG_INTERFACE
01848    fprintf (stderr, "window_rangemax_variable: start\n");
01849    #endif
01850    i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01851    input->variable[i].rangemax = gtk_spin_button_get_value (window->spin_max);
01852    #if DEBUG_INTERFACE
01853    fprintf (stderr, "window_rangemax_variable: end\n");
01854    #endif
01855 }
```

5.11.3.30 window_rangemaxabs_variable()

```
static void window_rangemaxabs_variable ( ) [static]
```

Function to update the variable rangemaxabs in the main window.

Definition at line 1879 of file interface.c.

```
01880 {
01881 unsigned int i;
01882 #if DEBUG_INTERFACE
01883 fprintf (stderr, "window_rangemaxabs_variable: start\n");
01884 #endif
01885 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01886 input->variable[i].rangemaxabs
01887 = gtk_spin_button_get_value (window->spin_maxabs);
01888 #if DEBUG_INTERFACE
01889 fprintf (stderr, "window_rangemaxabs_variable: end\n");
01890 #endif
01891 }
```

5.11.3.31 window_rangemin_variable()

```
static void window_rangemin_variable ( ) [static]
```

Function to update the variable rangemin in the main window.

Definition at line 1827 of file interface.c.

```
01828 {
01829    unsigned int i;
01830    #if DEBUG_INTERFACE
01831    fprintf (stderr, "window_rangemin_variable: start\n");
01832    #endif
01833    i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01834    input->variable[i].rangemin = gtk_spin_button_get_value (window->spin_min);
01835    #if DEBUG_INTERFACE
01836    fprintf (stderr, "window_rangemin_variable: end\n");
01837    #endif
01838 }
```

5.11.3.32 window_rangeminabs_variable()

```
static void window_rangeminabs_variable ( ) [static]
```

Function to update the variable rangeminabs in the main window.

Definition at line 1861 of file interface.c.

```
01862 {
01863    unsigned int i;
01864 #if DEBUG_INTERFACE
01865    fprintf (stderr, "window_rangeminabs_variable: start\n");
01866 #endif
01867    i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01868    input->variable[i].rangeminabs
01869    = gtk_spin_button_get_value (window->spin_minabs);
01870 #if DEBUG_INTERFACE
01871    fprintf (stderr, "window_rangeminabs_variable: end\n");
01872 #endif
01873 }
```

5.11.3.33 window_read()

Function to read the input data of a file.

Returns

1 on succes, 0 on error.

Parameters

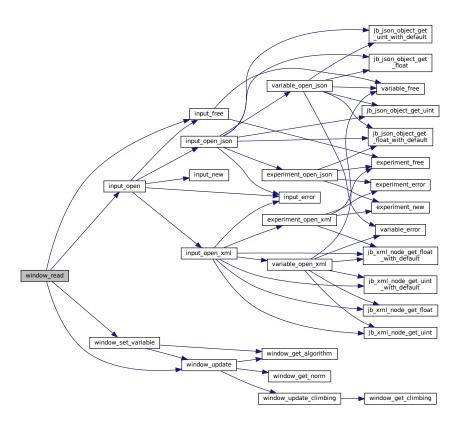
filename File name.

Definition at line 1953 of file interface.c.

```
01954 {
01955
        unsigned int i;
01956 #if DEBUG_INTERFACE
       fprintf (stderr, "window_read: start\n");
fprintf (stderr, "window_read: file name=%s\n", filename);
01957
01958
01959 #endif
01960
01961
        // Reading new input file
        input_free ();
input->result = input->variables = NULL;
01962
01963
        if (!input_open (filename))
01964
01965
01966 #if DEBUG_INTERFACE
01967
            fprintf (stderr, "window_read: end\n");
01968 #endif
01969
           return 0;
01970
01971
01972
        // Setting GTK+ widgets data
        gtk_entry_set_text (window->entry_result, input->result);
01974
        gtk_entry_set_text (window->entry_variables, input->variables);
01975
        gtk_button_set_label (window->button_simulator, input->simulator);
01976
        gtk_check_button_set_active (window->check_evaluator,
01977
                                       (size_t) input->evaluator);
01978
        if (input->evaluator)
01979
          gtk_button_set_label (window->button_evaluator, input->evaluator);
01980
        gtk_check_button_set_active (window->button_algorithm[input->algorithm],
01981
                                       TRUE);
01982
        switch (input->algorithm)
```

```
01983
01984
           case ALGORITHM_MONTE_CARLO:
01985
              gtk_spin_button_set_value (window->spin_simulations,
01986
                                              (gdouble) input->nsimulations);
01987
              // fallthrough
01988
           case ALGORITHM_SWEEP:
           case ALGORITHM_ORTHOGONAL:
01989
01990
              gtk_spin_button_set_value (window->spin_iterations,
01991
                                              (gdouble) input->niterations);
01992
              gtk_spin_button_set_value (window->spin_bests, (gdouble) input->nbest);
              gtk_spin_button_set_value (window->spin_tolerance, input->tolerance);
01993
              gtk_check_button_set_active (window->check_climbing, input->nsteps);
01994
01995
              if (input->nsteps)
01996
01997
                  gtk_check_button_set_active
01998
                     (window->button_climbing[input->climbing], TRUE);
01999
                  gtk_spin_button_set_value (window->spin_steps,
02000
                                                   (gdouble) input->nsteps);
                  gtk_spin_button_set_value (window->spin_relaxation,
02001
02002
                                                   (gdouble) input->relaxation);
02003
                  switch (input->climbing)
02004
                     case CLIMBING METHOD RANDOM:
02005
02006
                       gtk_spin_button_set_value (window->spin_estimates,
02007
                                                        (gdouble) input->nestimates);
02008
02009
02010
             break;
02011
           default:
02012
              gtk_spin_button_set_value (window->spin_population,
02013
                                              (gdouble) input->nsimulations);
02014
              gtk_spin_button_set_value (window->spin_generations,
02015
                                              (gdouble) input->niterations);
02016
              gtk_spin_button_set_value (window->spin_mutation, input->mutation_ratio);
02017
              gtk_spin_button_set_value (window->spin_reproduction,
02018
                                              input->reproduction_ratio);
              gtk_spin_button_set_value (window->spin_adaptation,
02019
02020
                                              input->adaptation_ratio);
02021
02022
         gtk_check_button_set_active (window->button_norm[input->norm], TRUE);
02023
         gtk_spin_button_set_value (window->spin_p, input->p);
         gtk_spin_button_set_value (window->spin_threshold, input->threshold);
g_signal_handler_block (window->combo_experiment, window->id_experiment);
gtk_combo_box_text_remove_all (window->combo_experiment);
02024
02025
02026
02027
              (i = 0; i < input->nexperiments; ++i)
02028
           gtk_combo_box_text_append_text (window->combo_experiment,
02029
                                                 input->experiment[i].name);
         g_signal_handler_unblock (window->combo_experiment(I).hame);
gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), 0);
g_signal_handler_block (window->combo_variable, window->id_variable);
g_signal_handler_block (window->experiment), window->id_variable);
02030
02031
02032
02034
         gtk_combo_box_text_remove_all (window->combo_variable);
02035
              (i = 0; i < input->nvariables; ++i)
           gtk_combo_box_text_append_text (window->combo_variable,
02036
02037
                                                 input->variable[i].name);
         g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
g_signal_handler_unblock (window->combo_variable, window->id_variable);
02038
02039
02040
         gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), 0);
02041
         window_set_variable ();
02042
         window_update ();
02043
02044 #if DEBUG_INTERFACE
02045
        fprintf (stderr, "window_read: end\n");
02046 #endif
         return 1;
02047
02048 }
```

Here is the call graph for this function:



5.11.3.34 window_remove_experiment()

```
static void window_remove_experiment ( ) [static]
```

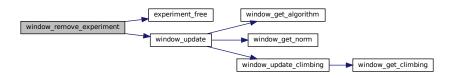
Function to remove an experiment in the main window.

Definition at line 1377 of file interface.c.

```
01378 {
01379 unsigned int i, j; 01380 #if DEBUG_INTERFACE
        fprintf (stderr, "window_remove_experiment: start\n");
01381
01382 #endif
01383
       i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01384
         g_signal_handler_block (window->combo_experiment, window->id_experiment);
         gtk_combo_box_text_remove (window->combo_experiment, i);
g_signal_handler_unblock (window->combo_experiment, window->id_experiment);
01385
01386
01387
         experiment_free (input->experiment + i, input->type);
         --input->nexperiments;
01388
01389
         for (j = i; j < input->nexperiments; ++j)
01390
         memcpy (input->experiment + j, input->experiment + j + 1,
                    sizeof (Experiment));
01391
         j = input->nexperiments - 1;
if (i > j)
01392
01393
01394
          i = j;
         for (j = 0; j < input->experiment->ninputs; ++j)
01395
01396
           g_signal_handler_block (window->button_template[j], window->id_input[j]);
01397
         gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), i);
        for (j = 0; j < input->experiment->ninputs; ++j)
  g_signal_handler_unblock (window->button_template[j], window->id_input[j]);
01398
01399
01400
         window_update ();
01401 #if DEBUG_INTERFACE
```

```
01402 fprintf (stderr, "window_remove_experiment: end\n"); 01403 #endif 01404 }
```

Here is the call graph for this function:



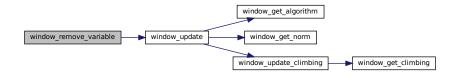
5.11.3.35 window_remove_variable()

```
static void window_remove_variable ( ) [static]
```

Function to remove a variable in the main window.

Definition at line 1715 of file interface.c.

```
01716 {
01717 unsigned int i, j;
01718 #if DEBUG_INTERFACE
        fprintf (stderr, "window_remove_variable: start\n");
01719
01720 #endif
01721
        i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01722
        g_signal_handler_block (window->combo_variable, window->id_variable);
01723
        gtk_combo_box_text_remove (window->combo_variable, i);
        g_signal_handler_unblock (window->combo_variable, window->id_variable);
xmlFree (input->variable[i].name);
01724
01725
01726
          -input->nvariables;
01727
        for (j = i; j < input->nvariables; ++j)
01728
          memcpy (input->variable + j, input->variable + j + 1, sizeof (Variable));
        j = input->nvariables - 1;
if (i > j)
01729
01730
01731
          i = j;
01732
        g_signal_handler_block (window->entry_variable, window->id_variable_label);
01733
        gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), i);
01734
        g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
01735 window_update ();
01736 #if DEBUG_INTERFACE
01737 fprintf (stderr, "window_remove_variable: end\n");
01738 #endif
01739 }
```



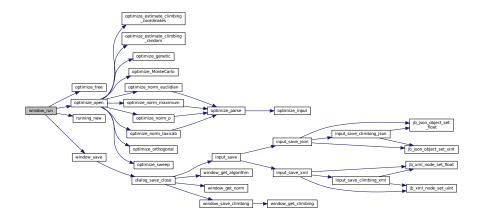
5.11.3.36 window_run()

```
static void window_run ( ) [static]
```

Function to run a optimization.

```
Definition at line 1019 of file interface.c.
```

```
01020 {
01021
        char *msg, *msg2, buffer[64], buffer2[64];
01022 unsigned int i;
01023 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: start\n");
01024
01025 #endif
      window_save ();
01026
01027
       running_new ();
01028
        jbw_process_pending ();
01029
       optimize_open ();
01030 #if DEBUG_INTERFACE
01031
       fprintf (stderr, "window_run: closing running dialog\n");
01032 #endif
01033
       gtk_spinner_stop (running->spinner);
01034
        gtk_window_destroy (GTK_WINDOW (running->dialog));
01035 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: displaying results\n");
01036
01037 #endif
01038
       snprintf (buffer, 64, "error = %.15le\n", optimize->error_old[0]);
01039
       msg2 = g\_strdup (buffer);
01040
       for (i = 0; i < optimize->nvariables; ++i, msg2 = msg)
01041
           01042
01043
           snprintf (buffer2, 64, buffer, optimize->value_old[i]);
msg = g_strconcat (msg2, buffer2, NULL);
01044
01045
01046
           g_free (msg2);
01047
       01048
01049
       msg = g_strconcat (msg2, buffer, NULL);
01050
01051
       g_free (msg2);
01052
       jbw_show_message_gtk (_("Best result"), msg, INFO_TYPE);
01053 g_free (msg);
01054 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: freeing memory\n");
01055
01056 #endif
01057
       optimize_free ();
01058 #if DEBUG_INTERFACE
01059
       fprintf (stderr, "window_run: end\n");
01060 #endif
01061 }
```



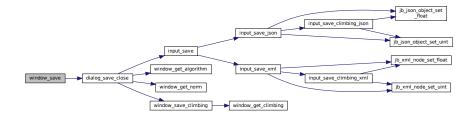
5.11.3.37 window_save()

```
static void window_save ( ) [static]
```

Function to save the input file.

```
Definition at line 959 of file interface.c.
```

```
00961
         GtkFileChooserDialog *dlg;
00962
         GtkFileFilter *filter1, *filter2;
00963
         char *buffer:
00964
00965 #if DEBUG_INTERFACE
        fprintf (stderr, "window_save: start\n");
00966
00967 #endif
00968
00969
         // Opening the saving dialog
00970
        dlg = (GtkFileChooserDialog *)
00971
           gtk_file_chooser_dialog_new (_("Save file"),
00972
                                             window->window,
00973
                                             GTK_FILE_CHOOSER_ACTION_SAVE,
00974
                                             _("_Cancel"), GTK_RESPONSE_CANCEL,
                                             _("_OK"), GTK_RESPONSE_OK, NULL);
00975
00976 #if !GTK4
00977
         gtk_file_chooser_set_do_overwrite_confirmation (GTK_FILE_CHOOSER (dlg), TRUE);
00978 #endi:
00979
         buffer = g_build_filename (input->directory, input->name, NULL);
00980
         gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (dlg), buffer);
00981
         g_free (buffer);
00982
00983
         // Adding XML filter
00984
         filter1 = (GtkFileFilter *) gtk_file_filter_new ();
00985
         gtk_file_filter_set_name (filter1, "XML");
         gtk_file_filter_add_pattern (filterl, "*.xml");
gtk_file_filter_add_pattern (filterl, "*.XML");
00986
00987
         gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter1);
00988
00989
00990
         // Adding JSON filter
00991
         filter2 = (GtkFileFilter *) gtk_file_filter_new ();
00992
         gtk_file_filter_set_name (filter2, "JSON");
        gtk_file_filter_add_pattern (filter2, "*.json");
gtk_file_filter_add_pattern (filter2, "*.json");
gtk_file_filter_add_pattern (filter2, "*.json");
gtk_file_filter_add_pattern (filter2, "*.js");
gtk_file_filter_add_pattern (filter2, "*.js");
00993
00994
00995
00996
00997
         gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter2);
00998
00999
         if (input->type == INPUT_TYPE_XML)
01000
           gtk_file_chooser_set_filter (GTK_FILE_CHOOSER (dlg), filter1);
01001
         else
01002
           gtk file chooser set filter (GTK FILE CHOOSER (dlg), filter2);
01003
01004
         // Connecting the close function
         g_signal_connect (dlg, "response", G_CALLBACK (dialog_save_close), NULL);
01005
01006
01007
         // Showing modal dialog
01008
        gtk_window_present (GTK_WINDOW (dlg));
01009
01010 #if DEBUG_INTERFACE
01011
        fprintf (stderr, "window_save: end\n");
01012 #endif
01013 }
```



5.11.3.38 window_save_climbing()

```
static void window_save_climbing ( ) [static]
```

Function to save the hill climbing method data in the input file.

Definition at line 821 of file interface.c.

```
00822
00823 #if DEBUG INTERFACE
        fprintf (stderr, "window_save_climbing: start\n");
00826
           (gtk_check_button_get_active (window->check_climbing))
00827
00828
             input->nsteps = gtk_spin_button_get_value_as_int (window->spin_steps);
            input >relaxation = gtk_spin_button_get_value (window->spin_relaxation);
switch (window_get_climbing ())
00829
00830
              case CLIMBING_METHOD_COORDINATES:
00832
00833
                input->climbing = CLIMBING_METHOD_COORDINATES;
00834
                break;
00835
               default:
               input->climbing = CLIMBING_METHOD_RANDOM;
input->nestimates
00836
00837
00838
                   = gtk_spin_button_get_value_as_int (window->spin_estimates);
00839
00840
          }
00841 else
          input->nsteps = 0;
00842
00843 #if DEBUG_INTERFACE
00844 fprintf (stderr, "window_save_climbing: end\n");
00845 #endif
00846 }
```

Here is the call graph for this function:



5.11.3.39 window_set_algorithm()

```
static void window_set_algorithm ( ) [static]
```

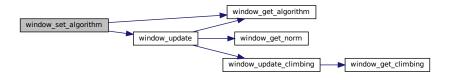
Function to avoid memory errors changing the algorithm.

Definition at line 1314 of file interface.c.

```
01315 {
01316
01317 #if DEBUG_INTERFACE
01318
       fprintf (stderr, "window_set_algorithm: startn");
01319 #endif
       i = window_get_algorithm ();
01320
01321
       switch (i)
01322
01323
         case ALGORITHM_SWEEP:
01324
          case ALGORITHM_ORTHOGONAL:
01325
          i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
            <u>if</u> (i < 0)
01326
01327
              i = 0;
01328
            gtk_spin_button_set_value (window->spin_sweeps,
01329
                                        (gdouble) input->variable[i].nsweeps);
```

```
01330
           break;
01331
          case ALGORITHM_GENETIC:
01332
           i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
           if (i < 0)
01333
01334
             i = 0:
01335
           gtk_spin_button_set_value (window->spin_bits,
01336
                                       (gdouble) input->variable[i].nbits);
01337
01338
       window_update ();
01339 #if DEBUG_INTERFACE
01340 fprintf (stderr, "window_set_algorithm: end\n");
01341 #endif
01342 }
```

Here is the call graph for this function:



5.11.3.40 window_set_experiment()

```
static void window_set_experiment ( ) [static]
```

Function to set the experiment data in the main window.

Definition at line 1348 of file interface.c.

```
01349 {
01350
          unsigned int i, j;
01351
          char *buffer1;
01352 #if DEBUG_INTERFACE
         fprintf (stderr, "window_set_experiment: start\n");
01353
01354 #endif
01355
         i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
          gtk_spin_button_set_value (window->spin_weight, input->experiment[i].weight);
buffer1 = gtk_combo_box_text_get_active_text (window->combo_experiment);
gtk_button_set_label (window->button_experiment, buffer1);
01356
01357
01358
01359
          g_free (buffer1);
01360
          for (j = 0; j < input->experiment->ninputs; ++j)
01361
               g_signal_handler_block (window->button_template[j], window->id_input[j]);
gtk_button_set_label (window->button_template[j],
01362
01363
                                           input->experiment[i].stencil[j]);
01364
01365
               g_signal_handler_unblock
01366
                  (window->button_template[j], window->id_input[j]);
01367
01368 #if DEBUG_INTERFACE
01369 fprintf (stderr, "window_set_experiment: end\n"); 01370 #endif
01371 }
```

5.11.3.41 window_set_variable()

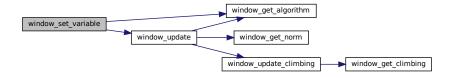
```
static void window_set_variable ( ) [static]
```

Function to set the variable data in the main window.

```
Definition at line 1642 of file interface.c.
```

```
01643 {
01644
        unsigned int i;
01645 #if DEBUG_INTERFACE
        fprintf (stderr, "window_set_variable: start\n");
01646
01647 #endif
        i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01648
        g_signal_handler_block (window->entry_variable, window->id_variable_label);
01649
01650
        gtk_entry_set_text (window->entry_variable, input->variable[i].name);
01651
        g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
        gtk_spin_button_set_value (window->spin_min, input->variable[i].rangemin);
gtk_spin_button_set_value (window->spin_max, input->variable[i].rangemax);
if (input->variable[i].rangeminabs != -G_MAXDOUBLE)
01652
01653
01654
01655
          {
            gtk_spin_button_set_value (window->spin_minabs,
01656
01657
                                         input->variable[i].rangeminabs);
01658
             gtk_check_button_set_active (window->check_minabs, 1);
01659
01660
        else
01661
         {
             gtk_spin_button_set_value (window->spin_minabs, -G_MAXDOUBLE);
01662
01663
            gtk_check_button_set_active (window->check_minabs, 0);
01664
        if (input->variable[i].rangemaxabs != G MAXDOUBLE)
01665
01666
01667
            gtk_spin_button_set_value (window->spin_maxabs,
01668
                                         input->variable[i].rangemaxabs);
01669
            gtk_check_button_set_active (window->check_maxabs, 1);
          1
01670
01671
        else
01672
         {
01673
            gtk_spin_button_set_value (window->spin_maxabs, G_MAXDOUBLE);
01674
            gtk_check_button_set_active (window->check_maxabs, 0);
01675
01676
        gtk_spin_button_set_value (window->spin_precision,
                                     input->variable[i].precision);
01677
01678
        gtk_spin_button_set_value (window->spin_steps, (gdouble) input->nsteps);
01679
        if (input->nsteps)
01680
          gtk_spin_button_set_value (window->spin_step, input->variable[i].step);
01681 #if DEBUG_INTERFACE
01682 fprintf (stderr, "window_set_variable: precision[u]=u^n, i,
01683
                  input->variable[i].precision);
01684 #endif
01685
       switch (window_get_algorithm ())
01686
         {
01687
          case ALGORITHM_SWEEP:
01688
          case ALGORITHM_ORTHOGONAL:
01689
            gtk_spin_button_set_value (window->spin_sweeps,
01690
                                         (gdouble) input->variable[i].nsweeps);
01691 #if DEBUG INTERFACE
           fprintf (stderr, "window_set_variable: nsweeps[%u]=%u\n", i,
01692
01693
                      input->variable[i].nsweeps);
01694 #endif
            break;
01695
          case ALGORITHM_GENETIC:
01696
01697
            gtk_spin_button_set_value (window->spin_bits,
01698
                                         (gdouble) input->variable[i].nbits);
01699 #if DEBUG_INTERFACE
01700
            fprintf (stderr, "window_set_variable: nbits[%u]=%u\n", i,
01701
                      input->variable[i].nbits);
01702 #endif
       break;
}
01703
01704
        window_update ();
01705
01706 #if DEBUG_INTERFACE
01707 fprintf (stderr, "window_set_variable: end\n");
01708 #endif
01709 }
```

Here is the call graph for this function:



5.11.3.42 window step variable()

```
static void window_step_variable ( ) [static]
```

Function to update the variable step in the main window.

Definition at line 1897 of file interface.c.

```
01898 {
01899    unsigned int i;
01900    #if DEBUG_INTERFACE
01901    i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01903    i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01904    input->variable[i].step = gtk_spin_button_get_value (window->spin_step);
01905    #if DEBUG_INTERFACE
01906    fprintf (stderr, "window_step_variable: end\n");
01907    #endif
01908 }
```

5.11.3.43 window_template_experiment()

Function to update the experiment i-th input template in the main window.

Parameters

```
data Callback data (i-th input template).
```

Definition at line 1605 of file interface.c.

```
01607 {
01608
       GtkFileChooserDialog *dlg;
01609
       GMainLoop *loop;
       const char *buffer;
01610
01611
       unsigned int i;
01612 #if DEBUG_INTERFACE
       fprintf (stderr, "window_template_experiment: start\n");
01613
01614 #endif
01615
       i = (size_t) data;
       buffer = gtk_button_get_label (window->button_template[i]);
01616
       dlg = (GtkFileChooserDialog *)
01617
01618
         gtk_file_chooser_dialog_new (_("Open template file"),
01619
                                       window->window,
```

```
01620
                                         GTK_FILE_CHOOSER_ACTION_OPEN,
                                         _("_Cancel"),
GTK_RESPONSE_CANCEL,
01621
01622
                                          _("_Open"), GTK_RESPONSE_ACCEPT, NULL);
01623
        gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (dlg), buffer);
01624
        g_signal_connect (dlg, "response",

G_CALLBACK (window_template_experiment_close), data);
01625
01626
01627
        gtk_window_present (GTK_WINDOW (dlg));
01628
        loop = g_main_loop_new (NULL, 0);
01629
        g_signal_connect_swapped (dlg, "destroy", G_CALLBACK (g_main_loop_quit),
01630
                                    loop);
        g_main_loop_run (loop);
01631
01632
        g_main_loop_unref (loop);
01633 #if DEBUG_INTERFACE
01634
        fprintf (stderr, "window_template_experiment: end\n");
01635 #endif
01636 }
```

Here is the call graph for this function:



5.11.3.44 window_template_experiment_close()

Function to close the experiment template dialog.

Parameters

dlg	GtkFileChooserDialg struct.
response <i>←</i> _id	Response identifier.
data	Function data.

Definition at line 1565 of file interface.c.

```
01570 {
01571
        GFile *file1, *file2;
01572
        char *buffer1, *buffer2;
01573 unsigned int i, j;
01574 #if DEBUG_INTERFACE
01575
        fprintf (stderr, "window_template_experiment_close: start\n");
01576 #endif
01577
        if (response_id == GTK_RESPONSE_OK)
01578
01579
             i = (size_t) data;
             j = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01580
             buffer1 = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
file1 = g_file_new_for_path (buffer1);
01581
01583
             file2 = g_file_new_for_path (input->directory);
01584
             buffer2 = g_file_get_relative_path (file2, file1);
01585
             if (input->type == INPUT_TYPE_XML)
```

```
input->experiment[j].stencil[i]
                 = (char *) xmlStrdup ((xmlChar *) buffer2);
01587
01588
01589
               input->experiment[j].stencil[i] = g_strdup (buffer2);
01590
             g free (buffer2);
             g_free (buffer2);
g_object_unref (file2);
g_object_unref (file1);
01591
01592
01593
             g_free (buffer1);
01594
01595
        gtk_window_destroy (GTK_WINDOW (dlg));
01596 #if DEBUG_INTERFACE
       fprintf (stderr, "window_template_experiment_close: end\n");
01597
01598 #endif
01599 }
```

5.11.3.45 window_update()

```
static void window_update ( ) [static]
```

Function to update the main window view.

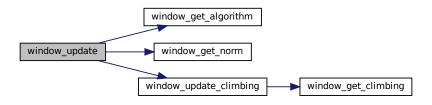
Definition at line 1162 of file interface.c.

```
01163 {
01164
       unsigned int i;
01165 #if DEBUG INTERFACE
       fprintf (stderr, "window_update: start\n");
01166
01167 #endif
       gtk_widget_set_sensitive
01169
         (GTK_WIDGET (window->button_evaluator),
01170
           gtk_check_button_get_active (window->check_evaluator));
01171
       gtk_widget_hide (GTK_WIDGET (window->label_simulations));
       gtk_widget_hide (GTK_WIDGET (window->spin simulations));
01172
01173
       gtk_widget_hide (GTK_WIDGET (window->label_iterations));
01174
       gtk_widget_hide (GTK_WIDGET (window->spin_iterations));
01175
       gtk_widget_hide (GTK_WIDGET (window->label_tolerance));
01176
       gtk_widget_hide (GTK_WIDGET (window->spin_tolerance));
01177
       gtk_widget_hide (GTK_WIDGET (window->label_bests));
       gtk_widget_hide (GTK_WIDGET (window->spin_bests));
01178
01179
       gtk_widget_hide (GTK_WIDGET (window->label_population));
01180
       gtk_widget_hide (GTK_WIDGET (window->spin_population));
       gtk_widget_hide (GTK_WIDGET (window->label_generations));
01181
01182
        gtk_widget_hide (GTK_WIDGET (window->spin_generations));
01183
        gtk_widget_hide (GTK_WIDGET (window->label_mutation));
       gtk_widget_hide (GTK_WIDGET (window->spin_mutation));
01184
       gtk_widget_hide (GTK_WIDGET (window->label_reproduction));
01185
01186
       gtk_widget_hide (GTK_WIDGET (window->spin_reproduction));
       gtk_widget_hide (GTK_WIDGET (window->label_adaptation));
01187
01188
       gtk_widget_hide (GTK_WIDGET (window->spin_adaptation));
01189
        gtk_widget_hide (GTK_WIDGET (window->label_sweeps));
01190
       gtk_widget_hide (GTK_WIDGET (window->spin_sweeps));
gtk_widget_hide (GTK_WIDGET (window->label_bits));
01191
01192
       gtk_widget_hide (GTK_WIDGET (window->spin_bits));
01193
       gtk_widget_hide (GTK_WIDGET (window->check_climbing));
01194
       gtk_widget_hide (GTK_WIDGET (window->grid_climbing));
01195
        gtk_widget_hide (GTK_WIDGET (window->label_step));
01196
        gtk_widget_hide (GTK_WIDGET (window->spin_step));
       gtk_widget_hide (GTK_WIDGET (window->label_p));
01197
       gtk_widget_hide (GTK_WIDGET (window->spin_p));
01198
01199
        i = gtk_spin_button_get_value_as_int (window->spin_iterations);
01200
       switch (window_get_algorithm ())
01201
01202
          case ALGORITHM MONTE CARLO:
            gtk_widget_show (GTK_WIDGET (window->label_simulations));
01203
01204
            gtk_widget_show (GTK_WIDGET (window->spin_simulations));
01205
            gtk_widget_show (GTK_WIDGET (window->label_iterations));
            gtk_widget_show (GTK_WIDGET (window->spin_iterations));
01206
01207
            if (i > 1)
01208
                gtk_widget_show (GTK_WIDGET (window->label_tolerance));
01209
                gtk_widget_show (GTK_WIDGET (window->spin_tolerance));
01210
                gtk_widget_show (GTK_WIDGET (window->label_bests));
01211
01212
                gtk_widget_show (GTK_WIDGET (window->spin_bests));
01213
01214
            window_update_climbing ();
01215
            break:
01216
          case ALGORITHM_SWEEP:
01217
          case ALGORITHM_ORTHOGONAL:
01218
            gtk_widget_show (GTK_WIDGET (window->label_iterations));
```

```
01219
            gtk_widget_show (GTK_WIDGET (window->spin_iterations));
01220
01221
              {
01222
                gtk_widget_show (GTK_WIDGET (window->label_tolerance));
                gtk_widget_show (GTK_WIDGET (window->spin_tolerance));
gtk_widget_show (GTK_WIDGET (window->label_bests));
01223
01224
                gtk_widget_show (GTK_WIDGET (window->spin_bests));
01225
01226
01227
            gtk_widget_show (GTK_WIDGET (window->label_sweeps));
01228
            gtk_widget_show (GTK_WIDGET (window->spin_sweeps));
            gtk_widget_show (GTK_WIDGET (window->check_climbing));
01229
01230
            window_update_climbing ();
01231
            break;
01232
01233
            gtk_widget_show (GTK_WIDGET (window->label_population));
01234
            gtk_widget_show (GTK_WIDGET (window->spin_population));
            gtk_widget_show (GTK_WIDGET (window->label_generations));
01235
            gtk_widget_show (GTK_WIDGET (window->spin_generations));
01236
            gtk_widget_show (GTK_WIDGET (window->label_mutation));
01238
            gtk_widget_show (GTK_WIDGET (window->spin_mutation));
01239
            gtk_widget_show (GTK_WIDGET (window->label_reproduction));
01240
            gtk_widget_show (GTK_WIDGET (window->spin_reproduction));
01241
            gtk_widget_show (GTK_WIDGET (window->label_adaptation));
            gtk_widget_show (GTK_WIDGET (window->spin_adaptation));
01242
01243
            gtk_widget_show (GTK_WIDGET (window->label_bits));
            gtk_widget_show (GTK_WIDGET (window->spin_bits));
01244
01245
01246
        {\tt gtk\_widget\_set\_sensitive}
01247
          (GTK_WIDGET (window->button_remove_experiment), input->nexperiments > 1);
01248
        gtk_widget_set_sensitive
01249
         (GTK WIDGET (window->button remove variable), input->nvariables > 1);
01250
        for (i = 0; i < input->experiment->ninputs; ++i)
01251
01252
            gtk_widget_show (GTK_WIDGET (window->check_template[i]));
01253
            gtk_widget_show (GTK_WIDGET (window->button_template[i]));
            gtk_widget_set_sensitive (GTK_WIDGET (window->check_template[i]), 0);
01254
            gtk_widget_set_sensitive (GTK_WIDGET (window->button_template[i]), 1);
01255
01256
            g_signal_handler_block
01257
              (window->check_template[i], window->id_template[i]);
01258
            g_signal_handler_block (window->button_template[i], window->id_input[i]);
01259
            gtk_check_button_set_active (window->check_template[i], 1);
            g_signal_handler_unblock (window->button_template[i],
01260
                                        window->id_input[i]);
01261
01262
            g_signal_handler_unblock (window->check_template[i],
01263
                                        window->id template[i]);
01264
01265
        if (i > 0)
01266
         {
            gtk widget set sensitive (GTK WIDGET (window->check template[i - 1]), 1);
01267
            gtk_widget_set_sensitive (GTK_WIDGET (window->button_template[i - 1]),
01268
                                        gtk_check_button_get_active
01270
                                        (window->check_template[i - 1]));
01271
        if (i < MAX_NINPUTS)</pre>
01272
01273
01274
            gtk widget show (GTK WIDGET (window->check template[i]));
            gtk_widget_show (GTK_WIDGET (window->button_template[i]));
            gtk_widget_set_sensitive (GTK_WIDGET (window->check_template[i]), 1);
01276
01277
            {\tt gtk\_widget\_set\_sensitive}
01278
               (GTK_WIDGET (window->button_template[i]),
               gtk_check_button_get_active (window->check_template[i]));
01279
01280
            g_signal_handler_block
01281
               (window->check_template[i], window->id_template[i]);
            g_signal_handler_block (window->button_template[i], window->id_input[i]);
01282
01283
            gtk_check_button_set_active (window->check_template[i], 0);
01284
            g_signal_handler_unblock (window->button_template[i],
01285
                                        window->id_input[i]);
01286
            g signal handler unblock (window->check template[i].
01287
                                        window->id_template[i]);
01288
        while (++i < MAX_NINPUTS)</pre>
01289
01290
            gtk_widget_hide (GTK_WIDGET (window->check_template[i]));
gtk_widget_hide (GTK_WIDGET (window->button_template[i]));
01291
01292
01293
01294
        gtk_widget_set_sensitive
01295
          (GTK_WIDGET (window->spin_minabs),
01296
           gtk_check_button_get_active (window->check_minabs));
01297
        {\tt gtk\_widget\_set\_sensitive}
01298
          (GTK WIDGET (window->spin maxabs).
01299
           gtk_check_button_get_active (window->check_maxabs));
01300
        if (window_get_norm () == ERROR_NORM_P)
01301
01302
            gtk_widget_show (GTK_WIDGET (window->label_p));
01303
            gtk_widget_show (GTK_WIDGET (window->spin_p));
01304
01305 #if DEBUG_INTERFACE
```

```
01306 fprintf (stderr, "window_update: end\n"); 01307 #endif 01308 }
```

Here is the call graph for this function:



5.11.3.46 window update climbing()

```
static void window_update_climbing ( ) [static]
```

Function to update hill climbing method widgets view in the main window.

Definition at line 1131 of file interface.c.

```
01132
01133 #if DEBUG_INTERFACE
01134
        fprintf (stderr, "window_update_climbing: start\n");
01135 #endif
01136
        gtk_widget_show (GTK_WIDGET (window->check_climbing));
01137
         if (gtk_check_button_get_active (window->check_climbing))
         {
01138
             gtk_widget_show (GTK_WIDGET (window->grid_climbing));
gtk_widget_show (GTK_WIDGET (window->label_step));
gtk_widget_show (GTK_WIDGET (window->spin_step));
01139
01141
01142
01143
        switch (window_get_climbing ())
01144
01145
           case CLIMBING_METHOD_COORDINATES:
01146
           gtk_widget_hide (GTK_WIDGET (window->label_estimates));
01147
             gtk_widget_hide (GTK_WIDGET (window->spin_estimates));
01148
01149
           default:
            gtk_widget_show (GTK_WIDGET (window->label_estimates));
01150
             gtk_widget_show (GTK_WIDGET (window->spin_estimates));
01151
01152
01153 #if DEBUG_INTERFACE
01154
       fprintf (stderr, "window_update_climbing: end\n");
01155 #endif
01156 }
```



5.11.3.47 window_update_variable()

```
static void window_update_variable ( ) [static]
```

Function to update the variable data in the main window.

Definition at line 1914 of file interface.c.

```
01916
01917 #if DEBUG_INTERFACE
      fprintf (stderr, "window_update_variable: start\n");
01918
01919 #endif
01920 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
       <u>if</u> (i < 0)
01921
01922
        i = 0;
01923
       switch (window_get_algorithm ())
01924
         case ALGORITHM_SWEEP:
01925
        case ALGORITHM ORTHOGONAL:
01926
01927
         input->variable[i].nsweeps
01928
             = gtk_spin_button_get_value_as_int (window->spin_sweeps);
01929 #if DEBUG_INTERFACE
01930
         fprintf (stderr, "window_update_variable: nsweeps[%d]=%u\n", i,
01931
                   input->variable[i].nsweeps);
01932 #endif
01933
          break:
01934
         case ALGORITHM_GENETIC:
        input->variable[i].nbits
01935
01936
             = gtk_spin_button_get_value_as_int (window->spin_bits);
01937 #if DEBUG_INTERFACE
      01938
01939
01940 #endif
01941
01942 #if DEBUG_INTERFACE
      fprintf (stderr, "window_update_variable: end\n");
01943
01944 #endif
01945 }
```

Here is the call graph for this function:



5.11.3.48 window_weight_experiment()

```
static void window_weight_experiment ( ) [static]
```

Function to update the experiment weight in the main window.

Definition at line 1526 of file interface.c.

```
01527 {
01528    unsigned int i;
01529    #if DEBUG_INTERFACE
01530    fprintf (stderr, "window_weight_experiment: start\n");
01531    #endif
01532    i = gtk_combo_box_get_active (GTK_COMBo_BOX (window->combo_experiment));
01533    input->experiment[i].weight = gtk_spin_button_get_value (window->spin_weight);
01534    #if DEBUG_INTERFACE
01535    fprintf (stderr, "window_weight_experiment: end\n");
01536    #endif
01537 }
```

5.11.4 Variable Documentation

5.11.4.1 logo

```
const char* logo[] [static]
```

Logo pixmap.

Definition at line 84 of file interface.c.

5.11.4.2 options

```
Options options[1] [static]
```

Options struct to define the options dialog.

Definition at line 163 of file interface.c.

5.11.4.3 running

```
Running running[1] [static]
```

Running struct to define the running dialog.

Definition at line 165 of file interface.c.

5.11.4.4 window

```
Window window[1]
```

Window struct to define the main interface window.

Definition at line 81 of file interface.c.

5.12 interface.c 193

5.12 interface.c

Go to the documentation of this file.

```
00001 /*
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00009
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are met:
00012
00013 1. Redistributions of source code must retain the above copyright notice,
{\tt 00014} this list of conditions and the following disclaimer.
00015
00016 2. Redistributions in binary form must reproduce the above copyright notice, 00017 this list of conditions and the following disclaimer in the \,
00018 documentation and/or other materials provided with the distribution.
00019
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF 00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN 00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING 00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
00031
00038 #define _GNU_SOURCE
00039 #include "config.h"
00040 #include <stdio.h>
00041 #include <stdlib.h>
00042 #include <string.h>
00043 #include <math.h>
00044 #include <gsl/gsl_rng.h>
00045 #include ibxml/parser.h>
00046 #include <libintl.h>
00047 #include <glib.h>
00048 #include <glib/gstdio.h>
00049 #include <json-glib/json-glib.h>
00050 #ifdef G_OS_WIN32
00051 #include <windows.h>
00052 #endif
00053 #if HAVE_MPI
00054 #include <mpi.h>
00055 #endif
00056 #include <gio/gio.h>
00057 #include <gtk/gtk.h>
00058 #include "jb/src/jb_xml.h"
00059 #include "jb/src/jb_json.h"
00060 #include "jb/src/jb_win.h"
00061 #include "genetic/genetic.h"
00062 #include "tools.h"
00063 #include "experiment.h"
00064 #include "variable.h"
00065 #include "input.h"
00066 #include "optimize.h"
00067 #include "interface.h"
00068
00069 #define DEBUG INTERFACE 1
00070
00075 #ifdef G_OS_WIN32
00076 #define INPUT_FILE "test-ga-win.xml"
00077 #else
00078 #define INPUT_FILE "test-ga.xml"
00079 #endif
08000
00081 Window window[1]:
00083
00084 static const char *logo[] = {
00085
       "32 32 3 1",
00086
                c None"
                     c #0000FF".
00087
                c #FF0000",
00088
00089
00090
00091
00092
00093
```

```
00095
00096
00097
                                        +++++
00098
                                        +++++
00099
00100
00101
             ++++
00102
             +++++
00103
             +++++
00104
              +++
00105
00106
00107
                          ++++
00108
                         +++++
00109
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121 };
00122
00123 /*
00124 const char * logo[] = {
00124 consc char \( \times \)
00125 "32 32 3 1",
00126 " c #FFF!
            c #FFFFFFFFFFF,
00127 ".
                c #0000000FFFF",
00127 .
00128 "X
00129 "
             c #FFFF00000000",
00130 "
00131 "
00132 "
00133 "
00134 "
00135 "
00136 "
                                        XXX
00137 "
                                      XXXXX
00138 "
                                      XXXXX
00139 "
                                      XXXXX
00140 "
            XXX
                           XXX
                                   XXX
00141 "
                                       XXXXX
           XXXXX
00142 "
           XXXXX
                                       XXXXX
00143 "
           XXXXX
                                       XXXXX
00144 "
00145 "
            XXX
                                         XXX
00146 "
                         XXX
00147 "
                       XXXXX
00148 "
                       XXXXX
00149 "
00150 "
                       XXXXX
                         XXX
00151 "
00152 "
00153 "
00154 "
00155 "
00156 "
00157 "
00158 "
00159 "
00160 "
00161 */
00162
00163 static Options options[1];
00165 static Running running[1];
00167
00171 static void
00172 input_save_climbing_xml (xmlNode * node)
00173 {
00174 #if DEBUG_INTERFACE
00175
        fprintf (stderr, "input_save_climbing_xml: start\n");
00176 #endif
00177
        if (input->nsteps)
00178
00179
             jb_xml_node_set_uint (node, (const xmlChar *) LABEL_NSTEPS,
                                     input->nsteps);
00180
             if (input->relaxation != DEFAULT_RELAXATION)
00181
00182
               jb_xml_node_set_float (node, (const xmlChar *) LABEL_RELAXATION,
                                         input->relaxation);
00183
00184
             switch (input->climbing)
               {
00185
```

5.12 interface.c 195

```
case CLIMBING_METHOD_COORDINATES:
              xmlSetProp (node, (const xmlChar *) LABEL_CLIMBING,
00187
00188
                            (const xmlChar *) LABEL_COORDINATES);
00189
               break:
00190
              default:
              00191
00192
00193
               jb_xml_node_set_uint (node, (const xmlChar *) LABEL_NESTIMATES,
00194
                                     input->nestimates);
00195
00196
00197 #if DEBUG_INTERFACE
00198
       fprintf (stderr, "input_save_climbing_xml: end\n");
00199 #endif
00200 }
00201
00205 static void
00206 input_save_climbing_json (JsonNode * node)
00208
       JsonObject *object;
00209 #if DEBUG_INTERFACE
       fprintf (stderr, "input_save_climbing_json: start\n");
00210
00211 #endif
       object = json_node_get_object (node);
00212
00213
       if (input->nsteps)
00214
00215
            jb_json_object_set_uint (object, LABEL_NSTEPS, input->nsteps);
00216
            if (input->relaxation != DEFAULT_RELAXATION)
00217
              jb_json_object_set_float (object, LABEL_RELAXATION, input->relaxation);
00218
            switch (input->climbing)
00219
             {
00220
             case CLIMBING_METHOD_COORDINATES:
00221
              json_object_set_string_member (object, LABEL_CLIMBING,
00222
                                              LABEL_COORDINATES);
00223
00224
              default:
               json_object_set_string_member (object, LABEL_CLIMBING, LABEL_RANDOM);
jb_json_object_set_uint (object, LABEL_NESTIMATES, input->nestimates);
00225
00227
00228
00229 #if DEBUG_INTERFACE
00230 fprintf (stderr, "input_save_climbing_json: end\n");
00231 #endif
00232 }
00233
00237 static inline void
00238 input_save_xml (xmlDoc * doc)
00239 {
       unsigned int i, j;
00240
00241
       char *buffer:
00242
       xmlNode *node, *child;
00243
       GFile *file, *file2;
00244
00247 #endif
00248
00249
        // Setting root XML node
00250
       node = xmlNewDocNode (doc, 0, (const xmlChar *) LABEL_OPTIMIZE, 0);
00251
       xmlDocSetRootElement (doc, node);
00252
00253
       // Adding properties to the root XML node
00254
       if (xmlStrcmp
00255
            ((const xmlChar *) input->result, (const xmlChar *) result_name))
00256
          xmlSetProp (node, (const xmlChar *) LABEL_RESULT_FILE,
00257
                      (xmlChar *) input->result);
00258
       if (xmlStrcmp
00259
            ((const xmlChar *) input->variables, (const xmlChar *) variables_name))
         xmlSetProp (node, (const xmlChar *) LABEL_VARIABLES_FILE,
00260
00261
                     (xmlChar *) input->variables);
00262
       file = g_file_new_for_path (input->directory);
00263
       file2 = g_file_new_for_path (input->simulator);
       buffer = g_file_get_relative_path (file, file2);
00264
00265
        q_object_unref (file2);
00266
       xmlSetProp (node, (const xmlChar *) LABEL_SIMULATOR, (xmlChar *) buffer);
00267
        g_free (buffer);
00268
        if (input->evaluator)
00269
00270
           file2 = g_file_new_for_path (input->evaluator);
00271
           buffer = g_file_get_relative_path (file, file2);
00272
            g_object_unref (file2);
00273
            if (xmlStrlen ((xmlChar *) buffer))
00274
              xmlSetProp (node, (const xmlChar *) LABEL_EVALUATOR,
00275
                          (xmlChar *) buffer);
00276
            g_free (buffer);
00277
00278
       if (input->seed != DEFAULT_RANDOM_SEED)
```

```
jb_xml_node_set_uint (node, (const xmlChar *) LABEL_SEED, input->seed);
00280
00281
         // Setting the algorithm
00282
         buffer = (char *) g_slice_alloc (64);
         switch (input->algorithm)
00283
00284
           case ALGORITHM_MONTE_CARLO:
00286
              xmlSetProp (node, (const xmlChar *) LABEL_ALGORITHM,
              (const xmlChar *) LABEL_MONTE_CARLO);
snprintf (buffer, 64, "%u", input->nsimulations);
00287
00288
              xmlSetProp (node, (const xmlChar *) LABEL_NSIMULATIONS,
00289
             (xmlChar *) buffer);
snprintf (buffer, 64, "%u", input->niterations);
xmlSetProp (node, (const xmlChar *) LABEL_NITERATIONS,
00290
00291
00292
00293
                            (xmlChar *) buffer);
00294
              snprintf (buffer, 64, "%.31g", input->tolerance);
              xmlSetProp (node, (const xmlChar *) LABEL_TOLERANCE, (xmlChar *) buffer);
00295
              xmllsetProp (node, (const xmlChar *) LABEL_NBEST, (xmlChar *) buffer);
xmlSetProp (node, (const xmlChar *) LABEL_NBEST, (xmlChar *) buffer);
00296
00298
              input_save_climbing_xml (node);
00299
             break;
00300
           case ALGORITHM SWEEP:
             00301
00302
00303
00304
00305
                            (xmlChar *) buffer);
00306
              snprintf (buffer, 64, "%.31g", input->tolerance);
00307
              xmlSetProp (node, (const xmlChar *) LABEL_TOLERANCE, (xmlChar *) buffer);
              snprintf (buffer, 64, "%u", input->nbest);
00308
              xmlSetProp (node, (const xmlChar *) LABEL_NBEST, (xmlChar *) buffer);
00309
00310
              input_save_climbing_xml (node);
00311
              break;
00312
           case ALGORITHM_ORTHOGONAL:
00313
              xmlSetProp (node, (const xmlChar *) LABEL_ALGORITHM,
              (const xmlChar *) LABEL_ORTHOGONAL);
snprintf (buffer, 64, "%u", input->niterations);
00314
00315
              xmlSetProp (node, (const xmlChar *) LABEL_NITERATIONS,
00317
                            (xmlChar *) buffer);
00318
              snprintf (buffer, 64, "%.31g", input->tolerance);
             xmlSetProp (node, (const xmlChar *) LABEL_TOLERANCE, (xmlChar *) buffer);
snprintf (buffer, 64, "%u", input->nbest);
xmlSetProp (node, (const xmlChar *) LABEL_NBEST, (xmlChar *) buffer);
input_save_climbing_xml (node);
00319
00320
00321
00322
00323
              break;
00324
           default:
00325
             xmlSetProp (node, (const xmlChar *) LABEL_ALGORITHM,
              (const xmlChar *) LABEL_GENETIC);
snprintf (buffer, 64, "%u", input->nsimulations);
00326
00327
             xmlSetProp (node, (const xmlChar *) LABEL_NPOPULATION,
00328
00329
                            (xmlChar *) buffer);
00330
              snprintf (buffer, 64, "%u", input->niterations);
00331
              xmlSetProp (node, (const xmlChar *) LABEL_NGENERATIONS,
              (xmlChar *) buffer);
snprintf (buffer, 64, "%.31g", input->mutation_ratio);
00332
00333
00334
              xmlSetProp (node, (const xmlChar *) LABEL_MUTATION, (xmlChar *) buffer);
              snprintf (buffer, 64, "%.31g", input->reproduction_ratio);
00336
              xmlSetProp (node, (const xmlChar *) LABEL_REPRODUCTION,
              (xmlChar *) buffer);
snprintf (buffer, 64, "%.31g", input->adaptation_ratio);
xmlSetProp (node, (const xmlChar *) LABEL_ADAPTATION, (xmlChar *) buffer);
00337
00338
00339
00340
             break;
00341
00342
         g_slice_free1 (64, buffer);
00343
          if (input->threshold != 0.)
00344
           jb_xml_node_set_float (node, (const xmlChar *) LABEL_THRESHOLD,
00345
                                      input->threshold);
00346
00347
         // Setting the experimental data
         for (i = 0; i < input->nexperiments; ++i)
00348
00349
00350
              child = xmlNewChild (node, 0, (const xmlChar *) LABEL_EXPERIMENT, 0);
              xmlSetProp (child, (const xmlChar *) LABEL_NAME,
00351
             (xm1Char *) input->experiment[i].name);
if (input->experiment[i].weight != 1.)
00352
00353
                jb_xml_node_set_float (child, (const xmlChar *) LABEL_WEIGHT,
00354
00355
                                           input->experiment[i].weight);
              for (j = 0; j < input->experiment->ninputs; ++j)
00356
               xmlSetProp (child, (const xmlChar *) stencil[j],
00357
00358
                              (xmlChar *) input->experiment[i].stencil[j]);
00359
00360
00361
         // Setting the variables data
00362
         for (i = 0; i < input->nvariables; ++i)
00363
             child = xmlNewChild (node, 0, (const xmlChar *) LABEL_VARIABLE, 0);
xmlSetProp (child, (const xmlChar *) LABEL_NAME,
00364
00365
```

5.12 interface.c 197

```
(xmlChar *) input->variable[i].name);
00367
            jb_xml_node_set_float (child, (const xmlChar *) LABEL_MINIMUM,
00368
                                  input->variable[i].rangemin);
            if (input->variable[i].rangeminabs != -G_MAXDOUBLE)
00369
             00370
00371
            jb_xml_node_set_float (child, (const xmlChar *) LABEL_MAXIMUM,
00372
00373
                                  input->variable[i].rangemax);
00374
            if (input->variable[i].rangemaxabs != G_MAXDOUBLE)
00375
              jb_xml_node_set_float (child, (const xmlChar *) LABEL_ABSOLUTE_MAXIMUM,
           input->variable[i].rangemaxabs);
if (input->variable[i].precision != DEFAULT_PRECISION)
00376
00377
             jb_xml_node_set_uint (child, (const xmlChar *) LABEL_PRECISION,
00378
00379
                                    input->variable[i].precision);
00380
            if (input->algorithm == ALGORITHM_SWEEP
00381
               || input->algorithm == ALGORITHM_ORTHOGONAL)
             00382
00383
           else if (input->algorithm == ALGORITHM_GENETIC)
00384
             jb_xml_node_set_uint (child, (const xmlChar *) LABEL_NBITS,
00385
00386
                                   input->variable[i].nbits);
00387
            if (input->nsteps)
             jb_xml_node_set_float (child, (const xmlChar *) LABEL_STEP,
00388
                                     input->variable[i].step);
00389
00390
00391
00392
       // Saving the error norm
00393
       switch (input->norm)
00394
00395
         case ERROR NORM MAXIMUM:
           xmlSetProp (node, (const xmlChar *) LABEL_NORM,
00396
00397
                        (const xmlChar *) LABEL_MAXIMUM);
00398
00399
         case ERROR_NORM_P:
00400
           xmlSetProp (node, (const xmlChar *) LABEL_NORM,
                       (const xmlChar *) LABEL_P);
00401
            jb_xml_node_set_float (node, (const xmlChar *) LABEL_P, input->p);
00402
           break;
00404
         case ERROR_NORM_TAXICAB:
00405
          xmlSetProp (node, (const xmlChar *) LABEL_NORM,
00406
                        (const xmlChar *) LABEL_TAXICAB);
00407
         }
00408
00409 #if DEBUG_INTERFACE
00410 fprintf (stderr, "input_save: end\n");
00411 #endif
00412 }
00413
00417 static inline void
00418 input_save_json (JsonGenerator * generator)
00419 {
00420
       unsigned int i, j;
00421
       char *buffer;
00422
       JsonNode *node, *child;
       JsonObject *object;
JsonArray *array;
00423
00424
       GFile *file, *file2;
00425
00426
00427 #if DEBUG_INTERFACE
00428 fprintf (stderr, "input_save_json: start\n");
00429 #endif
00430
00431
        // Setting root JSON node
       object = json_object_new ();
00432
00433
       node = json_node_new (JSON_NODE_OBJECT);
00434
        json_node_set_object (node, object);
00435
        json_generator_set_root (generator, node);
00436
00437
        // Adding properties to the root JSON node
00438
       if (strcmp (input->result, result_name))
00439
         json_object_set_string_member (object, LABEL_RESULT_FILE, input->result);
00440
       if (strcmp (input->variables, variables_name))
         json_object_set_string_member (object, LABEL_VARIABLES_FILE,
00441
00442
                                        input->variables);
00443
       file = g_file_new_for_path (input->directory);
       file2 = g_file_new_for_path (input->simulator);
00444
00445
       buffer = g_file_get_relative_path (file, file2);
00446
       g_object_unref (file2);
00447
        json_object_set_string_member (object, LABEL_SIMULATOR, buffer);
00448
        g_free (buffer):
00449
       if (input->evaluator)
00450
00451
            file2 = g_file_new_for_path (input->evaluator);
00452
            buffer = g_file_get_relative_path (file, file2);
00453
            g_object_unref (file2);
00454
            if (strlen (buffer))
00455
             json_object_set_string_member (object, LABEL_EVALUATOR, buffer);
```

```
g_free (buffer);
00457
00458
          if (input->seed != DEFAULT_RANDOM_SEED)
00459
            jb_json_object_set_uint (object, LABEL_SEED, input->seed);
00460
00461
          // Setting the algorithm
          buffer = (char *) g_slice_alloc (64);
00462
00463
          switch (input->algorithm)
00464
00465
            case ALGORITHM MONTE CARLO:
               json_object_set_string_member (object, LABEL_ALGORITHM,
00466
                                                     LABEL_MONTE_CARLO);
00467
               snprintf (buffer, 64, "%u", input->nsimulations);
00468
               json_object_set_string_member (object, LABEL_NSIMULATIONS, buffer);
00469
00470
               snprintf (buffer, 64, "%u", input->niterations);
              json_object_set_string_member (object, LABEL_NITERATIONS, buffer);
snprintf (buffer, 64, "%.31g", input->tolerance);
json_object_set_string_member (object, LABEL_TOLERANCE, buffer);
snprintf (buffer, 64, "%u", input->nbest);
00471
00472
00473
00475
               json_object_set_string_member (object, LABEL_NBEST, buffer);
00476
               input_save_climbing_json (node);
               break;
00477
            case ALGORITHM_SWEEP:
00478
               json_object_set_string_member (object, LABEL_ALGORITHM, LABEL_SWEEP);
snprintf (buffer, 64, "%u", input->niterations);
00479
00480
               json_object_set_string_member (object, LABEL_NITERATIONS, buffer);
00481
               snprintf (buffer, 64, "%.31g", input->tolerance);
json_object_set_string_member (object, LABEL_TOLERANCE, buffer);
snprintf (buffer, 64, "%u", input->nbest);
00482
00483
00484
               json_object_set_string_member (object, LABEL_NBEST, buffer);
00485
00486
               input save climbing ison (node):
00487
               break;
00488
             case ALGORITHM_ORTHOGONAL:
00489
               json_object_set_string_member (object, LABEL_ALGORITHM, LABEL_ORTHOGONAL);
               snprintf (buffer, 64, "%u", input->niterations);
json_object_set_string_member (object, LABEL_NITERATIONS, buffer);
snprintf (buffer, 64, "%.31g", input->tolerance);
00490
00491
00492
               json_object_set_string_member (object, LABEL_TOLERANCE, buffer);
00494
               snprintf (buffer, 64, "%u", input->nbest);
00495
               json_object_set_string_member (object, LABEL_NBEST, buffer);
00496
               input_save_climbing_json (node);
00497
              break;
00498
            default:
00499
               json_object_set_string_member (object, LABEL_ALGORITHM, LABEL_GENETIC);
               snprintf (buffer, 64, "%u", input->nsimulations);
00500
00501
               json_object_set_string_member (object, LABEL_NPOPULATION, buffer);
00502
               snprintf (buffer, 64, "%u", input->niterations);
               json_object_set_string_member (object, LABEL_NGENERATIONS, buffer); snprintf (buffer, 64, "%.31g", input->mutation_ratio); json_object_set_string_member (object, LABEL_MUTATION, buffer); snprintf (buffer, 64, "%.31g", input->reproduction_ratio); json_object_set_string_member (object, LABEL_REPRODUCTION, buffer);
00503
00504
00505
00506
00507
               snprintf (buffer, 64, "%.31g", input->adaptation_ratio);
json_object_set_string_member (object, LABEL_ADAPTATION, buffer);
00508
00509
00510
               break:
00511
00512
         g_slice_free1 (64, buffer);
00513
             (input->threshold != 0.)
00514
             jb_json_object_set_float (object, LABEL_THRESHOLD, input->threshold);
00515
00516
          // Setting the experimental data
00517
         array = json_array_new ();
00518
          for (i = 0; i < input->nexperiments; ++i)
00519
00520
               child = json_node_new (JSON_NODE_OBJECT);
00521
               object = json_node_get_object (child);
               json_object_set_string_member (object, LABEL_NAME,
00522
00523
                                                      input->experiment[i].name);
               if (input->experiment[i].weight != 1.)
00524
00525
                 jb_json_object_set_float (object, LABEL_WEIGHT,
00526
                                                  input->experiment[i].weight);
00527
               for (j = 0; j < input->experiment->ninputs; ++j)
00528
                 json_object_set_string_member (object, stencil[j],
00529
                                                         input->experiment[i].stencil[j]);
00530
               json_array_add_element (array, child);
00531
00532
          json_object_set_array_member (object, LABEL_EXPERIMENTS, array);
00533
00534
          // Setting the variables data
         array = json_array_new ();
for (i = 0; i < input->nvariables; ++i)
00535
00536
00537
               child = json_node_new (JSON_NODE_OBJECT);
object = json_node_get_object (child);
00538
00539
               json_object_set_string_member (object, LABEL_NAME,
00540
00541
                                                      input->variable[i].name);
00542
               jb_json_object_set_float (object, LABEL_MINIMUM,
```

5.12 interface.c 199

```
input->variable[i].rangemin);
00544
            if (input->variable[i].rangeminabs != -G_MAXDOUBLE)
00545
              jb_json_object_set_float (object, LABEL_ABSOLUTE_MINIMUM,
00546
                                        input->variable[i].rangeminabs);
00547
            00548
            if (input->variable[i].rangemaxabs != G_MAXDOUBLE)
00550
              jb_json_object_set_float (object, LABEL_ABSOLUTE_MAXIMUM,
00551
                                        input->variable[i].rangemaxabs);
00552
            if (input->variable[i].precision != DEFAULT_PRECISION)
              jb_json_object_set_uint (object, LABEL_PRECISION,
00553
00554
                                       input->variable[i].precision);
00555
            if (input->algorithm == ALGORITHM_SWEEP
00556
                || input->algorithm == ALGORITHM_ORTHOGONAL)
00557
              jb_json_object_set_uint (object, LABEL_NSWEEPS,
                                       input->variable[i].nsweeps);
00558
            else if (input->algorithm == ALGORITHM GENETIC)
            jb_json_object_set_uint (object, LABEL_NBITS, input->variable[i].nbits);
if (input->nsteps)
00559
00560
00561
              (input->nsteps)
              jb_json_object_set_float (object, LABEL_STEP, input->variable[i].step);
00562
00563
            json_array_add_element (array, child);
00564
00565
        json_object_set_array_member (object, LABEL_VARIABLES, array);
00566
00567
        // Saving the error norm
        switch (input->norm)
00568
00569
00570
          case ERROR_NORM_MAXIMUM:
00571
            json_object_set_string_member (object, LABEL_NORM, LABEL_MAXIMUM);
00572
            break:
00573
          case ERROR_NORM_P:
00574
            json_object_set_string_member (object, LABEL_NORM, LABEL_P);
00575
            jb_json_object_set_float (object, LABEL_P, input->p);
00576
00577
          case ERROR_NORM_TAXICAB:
            json_object_set_string_member (object, LABEL_NORM, LABEL_TAXICAB);
00578
00579
00581 #if DEBUG_INTERFACE
00582
       fprintf (stderr, "input_save_json: end\n");
00583 #endif
00584 }
00585
00589 static inline void
00590 input_save (char *filename)
00591 {
        xmlDoc *doc;
00592
00593
       JsonGenerator *generator;
00594
00595 #if DEBUG_INTERFACE
00596
       fprintf (stderr, "input_save: start\n");
00597 #endif
00598
00599
        // Getting the input file directory
00600
       input->name = g_path_get_basename (filename);
        input->directory = g_path_get_dirname (filename);
00601
00602
00603
        if (input->type == INPUT_TYPE_XML)
00604
            \ensuremath{//} Opening the input file
00605
            doc = xmlNewDoc ((const xmlChar *) "1.0");
00606
00607
            input save xml (doc);
00608
00609
            // Saving the XML file
00610
            xmlSaveFormatFile (filename, doc, 1);
00611
00612
            // Freeing memory
00613
           xmlFreeDoc (doc);
00614
00615
        else
00616
00617
            // Opening the input file
00618
            generator = json_generator_new ();
00619
            json_generator_set_pretty (generator, TRUE);
00620
            input_save_json (generator);
00621
00622
            // Saving the JSON file
00623
            json_generator_to_file (generator, filename, NULL);
00624
00625
            // Freeing memory
            g_object_unref (generator);
00626
00627
00628
00629 #if DEBUG_INTERFACE
       fprintf (stderr, "input_save: end\n");
00630
00631 #endif
00632 }
```

```
00633
00637 static void
00638 dialog_options_close (GtkDialog * dlg,
00639
                              int response_id)
00640 4
00641 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_options_close: start\n");
00642
00643 #endif
00644
       if (response_id == GTK_RESPONSE_OK)
00645
00646
             input->seed
00647
               = (unsigned long int) gtk_spin_button_get_value (options->spin_seed);
00648
             nthreads = gtk_spin_button_get_value_as_int (options->spin_threads);
             nthreads_climbing
00649
00650
               = gtk_spin_button_get_value_as_int (options->spin_climbing);
00651
00652
        gtk_window_destroy (GTK_WINDOW (dlg));
00653 #if DEBUG_INTERFACE
00654 fprintf (stderr, "dialog_options_close: end\n");
00655 #endif
00656 }
00657
00661 static void
00662 options_new ()
00663 {
00664 #if DEBUG_INTERFACE
        fprintf (stderr, "options_new: start\n");
00665
00666 #endif
        options->label_seed = (GtkLabel *)
00667
          gtk_label_new (_("Pseudo-random numbers generator seed"));
00668
00669
        options->spin_seed = (GtkSpinButton *)
00670
          gtk_spin_button_new_with_range (0., (gdouble) G_MAXULONG, 1.);
00671
        gtk_widget_set_tooltip_text
00672
          (GTK_WIDGET (options->spin_seed),
00673
           _("Seed to init the pseudo-random numbers generator"));
        gtk_spin_button_set_value (options->spin_seed, (gdouble) input->seed);
options->label_threads = (GtkLabel *)
00674
00675
00676
          gtk_label_new (_("Threads number for the stochastic algorithm"));
00677
        options->spin_threads
00678
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 64., 1.);
00679
        gtk_widget_set_tooltip_text
00680
          (GTK_WIDGET (options->spin_threads),
           _("Number of threads to perform the calibration/optimization for " \,
00681
00682
              "the stochastic algorithm"));
        gtk_spin_button_set_value (options->spin_threads, (gdouble) nthreads);
00683
00684
        options->label_climbing = (GtkLabel *)
00685
          gtk_label_new (_("Threads number for the hill climbing method"));
        options->spin_climbing =
00686
00687
          (GtkSpinButton *) gtk_spin_button_new_with_range (1., 64., 1.);
        gtk_widget_set_tooltip_text
  (GTK_WIDGET (options->spin_climbing),
00688
00689
00690
           _("Number of threads to perform the calibration/optimization for the "
00691
              "hill climbing method"));
00692
        gtk_spin_button_set_value (options->spin_climbing,
00693
                                     (gdouble) nthreads_climbing);
        options->grid = (GtkGrid *) gtk_grid_new ();
gtk_grid_attach (options->grid, GTK_WIDGET (options->label_seed), 0, 0, 1, 1);
00694
00695
00696
        gtk_grid_attach (options->grid, GTK_WIDGET (options->spin_seed), 1, 0, 1, 1);
00697
        gtk_grid_attach (options->grid, GTK_WIDGET (options->label_threads),
        0, 1, 1, 1);
gtk_grid_attach (options->grid, GTK_WIDGET (options->spin_threads),
00698
00699
00700
                           1, 1, 1, 1);
00701
        gtk_grid_attach (options->grid, GTK_WIDGET (options->label_climbing), 0, 2, 1,
00702
00703
        gtk_grid_attach (options->grid, GTK_WIDGET (options->spin_climbing), 1, 2, 1,
00704
00705 #if !GTK4
00706
        gtk widget show all (GTK WIDGET (options->grid));
00707 #else
00708
        gtk_widget_show (GTK_WIDGET (options->grid));
00709 #endif
        options->dialog = (GtkDialog *)
00710
00711
          gtk_dialog_new_with_buttons (_("Options"),
00712
                                          window->window.
00713
                                         GTK_DIALOG_MODAL,
                                         _("_OK"), GTK_RESPONSE_OK,
00714
00715
                                          ("_Cancel"), GTK_RESPONSE_CANCEL, NULL);
00716
        gtk_box_append (GTK_BOX (gtk_dialog_get_content_area (options->dialog)),
       GTK_WIDGET (options->grid));
g_signal_connect (options->dialog, "response",
00717
00718
                           G_CALLBACK (dialog_options_close), NULL);
00719
        gtk_window_present (GTK_WINDOW (options->dialog));
00721 #if DEBUG_INTERFACE
       fprintf (stderr, "options_new: end\n");
00722
00723 #endif
00724 }
00725
```

```
00729 static inline void
00730 running_new ()
00731
00732 #if DEBUG_INTERFACE
        fprintf (stderr, "running_new: start\n");
00733
00734 #endif
       running->label = (GtkLabel *) gtk_label_new (_("Calculating ..."));
00736
        running->spinner = (GtkSpinner *) gtk_spinner_new ();
00737
        running->grid = (GtkGrid *) gtk_grid_new ();
00738
        gtk_grid_attach (running->grid, GTK_WIDGET (running->label), 0, 0, 1, 1);
        gtk_grid_attach (running->grid, GTK_WIDGET (running->spinner), 0, 1, 1, 1);
00739
00740
        running->dialog = (GtkDialog *)
          gtk_dialog_new_with_buttons (_("Calculating"),
00741
                                          window->window, GTK_DIALOG_MODAL, NULL, NULL);
00742
00743
        gtk_window_set_child (GTK_WINDOW
00744
                                (gtk_dialog_get_content_area (running->dialog)),
00745
                                GTK_WIDGET (running->grid));
00746
        gtk_spinner_start (running->spinner);
00747 #if !GTK4
00748
       gtk_widget_show_all (GTK_WIDGET (running->dialog));
00749 #else
00750
       gtk_widget_show (GTK_WIDGET (running->dialog));
00751 #endif
00752 #if DEBUG_INTERFACE
00753
        fprintf (stderr, "running_new: end\n");
00754 #endif
00755 }
00756
00762 static unsigned int
00763 window_get_algorithm ()
00764 {
00765
        unsigned int i;
00766 #if DEBUG_INTERFACE
00767
       fprintf (stderr, "window_get_algorithm: start\n");
00768 #endif
00769
       i = jbw_array_buttons_get_active (window->button_algorithm, NALGORITHMS);
00770 #if DEBUG_INTERFACE
00771 fprintf (stderr, "window_get_algorithm: u\n", i); 00772 fprintf (stderr, "window_get_algorithm: end\n"):
00773 #endif
00774 return i;
00775 }
00776
00782 static unsigned int
00783 window_get_climbing ()
00784 {
00785
       unsigned int i;
00786 #if DEBUG_INTERFACE
       fprintf (stderr, "window_get_climbing: start\n");
00787
00788 #endif
        i = jbw_array_buttons_get_active (window->button_climbing, NCLIMBINGS);
00790 #if DEBUG_INTERFACE
00791 fprintf (stderr, "window_get_climbing: u^n, i); 00792 fprintf (stderr, "window_get_climbing: end\n");
00793 #endif
00794
        return i;
00795 }
00796
00802 static unsigned int
00803 window_get_norm ()
00804 {
00805 unsigned int i;
00806 #if DEBUG_INTERFACE
       fprintf (stderr, "window_get_norm: start\n");
00807
00808 #endif
00809
       i = jbw_array_buttons_get_active (window->button_norm, NNORMS);
00810 #if DEBUG_INTERFACE
00811 fprintf (stderr, "window_get_norm: %u\n", i);
00812 fprintf (stderr, "window_get_norm: end\n");
00813 #endif
00814
       return i;
00815 }
00816
00820 static void
00821 window_save_climbing ()
00822 {
00823 #if DEBUG_INTERFACE
00824
       fprintf (stderr, "window_save_climbing: start\n");
00825 #endif
00826
        if (gtk check button get active (window->check climbing))
00827
00828
             input->nsteps = gtk_spin_button_get_value_as_int (window->spin_steps);
             input->relaxation = gtk_spin_button_get_value (window->spin_relaxation);
00829
00830
             switch (window_get_climbing ())
00831
               case CLIMBING_METHOD_COORDINATES:
00832
00833
                 input->climbing = CLIMBING_METHOD_COORDINATES;
```

```
break;
00835
00836
                input->climbing = CLIMBING_METHOD_RANDOM;
00837
                input->nestimates
00838
                   = gtk_spin_button_get_value_as_int (window->spin_estimates);
00839
              }
00841
        else
00842
         input->nsteps = 0;
00843 #if DEBUG_INTERFACE
       fprintf (stderr, "window_save_climbing: end\n");
00844
00845 #endif
00846 }
00847
00851 static void
00852 dialog_save_close (GtkFileChooserDialog * dlg,
00854
                          int response_id)
00855 {
       GtkFileFilter *filter1;
00857
        char *buffer;
00858 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_save_close: start\n");
00859
00860 #endif
       // If OK response then saving
if (response_id == GTK_RESPONSE_OK)
00861
00862
00863
00864
            // Setting input file type
            filter1 = gtk_file_chooser_get_filter (GTK_FILE_CHOOSER (dlg));
buffer = (char *) gtk_file_filter_get_name (filter1);
if (!strcmp (buffer, "XML"))
  input->type = INPUT_TYPE_XML;
00865
00866
00867
00868
00869
            else
00870
              input->type = INPUT_TYPE_JSON;
00871
00872
            // Adding properties to the root XML node
00873
            input->simulator
00874
              = g_strdup (gtk_button_get_label (window->button_simulator));
            if (gtk_check_button_get_active (window->check_evaluator))
00876
              input->evaluator
00877
                 = g_strdup (gtk_button_get_label (window->button_evaluator));
00878
00879
              input->evaluator = NULL;
            if (input->type == INPUT_TYPE_XML)
00880
00881
              {
00882
                 input->result
00883
                    = (char *) xmlStrdup ((const xmlChar *)
00884
                                          gtk_entry_get_text (window->entry_result));
00885
                 input->variables
00886
                   = (char *) xmlStrdup ((const xmlChar *)
00887
                                          gtk_entry_get_text (window->entry_variables));
00888
              }
00889
00890
00891
                 input->result = g_strdup (gtk_entry_get_text (window->entry_result));
00892
                 input->variables =
00893
                   g strdup (gtk entry get text (window->entry variables));
00894
00895
00896
            // Setting the algorithm
00897
            switch (window_get_algorithm ())
00898
              {
              case ALGORITHM_MONTE_CARLO:
00899
00900
                 input->algorithm = ALGORITHM_MONTE_CARLO;
00901
                 input->nsimulations
00902
                   = gtk_spin_button_get_value_as_int (window->spin_simulations);
00903
                 input->niterations
                   = gtk_spin_button_get_value_as_int (window->spin_iterations);
00904
00905
                 input->tolerance = gtk_spin_button_get_value (window->spin_tolerance);
00906
                 input->nbest = gtk_spin_button_get_value_as_int (window->spin_bests);
00907
                 window_save_climbing ();
00908
                 break;
00909
               case ALGORITHM SWEEP:
00910
                 input->algorithm = ALGORITHM_SWEEP;
00911
                 input->niterations
00912
                   = gtk_spin_button_get_value_as_int (window->spin_iterations);
00913
                 input->tolerance = gtk_spin_button_get_value (window->spin_tolerance);
00914
                 input->nbest = gtk_spin_button_get_value_as_int (window->spin_bests);
00915
                 window_save_climbing ();
              break;
case ALGORITHM ORTHOGONAL:
00916
00917
                 input->algorithm = ALGORITHM_ORTHOGONAL;
00918
                 input->niterations
00920
                   = gtk_spin_button_get_value_as_int (window->spin_iterations);
00921
                 input->tolerance = gtk_spin_button_get_value (window->spin_tolerance);
                 input->nbest = gtk_spin_button_get_value_as_int (window->spin_bests);
00922
00923
                 window_save_climbing ();
00924
                 break:
```

```
default:
00926
                 input->algorithm = ALGORITHM_GENETIC;
00927
                 input->nsimulations
00928
                   = gtk_spin_button_get_value_as_int (window->spin_population);
00929
                 input->niterations
00930
                   = gtk spin button get value as int (window->spin generations);
00931
                 input->mutation_ratio
00932
                    gtk_spin_button_get_value (window->spin_mutation);
                 input->reproduction_ratio
00933
00934
                    = gtk_spin_button_get_value (window->spin_reproduction);
                 input->adaptation_ratio
00935
                   = gtk_spin_button_get_value (window->spin_adaptation);
00936
00937
00938
             input->norm = window_get_norm ();
00939
             input->p = gtk_spin_button_get_value (window->spin_p);
00940
             input->threshold = gtk_spin_button_get_value (window->spin_threshold);
00941
00942
             // Saving the XML file
00943
             buffer = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
00944
             input_save (buffer);
00945
00946
             // Closing and freeing memory
00947
             g_free (buffer);
00948
00949
        gtk_window_destroy (GTK_WINDOW (dlg));
00950 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_save_close: end\n");
00951
00952 #endif
00953 }
00954
00958 static void
00959 window_save ()
00960 {
00961
        GtkFileChooserDialog *dlg;
00962
        GtkFileFilter *filter1, *filter2;
00963
        char *buffer:
00964
00965 #if DEBUG_INTERFACE
00966
        fprintf (stderr, "window_save: start\n");
00967 #endif
00968
00969
         // Opening the saving dialog
00970
        dlg = (GtkFileChooserDialog *)
00971
          gtk_file_chooser_dialog_new (_("Save file"),
00972
                                           window->window,
00973
                                          GTK_FILE_CHOOSER_ACTION_SAVE,
00974
                                          _("_Cancel"), GTK_RESPONSE_CANCEL,
00975
                                           _("_OK"), GTK_RESPONSE_OK, NULL);
00976 #if !GTK4
00977
        gtk file chooser set do overwrite confirmation (GTK FILE CHOOSER (dlg), TRUE);
00978 #endi
00979
      buffer = g_build_filename (input->directory, input->name, NULL);
00980
        gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (dlg), buffer);
00981
        g_free (buffer);
00982
00983
         // Adding XML filter
        filter1 = (GtkFileFilter *) gtk_file_filter_new ();
00984
00985
        gtk_file_filter_set_name (filter1, "XML");
        gtk_file_filter_add_pattern (filterl, "*.xml");
gtk_file_filter_add_pattern (filterl, "*.XML");
00986
00987
        gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter1);
00988
00989
00990
         // Adding JSON filter
00991
        filter2 = (GtkFileFilter *) gtk_file_filter_new ();
00992
        gtk_file_filter_set_name (filter2, "JSON");
        gtk_file_filter_add_pattern (filter2, "*.json");
gtk_file_filter_add_pattern (filter2, "*.JSON");
gtk_file_filter_add_pattern (filter2, "*.js");
gtk_file_filter_add_pattern (filter2, "*.js");
00993
00994
00995
00996
00997
        gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter2);
00998
00999
        if (input->type == INPUT_TYPE_XML)
          gtk_file_chooser_set_filter (GTK_FILE_CHOOSER (dlg), filter1);
01000
01001
01002
          gtk file chooser set filter (GTK FILE CHOOSER (dlg), filter2);
01003
        // Connecting the close function
g_signal_connect (dlg, "response", G_CALLBACK (dialog_save_close), NULL);
01004
01005
01006
01007
        // Showing modal dialog
01008
        gtk_window_present (GTK_WINDOW (dlg));
01009
01010 #if DEBUG_INTERFACE
        fprintf (stderr, "window_save: end\n");
01011
01012 #endif
01013 }
01014
```

```
01018 static void
01019 window_run ()
01020 {
        char *msg, *msg2, buffer[64], buffer2[64];
01021
01022 unsigned int i;
01023 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: start\n");
01025 #endif
01026 window_save ();
01027
       running new ();
01028
       jbw_process_pending ();
01029
        optimize_open ();
01030 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: closing running dialog\n");
01031
01032 #endif
01033 gtk_spinner_stop (running->spinner);
01034
        gtk_window_destroy (GTK_WINDOW (running->dialog));
01035 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: displaying results\n");
01036
01037 #endif
01038
        snprintf (buffer, 64, "error = %.15le\n", optimize->error_old[0]);
01039
        msg2 = g\_strdup (buffer);
        for (i = 0; i < optimize->nvariables; ++i, msg2 = msg)
01040
01041
01042
            snprintf (buffer, 64, "%s = %s\n",
                      input->variable[i].name, format[input->variable[i].precision]);
01043
01044
            snprintf (buffer2, 64, buffer, optimize->value_old[i]);
01045
            msg = g_strconcat (msg2, buffer2, NULL);
01046
            g_free (msg2);
         1
01047
       01048
01049
01050
       msg = g_strconcat (msg2, buffer, NULL);
01051
        g_free (msg2);
01052
        jbw_show_message_gtk (_("Best result"), msg, INFO_TYPE);
01053 g_free (msg);
01054 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: freeing memory\n");
01056 #endif
01057
       optimize_free ();
01058 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: end\n");
01059
01060 #endif
01061 }
01062
01066 static void
01067 window_help ()
01068 {
01069
       char *buffer, *buffer2;
01070 #if DEBUG_INTERFACE
       fprintf (stderr, "window_help: start\n");
01072 #endif
01073
       buffer2 = g_build_filename (window->application_directory, "..", "manuals",
01074
                                      _("user-manual.pdf"), NULL);
01075 buffer = g_filename_to_uri (buffer2, NULL, NULL);
01076
       g_free (buffer2);
01077 #if GTK4
01078
       gtk_show_uri (window->window, buffer, GDK_CURRENT_TIME);
01079 #else
       gtk_show_uri_on_window (window->window, buffer, GDK_CURRENT_TIME, NULL);
01080
01081 #endif
01082 #if DEBUG_INTERFACE
01083
       fprintf (stderr, "window_help: uri=%s\n", buffer);
01084 #endif
01085
       g_free (buffer);
01086 #if DEBUG_INTERFACE
       fprintf (stderr, "window_help: end\n");
01087
01088 #endif
01089 }
01090
01094 static void
01095 window_about ()
01096 {
01097
       static const gchar *authors[] = {
01098
           "Javier Burguete Tolosa <jburguete@eead.csic.es>",
         "Borja Latorre Garcés <borja.latorre@csic.es>",
01099
01100
         NULL
01101
01102 #if DEBUG_INTERFACE
       fprintf (stderr, "window_about: start\n");
01103
01104 #endif
01105
       gtk_show_about_dialog
         (window->window,
  "program_name", "MPCOTool",
01106
01107
01108
           "comments",
          _("The Multi-Purposes Calibration and Optimization Tool.\n" "A software to perform calibrations or optimizations of empirical " \,
01109
01110
```

```
"parameters"),
01112
           "authors", authors,
01113
           "translator-credits"
           "Javier Burguete Tolosa <jburguete@eead.csic.es> "
01114
           "(english, french and spanish)\n"
01115
           "Uğur Çayoğlu (german)",
01116
           "version", "4.4.6",
"copyright", "Copyright 2012-2023 Javier Burguete Tolosa",
01117
01118
           "logo", window->logo,
01119
           "website", "https://github.com/jburguete/mpcotool",
01120
           "license-type", GTK_LICENSE_BSD, NULL);
01121
01122 #if DEBUG INTERFACE
       fprintf (stderr, "window_about: end\n");
01123
01124 #endif
01125 }
01126
01130 static void
01131 window_update_climbing ()
01133 #if DEBUG_INTERFACE
       fprintf (stderr, "window_update_climbing: start\n");
01134
01135 #endif
01136
       gtk_widget_show (GTK_WIDGET (window->check_climbing));
01137
        if (gtk_check_button_get_active (window->check_climbing))
01138
01139
            gtk_widget_show (GTK_WIDGET (window->grid_climbing));
01140
            gtk_widget_show (GTK_WIDGET (window->label_step));
01141
            gtk_widget_show (GTK_WIDGET (window->spin_step));
01142
01143
       switch (window_get_climbing ())
01144
01145
         case CLIMBING_METHOD_COORDINATES:
           gtk_widget_hide (GTK_WIDGET (window->label_estimates));
01146
01147
            gtk_widget_hide (GTK_WIDGET (window->spin_estimates));
01148
01149
          default:
           gtk_widget_show (GTK_WIDGET (window->label_estimates));
01150
            gtk_widget_show (GTK_WIDGET (window->spin_estimates));
01151
01153 #if DEBUG_INTERFACE
01154 fprintf (stderr, "window_update_climbing: end\n");
01155 #endif
01156 }
01157
01161 static void
01162 window_update ()
01163 {
01164
       unsigned int i;
01165 #if DEBUG_INTERFACE
       fprintf (stderr, "window_update: start\n");
01166
01167 #endif
       gtk_widget_set_sensitive
01168
01169
          (GTK_WIDGET (window->button_evaluator),
01170
           gtk_check_button_get_active (window->check_evaluator));
01171
       gtk_widget_hide (GTK_WIDGET (window->label_simulations));
       gtk_widget_hide (GTK_WIDGET (window->spin_simulations));
01172
       gtk_widget_hide (GTK_WIDGET (window->label_iterations));
01174
       gtk_widget_hide (GTK_WIDGET (window->spin_iterations));
01175
       gtk_widget_hide (GTK_WIDGET (window->label_tolerance));
01176
        gtk_widget_hide (GTK_WIDGET (window->spin_tolerance));
       gtk_widget_hide (GTK_WIDGET (window->label_bests));
01177
01178
       gtk widget hide (GTK WIDGET (window->spin bests));
01179
       gtk_widget_hide (GTK_WIDGET (window->label_population));
01180
       gtk_widget_hide (GTK_WIDGET (window->spin_population));
01181
       gtk_widget_hide (GTK_WIDGET (window->label_generations));
01182
        gtk_widget_hide (GTK_WIDGET (window->spin_generations));
01183
        gtk_widget_hide (GTK_WIDGET (window->label_mutation));
       gtk widget hide (GTK WIDGET (window->spin mutation));
01184
01185
       qtk_widget_hide (GTK_WIDGET (window->label_reproduction));
01186
       gtk_widget_hide (GTK_WIDGET (window->spin_reproduction));
01187
       gtk_widget_hide (GTK_WIDGET (window->label_adaptation));
01188
        gtk_widget_hide (GTK_WIDGET (window->spin_adaptation));
01189
        gtk_widget_hide (GTK_WIDGET (window->label_sweeps));
       gtk_widget_hide (GTK_WIDGET (window->spin_sweeps));
01190
       gtk_widget_hide (GTK_WIDGET (window->label_bits));
01191
01192
       gtk_widget_hide (GTK_WIDGET (window->spin_bits));
01193
       gtk_widget_hide (GTK_WIDGET (window->check_climbing));
01194
        gtk_widget_hide (GTK_WIDGET (window->grid_climbing));
01195
        gtk_widget_hide (GTK_WIDGET (window->label_step));
        gtk_widget_hide (GTK_WIDGET (window->spin_step));
01196
        gtk widget hide (GTK WIDGET (window->label p));
01197
01198
        gtk_widget_hide (GTK_WIDGET (window->spin_p));
01199
        i = gtk_spin_button_get_value_as_int (window->spin_iterations);
01200
        switch (window_get_algorithm ())
01201
          case ALGORITHM MONTE CARLO:
01202
01203
            gtk_widget_show (GTK_WIDGET (window->label_simulations));
```

```
gtk_widget_show (GTK_WIDGET (window->spin_simulations));
            gtk_widget_show (GTK_WIDGET (window->label_iterations));
01205
01206
            gtk_widget_show (GTK_WIDGET (window->spin_iterations));
01207
            if (i > 1)
01208
              {
01209
                qtk_widget_show (GTK_WIDGET (window->label_tolerance));
                gtk_widget_show (GTK_WIDGET (window->spin_tolerance));
01210
                gtk_widget_show (GTK_WIDGET (window->label_bests));
01211
01212
                gtk_widget_show (GTK_WIDGET (window->spin_bests));
01213
01214
            window_update_climbing ();
01215
            break:
          case ALGORITHM_SWEEP:
01216
01217
          case ALGORITHM_ORTHOGONAL:
01218
            gtk_widget_show (GTK_WIDGET (window->label_iterations));
01219
            gtk_widget_show (GTK_WIDGET (window->spin_iterations));
01220
            if (i > 1)
01221
             {
                gtk_widget_show (GTK_WIDGET (window->label_tolerance));
                gtk_widget_show (GTK_WIDGET (window->spin_tolerance));
01223
                gtk_widget_show (GTK_WIDGET (window->label_bests));
01224
01225
                gtk_widget_show (GTK_WIDGET (window->spin_bests));
01226
            gtk_widget_show (GTK_WIDGET (window->label_sweeps));
gtk_widget_show (GTK_WIDGET (window->spin_sweeps));
01227
01228
            gtk_widget_show (GTK_WIDGET (window->check_climbing));
01229
            window_update_climbing ();
01230
01231
            break;
01232
          default:
01233
            gtk_widget_show (GTK_WIDGET (window->label_population));
            gtk_widget_show (GTK_WIDGET (window->spin_population));
01234
01235
            gtk_widget_show (GTK_WIDGET (window->label_generations));
01236
            gtk_widget_show (GTK_WIDGET (window->spin_generations));
01237
            gtk_widget_show (GTK_WIDGET (window->label_mutation));
01238
            gtk_widget_show (GTK_WIDGET (window->spin_mutation));
            gtk_widget_show (GTK_WIDGET (window->label_reproduction));
01239
            gtk_widget_show (GTK_WIDGET (window->spin_reproduction));
01240
01241
            gtk_widget_show (GTK_WIDGET (window->label_adaptation));
01242
            gtk_widget_show (GTK_WIDGET (window->spin_adaptation));
01243
            gtk_widget_show (GTK_WIDGET (window->label_bits));
01244
            gtk_widget_show (GTK_WIDGET (window->spin_bits));
01245
01246
        gtk_widget_set_sensitive
01247
          (GTK_WIDGET (window->button_remove_experiment), input->nexperiments > 1);
        gtk_widget_set_sensitive
01248
01249
          (GTK_WIDGET (window->button_remove_variable), input->nvariables > 1);
01250
        for (i = 0; i < input->experiment->ninputs; ++i)
01251
            gtk widget show (GTK WIDGET (window->check template[i]));
01252
            gtk_widget_show (GTK_WIDGET (window->button_template[i]));
01253
            gtk_widget_set_sensitive (GTK_WIDGET (window->check_template[i]), 0);
01255
            gtk_widget_set_sensitive (GTK_WIDGET (window->button_template[i]), 1);
01256
            g_signal_handler_block
01257
              (window->check_template[i], window->id_template[i]);
01258
            g_signal_handler_block (window->button_template[i], window->id_input[i]);
            gtk_check_button_set_active (window->check_template[i], 1);
01259
            g_signal_handler_unblock (window->button_template[i],
01260
                                        window->id_input[i]);
01261
01262
            g_signal_handler_unblock (window->check_template[i],
01263
                                        window->id_template[i]);
01264
01265
        if (i > 0)
01266
         {
            gtk_widget_set_sensitive (GTK_WIDGET (window->check_template[i - 1]), 1);
01267
01268
            gtk_widget_set_sensitive (GTK_WIDGET (window->button_template[i - 1]),
01269
                                       gtk_check_button_get_active
01270
                                        (window->check template[i - 1]));
01271
01272
        if (i < MAX_NINPUTS)</pre>
01274
            gtk_widget_show (GTK_WIDGET (window->check_template[i]));
            gtk_widget_show (GTK_WIDGET (window->button_template[i]));
01275
01276
            gtk_widget_set_sensitive (GTK_WIDGET (window->check_template[i]), 1);
            gtk_widget_set_sensitive
01277
01278
              (GTK_WIDGET (window->button_template[i]),
               gtk_check_button_get_active (window->check_template[i]));
01279
01280
            g_signal_handler_block
01281
              (window->check_template[i], window->id_template[i]);
            g_signal_handler_block (window->button_template[i], window->id_input[i]);
gtk_check_button_set_active (window->check_template[i], 0);
01282
01283
            g_signal_handler_unblock (window->button_template[i],
01284
                                       window->id_input[i]);
            g_signal_handler_unblock (window->check_template[i],
01286
01287
                                        window->id_template[i]);
01288
        while (++i < MAX NINPUTS)
01289
01290
```

```
gtk_widget_hide (GTK_WIDGET (window->check_template[i]));
01292
            gtk_widget_hide (GTK_WIDGET (window->button_template[i]));
01293
01294
        {\tt gtk\_widget\_set\_sensitive}
01295
          (GTK_WIDGET (window->spin_minabs),
01296
           gtk_check_button_get_active (window->check_minabs));
01297
        gtk_widget_set_sensitive
01298
          (GTK_WIDGET (window->spin_maxabs),
01299
           gtk_check_button_get_active (window->check_maxabs));
01300
        if (window_get_norm () == ERROR_NORM_P)
01301
         {
            gtk_widget_show (GTK_WIDGET (window->label_p));
01302
            gtk_widget_show (GTK_WIDGET (window->spin_p));
01303
01304
01305 #if DEBUG_INTERFACE
       fprintf (stderr, "window_update: end\n");
01306
01307 #endif
01308 }
01309
01313 static void
01314 window_set_algorithm ()
01315 {
01316
       int i;
01317 #if DEBUG_INTERFACE
01318
       fprintf (stderr, "window_set_algorithm: start\n");
01319 #endif
01320
       i = window_get_algorithm ();
01321
       switch (i)
01322
01323
          case ALGORITHM SWEEP:
01324
          case ALGORITHM_ORTHOGONAL:
01325
           i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01326
            if (i < 0)
01327
              i = 0;
01328
            gtk_spin_button_set_value (window->spin_sweeps,
01329
                                         (gdouble) input->variable[i].nsweeps);
01330
            break;
          case ALGORITHM_GENETIC:
01331
01332
           i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01333
            if (i < 0)
01334
              i = 0;
01335
            gtk_spin_button_set_value (window->spin_bits,
01336
                                         (gdouble) input->variable[i].nbits);
01337
       window_update ();
01338
01339 #if DEBUG_INTERFACE
01340 fprintf (stderr, "window_set_algorithm: end\n");
01341 #endif
01342 }
01343
01347 static void
01348 window_set_experiment ()
01349 {
01350
       unsigned int i, j;
01351
        char *buffer1;
01352 #if DEBUG_INTERFACE
       fprintf (stderr, "window_set_experiment: start\n");
01353
01354 #endif
01355 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
       gtk_spin_button_set_value (window->spin_weight, input->experiment[i].weight);
buffer1 = gtk_combo_box_text_get_active_text (window->combo_experiment);
01356
01357
01358
        gtk button set label (window->button experiment, bufferl);
01359
        g_free (buffer1);
        for (j = 0; j < input->experiment->ninputs; ++j)
01360
01361
01362
            g_signal_handler_block (window->button_template[j], window->id_input[j]);
            gtk_button_set_label (window->button_template[j],
01363
                                    input->experiment[i].stencil[j]);
01364
01365
            g_signal_handler_unblock
01366
              (window->button_template[j], window->id_input[j]);
01367
01368 #if DEBUG_INTERFACE
01369 fprintf (stderr, "window_set_experiment: end\n");
01370 #endif
01371 }
01372
01376 static void
01377 window_remove_experiment ()
01378 {
01379
       unsigned int i, j;
01380 #if DEBUG_INTERFACE
01381
        fprintf (stderr, "window_remove_experiment: start\n");
01382 #endif
01383
       i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01384
        g_signal_handler_block (window->combo_experiment, window->id_experiment);
       gtk_combo_box_text_remove (window->combo_experiment, i);
g_signal_handler_unblock (window->combo_experiment, window->id_experiment);
01385
01386
```

```
experiment_free (input->experiment + i, input->type);
        --input->nexperiments;
01388
01389
        for (j = i; j < input->nexperiments; ++j)
         memcpy (input->experiment + j, input->experiment + j + 1,
01390
                   sizeof (Experiment));
01391
01392
        j = input->nexperiments - 1;
01393
        if (i > j)
01394
          i = j;
01395
        for (j = 0; j < input->experiment->ninputs; ++j)
01396
          g_signal_handler_block (window->button_template[j], window->id_input[j]);
        gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), i);
01397
01398
        for (j = 0; j < input->experiment->ninputs; ++j)
01399
          g_signal_handler_unblock (window->button_template[j], window->id_input[j]);
        window_update ();
01400
01401 #if DEBUG_INTERFACE
01402
        fprintf (stderr, "window_remove_experiment: end\n");
01403 #endif
01404 }
01409 static void
01410 window add experiment ()
01411 {
01412
        unsigned int i, j;
01413 #if DEBUG_INTERFACE
        fprintf (stderr, "window_add_experiment: start\n");
01414
01415 #endif
        i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01416
01417
        g_signal_handler_block (window->combo_experiment, window->id_experiment);
01418
        gtk_combo_box_text_insert_text
01419
        (window->combo_experiment, i, input->experiment[i].name);
g_signal_handler_unblock (window->combo_experiment, window->id_experiment);
01420
01421
         input->experiment = (Experiment *) g_realloc
01422
           (input->experiment, (input->nexperiments + 1) * sizeof (Experiment));
01423
         for (j = input->nexperiments - 1; j > i; --j)
          memcpy (input->experiment + j + 1, input->experiment + j,
01424
        memcpy (Input-Sexperiment();
    sizeof (Experiment());
input->experiment[j + 1].weight = input->experiment[j].weight;
input->experiment[j + 1].ninputs = input->experiment[j].ninputs;
01425
01426
           (input->type == INPUT_TYPE_XML)
01428
01429
01430
             input->experiment[j + 1].name
              = (char *) xmlStrdup ((xmlChar *) input->experiment[j].name);
01431
             for (j = 0; j < input->experiment->ninputs; ++j)
  input->experiment[i + 1].stencil[j]
01432
01433
                 = (char *) xmlStrdup ((xmlChar *) input->experiment[i].stencil[j]);
01434
01435
01436
        else
01437
             input->experiment[j + 1].name = g_strdup (input->experiment[j].name);
for (j = 0; j < input->experiment->ninputs; ++j)
01438
01439
               input->experiment[i + 1].stencil[j]
01440
01441
                 = g_strdup (input->experiment[i].stencil[j]);
01442
01443
        ++input->nexperiments;
        for (j = 0; j < input->experiment->ninputs; ++j)
01444
          g_signal_handler_block (window->button_template[j], window->id_input[j]);
01445
        gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), i + 1);
01446
        for (j = 0; j < input->experiment->ninputs; ++j)
01447
01448
          g_signal_handler_unblock (window->button_template[j], window->id_input[j]);
01449 window_update ();
01450 #if DEBUG_INTERFACE
01451 fprintf (stderr, "window_add_experiment: end\n");
01452 #endif
01453 }
01454
01458 static void
01459 dialog_name_experiment_close (GtkFileChooserDialog \star dlg,
01461
                                       int response id.
                                       void *data)
01462
01463 {
01464
        char *buffer;
01465
        unsigned int i;
01466 #if DEBUG_INTERFACE
        fprintf (stderr, "window_name_experiment_close: start\n");
01467
01468 #endif
01469
      i = (size_t) data;
01470
        if (response_id == GTK_RESPONSE_OK)
01471
01472
             buffer = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
01473
             g signal handler block (window->combo experiment, window->id experiment);
             gtk_combo_box_text_remove (window->combo_experiment, i);
01474
             gtk_combo_box_text_insert_text (window->combo_experiment, i, buffer);
01476
             gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), i);
01477
             g_signal_handler_unblock (window->combo_experiment,
01478
                                         window->id_experiment);
             g_free (buffer);
01479
01480
```

```
01481 #if DEBUG_INTERFACE
       fprintf (stderr, "window_name_experiment_close: end\n");
01483 #endif
01484 }
01485
01489 static void
01490 window_name_experiment ()
01491 {
01492
        GtkFileChooserDialog *dlg;
01493
       GMainLoop *loop;
       const char *buffer;
01494
01495
       unsigned int i:
01496 #if DEBUG_INTERFACE
01497
       fprintf (stderr, "window_name_experiment: start\n");
01498 #endif
01499
       i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01500
       buffer = gtk_button_get_label (window->button_experiment);
        dlg = (GtkFileChooserDialog *)
01501
01502
         gtk_file_chooser_dialog_new (_("Open experiment file"),
01503
                                       window->window
01504
                                       GTK_FILE_CHOOSER_ACTION_OPEN,
01505
                                        ("_Cancel"),
                                       GTK_RESPONSE_CANCEL,
01506
       __("_Open"), GTK_RESPONSE_OK, NULL); gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (dlg), buffer);
01507
01508
       01509
01510
01511
        gtk_window_present (GTK_WINDOW (dlg));
       loop = g_main_loop_new (NULL, 0);
g_signal_connect_swapped (dlg, "destroy", G_CALLBACK (g_main_loop_quit),
01512
01513
01514
                                  loop);
01515
       g_main_loop_run (loop);
01516
        g_main_loop_unref (loop);
01517 #if DEBUG_INTERFACE
       fprintf (stderr, "window_name_experiment: end\n");
01518
01519 #endif
01520 }
01521
01525 static void
01526 window_weight_experiment ()
01527 {
01528
       unsigned int i;
01529 #if DEBUG_INTERFACE
01530
       fprintf (stderr, "window_weight_experiment: start\n");
01531 #endif
01532 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01533
       input->experiment[i].weight = gtk_spin_button_get_value (window->spin_weight);
01534 #if DEBUG_INTERFACE
       fprintf (stderr, "window_weight_experiment: end\n");
01535
01536 #endif
01537 }
01538
01542 static void
01543 window_inputs_experiment ()
01544 {
01545
        unsigned int j;
01546 #if DEBUG_INTERFACE
       fprintf (stderr, "window_inputs_experiment: start\n");
01547
01548 #endif
       j = input->experiment->ninputs - 1;
01549
       if (j && !gtk_check_button_get_active (window->check_template[j]))
01550
01551
         --input->experiment->ninputs;
       if (input->experiment->ninputs < MAX_NINPUTS</pre>
            && gtk_check_button_get_active (window->check_template[j]))
01553
01554
         ++input->experiment->ninputs;
01555
       window_update ();
01556 #if DEBUG INTERFACE
       fprintf (stderr, "window_inputs_experiment: end\n");
01557
01558 #endif
01559 }
01560
01564 static void
01565 window_template_experiment_close (GtkFileChooserDialog * dlg,
01567
                                        int response_id,
                                        void *data)
01569
01570 {
01571
       GFile *file1, *file2;
01572
       char *buffer1, *buffer2;
01573 unsigned int i, j;
01574 #if DEBUG_INTERFACE
01575
       fprintf (stderr, "window_template_experiment_close: start\n");
01576 #endif
01577
          (response_id == GTK_RESPONSE_OK)
01578
01579
            i = (size_t) data;
            j = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01580
01581
            buffer1 = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
```

```
file1 = g_file_new_for_path (buffer1);
            file2 = g_file_new_for_path (input->directory);
01583
01584
            buffer2 = g_file_get_relative_path (file2, file1);
            if (input->type == INPUT_TYPE_XML)
01585
01586
              input->experiment[j].stencil[i]
01587
                = (char *) xmlStrdup ((xmlChar *) buffer2);
01588
01589
              input->experiment[j].stencil[i] = g_strdup (buffer2);
01590
            g_free (buffer2);
            g_object_unref (file2);
g_object_unref (file1);
01591
01592
            g_free (buffer1);
01593
01594
01595
        gtk_window_destroy (GTK_WINDOW (dlg));
01596 #if DEBUG_INTERFACE
       fprintf (stderr, "window_template_experiment_close: end\n");
01597
01598 #endif
01599 }
01600
01604 static void
01605 window_template_experiment (void *data)
01607 {
01608
        GtkFileChooserDialog *dlg;
01609
        GMainLoop *loop;
        const char *buffer;
01610
01611
        unsigned int i;
01612 #if DEBUG_INTERFACE
01613
       fprintf (stderr, "window_template_experiment: start\n");
01614 #endif
01615
        i = (size t) data;
01616
        buffer = qtk_button_qet_label (window->button_template[i]);
01617
        dlg = (GtkFileChooserDialog *)
01618
          gtk_file_chooser_dialog_new (_("Open template file"),
01619
                                         window->window,
01620
                                        GTK_FILE_CHOOSER_ACTION_OPEN,
                                         ("_Cancel"),
01621
                                        GTK_RESPONSE_CANCEL,
01622
                                        _("_Open"), GTK_RESPONSE_ACCEPT, NULL);
01623
01624
        gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (dlg), buffer);
01625
        g_signal_connect (dlg, "response",
01626
                           G_CALLBACK (window_template_experiment_close), data);
        gtk_window_present (GTK_WINDOW (dlg));
01627
01628
        loop = g main loop new (NULL, 0);
01629
       g_signal_connect_swapped (dlg, "destroy", G_CALLBACK (g_main_loop_quit),
01630
                                   loop);
01631
        g_main_loop_run (loop);
01632
       g_main_loop_unref (loop);
01633 #if DEBUG INTERFACE
       fprintf (stderr, "window_template_experiment: end\n");
01634
01635 #endif
01636 }
01637
01641 static void
01642 window_set_variable ()
01643 {
01644
        unsigned int i;
01645 #if DEBUG_INTERFACE
       fprintf (stderr, "window_set_variable: start\n");
01646
01647 #endif
01648
       i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01649
        g_signal_handler_block (window->entry_variable, window->id_variable_label);
        gtk_entry_set_text (window->entry_variable, input->variable[i].name);
01650
01651
        g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
        gtk_spin_button_set_value (window->spin_min, input->variable[i].rangemin);
gtk_spin_button_set_value (window->spin_max, input->variable[i].rangemax);
01652
01653
01654
        if (input->variable[i].rangeminabs != -G_MAXDOUBLE)
01655
01656
            gtk_spin_button_set_value (window->spin_minabs,
                                         input->variable[i].rangeminabs);
01657
01658
            gtk_check_button_set_active (window->check_minabs, 1);
01659
01660
        else
01661
         {
            gtk_spin_button_set_value (window->spin_minabs, -G_MAXDOUBLE);
01662
01663
            gtk check button set active (window->check minabs, 0);
01664
01665
        if (input->variable[i].rangemaxabs != G_MAXDOUBLE)
01666
01667
            gtk_spin_button_set_value (window->spin_maxabs,
01668
                                         input->variable[i].rangemaxabs);
            gtk_check_button_set_active (window->check_maxabs, 1);
01669
01670
01671
        else
01672
01673
            gtk_spin_button_set_value (window->spin_maxabs, G_MAXDOUBLE);
01674
            gtk_check_button_set_active (window->check_maxabs, 0);
01675
```

```
gtk_spin_button_set_value (window->spin_precision,
                                    input->variable[i].precision);
01677
01678
        gtk_spin_button_set_value (window->spin_steps, (gdouble) input->nsteps);
       if (input->nsteps)
01679
01680
         gtk_spin_button_set_value (window->spin_step, input->variable[i].step);
01681 #if DEBUG_INTERFACE
01682 fprintf (stderr, "window_set_variable: precision[%u]=%u\n", i,
                 input->variable[i].precision);
01683
01684 #endif
01685
       switch (window_get_algorithm ())
        {
01686
01687
          case ALGORITHM SWEEP:
01688
         case ALGORITHM_ORTHOGONAL:
01689
           gtk_spin_button_set_value (window->spin_sweeps,
01690
                                        (gdouble) input->variable[i].nsweeps);
01691 #if DEBUG_INTERFACE
           fprintf (stderr, "window_set_variable: nsweeps[%u]=%u\n", i,
01692
                     input->variable[i].nsweeps);
01693
01694 #endif
01695
           break;
01696
          case ALGORITHM_GENETIC:
01697
           gtk_spin_button_set_value (window->spin_bits,
01698
                                        (gdouble) input->variable[i].nbits);
01699 #if DEBUG_INTERFACE
       fprintf (stderr, "window_set_variable: nbits[%u]=%u\n", i,
01700
                    input->variable[i].nbits);
01701
01702 #endif
01703 break;
01704
       window_update ();
01705
01706 #if DEBUG_INTERFACE
01707
       fprintf (stderr, "window_set_variable: end\n");
01708 #endif
01709 }
01710
01714 static void
01715 window_remove_variable ()
01716 {
01717
        unsigned int i, j;
01718 #if DEBUG_INTERFAC
01719
       fprintf (stderr, "window_remove_variable: start\n");
01720 #endif
01721 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01722
       g_signal_handler_block (window->combo_variable, window->id_variable);
       gtk_combo_box_text_remove (window->combo_variable, i);
01723
01724
        g_signal_handler_unblock (window->combo_variable, window->id_variable);
01725
        xmlFree (input->variable[i].name);
01726
        --input->nvariables;
       for (j = i; j < input->nvariables; ++j)
  memcpy (input->variable + j, input->variable + j + 1, sizeof (Variable));
01727
01728
        j = input->nvariables - 1;
01729
01730
       if (i > j)
         i = j;
01731
01732
       g_signal_handler_block (window->entry_variable, window->id_variable_label);
       gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), i);
01733
01734
       g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
01735
        window_update ();
01736 #if DEBUG_INTERFACE
01737
       fprintf (stderr, "window_remove_variable: end\n");
01738 #endif
01739 }
01740
01744 static void
01745 window add variable ()
01746 {
       unsigned int i, j;
01747
01748 #if DEBUG_INTERFACE
       fprintf (stderr, "window_add_variable: start\n");
01749
01750 #endif
01751
       i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01752
        g_signal_handler_block (window->combo_variable, window->id_variable);
01753
        gtk_combo_box_text_insert_text (window->combo_variable, i,
01754
                                         input->variable[i].name);
        g_signal_handler_unblock (window->combo_variable, window->id_variable);
01755
01756
        input->variable = (Variable *) g_realloc
01757
          (input->variable, (input->nvariables + 1) * sizeof (Variable));
01758
        for (j = input->nvariables - 1; j > i; --j)
        memcpy (input->variable + j + 1, input->variable + j, sizeof (Variable));
memcpy (input->variable + j + 1, input->variable + j, sizeof (Variable));
01759
01760
01761
        if (input->type == INPUT_TYPE_XML)
01762
         input->variable[j + 1].name
01763
            = (char *) xmlStrdup ((xmlChar *) input->variable[j].name);
01764
01765
          input->variable[j + 1].name = g_strdup (input->variable[j].name);
01766
        ++input->nvariables;
        g_signal_handler_block (window->entry_variable, window->id_variable_label);
01767
01768
       gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), i + 1);
```

```
g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
01770
         window_update ();
01771 #if DEBUG_INTERFACE
01772 fprintf (stderr, "window_add_variable: end\n");
01773 #endif
01774 }
01775
01779 static void
01780 window_label_variable ()
01781 {
01782
        unsigned int i:
01783
        const char *buffer:
01784 #if DEBUG_INTERFACE
01785
        fprintf (stderr, "window_label_variable: start\n");
01786 #endif
01787     i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01788     buffer = gtk_entry_get_text (window->entry_variable);
01789     g_signal_handler_block (window->combo_variable, window->id_variable);
        gtk_combo_box_text_remove (window->combo_variable, i);
        gtk_combo_box_text_insert_text (window->combo_variable, i, buffer);
01791
01792
        gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), i);
01793
         g_signal_handler_unblock (window->combo_variable, window->id_variable);
01794 #if DEBUG_INTERFACE
01795 fprintf (stderr, "window_label_variable: end\n");
01796 #endif
01797 }
01798
01802 static void
01803 window_precision_variable ()
01804 {
01805
        unsigned int i:
01806 #if DEBUG_INTERFACE
        fprintf (stderr, "window_precision_variable: start\n");
01807
01808 #endif
01809
        i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01810
        input->variable[i].precision
          = (unsigned int) gtk_spin_button_get_value_as_int (window->spin_precision);
01811
        gtk_spin_button_set_digits (window->spin_min, input->variable[i].precision); gtk_spin_button_set_digits (window->spin_max, input->variable[i].precision);
01812
01813
01814
        gtk_spin_button_set_digits (window->spin_minabs,
01815
                                       input->variable[i].precision);
       gtk_spin_button_set_digits (window->spin_maxabs,
01816
                                       input->variable[i].precision);
01817
01818 #if DEBUG_INTERFACE
01819 fprintf (stderr, "window_precision_variable: end\n");
01820 #endif
01821 }
01822
01826 static void
01827 window_rangemin_variable ()
01828 {
        unsigned int i;
01829
01830 #if DEBUG_INTERFACE
01831
       fprintf (stderr, "window_rangemin_variable: start\n");
01832 #endif
01833 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01834 input->variable[i].rangemin = gtk_spin_button_get_value (window->spin
        input->variable[i].rangemin = gtk_spin_button_get_value (window->spin_min);
01835 #if DEBUG_INTERFACE
01836 fprintf (stderr, "window_rangemin_variable: end\n");
01837 #endif
01838 }
01839
01843 static void
01844 window_rangemax_variable ()
01845 {
01846
        unsigned int i;
01847 #if DEBUG_INTERFACE
       fprintf (stderr, "window_rangemax_variable: start\n");
01848
01849 #endif
01850 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
        input->variable[i].rangemax = gtk_spin_button_get_value (window->spin_max);
01851
01852 #if DEBUG INTERFACE
       fprintf (stderr, "window_rangemax_variable: end\n");
01853
01854 #endif
01855 }
01856
01860 static void
01861 window_rangeminabs_variable ()
01862 {
01863
        unsigned int i:
01864 #if DEBUG_INTERFACE
01865
        fprintf (stderr, "window_rangeminabs_variable: start\n");
01866 #endif
01867
       i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01868 input->variable[i].rangeminabs
01869
          = gtk_spin_button_get_value (window->spin_minabs);
01870 #if DEBUG_INTERFACE
```

```
fprintf (stderr, "window_rangeminabs_variable: end\n");
01872 #endif
01873 }
01874
01878 static void
01879 window_rangemaxabs_variable ()
01880 {
01881
        unsigned int i;
01882 #if DEBUG_INTERFACE
       fprintf (stderr, "window_rangemaxabs_variable: start\n");
01883
01884 #endif
01885 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01886
       input->variable[i].rangemaxabs
          = gtk_spin_button_get_value (window->spin_maxabs);
01887
01888 #if DEBUG_INTERFACE
01889
       fprintf (stderr, "window_rangemaxabs_variable: end\n");
01890 #endif
01891 }
01892
01896 static void
01897 window_step_variable ()
01898 {
01899
       unsigned int i;
01900 #if DEBUG_INTERFACE
01901
       fprintf (stderr, "window_step_variable: start\n");
01902 #endif
      i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01903
01904
       input->variable[i].step = gtk_spin_button_get_value (window->spin_step);
01905 #if DEBUG_INTERFACE
       fprintf (stderr, "window_step_variable: end\n");
01906
01907 #endif
01908 }
01909
01913 static void
01914 window_update_variable ()
01915 {
01916
       int i;
01917 #if DEBUG_INTERFACE
01918
       fprintf (stderr, "window_update_variable: start\n");
01919 #endif
01920
       i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01921
       if (i < 0)
         i = 0;
01922
01923
       switch (window_get_algorithm ())
01924
01925
          case ALGORITHM_SWEEP:
01926
         case ALGORITHM_ORTHOGONAL:
01927
           input->variable[i].nsweeps
01928 = gtk_spin_button_get_value_as_int (window->spin_sweeps);
01929 #if DEBUG_INTERFACE
01930
           fprintf (stderr, "window_update_variable: nsweeps[%d]=%u\n", i,
01931
                     input->variable[i].nsweeps);
01932 #endif
01933
           break;
          case ALGORITHM_GENETIC:
01934
          input->variable[i].nbits
01935
01936
              = gtk_spin_button_get_value_as_int (window->spin_bits);
01937 #if DEBUG_INTERFACE
01938
       fprintf (stderr, "window_update_variable: nbits[%d]=%u\n", i,
01939
                     input->variable[i].nbits);
01940 #endif
01941
01942 #if DEBUG_INTERFACE
01943
       fprintf (stderr, "window_update_variable: end\n");
01944 #endif
01945 }
01946
01952 static int
01953 window_read (char *filename)
01954 {
01955
        unsigned int i;
01956 #if DEBUG_INTERFACE
01957 fprintf (stderr, "window_read: start\n");
01958 fprintf (stderr, "window_read: file name=%s\n", filename);
01959 #endif
01960
01961
       // Reading new input file
       input_free ();
input->result = input->variables = NULL;
01962
01963
01964
       if (!input_open (filename))
01965
01966 #if DEBUG_INTERFACE
01967
           fprintf (stderr, "window_read: end\n");
01968 #endif
01969
            return 0;
01970
          }
01971
```

```
01972
         // Setting GTK+ widgets data
01973
         gtk_entry_set_text (window->entry_result, input->result);
        gtk_entry_set_text (window->entry_variables, input->variables);
gtk_button_set_label (window->button_simulator, input->simulator);
01974
01975
01976
        gtk_check_button_set_active (window->check_evaluator,
01977
                                          (size_t) input->evaluator);
01978
         if (input->evaluator)
01979
           gtk_button_set_label (window->button_evaluator, input->evaluator);
01980
         gtk_check_button_set_active (window->button_algorithm[input->algorithm],
01981
                                          TRUE);
01982
         switch (input->algorithm)
01983
01984
           case ALGORITHM_MONTE_CARLO:
01985
             gtk_spin_button_set_value (window->spin_simulations,
01986
                                            (gdouble) input->nsimulations);
01987
             // fallthrough
           case ALGORITHM_SWEEP:
01988
           case ALGORITHM_ORTHOGONAL:
01989
01990
             gtk_spin_button_set_value (window->spin_iterations,
01991
                                            (gdouble) input->niterations);
01992
             gtk_spin_button_set_value (window->spin_bests, (gdouble) input->nbest);
01993
             gtk_spin_button_set_value (window->spin_tolerance, input->tolerance);
             gtk_check_button_set_active (window->check_climbing, input->nsteps);
01994
01995
             if (input->nsteps)
01996
               {
01997
                  gtk_check_button_set_active
01998
                    (window->button_climbing[input->climbing], TRUE);
01999
                  gtk_spin_button_set_value (window->spin_steps,
02000
                                                 (gdouble) input->nsteps);
                  gtk_spin_button_set_value (window->spin_relaxation,
02001
02002
                                                 (gdouble) input->relaxation);
02003
                  switch (input->climbing)
02004
02005
                    case CLIMBING_METHOD_RANDOM:
02006
                      gtk_spin_button_set_value (window->spin_estimates,
02007
                                                     (gdouble) input->nestimates);
02008
                    }
02009
               }
02010
             break;
02011
           default:
02012
             gtk_spin_button_set_value (window->spin_population,
                                            (gdouble) input->nsimulations);
02013
02014
             gtk_spin_button_set_value (window->spin_generations,
                                            (gdouble) input->niterations);
02015
             gtk_spin_button_set_value (window->spin_mutation, input->mutation_ratio);
02016
02017
             gtk_spin_button_set_value (window->spin_reproduction,
02018
                                            input->reproduction_ratio);
02019
             gtk_spin_button_set_value (window->spin_adaptation,
                                            input->adaptation_ratio);
02020
02021
02022
         gtk_check_button_set_active (window->button_norm[input->norm], TRUE);
02023
         gtk_spin_button_set_value (window->spin_p, input->p);
02024
         gtk_spin_button_set_value (window->spin_threshold, input->threshold);
         g_signal_handler_block (window->combo_experiment, window->id_experiment);
gtk_combo_box_text_remove_all (window->combo_experiment);
02025
02026
02027
         for (i = 0; i < input->nexperiments; ++i)
           gtk_combo_box_text_append_text (window->combo_experiment,
02028
02029
                                               input->experiment[i].name);
02030
         g_signal_handler_unblock (window->combo_experiment, window->id_experiment);
02031
         gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), 0);
        g_signal_handler_block (window->combo_variable, window->id_variable);
g_signal_handler_block (window->entry_variable, window->id_variable_label);
02032
02033
02034
         gtk_combo_box_text_remove_all (window->combo_variable);
02035
              (i = 0; i < input->nvariables; ++i)
02036
           gtk_combo_box_text_append_text (window->combo_variable,
02037
                                               input->variable[i].name);
        g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
g_signal_handler_unblock (window->combo_variable, window->id_variable);
gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), 0);
02038
02039
02040
02041
         window_set_variable ();
02042
        window_update ();
02043
02044 #if DEBUG_INTERFACE
02045 fprintf (stderr, "window_read: end\n");
02046 #endif
02047
        return 1;
02048 }
02049
02053 static void
02054 dialog_open_close (GtkFileChooserDialog * dlg,
02056
                            int response id)
02057 {
02058
       char *buffer, *directory, *name;
02059
       GFile *file;
02060
02061 #if DEBUG INTERFACE
        fprintf (stderr, "dialog_open_close: start\n");
02062
```

```
02063 #endif
02064
02065
         // Saving a backup of the current input file
02066
        directory = g_strdup (input->directory);
02067
        name = g_strdup (input->name);
02068
02069
         // If OK saving
02070
        if (response_id == GTK_RESPONSE_OK)
02071
02072
02073
             // Traying to open the input file
             file = gtk_file_chooser_get_file (GTK_FILE_CHOOSER (dlg));
buffer = g_file_get_path (file);
02074
02075
02076 #if DEBUG_INTERFACE
02077
             fprintf (stderr, "dialog_open_close: file name=%s\n", buffer);
02078 #endif
             if (!window_read (buffer))
02079
02080
02081 #if DEBUG_INTERFACE
                  fprintf (stderr, "dialog_open_close: error reading input file\n");
02083 #endif
02084
                 g_free (buffer);
02085
                 // Reading backup file on error
buffer = g_build_filename (directory, name, NULL);
02086
02087
                  input->result = input->variables = NULL;
02088
02089
                  if (!input_open (buffer))
02090
02091
                       // Closing on backup file reading error
02092
02093 #if DEBUG_INTERFACE
02094
                    fprintf (stderr,
02095
                                 "dialog_open_close: error reading backup file\n");
02096 #endif
                  }
02097
02098
02099
             g free (buffer);
             g_object_unref (file);
02100
02101
02102
02103
        // Freeing and closing
02104
       g_free (name);
02105
        a free (directory):
02106
        gtk_window_destroy (GTK_WINDOW (dlg));
02107
02108 #if DEBUG_INTERFACE
02109 fprintf (stderr, "dialog_open_close: end\n");
02110 #endif
02111
02112 }
02113
02117 static void
02118 window_open ()
02119 {
        GtkFileChooserDialog *dlg;
02120
02121
        GtkFileFilter *filter;
02122
02123 #if DEBUG_INTERFACE
02124 fprintf (stderr, "window_open: start\n");
02125 #endif
02126
02127
         // Opening dialog
02128
        dlg = (GtkFileChooserDialog *)
02129
         gtk_file_chooser_dialog_new (_("Open input file"),
02130
                                             window->window.
02131
                                            GTK_FILE_CHOOSER_ACTION_OPEN,
                                            _("_Cancel"), GTK_RESPONSE_CANCEL, _("_OK"), GTK_RESPONSE_OK, NULL);
02132
02133
02134
         // Adding XML filter
02136
         filter = (GtkFileFilter *) gtk_file_filter_new ();
02137
         gtk_file_filter_set_name (filter, "XML");
         gtk_file_filter_add_pattern (filter, "*.xml");
gtk_file_filter_add_pattern (filter, "*.XML");
02138
02139
         gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter);
02140
02141
         // Adding JSON filter
02142
         filter = (GtkFileFilter *) gtk_file_filter_new ();
gtk_file_filter_set_name (filter, "JSON");
02143
02144
         gtk_file_filter_add_pattern (filter, "*.json");
gtk_file_filter_add_pattern (filter, "*.JSON");
02145
02146
         gtk_file_filter_add_pattern (filter, "*.js");
gtk_file_filter_add_pattern (filter, "*.JS");
02147
02148
02149
         gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter);
02150
        // Connecting the close function
g_signal_connect (dlg, "response", G_CALLBACK (dialog_open_close), NULL);
02151
02152
```

```
// Showing modal dialog
02154
02155
        gtk_window_present (GTK_WINDOW (dlg));
02156
02157 #if DEBUG_INTERFACE
       fprintf (stderr, "window_open: end\n");
02158
02159 #endif
02160 }
02161
02165 static void
02166 dialog_simulator_close (GtkFileChooserDialog * dlg,
02168
                                int response id)
02169 {
02170 GFile *file1, *file2;
02171 char *buffer1, *buffer2;
02172 #if DEBUG_INTERFACE
02173 fprintf (stderr, "dialog_simulator_close: start\n");
02174 #endif
       if (response_id == GTK_RESPONSE_OK)
02176
          {
02177
            buffer1 = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
02178
            file1 = g_file_new_for_path (buffer1);
            file2 = g_file_new_for_path (input->directory);
buffer2 = g_file_get_relative_path (file2, file1);
02179
02180
02181
            input->simulator = g_strdup (buffer2);
02182
            g_free (buffer2);
02183
            g_object_unref (file2);
02184
             g_object_unref (file1);
02185
            g_free (buffer1);
02186
02187
        gtk_window_destroy (GTK_WINDOW (dlg));
02188 #if DEBUG_INTERFACE
02189
       fprintf (stderr, "dialog_simulator_close: end\n");
02190 #endif
02191 }
02192
02196 static void
02197 dialog_simulator ()
02198 {
02199
        GtkFileChooserDialog *dlg;
02200 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_simulator: start\n");
02201
02202 #endif
02203
       dlg = (GtkFileChooserDialog *)
         gtk_file_chooser_dialog_new (_("Open simulator file"),
02204
02205
                                          window->window.
02206
                                         GTK FILE CHOOSER ACTION OPEN.
                                         _("_Cancel"), GTK_RESPONSE_CANCEL,
_("_Open"), GTK_RESPONSE_ACCEPT, NULL);
02207
02208
       q_signal_connect (dlq, "response", G_CALLBACK (dialog_simulator_close), NULL);
02209
        gtk_window_present (GTK_WINDOW (dlg));
02211 #if DEBUG_INTERFACE
02212
       fprintf (stderr, "dialog_simulator: end\n");
02213 #endif
02214 }
02215
02219 static void
02220 dialog_evaluator_close (GtkFileChooserDialog * dlg,
02222
                                int response_id)
02223 {
02224 GFile *file1, *file2;
        char *buffer1, *buffer2;
02225
02226 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_evaluator_close: start\n");
02227
02228 #endif
02229
       if (response_id == GTK_RESPONSE_OK)
02230
02231
            buffer1 = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
02232
            file1 = g_file_new_for_path (buffer1);
             file2 = g_file_new_for_path (input->directory);
02234
            buffer2 = g_file_get_relative_path (file2, file1);
02235
            input->evaluator = g_strdup (buffer2);
02236
            g_free (buffer2);
            g_object_unref (file2);
g_object_unref (file1);
02237
02238
02239
            g_free (buffer1);
02240
02241
       gtk_window_destroy (GTK_WINDOW (dlg));
02242 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_evaluator_close: end\n");
02243
02244 #endif
02245 }
02246
02250 static void
02251 dialog_evaluator ()
02252 {
02253
       GtkFileChooserDialog *dlg;
```

```
02254 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_evaluator: start\n");
02255
02256 #endif
02257
       dlg = (GtkFileChooserDialog *)
          {\tt gtk\_file\_chooser\_dialog\_new (\_("Open evaluator file"),}
02258
02259
                                         window->window.
02260
                                        GTK_FILE_CHOOSER_ACTION_OPEN,
02261
                                        _("_Cancel"), GTK_RESPONSE_CANCEL,
        __("_Open"), GTK_RESPONSE_ACCEPT, NULL);
g_signal_connect (dlg, "response", G_CALLBACK (dialog_evaluator_close), NULL);
02262
02263
        gtk_window_present (GTK_WINDOW (dlg));
02264
02265 #if DEBUG INTERFACE
02266
       fprintf (stderr, "dialog_evaluator: end\n");
02267 #endif
02268 }
02269
02273 void
02274 window_new (GtkApplication * application)
02275 {
02276
        unsigned int i;
02277
        char *buffer, *buffer2, buffer3[64];
02278
        const char *label_algorithm[NALGORITHMS] = {
          "_Monte-Carlo", _("_Sweep"), _("_Genetic"), _("_Orthogonal")
02279
02280
02281
        const char *tip_algorithm[NALGORITHMS] = {
          _("Monte-Carlo brute force algorithm"),
02282
02283
          _("Sweep brute force algorithm"),
02284
          _("Genetic algorithm"),
02285
          _("Orthogonal sampling brute force algorithm"),
02286
        const char *label_climbing[NCLIMBINGS] = {
02287
02288
          _("_Coordinates climbing"), _("_Random climbing")
02289
02290
        const char *tip_climbing[NCLIMBINGS] = {
02291
         _("Coordinates climbing estimate method"),
02292
          _("Random climbing estimate method")
02293
        };
02294
        const char *label_norm[NNORMS] = { "L2", "L", "Lp", "L1" };
        const char *tip_norm[NNORMS] =
02295
02296
         _("Euclidean error norm (L2)"),
02297
          _("Maximum error norm (L)"),
          _("P error norm (Lp)"),
02298
          _("Taxicab error norm (L1)")
02299
02300
02301 #if !GTK4
02302
        const char *close = "delete-event";
02303 #else
       const char *close = "close-request";
02304
02305 #endif
02306
02307 #if DEBUG_INTERFACE
02308
       fprintf (stderr, "window_new: start\n");
02309 #endif
02310
02311
        // Creating the window
02312
        window->window = window parent = main window
02313
          = (GtkWindow *) gtk_application_window_new (application);
02314
02315
        // Finish when closing the window
02316
        g_signal_connect_swapped (window->window, close,
                                   G_CALLBACK (g_application_quit),
02317
02318
                                   G_APPLICATION (application));
02319
02320
        // Setting the window title
02321
        gtk_window_set_title (window->window, "MPCOTool");
02322
02323
        // Creating the open button
02324
       window->button_open = (GtkButton *)
02325 #if !GTK4
02326
          gtk_button_new_from_icon_name ("document-open", GTK_ICON_SIZE_BUTTON);
02327 #else
02328
          gtk_button_new_from_icon_name ("document-open");
02329 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_open),
02330
02331
        _("Open a case"));
g_signal_connect (window->button_open, "clicked", window_open, NULL);
02332
02333
02334
        // Creating the save button
02335
        window->button_save = (GtkButton *)
02336 #if !GTK4
02337
          gtk_button_new_from_icon_name ("document-save", GTK_ICON_SIZE_BUTTON);
02338 #else
02339
          gtk_button_new_from_icon_name ("document-save");
02340 #endif
02341
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_save),
        _("Save the case"));
g_signal_connect (window->button_save, "clicked", (GCallback) window_save,
02342
02343
```

```
02344
                            NULL);
02345
02346
         // Creating the run button
02347
        window->button_run = (GtkButton *)
02348 #if !GTK4
02349
           gtk button new from icon name ("system-run", GTK ICON SIZE BUTTON);
02350 #else
02351
          gtk_button_new_from_icon_name ("system-run");
02352 #endif
02353
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_run),
        __("Run the optimization"));
g_signal_connect (window->button_run, "clicked", window_run, NULL);
02354
02355
02356
02357
         // Creating the options button
02358
        window->button_options = (GtkButton *)
02359 #if !GTK4
           gtk_button_new_from_icon_name ("preferences-system", GTK_ICON_SIZE_BUTTON);
02360
02361 #else
02362
          gtk_button_new_from_icon_name ("preferences-system");
02363 #endif
02364
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_options),
02365
                                          _("Edit the case"));
        g_signal_connect (window->button_options, "clicked", options_new, NULL);
02366
02367
02368
        // Creating the help button
02369
        window->button_help = (GtkButton *)
02370 #if !GTK4
02371
           gtk_button_new_from_icon_name ("help-browser", GTK_ICON_SIZE_BUTTON);
02372 #else
          gtk_button_new_from_icon_name ("help-browser");
02373
02374 #endif
02375
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_help),
                                                                               _("Help"));
        g_signal_connect (window->button_help, "clicked", window_help, NULL);
02376
02377
02378
         \ensuremath{//} Creating the about button
02379
        window->button_about = (GtkButton *)
02380 #if !GTK4
02381
          gtk_button_new_from_icon_name ("help-about", GTK_ICON_SIZE_BUTTON);
02382 #else
           gtk_button_new_from_icon_name ("help-about");
02383
02384 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_about), _("About"));
g_signal_connect (window->button_about, "clicked", window_about, NULL);
02385
02386
02387
02388
         // Creating the exit button
        window->button_exit = (GtkButton *)
02389
02390 #if !GTK4
02391
           gtk_button_new_from_icon_name ("application-exit", GTK_ICON_SIZE_BUTTON);
02392 #else
02393
          gtk button new from icon name ("application-exit");
02394 #endif
02395
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_exit), _("Exit"));
02396
        g_signal_connect_swapped (window->button_exit, "clicked")
02397
                                     G\_CALLBACK (g\_application\_quit),
02398
                                     G_APPLICATION (application));
02399
02400
        // Creating the buttons bar
02401
         window->box_buttons = (GtkBox *) gtk_box_new (GTK_ORIENTATION_HORIZONTAL, 0);
02402
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_open));
02403
         gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_save));
02404
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_run));
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_options));
gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_help));
02405
02406
02407
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_about));
02408
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_exit));
02409
02410
        \ensuremath{//} Creating the simulator program label and entry
        window->label_simulator = (GtkLabel *) gtk_label_new (_("Simulator program"));
window->button_simulator = (GtkButton *)
02411
02412
02413
          gtk_button_new_with_mnemonic (_("Simulator program"));
02414
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_simulator),
02415
                                         _("Simulator program executable file"));
        gtk_widget_set_hexpand (GTK_WIDGET (window->button_simulator), TRUE);
g_signal_connect (window->button_simulator, "clicked",
02416
02417
                            G_CALLBACK (dialog_simulator), NULL);
02418
02419
02420
         // Creating the evaluator program label and entry
02421
        window->check_evaluator = (GtkCheckButton *)
02422
          gtk_check_button_new_with_mnemonic (_("_Evaluator program"));
        g_signal_connect (window->check_evaluator, "toggled", window_update, NULL);
02423
        window->button_evaluator = (GtkButton *)
02424
02425
           gtk_button_new_with_mnemonic (_("Evaluator program"));
02426
        gtk_widget_set_tooltip_text
02427
           (GTK_WIDGET (window->button_evaluator),
        _("Optional evaluator program executable file"));
g_signal_connect (window->button_evaluator, "clicked"
02428
02429
02430
                            G_CALLBACK (dialog_evaluator), NULL);
```

```
02431
        // Creating the results files labels and entries
02432
        window->label_result = (GtkLabel *) gtk_label_new (_("Result file"));
window->entry_result = (GtkEntry *) gtk_entry_new ();
02433
02434
02435
        gtk_widget_set_tooltip_text
        (GTK_WIDGET (window->entry_result), _("Best results file"));
window->label_variables = (GtkLabel *) gtk_label_new (_("Variables file"));
02436
02437
02438
        window->entry_variables = (GtkEntry *) gtk_entry_new ();
02439
        gtk_widget_set_tooltip_text
02440
           (GTK_WIDGET (window->entry_variables), _("All simulated results file"));
02441
02442
        // Creating the files grid and attaching widgets
        window->grid_files = (GtkGrid *) gtk_grid_new ();
gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_simulator),
02443
02444
02445
                           0, 0, 1, 1);
02446
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_simulator),
02447
                           1, 0, 1, 1);
02448
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->check_evaluator),
02449
                           0, 1, 1, 1);
02450
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_evaluator),
02451
                           1, 1, 1, 1);
02452
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_result),
02453
                           0, 2, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->entry_result),
02454
02455
                           1, 2, 1, 1);
02456
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_variables),
02457
                           0, 3, 1, 1);
02458
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->entry_variables),
02459
                           1, 3, 1, 1);
02460
02461
        // Creating the algorithm properties
02462
        window->label_simulations
                                      = (GtkLabel *) gtk_label_new
02463
           (_("Simulations number"));
02464
        window->spin_simulations
02465
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02466
        {\tt gtk\_widget\_set\_tooltip\_text}
02467
           (GTK_WIDGET (window->spin_simulations),
            _("Number of simulations to perform for each iteration"));
02468
02469
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_simulations), TRUE);
02470
        window->label_iterations = (GtkLabel *)
02471
          gtk_label_new (_("Iterations number"));
        window->spin iterations
02472
02473
          = (GtkSpinButton \star) gtk_spin_button_new_with_range (1., 1.e6, 1.);
02474
        gtk_widget_set_tooltip_text
02475
           (GTK_WIDGET (window->spin_iterations), _("Number of iterations"));
02476
        g_signal_connect
02477
           (window->spin_iterations, "value-changed", window_update, NULL);
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_iterations), TRUE);
02478
        window->label_tolerance = (GtkLabel *) gtk_label_new (_("Tolerance"));
window->spin_tolerance =
02479
02480
02481
           (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02482
        gtk_widget_set_tooltip_text
02483
           (GTK_WIDGET (window->spin_tolerance),
02484
            _("Tolerance to set the variable interval on the next iteration"));
02485
        window->label_bests = (GtkLabel *) gtk_label_new (_("Bests number"));
        window->spin_bests
02486
02487
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e6, 1.);
02488
        gtk_widget_set_tooltip_text
02489
           (GTK_WIDGET (window->spin_bests),
           _("Number of best simulations used to set the variable interval " \,
02490
02491
              "on the next iteration"));
02492
        window->label_population
        = (GtkLabel *) gtk_label_new (_("Population number")); window->spin_population
02493
02494
02495
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02496
        gtk_widget_set_tooltip_text
02497
           (GTK_WIDGET (window->spin_population),
        _("Number of population for the genetic algorithm"));
gtk_widget_set_hexpand (GTK_WIDGET (window->spin_population), TRUE);
02498
02499
02500
        window->label_generations
02501
           = (GtkLabel *) gtk_label_new (_("Generations number"));
02502
        window->spin_generations
02503
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e6, 1.);
02504
        {\tt gtk\_widget\_set\_tooltip\_text}
          (GTK_WIDGET (window->spin_generations),
_("Number of generations for the genetic algorithm"));
02505
02506
02507
        window->label_mutation = (GtkLabel *) gtk_label_new (_("Mutation ratio"));
02508
        window->spin_mutation
02509
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
        gtk_widget_set_tooltip_text
02510
02511
           (GTK WIDGET (window->spin mutation),
02512
            _("Ratio of mutation for the genetic algorithm"));
02513
        window->label_reproduction
02514
           = (GtkLabel *) gtk_label_new (_("Reproduction ratio"));
02515
        window->spin_reproduction
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02516
02517
        gtk_widget_set_tooltip_text
```

```
(GTK_WIDGET (window->spin_reproduction),
            _("Ratio of reproduction for the genetic algorithm"));
02519
02520
        window->label_adaptation = (GtkLabel *) gtk_label_new (_("Adaptation ratio"));
02521
        window->spin_adaptation
02522
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
        gtk_widget_set_tooltip_text
  (GTK_WIDGET (window->spin_adaptation),
02523
            _("Ratio of adaptation for the genetic algorithm"));
02525
        window->label_threshold = (GtkLabel *) gtk_label_new (_("Threshold"));
window->spin_threshold = (GtkSpinButton *)
02526
02527
          gtk_spin_button_new_with_range (-G_MAXDOUBLE, G_MAXDOUBLE,
02528
                                             precision[DEFAULT_PRECISION]);
02529
02530
        gtk widget set tooltip text
02531
           (GTK_WIDGET (window->spin_threshold),
02532
           _("Threshold in the objective function to finish the simulations"));
02533
        window->scrolled_threshold = (GtkScrolledWindow *)
02534 #if !GTK4
02535
          gtk scrolled window new (NULL, NULL);
02536 #else
          gtk_scrolled_window_new ();
02538 #endif
02539
        gtk_scrolled_window_set_child (window->scrolled_threshold,
02540
                                          GTK_WIDGET (window->spin_threshold));
          gtk_widget_set_hexpand (GTK_WIDGET (window->scrolled_threshold), TRUE);
gtk_widget_set_halign (GTK_WIDGET (window->scrolled_threshold),
02541 //
02542 //
                                          GTK_ALIGN_FILL);
02543 //
02544
02545
         // Creating the hill climbing method properties
02546
        window->check_climbing = (GtkCheckButton *)
        gtk_check_button_new_with_mnemonic (_("-Hill climbing method"));
g_signal_connect (window->check_climbing, "toggled", window_update, NULL);
02547
02548
02549
        window->grid_climbing = (GtkGrid *) gtk_grid_new ();
02550 #if !GTK4
02551
        window->button_climbing[0] = (GtkRadioButton *)
02552
          gtk_radio_button_new_with_mnemonic (NULL, label_climbing[0]);
02553 #else
        window->button climbing[0] = (GtkCheckButton *)
02554
          gtk_check_button_new_with_mnemonic (label_climbing[0]);
02556 #endif
02557
        gtk_grid_attach (window->grid_climbing,
02558
                           GTK_WIDGET (window->button_climbing[0]), 0, 0, 1, 1);
        g_signal_connect (window->button_climbing[0], "toggled", window_update, NULL);
for (i = 0; ++i < NCLIMBINGS;)</pre>
02559
02560
02561
02562 #if !GTK4
02563
             window->button_climbing[i] = (GtkRadioButton *)
02564
               gtk_radio_button_new_with_mnemonic
02565
               (gtk_radio_button_get_group (window->button_climbing[0]),
02566
                label_climbing[i]);
02567 #else
02568
            window->button_climbing[i] = (GtkCheckButton *)
02569
               gtk_check_button_new_with_mnemonic (label_climbing[i]);
02570
             gtk_check_button_set_group (window->button_climbing[i]
02571
                                           window->button_climbing[0]);
02572 #endif
02573
            gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_climbing[i]),
02574
                                            tip_climbing[i]);
             gtk_grid_attach (window->grid_climbing,
02575
             GTK_WIDGET (window->button_climbing[i]), 0, i, 1, 1);
g_signal_connect (window->button_climbing[i], "toggled", window_update,
02576
02577
                                NULT.):
02578
02579
02580
        window->label_steps = (GtkLabel *) gtk_label_new (_("Steps number"));
        window->spin_steps = (GtkSpinButton *)
  gtk_spin_button_new_with_range (1., 1.e12, 1.);
02581
02582
02583
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_steps), TRUE);
02584
        window->label_estimates
          = (GtkLabel *) gtk_label_new (_("Climbing estimates number"));
02585
        window->spin_estimates = (GtkSpinButton *)
02586
          gtk_spin_button_new_with_range (1., 1.e3, 1.);
02588
        window->label_relaxation
02589
          = (GtkLabel *) gtk_label_new (_("Relaxation parameter"));
        window->spin_relaxation = (GtkSpinButton *)
02590
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->label_steps),
02591
02592
                           0, NCLIMBINGS, 1, 1);
02593
02594
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_steps),
02595
                           1, NCLIMBINGS, 1, 1);
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->label_estimates),
02596
                           0, NCLIMBINGS + 1, 1, 1);
02597
        \verb|gtk_grid_attach| is $$ \dot{u}$ indow->grid_climbing, $$ GTK_WIDGET (window->spin_estimates), $$
02598
02599
                           1, NCLIMBINGS + 1, 1, 1);
        02600
02601
        {\tt gtk\_grid\_attach~(window->grid\_climbing,~GTK\_WIDGET~(window->spin\_relaxation),}
02602
                           1, NCLIMBINGS + 2, 1, 1);
02603
02604
```

```
// Creating the array of algorithms
02606
       window->grid_algorithm = (GtkGrid *) gtk_grid_new ();
02607 #if !GTK4
02608
       window->button_algorithm[0] = (GtkRadioButton *)
02609
         gtk_radio_button_new_with_mnemonic (NULL, label_algorithm[0]);
02610 #else
02611
       window->button_algorithm[0] = (GtkCheckButton *)
         gtk_check_button_new_with_mnemonic (label_algorithm[0]);
02612
02613 #endif
02614
       gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_algorithm[0]),
02615
                                     tip_algorithm[0]);
       gtk_grid_attach (window->grid_algorithm,
02616
                        GTK_WIDGET (window->button_algorithm[0]), 0, 0, 1, 1);
02617
02618
       g_signal_connect (window->button_algorithm[0], "toggled",
02619
                          window_set_algorithm, NULL);
02620
       for (i = 0; ++i < NALGORITHMS;)</pre>
02621
02622 #if !GTK4
02623
           window->button_algorithm[i] = (GtkRadioButton *)
02624
             gtk_radio_button_new_with_mnemonic
02625
              (gtk_radio_button_get_group (window->button_algorithm[0]),
02626
               label_algorithm[i]);
02627 #else
           window->button algorithm[i] = (GtkCheckButton *)
02628
             qtk_check_button_new_with_mnemonic (label_algorithm[i]);
02629
            gtk_check_button_set_group (window->button_algorithm[i],
02630
                                        window->button_algorithm[0]),
02631
02632 #endif
02633
              gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_algorithm[i]),
02634
                                           tip_algorithm[i]);
02635
            gtk_grid_attach (window->grid_algorithm,
02636
                             GTK_WIDGET (window->button_algorithm[i]), 0, i, 1, 1);
            g_signal_connect (window->button_algorithm[i], "toggled",
02637
02638
                              window_set_algorithm, NULL);
02639
       gtk_grid_attach (window->grid_algorithm,
02640
                        GTK WIDGET (window->label simulations),
02641
                         0, NALGORITHMS, 1, 1);
02642
02643
       gtk_grid_attach (window->grid_algorithm,
02644
                         GTK_WIDGET (window->spin_simulations), 1, NALGORITHMS, 1, 1);
02645
       gtk_grid_attach (window->grid_algorithm,
                         GTK_WIDGET (window->label_iterations),
02646
                         0, NALGORITHMS + 1, 1, 1);
02647
02648
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_iterations),
                         1, NALGORITHMS + 1, 1, 1);
02649
02650
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_tolerance),
02651
                         0, NALGORITHMS + 2, 1, 1);
       02652
02653
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_bests),
02654
02655
                         0, NALGORITHMS + 3, 1, 1);
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_bests),
02656
02657
                         1, NALGORITHMS + 3, 1, 1);
02658
       gtk_grid_attach (window->grid_algorithm
02659
                         GTK_WIDGET (window->label_population),
                         0, NALGORITHMS + 4, 1, 1);
02660
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_population),
02661
                         1, NALGORITHMS + 4, 1, 1);
02662
02663
       gtk_grid_attach (window->grid_algorithm,
                        GTK_WIDGET (window->label_
0, NALGORITHMS + 5, 1, 1);
02664
                                                  _generations),
02665
       gtk_grid_attach (window->grid_algorithm,
02666
02667
                        GTK_WIDGET (window->spin_generations),
                         1, NALGORITHMS + 5, 1, 1);
02668
02669
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_mutation),
02670
                         0, NALGORITHMS + 6, 1, 1);
02671
       02672
02673
       gtk_grid_attach (window->grid_algorithm,
                         GTK_WIDGET (window->label_reproduction),
02675
                         0, NALGORITHMS + 7, 1, 1);
02676
       gtk_grid_attach (window->grid_algorithm,
02677
                         GTK_WIDGET (window->spin_reproduction),
02678
                         1, NALGORITHMS + 7, 1, 1);
02679
       gtk_grid_attach (window->grid_algorithm,
                         GTK_WIDGET (window->label_adaptation),
02680
                         0, NALGORITHMS + 8, 1, 1);
02681
02682
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_adaptation),
02683
                         1, NALGORITHMS + 8, 1, 1);
       \verb|gtk_grid_attach| (\verb|window->grid_algorithm|, GTK_WIDGET| (\verb|window->check_climbing)|, \\
02684
       0, NALGORITHMS + 9, 2, 1);
gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->grid_climbing),
02685
02686
                         0, NALGORITHMS + 10, 2, 1);
02687
02688
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_threshold),
02689
                         0, NALGORITHMS + 11, 1, 1);
02690
       gtk_grid_attach (window->grid_algorithm,
                         GTK_WIDGET (window->scrolled_threshold),
02691
```

```
1, NALGORITHMS + 11, 1, 1);
         window->frame_algorithm = (GtkFrame *) gtk_frame_new (_("Algorithm"));
02693
02694
         gtk_frame_set_child (window->frame_algorithm,
02695
                                 GTK_WIDGET (window->grid_algorithm));
02696
02697
         // Creating the variable widgets
         window->combo_variable = (GtkComboBoxText *) gtk_combo_box_text_new ();
02698
02699
         {\tt gtk\_widget\_set\_tooltip\_text}
         (GTK_WIDGET (window->combo_variable), _("Variables selector"));
window->id_variable = g_signal_connect
  (window->combo_variable, "changed", window_set_variable, NULL);
02700
02701
02702
02703 #if !GTK4
02704
         window->button_add_variable = (GtkButton *)
           gtk_button_new_from_icon_name ("list-add", GTK_ICON_SIZE_BUTTON);
02705
02706 #else
02707
         window -> button\_add\_variable = (GtkButton *)
02708
           gtk_button_new_from_icon_name ("list-add");
02709 #endif
02710
         g_signal_connect (window->button_add_variable, "clicked", window_add_variable,
02711
                              NULL);
         gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_add_variable),
02712
02713
                                           _("Add variable"));
02714 #if !GTK4
02715
         window->button remove variable = (GtkButton *)
02716
           qtk_button_new_from_icon_name ("list-remove", GTK_ICON_SIZE_BUTTON);
02717 #else
         window->button\_remove\_variable = (GtkButton *)
02718
02719
           gtk_button_new_from_icon_name ("list-remove");
02720 #endif
02721
         g_signal_connect (window->button_remove_variable, "clicked",
02722
                              window remove variable, NULL);
02723
         gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_remove_variable),
02724
                                           _("Remove variable"));
02725
         window->label_variable = (GtkLabel *) gtk_label_new (_("Name"));
         window->entry_variable = (GtkEntry *) gtk_entry_new ();
02726
02727
         gtk_widget_set_tooltip_text
         (GTK_WIDGET (window->entry_variable), _("Variable name"));
gtk_widget_set_hexpand (GTK_WIDGET (window->entry_variable), TRUE);
02728
02729
         window->id_variable_label = g_signal_connect
  (window->entry_variable, "changed", window_label_variable, NULL);
02730
02731
02732
         window->label_min = (GtkLabel *) gtk_label_new (_("Minimum"));
         window->spin_min = (GtkSpinButton *) gtk_spin_button_new_with_range
02733
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02734
02735
         gtk_widget_set_tooltip_text
02736
           (GTK_WIDGET (window->spin_min), _("Minimum initial value of the variable"));
02737
         window->scrolled_min = (GtkScrolledWindow *)
02738 #if !GTK4
02739
           gtk_scrolled_window_new (NULL, NULL);
02740 #else
02741
           gtk scrolled window new ();
02742 #endif
02743
         gtk_scrolled_window_set_child (window->scrolled_min,
02744
                                             GTK_WIDGET (window->spin_min));
02745
         g_signal_connect (window->spin_min, "value-changed",
02746
                              window_rangemin_variable, NULL);
         window->label_max = (GtkLabel *) gtk_label_new (_("Maximum"));
window->spin_max = (GtkSpinButton *) gtk_spin_button_new_with_range
02747
02748
02749
            (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02750
         gtk_widget_set_tooltip_text
         (GTK_WIDGET (window->spin_max), _("Maximum window->scrolled_max = (GtkScrolledWindow *)
02751
                                                 _("Maximum initial value of the variable"));
02752
02753 #if !GTK4
02754
           gtk_scrolled_window_new (NULL, NULL);
02755 #else
02756
           gtk_scrolled_window_new ();
02757 #endif
02758
         gtk_scrolled_window_set_child (window->scrolled_max,
                                             GTK_WIDGET (window->spin_max));
02759
02760
         g_signal_connect (window->spin_max, "value-changed",
02761
                              window_rangemax_variable, NULL);
         window->check_minabs = (GtkCheckButton *)
  gtk_check_button_new_with_mnemonic (_("_Absolute minimum"));
g_signal_connect (window->check_minabs, "toggled", window_update, NULL);
window->spin_minabs = (GtkSpinButton *) gtk_spin_button_new_with_range
02762
02763
02764
02765
02766
            (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02767
         gtk_widget_set_tooltip_text
02768
           (GTK_WIDGET (window->spin_minabs),
02769
            _("Minimum allowed value of the variable"));
02770
         window->scrolled_minabs = (GtkScrolledWindow *)
02771 #if !GTK4
02772
           gtk scrolled window new (NULL, NULL);
02773 #else
02774
           gtk_scrolled_window_new ();
02775 #endif
02776
         gtk_scrolled_window_set_child (window->scrolled_minabs,
02777
                                             GTK_WIDGET (window->spin_minabs));
         g_signal_connect (window->spin_minabs, "value-changed",
02778
```

```
window_rangeminabs_variable, NULL);
        window->check_maxabs = (GtkCheckButton *)
02780
        gtk_check_button_new_with_mnemonic (_("_Absolute maximum"));
g_signal_connect (window->check_maxabs, "toggled", window_update, NULL);
02781
02782
        window->spin_maxabs = (GtkSpinButton *) gtk_spin_button_new_with_range
02783
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02784
02785
        gtk_widget_set_tooltip_text
02786
           (GTK_WIDGET (window->spin_maxabs),
02787
           _("Maximum allowed value of the variable"));
02788
        window->scrolled_maxabs = (GtkScrolledWindow *)
02789 #if !GTK4
02790
          gtk scrolled window new (NULL, NULL);
02791 #else
02792
          gtk_scrolled_window_new ();
02793 #endif
02794
        gtk_scrolled_window_set_child (window->scrolled_maxabs,
02795
                                         GTK_WIDGET (window->spin_maxabs));
        g_signal_connect (window->spin_maxabs, "value-changed", window_rangemaxabs_variable, NULL);
02796
02797
02798
        window->label_precision = (GtkLabel *) gtk_label_new (_("Precision digits"));
        window->spin_precision = (GtkSpinButton *)
02799
02800
          gtk_spin_button_new_with_range (0., (gdouble) DEFAULT_PRECISION, 1.);
02801
        gtk_widget_set_tooltip_text
02802
           (GTK_WIDGET (window->spin_precision),
02803
           _("Number of precision floating point digits\n"
              "0 is for integer numbers"));
02804
02805
        g_signal_connect (window->spin_precision, "value-changed",
02806
                           window_precision_variable, NULL);
02807
        window->label_sweeps = (GtkLabel *) gtk_label_new (_("Sweeps number"));
02808
        window->spin_sweeps =
        (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
gtk_widget_set_tooltip_text (GTK_WIDGET (window->spin_sweeps),
02809
02810
02811
                                       _("Number of steps sweeping the variable"));
02812
        g_signal_connect (window->spin_sweeps, "value-changed",
02813
                           window_update_variable, NULL);
        window->label_bits = (GtkLabel *) gtk_label_new (_("Bits number"));
02814
        window->spin_bits
02815
02816
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 64., 1.);
02817
        gtk_widget_set_tooltip_text
02818
          (GTK_WIDGET (window->spin_bits),
02819
            _("Number of bits to encode the variable"));
02820
        g_signal_connect
          (window->spin_bits, "value-changed", window_update_variable, NULL);
02821
        window->label_step = (GtkLabel *) gtk_label_new (_("Step size"));
window->spin_step = (GtkSpinButton *) gtk_spin_button_new_with_range
02822
02823
02824
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02825
        gtk_widget_set_tooltip_text
02826
          (GTK_WIDGET (window->spin_step),
        __("Initial step size for the hill climbing method"));
window->scrolled_step = (GtkScrolledWindow *)
02827
02828
02829 #if !GTK4
02830
          gtk_scrolled_window_new (NULL, NULL);
02831 #else
02832
          gtk_scrolled_window_new ();
02833 #endif
02834
        gtk_scrolled_window_set_child (window->scrolled_step,
                                         GTK_WIDGET (window->spin_step));
02835
02836
02837
           (window->spin_step, "value-changed", window_step_variable, NULL);
02838
        window->grid_variable = (GtkGrid *) gtk_grid_new ();
        gtk_grid_attach (window->grid_variable,
02839
02840
                          GTK_WIDGET (window->combo_variable), 0, 0, 2, 1);
02841
        gtk_grid_attach (window->grid_variable,
                          GTK_WIDGET (window->button_add_variable), 2, 0, 1, 1);
02842
02843
        gtk_grid_attach (window->grid_variable,
02844
                          GTK_WIDGET (window->button_remove_variable), 3, 0, 1, 1);
02845
        02846
02847
        gtk_grid_attach (window->grid_variable,
                          GTK_WIDGET (window->entry_variable), 1, 1, 3, 1);
02849
        gtk_grid_attach (window->grid_variable,
02850
                          GTK_WIDGET (window->label_min), 0, 2, 1, 1);
02851
        gtk_grid_attach (window->grid_variable,
                          GTK_WIDGET (window->scrolled_min), 1, 2, 3, 1);
02852
        gtk_grid_attach (window->grid_variable,
02853
                          GTK_WIDGET (window->label_max), 0, 3, 1, 1);
02854
        gtk_grid_attach (window->grid_variable,
02855
02856
                          GTK_WIDGET (window->scrolled_max), 1, 3, 3, 1);
02857
        gtk_grid_attach (window->grid_variable,
                          GTK WIDGET (window->check_minabs), 0, 4, 1, 1);
02858
02859
        gtk_grid_attach (window->grid_variable,
02860
                          GTK_WIDGET (window->scrolled_minabs), 1, 4, 3, 1);
        gtk_grid_attach (window->grid_variable,
02861
02862
                          GTK_WIDGET (window->check_maxabs), 0, 5, 1, 1);
02863
        gtk_grid_attach (window->grid_variable,
02864
                          GTK_WIDGET (window->scrolled_maxabs), 1, 5, 3, 1);
02865
        gtk grid attach (window->grid variable,
```

```
02866
                          GTK_WIDGET (window->label_precision), 0, 6, 1, 1);
02867
        gtk_grid_attach (window->grid_variable,
02868
                          GTK_WIDGET (window->spin_precision), 1, 6, 3, 1);
        gtk_grid_attach (window->grid_variable,
02869
02870
                          GTK WIDGET (window->label sweeps), 0, 7, 1, 1);
02871
        gtk grid attach (window->grid variable,
                          GTK_WIDGET (window->spin_sweeps), 1, 7, 3, 1);
        gtk_grid_attach (window->grid_variable,
02873
02874
                          GTK_WIDGET (window->label_bits), 0, 8, 1, 1);
02875
        gtk_grid_attach (window->grid_variable,
                          GTK_WIDGET (window->spin_bits), 1, 8, 3, 1);
02876
02877
        gtk_grid_attach (window->grid_variable,
02878
                          GTK_WIDGET (window->label_step), 0, 9, 1, 1);
02879
        gtk_grid_attach (window->grid_variable,
02880
                          GTK_WIDGET (window->scrolled_step), 1, 9, 3, 1);
02881
        window->frame_variable = (GtkFrame *) gtk_frame_new (_("Variable"));
        gtk_frame_set_child (window->frame_variable,
02882
                              GTK WIDGET (window->grid variable));
02883
02884
02885
        // Creating the experiment widgets
        window->combo_experiment = (GtkComboBoxText *) gtk_combo_box_text_new ();
02886
02887
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->combo_experiment),
02888
                                      _("Experiment selector"));
        window->id_experiment = g_signal_connect
  (window->combo_experiment, "changed", window_set_experiment, NULL);
02889
02890
02891 #if !GTK4
        window->button\_add\_experiment = (GtkButton *)
02892
02893
          gtk_button_new_from_icon_name ("list-add", GTK_ICON_SIZE_BUTTON);
02894 #else
02895
        window->button_add_experiment = (GtkButton *)
02896
          gtk_button_new_from_icon_name ("list-add");
02897 #endif
02898
02899
          (window->button_add_experiment, "clicked", window_add_experiment, NULL);
02900
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_add_experiment),
                                      _("Add experiment"));
02901
02902 #if !GTK4
02903
        window->button_remove_experiment = (GtkButton *)
02904
          gtk_button_new_from_icon_name ("list-remove", GTK_ICON_SIZE_BUTTON);
02905 #else
02906
        window->button_remove_experiment = (GtkButton *)
          gtk_button_new_from_icon_name ("list-remove");
02907
02908 #endif
02909
        q_signal_connect (window->button_remove_experiment, "clicked",
02910
                           window_remove_experiment, NULL);
02911
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_remove_experiment),
02912
                                      _("Remove experiment"));
02913
        window->label_experiment
02914
          = (GtkLabel *) qtk_label_new (_("Experimental data file"));
02915
        window->button experiment = (GtkButton *)
          gtk_button_new_with_mnemonic (_("Experimental data file"));
02916
02917
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_experiment),
02918
                                      _("Experimental data file"));
02919
        g_signal_connect (window->button_experiment, "clicked",
        window_name_experiment, NULL);
gtk_widget_set_hexpand (GTK_WIDGET (window->button_experiment), TRUE);
02920
02921
02922
        window->label_weight = (GtkLabel *) gtk_label_new (_("Weight"));
02923
        window->spin weight
02924
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02925
        {\tt gtk\_widget\_set\_tooltip\_text}
          (GTK_WIDGET (window->spin_weight),
02926
02927
           _("Weight factor to build the objective function"));
02928
        q_signal_connect
02929
          (window->spin_weight, "value-changed", window_weight_experiment, NULL);
        window->grid_experiment = (GtkGrid *) gtk_grid_new ();
02930
02931
        gtk_grid_attach (window->grid_experiment,
02932
                          GTK_WIDGET (window->combo_experiment), 0, 0, 2, 1);
02933
        gtk_grid_attach (window->grid_experiment,
02934
                          GTK_WIDGET (window->button_add_experiment), 2, 0, 1, 1);
        gtk_grid_attach (window->grid_experiment,
02936
                          GTK_WIDGET (window->button_remove_experiment), 3, 0, 1, 1);
02937
        gtk_grid_attach (window->grid_experiment,
02938
                          GTK_WIDGET (window->label_experiment), 0, 1, 1, 1);
02939
        gtk_grid_attach (window->grid_experiment,
02940
                          GTK_WIDGET (window->button_experiment), 1, 1, 3, 1);
02941
        gtk_grid_attach (window->grid_experiment,
02942
                          GTK_WIDGET (window->label_weight), 0, 2, 1, 1);
02943
        gtk_grid_attach (window->grid_experiment,
                          GTK_WIDGET (window->spin_weight), 1, 2, 3, 1);
02944
02945
        for (i = 0; i < MAX NINPUTS; ++i)</pre>
02946
            snprintf (buffer3, 64, "%s %u", _("Input template"), i + 1);
window->check_template[i] = (GtkCheckButton *)
02947
02948
02949
              gtk_check_button_new_with_label (buffer3);
02950
            window->id_template[i]
              = g_signal_connect (window->check_template[i], "toggled",
02951
02952
                                   window inputs experiment, NULL);
```

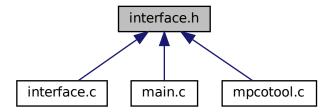
```
gtk_grid_attach (window->grid_experiment,
            GTK_WIDGET (window->check_template[i]), 0, 3 + i, 1, 1); window->button_template[i] = (GtkButton *)
02954
02955
              gtk_button_new_with_mnemonic (_("Input template"));
02956
02957
            gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_template[i]),
02958
02959
                                          _("Experimental input template file"));
02960
            window->id_input[i] =
02961
              g_signal_connect_swapped (window->button_template[i], "clicked",
02962
                                          (GCallback) window_template_experiment,
                                         (void *) (size_t) i);
02963
            gtk_grid_attach (window->grid_experiment,
02964
02965
                              GTK_WIDGET (window->button_template[i]), 1, 3 + i, 3, 1);
02966
02967
        window->frame_experiment = (GtkFrame *) gtk_frame_new (_("Experiment"));
02968
        gtk_frame_set_child (window->frame_experiment,
02969
                              GTK_WIDGET (window->grid_experiment));
02970
02971
        // Creating the error norm widgets
02972
        window->frame_norm = (GtkFrame *) gtk_frame_new (_("Error norm"));
02973
        window->grid_norm = (GtkGrid *) gtk_grid_new ();
02974
        gtk_frame_set_child (window->frame_norm, GTK_WIDGET (window->grid_norm));
02975 #if !GTK4
        window->button_norm[0] = (GtkRadioButton *)
02976
02977
          gtk_radio_button_new_with_mnemonic (NULL, label_norm[0]);
02978 #else
02979
        window->button_norm[0] = (GtkCheckButton *)
02980
          gtk_check_button_new_with_mnemonic (label_norm[0]);
02981 #endif
02982
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_norm[0]),
02983
                                      tip norm[0]);
02984
        gtk_grid_attach (window->grid_norm,
02985
                         GTK_WIDGET (window->button_norm[0]), 0, 0, 1, 1);
02986
        g_signal_connect (window->button_norm[0], "toggled", window_update, NULL);
02987
       for (i = 0; ++i < NNORMS;)</pre>
02988
02989 #if !GTK4
02990
            window->button_norm[i] = (GtkRadioButton *)
02991
              gtk_radio_button_new_with_mnemonic
02992
               (gtk_radio_button_get_group (window->button_norm[0]), label_norm[i]);
02993 #else
02994
            window->button norm[i] = (GtkCheckButton *)
              gtk_check_button_new_with_mnemonic (label_norm[i]);
02995
02996
            gtk_check_button_set_group (window->button_norm[i]
02997
                                         window->button_norm[0]);
02998 #endif
02999
            gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_norm[i]),
0.3000
                                           tip_norm[i]);
            gtk_grid_attach (window->grid_norm,
03001
                              GTK_WIDGET (window->button_norm[i]), 0, i, 1, 1);
03002
03003
            g_signal_connect (window->button_norm[i], "toggled", window_update, NULL);
03004
03005
        window->label_p = (GtkLabel *) gtk_label_new (_("P parameter"));
03006
        gtk_grid_attach (window->grid_norm, GTK_WIDGET (window->label_p), 1, 1, 1, 1);
        window->spin_p = (GtkSpinButton *)
03007
03008
          gtk_spin_button_new_with_range (-G_MAXDOUBLE, G_MAXDOUBLE, 0.01);
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->spin_p),
03009
03010
                                       _("P parameter for the P error norm"));
03011
        window->scrolled_p = (GtkScrolledWindow *)
03012 #if !GTK4
03013
          gtk scrolled window new (NULL, NULL);
03014 #else
03015
          gtk_scrolled_window_new ();
03016 #endif
03017
        gtk_scrolled_window_set_child (window->scrolled_p,
03018
                                        GTK_WIDGET (window->spin_p));
        gtk_widget_set_hexpand (GTK_WIDGET (window->scrolled_p), TRUE);
gtk_widget_set_halign (GTK_WIDGET (window->scrolled_p), GTK_ALIGN_FILL);
03019
03020
03021
        gtk_grid_attach (window->grid_norm, GTK_WIDGET (window->scrolled_p),
03022
                          1, 2, 1, 2);
03023
03024
        \ensuremath{//} Creating the grid and attaching the widgets to the grid
03025
        window->grid = (GtkGrid *) gtk_grid_new ();
        {\tt gtk\_grid\_attach~(window->grid,~GTK\_WIDGET~(window->box\_buttons),~0,~0,~3,~1);}
03026
        gtk_grid_attach (window->grid, GTK_WIDGET (window->grid_files), 0, 1, 1, 1);
03027
03028
        gtk_grid_attach (window->grid,
03029
                          GTK_WIDGET (window->frame_algorithm), 0, 2, 1, 1);
03030
        gtk_grid_attach (window->grid,
03031
                          GTK_WIDGET (window->frame_variable), 1, 2, 1, 1);
03032
        gtk grid attach (window->grid,
                         GTK_WIDGET (window->frame_experiment), 2, 2, 1, 1);
03033
        gtk_grid_attach (window->grid, GTK_WIDGET (window->frame_norm), 1, 1, 2, 1);
03034
        gtk_window_set_child (window->window, GTK_WIDGET (window->grid));
03035
03036
03037
        // Setting the window logo
03038
        window->logo = gdk_pixbuf_new_from_xpm_data (logo);
03039 #if !GTK4
```

```
gtk_window_set_icon (window->window, window->logo);
03041 #endif
03042
03043
         // Showing the window
03044 #if !GTK4
03045
         gtk_widget_show_all (GTK_WIDGET (window->window));
03046 #else
03047
         gtk_widget_show (GTK_WIDGET (window->window));
03048 #endif
03049
03050
         // In GTK+ 3.16 and 3.18 the default scrolled size is wrong
03051 #if GTK_MINOR_VERSION >= 16
        gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_min), -1, 40);
03052
03053
         gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_max), -1, 40);
03054
         gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_minabs), -1, 40);
         gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_maxabs), -1, 40);
gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_step), -1, 40);
03055
03056
        gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_p), -1, 40);
gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_threshold), -1, 40);
03057
03058
03059 #endif
03060
03061
         // Reading initial example
        input_new ();
buffer2 = g_get_current_dir ();
buffer = g_build_filename (buffer2, "..", "tests", "test1", INPUT_FILE, NULL);
03062
03063
03064
03065
        g_free (buffer2);
03066
        window_read (buffer);
        g_free (buffer);
03067
03068
03069 #if DEBUG INTERFACE
03070 fprintf (stderr, "window_new: start\n");
03071 #endif
03072 }
```

5.13 interface.h File Reference

Header file to define the graphical interface functions.

This graph shows which files directly or indirectly include this file:



Data Structures

struct Options

Struct to define the options dialog.

struct Running

Struct to define the running dialog.

struct Window

Struct to define the main window.

Macros

#define MAX_LENGTH (DEFAULT_PRECISION + 8)
 Max length of texts allowed in GtkSpinButtons.

Functions

void window_new (GtkApplication *application)

Variables

Window window [1]

Window struct to define the main interface window.

5.13.1 Detailed Description

Header file to define the graphical interface functions.

Authors

Javier Burguete.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file interface.h.

5.13.2 Macro Definition Documentation

5.13.2.1 MAX_LENGTH

```
#define MAX_LENGTH (DEFAULT_PRECISION + 8)
```

Max length of texts allowed in GtkSpinButtons.

Definition at line 42 of file interface.h.

5.13.3 Function Documentation

5.13.3.1 window_new()

Function to open the main window.

Parameters

application | GtkApplication struct.

Definition at line 2274 of file interface.c.

```
02276
        unsigned int i;
02277
        char *buffer, *buffer2, buffer3[64];
        const char *label_algorithm[NALGORITHMS] = {
   "_Monte-Carlo", _("_Sweep"), _("_Genetic"), _("_Orthogonal")
02278
02279
02280
        const char *tip_algorithm[NALGORITHMS] = {
         _("Monte-Carlo brute force algorithm"),
02282
          _("Sweep brute force algorithm"),
02283
02284
          _("Genetic algorithm"),
          \_("Orthogonal sampling brute force algorithm"),
02285
02286
02287
        const char *label_climbing[NCLIMBINGS] = {
02288
          _("_Coordinates climbing"), _("_Random climbing")
02289
02290
        const char *tip_climbing[NCLIMBINGS] = {
02291
         _("Coordinates climbing estimate method"),
          _("Random climbing estimate method")
02292
02293
        };
02294
        const char *label_norm[NNORMS] = { "L2", "L", "Lp", "L1" };
02295
        const char *tip_norm[NNORMS] =
         _("Euclidean error norm (L2)"),
02296
         _("Maximum error norm (L)"),
_("P error norm (Lp)"),
02297
02298
          _("Taxicab error norm (L1)")
02299
02300
02301 #if !GTK4
02302
        const char *close = "delete-event";
02303 #else
       const char *close = "close-request";
02304
02305 #endif
02306
02307 #if DEBUG_INTERFACE
02308
        fprintf (stderr, "window_new: start\n");
02309 #endif
02310
02311
        // Creating the window
02312
        window->window = window_parent = main_window
02313
          = (GtkWindow *) gtk_application_window_new (application);
02314
02315
        // Finish when closing the window
02316
        g_signal_connect_swapped (window->window, close,
02317
                                    G_CALLBACK (g_application_quit),
                                    G_APPLICATION (application));
02318
02319
02320
        // Setting the window title
02321
        gtk_window_set_title (window->window, "MPCOTool");
02322
02323
        \ensuremath{//} Creating the open button
        window->button_open = (GtkButton *)
02324
02325 #if !GTK4
02326
          gtk_button_new_from_icon_name ("document-open", GTK_ICON_SIZE_BUTTON);
02327 #else
02328
          gtk_button_new_from_icon_name ("document-open");
02329 #endi:
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_open),
02330
02331
                                        _("Open a case"));
        g_signal_connect (window->button_open, "clicked", window_open, NULL);
02332
02333
02334
        // Creating the save button
02335
        window->button_save = (GtkButton *)
02336 #if !GTK4
02337
          gtk button new from icon name ("document-save", GTK ICON SIZE BUTTON);
02338 #else
02339
          gtk_button_new_from_icon_name ("document-save");
02340 #endi:
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_save),
02341
        _("Save the case"));
g_signal_connect (window->button_save, "clicked", (GCallback) window_save,
02342
02343
02344
                           NULL);
02345
02346
        // Creating the run button
02347
        window->button_run = (GtkButton *)
02348 #if !GTK4
02349
          gtk button new from icon name ("system-run", GTK ICON SIZE BUTTON);
02350 #else
          gtk_button_new_from_icon_name ("system-run");
02352 #endif
```

```
gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_run),
02354
                                           _("Run the optimization"));
02355
         g_signal_connect (window->button_run, "clicked", window_run, NULL);
02356
02357
         // Creating the options button
02358
         window->button_options = (GtkButton *)
02359 #if !GTK4
02360
           qtk_button_new_from_icon_name ("preferences-system", GTK_ICON_SIZE_BUTTON);
02361 #else
02362
           gtk_button_new_from_icon_name ("preferences-system");
02363 #endif
         gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_options),
02364
02365
                                           ("Edit the case"));
02366
         g_signal_connect (window->button_options, "clicked", options_new, NULL);
02367
         \ensuremath{//} Creating the help button
02368
02369
         window->button_help = (GtkButton *)
02370 #if !GTK4
02371
           gtk_button_new_from_icon_name ("help-browser", GTK_ICON_SIZE_BUTTON);
02372 #else
02373
           gtk button new from icon name ("help-browser");
02374 #endif
         gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_help), _("Help"));
g_signal_connect (window->button_help, "clicked", window_help, NULL);
02375
02376
02377
02378
         // Creating the about button
02379
         window->button_about = (GtkButton *)
02380 #if !GTK4
02381
           gtk_button_new_from_icon_name ("help-about", GTK_ICON_SIZE_BUTTON);
02382 #else
02383
           gtk_button_new_from_icon_name ("help-about");
02384 #endif
02385
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_about), _("About"));
02386
         g_signal_connect (window->button_about, "clicked", window_about, NULL);
02387
02388
         // Creating the exit button
02389
        window->button_exit = (GtkButton *)
02390 #if !GTK4
02391
           gtk_button_new_from_icon_name ("application-exit", GTK_ICON_SIZE_BUTTON);
02392 #else
02393
           gtk_button_new_from_icon_name ("application-exit");
02394 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_exit), _("Exit"));
02395
         g_signal_connect_swapped (window->button_exit, "clicked",
02396
                                       G_CALLBACK (g_application_quit),
02397
                                       G_APPLICATION (application));
02398
02399
02400
         \ensuremath{//} Creating the buttons bar
         window->box_buttons = (GtkBox *) gtk_box_new (GTK_ORIENTATION_HORIZONTAL, 0);
02401
         gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_open));
gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_save));
02402
02403
02404
         gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_run));
02405
         gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_options));
         gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_help));
gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_about));
02406
02407
02408
         gtk box append (window->box buttons, GTK WIDGET (window->button exit));
02409
02410
         // Creating the simulator program label and entry
02411
         window->label_simulator = (GtkLabel *) gtk_label_new (_("Simulator program"));
         window->button_simulator = (GtkButton *)
02412
         gtk_button_new_with_mnemonic (_("Simulator program"));
gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_simulator),
02413
02414
02415
                                           _("Simulator program executable file"));
02416
         gtk_widget_set_hexpand (GTK_WIDGET (window->button_simulator), TRUE);
02417
         g_signal_connect (window->button_simulator, "clicked",
02418
                             G_CALLBACK (dialog_simulator), NULL);
02419
02420
        // Creating the evaluator program label and entry
window->check_evaluator = (GtkCheckButton *)
02421
02422
           gtk_check_button_new_with_mnemonic (_("_Evaluator program"));
         g_signal_connect (window->check_evaluator, "toggled", window_update, NULL);
window->button_evaluator = (GtkButton *)
02423
02424
02425
           gtk_button_new_with_mnemonic (_("Evaluator program"));
02426
         gtk_widget_set_tooltip_text
  (GTK_WIDGET (window->button_evaluator),
02427
02428
             _("Optional evaluator program executable file"));
         g_signal_connect (window->button_evaluator, "clicked"
02429
02430
                             G_CALLBACK (dialog_evaluator), NULL);
02431
02432
         // Creating the results files labels and entries
         window->label_result = (GtkLabel *) gtk_label_new (_("Result file"));
window->entry_result = (GtkEntry *) gtk_entry_new ();
02433
02434
         gtk_widget_set_tooltip_text
02435
02436
           (GTK_WIDGET (window->entry_result), _("Best results file"));
         window->label_variables = (GtkLabel *) gtk_label_new (_("Variables file"));
window->entry_variables = (GtkEntry *) gtk_entry_new ();
02437
02438
02439
         atk widget set tooltip text
```

```
02440
          (GTK_WIDGET (window->entry_variables), _("All simulated results file"));
02441
02442
        // Creating the files grid and attaching widgets
02443
        window->grid_files = (GtkGrid *) gtk_grid_new ();
        {\tt gtk\_grid\_attach~(window->grid\_files,~GTK\_WIDGET~(window->label\_simulator),}
02444
02445
                          0, 0, 1, 1);
02446
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_simulator),
02447
                          1, 0, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->check_evaluator),
02448
02449
                          0, 1, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_evaluator),
02450
02451
                          1, 1, 1, 1);
02452
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_result),
02453
                          0, 2, 1, 1);
02454
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->entry_result),
02455
                          1, 2, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_variables),
02456
                          0, 3, 1, 1);
02457
02458
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->entry_variables),
02459
                          1, 3, 1, 1);
02460
02461
        // Creating the algorithm properties
02462
        window->label_simulations = (GtkLabel *) gtk_label_new
          (_("Simulations number"));
02463
02464
        window->spin_simulations
02465
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02466
        {\tt gtk\_widget\_set\_tooltip\_text}
02467
          (GTK_WIDGET (window->spin_simulations),
02468
            ("Number of simulations to perform for each iteration"));
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_simulations), TRUE);
02469
02470
        window->label iterations = (GtkLabel *)
02471
          gtk_label_new (_("Iterations number"));
02472
        window->spin_iterations
02473
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e6, 1.);
02474
        {\tt gtk\_widget\_set\_tooltip\_text}
02475
          (GTK_WIDGET (window->spin_iterations), _("Number of iterations"));
02476
        g_signal_connect
02477
          (window->spin_iterations, "value-changed", window_update, NULL);
02478
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_iterations), TRUE);
02479
        window->label_tolerance = (GtkLabel *) gtk_label_new (_("Tolerance"));
02480
        window->spin_tolerance =
02481
          (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
        gtk_widget_set_tooltip_text
02482
02483
          (GTK_WIDGET (window->spin_tolerance),
           _("Tolerance to set the variable interval on the next iteration"));
02484
02485
        window->label_bests = (GtkLabel *) gtk_label_new (_("Bests number"));
02486
        window->spin_bests
02487
          = (GtkSpinButton \star) gtk_spin_button_new_with_range (1., 1.e6, 1.);
        gtk_widget_set_tooltip_text
  (GTK_WIDGET (window->spin_bests),
02488
02489
           _("Number of best simulations used to set the variable interval "
02490
02491
             "on the next iteration"));
02492
        window->label_population
02493
          = (GtkLabel *) gtk_label_new (_("Population number"));
        window->spin_population
02494
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02495
        gtk_widget_set_tooltip_text
02496
          (GTK_WIDGET (window->spin_population),
02497
02498
           _("Number of population for the genetic algorithm"));
02499
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_population), TRUE);
02500
        window->label_generations
02501
          = (GtkLabel *) gtk_label_new (_("Generations number"));
02502
        window->spin_generations
02503
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e6, 1.);
        gtk_widget_set_tooltip_text
02504
02505
          (GTK_WIDGET (window->spin_generations),
02506
            _("Number of generations for the genetic algorithm"));
        window->label_mutation = (GtkLabel *) gtk_label_new (_("Mutation ratio"));
02507
02508
        window->spin_mutation
02509
           = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02510
        gtk_widget_set_tooltip_text
02511
          (GTK_WIDGET (window->spin_mutation),
02512
            _("Ratio of mutation for the genetic algorithm"));
02513
        window->label_reproduction
        = (GtkLabel *) gtk_label_new (_("Reproduction ratio"));
window->spin_reproduction
02514
02515
02516
           = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02517
        gtk_widget_set_tooltip_text
02518
          (GTK_WIDGET (window->spin_reproduction),
            _("Ratio of reproduction for the genetic algorithm"));
02519
02520
        window->label_adaptation = (GtkLabel *) gtk_label_new (_("Adaptation ratio"));
02521
        window->spin_adaptation
02522
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02523
        gtk_widget_set_tooltip_text
02524
          (GTK_WIDGET (window->spin_adaptation),
        _("Ratio of adaptation for the genetic algorithm"));
window->label_threshold = (GtkLabel *) gtk_label_new (_("Threshold"));
02525
02526
```

```
window->spin_threshold = (GtkSpinButton *)
02528
          gtk_spin_button_new_with_range (-G_MAXDOUBLE, G_MAXDOUBLE,
02529
                                            precision[DEFAULT_PRECISION]);
02530
        {\tt gtk\_widget\_set\_tooltip\_text}
          (GTK_WIDGET (window->spin_threshold),
02531
            _("Threshold in the objective function to finish the simulations"));
02532
02533
        window->scrolled_threshold = (GtkScrolledWindow *)
02534 #if !GTK4
02535
          gtk_scrolled_window_new (NULL, NULL);
02536 #else
02537
          gtk_scrolled_window_new ();
02538 #endif
02539
        gtk_scrolled_window_set_child (window->scrolled_threshold,
02540
                                         GTK_WIDGET (window->spin_threshold));
02541 //
          gtk_widget_set_hexpand (GTK_WIDGET (window->scrolled_threshold), TRUE);
02542 // gtk_widget_set_halign (GTK_WIDGET (window->scrolled_threshold),
02543 //
                                         GTK ALIGN FILL):
02544
02545
        // Creating the hill climbing method properties
02546
        window->check_climbing = (GtkCheckButton *)
02547
          gtk_check_button_new_with_mnemonic (_("_Hill climbing method"));
02548
        g_signal_connect (window->check_climbing, "toggled", window_update, NULL);
02549
        window->grid_climbing = (GtkGrid *) gtk_grid_new ();
02550 #if !GTK4
02551
        window->button_climbing[0] = (GtkRadioButton *)
02552
          gtk_radio_button_new_with_mnemonic (NULL, label_climbing[0]);
02553 #else
02554
       window->button_climbing[0] = (GtkCheckButton *)
02555
          gtk_check_button_new_with_mnemonic (label_climbing[0]);
02556 #endif
02557
        gtk_grid_attach (window->grid_climbing,
02558
                          GTK_WIDGET (window->button_climbing[0]), 0, 0, 1, 1);
02559
        g_signal_connect (window->button_climbing[0], "toggled", window_update, NULL);
02560
        for (i = 0; ++i < NCLIMBINGS;)</pre>
02561
02562 #if !GTK4
            window->button climbing[i] = (GtkRadioButton *)
02563
02564
              gtk_radio_button_new_with_mnemonic
02565
               (gtk_radio_button_get_group (window->button_climbing[0]),
02566
                label_climbing[i]);
02567 #else
02568
            window->button climbing[i] = (GtkCheckButton *)
              atk check button new_with_mnemonic (label_climbing[i]);
02569
02570
            gtk_check_button_set_group (window->button_climbing[i],
02571
                                          window->button_climbing[0]);
02572 #endif
02573
            gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_climbing[i]),
02574
                                           tip_climbing[i]);
            gtk_grid_attach (window->grid_climbing,
02575
                              GTK_WIDGET (window->button_climbing[i]), 0, i, 1, 1);
02576
            g_signal_connect (window->button_climbing[i], "toggled", window_update,
02578
                                NULL);
02579
        window->label_steps = (GtkLabel *) gtk_label_new (_("Steps number"));
window->spin_steps = (GtkSpinButton *)
02580
02581
        gtk_spin_button_new_with_range (1., 1.e12, 1.);
gtk_widget_set_hexpand (GTK_WIDGET (window->spin_steps), TRUE);
02582
02583
02584
        window->label_estimates
02585
          = (GtkLabel *) gtk_label_new (_("Climbing estimates number"));
02586
        window->spin_estimates = (GtkSpinButton *)
          gtk_spin_button_new_with_range (1., 1.e3, 1.);
02587
02588
        window->label_relaxation
02589
          = (GtkLabel *) gtk_label_new (_("Relaxation parameter"));
        window->spin_relaxation = (GtkSpinButton *)
02590
02591
          gtk_spin_button_new_with_range (0., 2., 0.001);
02592
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->label_steps),
02593
        0, NCLIMBINGS, 1, 1);
gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_steps),
02594
02595
                          1, NCLIMBINGS, 1, 1);
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->label_estimates),
02596
02597
                          0, NCLIMBINGS + 1, 1, 1);
02598
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_estimates),
        1, NCLIMBINGS + 1, 1, 1);
gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->label_relaxation),
02599
02600
        0, NCLIMBINGS + 2, 1, 1);
gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_relaxation),
1, NCLIMBINGS + 2, 1, 1);
02601
02602
02603
02604
02605
        // Creating the array of algorithms
        window->grid algorithm = (GtkGrid *) gtk grid new ();
02606
02607 #if !GTK4
02608
        window->button_algorithm[0] = (GtkRadioButton *)
02609
          gtk_radio_button_new_with_mnemonic (NULL, label_algorithm[0]);
02610 #else
02611
        window->button\_algorithm[0] = (GtkCheckButton *)
02612
          gtk_check_button_new_with_mnemonic (label_algorithm[0]);
02613 #endif
```

```
gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_algorithm[0]),
                                      tip_algorithm[0]);
02615
02616
        gtk_grid_attach (window->grid_algorithm,
        GTK_WIDGET (window->button_algorithm[0]), 0, 0, 1, 1);
g_signal_connect (window->button_algorithm[0], "toggled",
02617
02618
02619
                           window set algorithm, NULL);
02620
        for (i = 0; ++i < NALGORITHMS;)</pre>
02621
02622 #if !GTK4
02623
            window->button_algorithm[i] = (GtkRadioButton *)
02624
              {\tt gtk\_radio\_button\_new\_with\_mnemonic}
02625
              (\texttt{gtk\_radio\_button\_get\_group} \ (\texttt{window->button\_algorithm[0]}) \, ,
02626
               label algorithm[i]);
02627 #else
02628
            window->button_algorithm[i] = (GtkCheckButton *)
02629
              gtk_check_button_new_with_mnemonic (label_algorithm[i]);
02630
            gtk_check_button_set_group (window->button_algorithm[i]
02631
                                          window->button_algorithm[0]),
02632 #endif
              gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_algorithm[i]),
                                             tip_algorithm[i]);
02634
02635
            gtk_grid_attach (window->grid_algorithm,
            GTK_WIDGET (window->button_algorithm[i]), 0, i, 1, 1); g_signal_connect (window->button_algorithm[i], "toggled",
02636
02637
                               window_set_algorithm, NULL);
02638
02639
        gtk_grid_attach (window->grid_algorithm,
02640
02641
                          GTK_WIDGET (window->label_simulations),
02642
                          0, NALGORITHMS, 1, 1);
02643
        gtk_grid_attach (window->grid_algorithm,
02644
                         GTK WIDGET (window->spin simulations), 1, NALGORITHMS, 1, 1);
02645
        gtk_grid_attach (window->grid_algorithm,
02646
                          GTK_WIDGET (window->label_iterations),
                          0, NALGORITHMS + 1, 1, 1);
02647
02648
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_iterations),
       1, NALGORITHMS + 1, 1, 1);
gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_tolerance),
02649
02650
                          0, NALGORITHMS + 2, 1, 1);
02651
02652
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_tolerance),
02653
                          1, NALGORITHMS + 2, 1, 1);
02654
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_bests),
                          0, NALGORITHMS + 3, 1, 1);
02655
       02656
02657
        gtk_grid_attach (window->grid_algorithm,
02658
02659
                          GTK_WIDGET (window->label_population),
02660
                          0, NALGORITHMS + 4, 1, 1);
        02661
02662
        gtk_grid_attach (window->grid_algorithm,
02663
02664
                          GTK_WIDGET (window->label_generations),
                          0, NALGORITHMS + 5, 1, 1);
02665
02666
        gtk_grid_attach (window->grid_algorithm,
02667
                          GTK_WIDGET (window->spin_generations),
       1, NALGORITHMS + 5, 1, 1);
gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_mutation),
02668
02669
                          0, NALGORITHMS + 6, 1, 1);
02670
02671
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_mutation),
02672
                          1, NALGORITHMS + 6, 1, 1);
02673
        gtk_grid_attach (window->grid_algorithm,
                         GTK_WIDGET (window->label_reproduction),
0, NALGORITHMS + 7, 1, 1);
02674
02675
02676
        gtk_grid_attach (window->grid_algorithm,
                          GTK_WIDGET (window->spin_reproduction),
02677
02678
                          1, NALGORITHMS + 7, 1, 1);
02679
        gtk_grid_attach (window->grid_algorithm,
02680
                          GTK_WIDGET (window->label_adaptation),
                          0, NALGORITHMS + 8, 1, 1);
02681
02682
        qtk_qrid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_adaptation),
                          1, NALGORITHMS + 8, 1, 1);
02684
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->check_climbing),
02685
                          0, NALGORITHMS + 9, 2, 1);
02686
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->grid_climbing),
       0, NALGORITHMS + 10, 2, 1);
gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_threshold),
02687
02688
                          0, NALGORITHMS + 11, 1, 1);
02689
02690
        gtk_grid_attach (window->grid_algorithm,
02691
                          GTK_WIDGET (window->scrolled_threshold),
02692
                         1, NALGORITHMS + 11, 1, 1);
        window->frame_algorithm = (GtkFrame *) gtk_frame_new (_("Algorithm"));
gtk_frame_set_child (window->frame_algorithm,
02693
02694
02695
                              GTK_WIDGET (window->grid_algorithm));
02696
02697
        // Creating the variable widgets
02698
        window->combo_variable = (GtkComboBoxText *) gtk_combo_box_text_new ();
02699
        gtk_widget_set_tooltip_text
02700
          (GTK WIDGET (window->combo variable), ("Variables selector"));
```

```
window->id_variable = q_signal_connect
           (window->combo_variable, "changed", window_set_variable, NULL);
02702
02703 #if !GTK4
02704
        window->button_add_variable = (GtkButton *)
          gtk_button_new_from_icon_name ("list-add", GTK_ICON_SIZE_BUTTON);
02705
02706 #else
02707
        window->button_add_variable = (GtkButton *)
02708
           gtk_button_new_from_icon_name ("list-add");
02709 #endif
02710
        g_signal_connect (window->button_add_variable, "clicked", window_add_variable,
02711
                             NULL);
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_add_variable),
02712
02713
                                         _("Add variable"));
02714 #if !GTK4
02715
        window->button_remove_variable = (GtkButton *)
02716
           gtk_button_new_from_icon_name ("list-remove", GTK_ICON_SIZE_BUTTON);
02717 #else
02718
        window->button_remove_variable = (GtkButton *)
gtk_button_new_from_icon_name ("list-remove");
02719
02720 #endif
02721
        g_signal_connect (window->button_remove_variable, "clicked",
02722
                             window_remove_variable, NULL);
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_remove_variable),
02723
                                          _("Remove variable"));
02724
        window->label_variable = (GtkLabel *) gtk_label_new (_("Name"));
window->entry_variable = (GtkEntry *) gtk_entry_new ();
02725
02726
02727
        gtk_widget_set_tooltip_text
02728
           (GTK_WIDGET (window->entry_variable), _("Variable name"));
02729
         gtk_widget_set_hexpand (GTK_WIDGET (window->entry_variable), TRUE);
        window->id_variable_label = g_signal_connect
  (window->entry_variable, "changed", window_label_variable, NULL);
02730
02731
02732
         window->label_min = (GtkLabel *) gtk_label_new (_("Minimum"));
02733
         window->spin_min = (GtkSpinButton *) gtk_spin_button_new_with_range
02734
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02735
         {\tt gtk\_widget\_set\_tooltip\_text}
02736
           (GTK_WIDGET (window->spin_min),
                                                ("Minimum initial value of the variable"));
         window->scrolled_min = (GtkScrolledWindow *)
02737
02738 #if !GTK4
02739
           gtk_scrolled_window_new (NULL, NULL);
02740 #else
02741
          gtk_scrolled_window_new ();
02742 #endif
        gtk_scrolled_window_set_child (window->scrolled_min,
02743
02744
                                           GTK_WIDGET (window->spin_min));
02745
        g_signal_connect (window->spin_min, "value-changed",
02746
                             window_rangemin_variable, NULL);
02747
        window->label_max = (GtkLabel *) gtk_label_new (_("Maximum"));
        window->spin_max = (GtkSpinButton *) gtk_spin_button_new_with_range
   (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02748
02749
02750
         gtk_widget_set_tooltip_text
02751
           (GTK_WIDGET (window->spin_max), _("Maximum initial value of the variable"));
02752
        window->scrolled_max = (GtkScrolledWindow *)
02753 #if !GTK4
02754
           gtk_scrolled_window_new (NULL, NULL);
02755 #else
02756
          gtk_scrolled_window_new ();
02757 #endif
02758
        gtk_scrolled_window_set_child (window->scrolled_max,
02759
                                           GTK_WIDGET (window->spin_max));
02760
        g_signal_connect (window->spin_max, "value-changed",
                             window_rangemax_variable, NULL);
02761
02762
        window->check minabs = (GtkCheckButton *)
        gtk_check_button_new_with_mnemonic (_("_Absolute minimum"));
g_signal_connect (window->check_minabs, "toggled", window_update, NULL);
02763
02764
02765
         window->spin_minabs = (GtkSpinButton *) gtk_spin_button_new_with_range
02766
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02767
         {\tt gtk\_widget\_set\_tooltip\_text}
02768
           (GTK WIDGET (window->spin minabs).
            _("Minimum allowed value of the variable"));
02769
         window->scrolled_minabs = (GtkScrolledWindow *)
02771 #if !GTK4
02772
           gtk_scrolled_window_new (NULL, NULL);
02773 #else
02774
          gtk_scrolled_window_new ();
02775 #endif
02776
        gtk_scrolled_window_set_child (window->scrolled_minabs,
02777
                                           GTK_WIDGET (window->spin_minabs));
02778
         g_signal_connect (window->spin_minabs, "value-changed",
02779
                             window_rangeminabs_variable, NULL);
02780
        window->check maxabs = (GtkCheckButton *)
        gtk_check_button_new_with_mnemonic (_("_Absolute maximum"));
g_signal_connect (window->check_maxabs, "toggled", window_update, NULL);
window->spin_maxabs = (GtkSpinButton *) gtk_spin_button_new_with_range
02781
02782
02783
02784
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02785
         gtk_widget_set_tooltip_text
02786
           (GTK_WIDGET (window->spin_maxabs),
            _("Maximum allowed value of the variable"));
02787
```

```
window->scrolled_maxabs = (GtkScrolledWindow *)
02789 #if !GTK4
02790
          gtk_scrolled_window_new (NULL, NULL);
02791 #else
02792
         gtk_scrolled_window_new ();
02793 #endif
02794
        gtk_scrolled_window_set_child (window->scrolled_maxabs,
02795
                                        GTK_WIDGET (window->spin_maxabs));
02796
        g_signal_connect (window->spin_maxabs, "value-changed",
02797
                          window_rangemaxabs_variable, NULL);
        window->label_precision = (GtkLabel *) gtk_label_new (_("Precision digits"));
window->spin_precision = (GtkSpinButton *)
02798
02799
02800
          gtk_spin_button_new_with_range (0., (gdouble) DEFAULT_PRECISION, 1.);
02801
        gtk_widget_set_tooltip_text
02802
          (GTK_WIDGET (window->spin_precision),
02803
           _("Number of precision floating point digits \n"
             "0 is for integer numbers"));
02804
        02805
02806
02807
        window->label_sweeps = (GtkLabel *) gtk_label_new (_("Sweeps number"));
        window->spin_sweeps =
02808
02809
          (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02810
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->spin_sweeps),
        _("Number of steps sweeping the variable"));
g_signal_connect (window->spin_sweeps, "value-changed",
02811
02812
                          window_update_variable, NULL);
02813
        window->label_bits = (GtkLabel *) gtk_label_new (_("Bits number"));
02814
02815
        window->spin_bits
02816
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 64., 1.);
02817
        gtk_widget_set_tooltip_text
  (GTK_WIDGET (window->spin_bits),
02818
02819
            _("Number of bits to encode the variable"));
02820
        q_signal_connect
02821
          (window->spin_bits, "value-changed", window_update_variable, NULL);
        window->label_step = (GtkLabel *) gtk_label_new (_("Step size"));
window->spin_step = (GtkSpinButton *) gtk_spin_button_new_with_range
02822
02823
          (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02824
02825
        gtk_widget_set_tooltip_text
02826
          (GTK_WIDGET (window->spin_step),
02827
           _("Initial step size for the hill climbing method"));
02828
        window->scrolled_step = (GtkScrolledWindow *)
02829 #if !GTK4
          gtk scrolled window new (NULL, NULL);
02830
02831 #else
02832
         gtk_scrolled_window_new ();
02833 #endif
02834
        gtk_scrolled_window_set_child (window->scrolled_step,
02835
                                        GTK_WIDGET (window->spin_step));
02836
        g_signal_connect
          (window->spin_step, "value-changed", window_step_variable, NULL);
02837
        window->grid_variable = (GtkGrid *) gtk_grid_new ();
02838
02839
        gtk_grid_attach (window->grid_variable,
02840
                          GTK_WIDGET (window->combo_variable), 0, 0, 2, 1);
02841
        gtk_grid_attach (window->grid_variable,
02842
                         GTK_WIDGET (window->button_add_variable), 2, 0, 1, 1);
02843
        gtk grid attach (window->grid variable,
                         GTK_WIDGET (window->button_remove_variable), 3, 0, 1, 1);
02844
        gtk_grid_attach (window->grid_variable,
02845
02846
                          GTK_WIDGET (window->label_variable), 0, 1, 1, 1);
02847
        gtk_grid_attach (window->grid_variable,
                         GTK WIDGET (window->entry_variable), 1, 1, 3, 1);
02848
02849
        gtk_grid_attach (window->grid_variable,
02850
                          GTK_WIDGET (window->label_min), 0, 2, 1, 1);
        gtk_grid_attach (window->grid_variable,
02851
02852
                          GTK_WIDGET (window->scrolled_min), 1, 2, 3, 1);
02853
        gtk_grid_attach (window->grid_variable,
02854
                          GTK_WIDGET (window->label_max), 0, 3, 1, 1);
02855
        gtk grid attach (window->grid variable,
02856
                         GTK_WIDGET (window->scrolled_max), 1, 3, 3, 1);
        gtk_grid_attach (window->grid_variable,
02858
                          GTK_WIDGET (window->check_minabs), 0, 4, 1, 1);
02859
        gtk_grid_attach (window->grid_variable,
02860
                          GTK_WIDGET (window->scrolled_minabs), 1, 4, 3, 1);
02861
        gtk_grid_attach (window->grid_variable,
                         GTK_WIDGET (window->check_maxabs), 0, 5, 1, 1);
02862
02863
        gtk_grid_attach (window->grid_variable,
02864
                          GTK_WIDGET (window->scrolled_maxabs), 1, 5, 3, 1);
02865
        gtk_grid_attach (window->grid_variable,
02866
                          GTK_WIDGET (window->label_precision), 0, 6, 1, 1);
02867
        gtk grid attach (window->grid variable,
                         GTK_WIDGET (window->spin_precision), 1, 6, 3, 1);
02868
02869
        gtk_grid_attach (window->grid_variable,
02870
                          GTK_WIDGET (window->label_sweeps), 0, 7, 1, 1);
02871
        gtk_grid_attach (window->grid_variable,
02872
                         GTK_WIDGET (window->spin_sweeps), 1, 7, 3, 1);
02873
        gtk_grid_attach (window->grid_variable,
02874
                          GTK_WIDGET (window->label_bits), 0, 8, 1, 1);
```

```
gtk_grid_attach (window->grid_variable,
                          GTK_WIDGET (window->spin_bits), 1, 8, 3, 1);
02876
02877
        gtk_grid_attach (window->grid_variable,
02878
                          GTK_WIDGET (window->label_step), 0, 9, 1, 1);
02879
        02880
        window->frame_variable = (GtkFrame *) gtk_frame_new (_("Variable"));
02881
        gtk_frame_set_child (window->frame_variable,
02882
02883
                              GTK_WIDGET (window->grid_variable));
02884
02885
        // Creating the experiment widgets
        window->combo_experiment = (GtkComboBoxText *) gtk_combo_box_text_new ();
02886
02887
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->combo_experiment),
                                       _("Experiment selector"));
02888
02889
        window->id_experiment = g_signal_connect
02890
          (window->combo_experiment, "changed", window_set_experiment, NULL);
02891 #if !GTK4
       window->button_add_experiment = (GtkButton *)
  gtk_button_new_from_icon_name ("list-add", GTK_ICON_SIZE_BUTTON);
02892
02893
02894 #else
02895
       window->button_add_experiment = (GtkButton *)
02896
          gtk_button_new_from_icon_name ("list-add");
02897 #endif
02898
        g_signal_connect
           (window->button_add_experiment, "clicked", window_add_experiment, NULL);
02899
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_add_experiment),
02900
02901
                                       _("Add experiment"));
02902 #if !GTK4
02903
        window->button_remove_experiment = (GtkButton *)
          gtk_button_new_from_icon_name ("list-remove", GTK_ICON_SIZE_BUTTON);
02904
02905 #else
02906
       window->button_remove_experiment = (GtkButton *)
gtk_button_new_from_icon_name ("list-remove");
02907
02908 #endif
02909
        g_signal_connect (window->button_remove_experiment, "clicked",
02910
                           window_remove_experiment, NULL);
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_remove_experiment),
02911
                                      _("Remove experiment"));
02912
02913
        window->label_experiment
02914
          = (GtkLabel *) gtk_label_new (_("Experimental data file"));
02915
        window->button_experiment = (GtkButton *)
          {\tt gtk\_button\_new\_with\_mnemonic\ (\_("Experimental\ data\ file"));}
02916
        {\tt gtk\_widget\_set\_tooltip\_text} \ \ ({\tt GTK\_WIDGET} \ \ ({\tt window->button\_experiment}) \ ,
02917
02918
                                       _("Experimental data file"));
        g_signal_connect (window->button_experiment, "clicked",
02919
02920
                           window_name_experiment, NULL);
02921
        gtk_widget_set_hexpand (GTK_WIDGET (window->button_experiment), TRUE);
02922
        window->label_weight = (GtkLabel *) gtk_label_new (_("Weight"));
02923
        window->spin weight
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02924
02925
        gtk_widget_set_tooltip_text
02926
          (GTK_WIDGET (window->spin_weight),
02927
            _("Weight factor to build the objective function"));
02928
        g_signal_connect
          (window->spin_weight, "value-changed", window_weight_experiment, NULL);
02929
        window->grid_experiment = (GtkGrid *) gtk_grid_new ();
02930
        gtk_grid_attach (window->grid_experiment,
02931
                          GTK_WIDGET (window->combo_experiment), 0, 0, 2, 1);
02932
02933
        gtk_grid_attach (window->grid_experiment,
02934
                          GTK_WIDGET (window->button_add_experiment), 2, 0, 1, 1);
02935
        gtk_grid_attach (window->grid_experiment,
02936
                          GTK_WIDGET (window->button_remove_experiment), 3, 0, 1, 1);
02937
        gtk_grid_attach (window->grid_experiment,
02938
                          GTK_WIDGET (window->label_experiment), 0, 1, 1, 1);
02939
        gtk_grid_attach (window->grid_experiment,
02940
                          GTK_WIDGET (window->button_experiment), 1, 1, 3, 1);
02941
        gtk_grid_attach (window->grid_experiment,
02942
                          GTK WIDGET (window->label weight), 0, 2, 1, 1);
        gtk_grid_attach (window->grid_experiment,
02943
                          GTK_WIDGET (window->spin_weight), 1, 2, 3, 1);
02945
        for (i = 0; i < MAX_NINPUTS; ++i)</pre>
02946
            snprintf (buffer3, 64, "%s %u", _("Input template"), i + 1);
window->check_template[i] = (GtkCheckButton *)
02947
02948
02949
              gtk check button new with label (buffer3);
02950
            window->id_template[i]
02951
              = g_signal_connect (window->check_template[i], "toggled",
02952
                                   window_inputs_experiment, NULL);
02953
            gtk_grid_attach (window->grid_experiment,
            GTK_WIDGET (window->check_template[i]), 0, 3 + i, 1, 1);
window->button_template[i] = (GtkButton *)
02954
02955
02956
              gtk_button_new_with_mnemonic (_("Input template"));
02957
02958
            gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_template[i]),
02959
                                          _("Experimental input template file"));
            window->id_input[i] =
02960
02961
              g signal connect swapped (window->button template[i], "clicked",
```

```
(GCallback) window_template_experiment,
02963
                                          (void *) (size_t) i);
02964
            gtk_grid_attach (window->grid_experiment,
02965
                              GTK_WIDGET (window->button_template[i]), 1, 3 + i, 3, 1);
02966
02967
        window->frame experiment = (GtkFrame *) gtk frame new ( ("Experiment"));
        gtk_frame_set_child (window->frame_experiment,
02968
                              GTK_WIDGET (window->grid_experiment));
02969
02970
02971
        // Creating the error norm widgets
        window->frame_norm = (GtkFrame *) gtk_frame_new (_("Error norm"));
window->grid_norm = (GtkGrid *) gtk_grid_new ();
02972
02973
02974
        gtk_frame_set_child (window->frame_norm, GTK_WIDGET (window->grid_norm));
02975 #if !GTK4
02976
        window->button_norm[0] = (GtkRadioButton *)
02977
          gtk_radio_button_new_with_mnemonic (NULL, label_norm[0]);
02978 #else
02979
       window->button norm[0] = (GtkCheckButton *)
          gtk_check_button_new_with_mnemonic (label_norm[0]);
02980
02981 #endif
02982
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_norm[0]),
02983
                                       tip_norm[0]);
        gtk_grid_attach (window->grid_norm,
02984
        GTK_WIDGET (window->button_norm[0]), 0, 0, 1, 1);
g_signal_connect (window->button_norm[0], "toggled", window_update, NULL);
02985
02986
        for (i = 0; ++i < NNORMS;)
02987
02988
02989 #if !GTK4
02990
            window->button_norm[i] = (GtkRadioButton *)
02991
              gtk_radio_button_new_with_mnemonic
02992
               (gtk_radio_button_get_group (window->button_norm[0]), label_norm[i]);
02993 #else
02994
            window->button_norm[i] = (GtkCheckButton *)
02995
              gtk_check_button_new_with_mnemonic (label_norm[i]);
02996
            gtk_check_button_set_group (window->button_norm[i],
02997
                                          window->button_norm[0]);
02998 #endif
            gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_norm[i]),
03000
                                           tip_norm[i]);
03001
            gtk_grid_attach (window->grid_norm,
            GTK_WIDGET (window->button_norm[i]), 0, i, 1, 1);
g_signal_connect (window->button_norm[i], "toggled", window_update, NULL);
03002
03003
03004
03005
        window->label_p = (GtkLabel *) gtk_label_new (_("P parameter"));
        gtk_grid_attach (window->grid_norm, GTK_WIDGET (window->label_p), 1, 1, 1, 1);
03006
03007
        window->spin_p = (GtkSpinButton *)
          gtk_spin_button_new_with_range (-G_MAXDOUBLE, G_MAXDOUBLE, 0.01);
03008
        03009
03010
        window->scrolled_p = (GtkScrolledWindow *)
03011
03012 #if !GTK4
03013
          gtk_scrolled_window_new (NULL, NULL);
03014 #else
03015
          gtk_scrolled_window_new ();
03016 #endif
        gtk_scrolled_window_set_child (window->scrolled_p,
03017
                                         GTK_WIDGET (window->spin_p));
03018
        gtk_widget_set_hexpand (GTK_WIDGET (window->scrolled_p), TRUE);
03019
03020
        gtk_widget_set_halign (GTK_WIDGET (window->scrolled_p), GTK_ALIGN_FILL);
03021
        gtk_grid_attach (window->grid_norm, GTK_WIDGET (window->scrolled_p),
03022
                          1, 2, 1, 2);
03023
03024
        // Creating the grid and attaching the widgets to the grid
        window->grid = (GtkGrid *) gtk_grid_new ();
03025
03026
        gtk_grid_attach (window->grid, GTK_WIDGET (window->box_buttons), 0, 0, 3, 1);
03027
        gtk_grid_attach (window->grid, GTK_WIDGET (window->grid_files), 0, 1, 1, 1);
        gtk_grid_attach (window->grid,
03028
03029
                          GTK WIDGET (window->frame algorithm), 0, 2, 1, 1);
03030
        gtk_grid_attach (window->grid,
03031
                          GTK_WIDGET (window->frame_variable), 1, 2, 1, 1);
03032
        gtk_grid_attach (window->grid,
03033
                          GTK_WIDGET (window->frame_experiment), 2, 2, 1, 1);
        gtk_grid_attach (window->grid, GTK_WIDGET (window->frame_norm), 1, 1, 2, 1);
gtk_window_set_child (window->window, GTK_WIDGET (window->grid));
03034
03035
03036
03037
        // Setting the window logo
03038
        window->logo = gdk_pixbuf_new_from_xpm_data (logo);
03039 #if !GTK4
03040
       gtk_window_set_icon (window->window, window->logo);
03041 #endif
03042
03043
        // Showing the window
03044 #if !GTK4
03045
       gtk_widget_show_all (GTK_WIDGET (window->window));
03046 #else
03047
       gtk_widget_show (GTK_WIDGET (window->window));
03048 #endif
```

5.14 interface.h 237

```
// In GTK+ 3.16 and 3.18 the default scrolled size is wrong
03050
03051 #if GTK_MINOR_VERSION >= 16
03052
        gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_min), -1, 40);
         gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_max), -1, 40);
gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_minabs), -1, 40);
gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_maxabs), -1, 40);
03053
03054
03056
         gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_step), -1, 40);
03057
         gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_p), -1, 40);
03058
         gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_threshold), -1, 40);
03059 #endif
03060
03061
         // Reading initial example
03062
        input_new ();
03063
        buffer2 = g_get_current_dir ();
        buffer = g_build_filename (buffer2, "..", "tests", "test1", INPUT_FILE, NULL);
03064
         g_free (buffer2);
03065
03066
        window read (buffer);
03067
        g_free (buffer);
03068
03069 #if DEBUG_INTERFACE
        fprintf (stderr, "window_new: start\n");
03070
03071 #endif
03072 }
```

5.13.4 Variable Documentation

5.13.4.1 window

```
Window window[1] [extern]
```

Window struct to define the main interface window.

Definition at line 81 of file interface.c.

5.14 interface.h

Go to the documentation of this file.

```
00001 /*
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00009
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are met:
00012
00013 1. Redistributions of source code must retain the above copyright notice,
00014 this list of conditions and the following disclaimer.
00015
00016 2. Redistributions in binary form must reproduce the above copyright notice, 00017 this list of conditions and the following disclaimer in the
00018 documentation and/or other materials provided with the distribution.
00019
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN 00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING 00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
```

```
00031
00038 #ifndef INTERFACE__H
00039 #define INTERFACE__H 1
00040
00041 #define MAX LENGTH (DEFAULT PRECISION + 8)
00043
00048 typedef struct
00049 {
00050
        GtkDialog *dialog;
        GtkGrid *grid;
GtkLabel *label_seed;
00051
00052
00054
        GtkSpinButton *spin_seed;
        GtkLabel *label_threads;
00056
00057
        GtkSpinButton *spin_threads;
00058
        GtkLabel *label_climbing;
00059
        GtkSpinButton *spin_climbing;
00060 } Options;
00061
00066 typedef struct
00067 {
        GtkDialog *dialog;
GtkLabel *label;
00068
00069
00070
        GtkSpinner *spinner;
00071
        GtkGrid *grid;
00072 } Running;
00073
00078 typedef struct
00079 {
08000
        GtkWindow *window;
00081
        GtkGrid *grid;
GtkBox *box_buttons;
00082
00083
        GtkButton *button_open;
00084
        GtkButton *button_save;
00085
        GtkButton *button_run;
00086
        GtkButton *button_options;
00087
        GtkButton *button_help;
00088
        GtkButton *button_about;
00089
        GtkButton *button_exit;
00090
        GtkGrid *grid_files;
00091
        GtkLabel *label_simulator;
00092
        GtkButton *button_simulator;
        GtkCheckButton *check_evaluator;
00093
       GtkButton *button_evaluator;
GtkLabel *label_result;
00094
00095
        GtkEntry *entry_result;
00096
00097
        GtkLabel *label_variables;
00098
        GtkEntry *entry_variables;
00099
        GtkFrame *frame_norm;
00100
        GtkGrid *grid_norm;
00101 #if !GTK4
        GtkRadioButton *button_norm[NNORMS];
00104 #else
00105
       GtkCheckButton *button_norm[NNORMS];
00107 #endif
        GtkLabel *label_p;
00108
00109
        GtkSpinButton *spin_p;
00110
        GtkScrolledWindow *scrolled_p;
00112
        GtkFrame *frame_algorithm;
00113
        GtkGrid *grid_algorithm;
00114 #if !GTK4
00115
        GtkRadioButton *button algorithm[NALGORITHMS];
00117 #else
00118
        GtkCheckButton *button_algorithm[NALGORITHMS];
00120 #endif
00121
        GtkLabel *label_simulations;
00122
        GtkSpinButton *spin_simulations;
00124
        GtkLabel *label iterations;
00125
        GtkSpinButton *spin_iterations;
        GtkLabel *label_tolerance;
00127
00128
        GtkSpinButton *spin_tolerance;
00129
        GtkLabel *label_bests;
00130
        GtkSpinButton *spin_bests;
        GtkLabel *label_population;
00131
00132
        GtkSpinButton *spin_population;
        GtkLabel *label_generations;
00134
00135
        GtkSpinButton *spin_generations;
00137
        GtkLabel *label_mutation;
00138
        GtkSpinButton *spin_mutation;
00139
        GtkLabel *label_reproduction;
        GtkSpinButton *spin_reproduction;
00140
        GtkLabel *label_adaptation;
00142
        GtkSpinButton *spin_adaptation;
00145
        GtkCheckButton *check_climbing;
00147
        GtkGrid *grid_climbing;
00149 #if !GTK4
       GtkRadioButton *button_climbing[NCLIMBINGS];
00150
00152 #else
```

5.15 main.c File Reference 239

```
GtkCheckButton *button_climbing[NCLIMBINGS];
00155 #endif
00156
        GtkLabel *label_steps;
00157
        GtkSpinButton *spin_steps;
00158
        GtkLabel *label estimates;
00159
        GtkSpinButton *spin estimates:
        GtkLabel *label_relaxation;
00161
00163
        GtkSpinButton *spin_relaxation;
00165
        GtkLabel *label_threshold;
00166
        GtkSpinButton *spin_threshold;
00167
        GtkScrolledWindow *scrolled_threshold;
00169
        GtkFrame *frame variable:
00170
        GtkGrid *grid_variable;
00171
        GtkComboBoxText *combo_variable;
00173
        GtkButton *button_add_variable;
        GtkButton *button_remove_variable;
GtkLabel *label_variable;
00174
00175
00176
        GtkEntry *entry_variable;
GtkLabel *label_min;
00177
00178
        GtkSpinButton *spin_min;
00179
        GtkScrolledWindow *scrolled_min;
00180
        GtkLabel *label_max;
00181
        GtkSpinButton *spin_max;
00182
        GtkScrolledWindow *scrolled max;
        GtkCheckButton *check_minabs;
GtkSpinButton *spin_minabs;
00183
00184
00185
        GtkScrolledWindow *scrolled_minabs;
00186
        GtkCheckButton *check_maxabs;
00187
        GtkSpinButton *spin_maxabs;
00188
        GtkScrolledWindow *scrolled_maxabs;
        GtkLabel *label_precision;
00189
00190
        GtkSpinButton *spin_precision;
00191
        GtkLabel *label_sweeps;
00192
        GtkSpinButton *spin_sweeps;
00193
        GtkLabel *label_bits;
00194
        GtkSpinButton *spin_bits;
00195
        GtkLabel *label_step;
00196
        GtkSpinButton *spin_step;
00197
        GtkScrolledWindow *scrolled_step;
00198
        GtkFrame *frame_experiment;
00199
        GtkGrid *grid_experiment;
        GtkComboBoxText *combo_experiment;
00200
00201
        GtkButton *button_add_experiment;
00202
        GtkButton *button_remove_experiment;
00203
        GtkLabel *label_experiment;
00204
        GtkButton *button_experiment;
00206
        GtkLabel *label_weight;
00207
        GtkSpinButton *spin_weight;
        GtkCheckButton *check_template[MAX_NINPUTS];
00208
00210
        GtkButton *button_template[MAX_NINPUTS];
00212
        GdkPixbuf *logo;
00213
        Experiment *experiment;
00214
        Variable *variable;
00215
        char *application_directory;
00216
       gulong id_experiment;
gulong id_experiment_name;
00217
00218
        gulong id_variable;
00219
        gulong id_variable_label;
00220
        gulong id_template[MAX_NINPUTS];
00222
        gulong id_input[MAX_NINPUTS];
        unsigned int nexperiments;
00224
00225
        unsigned int nvariables;
00226 } Window;
00228 // Global variables
00229 extern Window window[1];
00230
00231 // Public functions
00232 void window_new (GtkApplication * application);
00234 #endif
```

5.15 main.c File Reference

Main source file.

```
#include "config.h"
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
#include <math.h>
#include <locale.h>
#include <gsl/gsl_rng.h>
#include <libxml/parser.h>
#include <libintl.h>
#include <glib.h>
#include <json-glib/json-glib.h>
#include <mpi.h>
#include <gio/gio.h>
#include <gtk/gtk.h>
#include "jb/src/jb_win.h"
#include "genetic/genetic.h"
#include "tools.h"
#include "experiment.h"
#include "variable.h"
#include "input.h"
#include "optimize.h"
#include "interface.h"
#include "mpcotool.h"
Include dependency graph for main.c:
```



Macros

• #define JBW 2

Functions

• int main (int argn, char **argc)

5.15.1 Detailed Description

Main source file.

Authors

Javier Burguete and Borja Latorre.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file main.c.

5.15.2 Macro Definition Documentation

5.15 main.c File Reference 241

5.15.2.1 JBW

```
#define JBW 2
```

Definition at line 59 of file main.c.

5.15.3 Function Documentation

5.15.3.1 main()

Main function

Returns

0 on succes, error code (>0) on error.

Definition at line 81 of file main.c.

```
O0082 {

00083 #if HAVE_GTK

00084 show_pending = jbw_process_pending;

00085 #endif

00086 jbw_init (&argn, &argc);

00087 return mpcotool (argn, argc);

00088 }
```

Here is the call graph for this function:



5.16 main.c

Go to the documentation of this file.

```
00001 /*
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00009
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are met:
00012
00013 1. Redistributions of source code must retain the above copyright notice,
{\tt 00014} this list of conditions and the following disclaimer.
00015
00016 2. Redistributions in binary form must reproduce the above copyright notice, 00017 this list of conditions and the following disclaimer in the \,
00018 documentation and/or other materials provided with the distribution.
00019
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF 00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING 00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
00031
00038 #define _GNU_SOURCE
00039 #include "config.h"
00040 #include <stdio.h>
00041 #include <stdlib.h>
00042 #include <string.h>
00043 #include <math.h>
00044 #include <locale.h>
00045 #include <gsl/gsl_rng.h>
00046 #include <libxml/parser.h>
00047 #include <libintl.h>
00048 #include <glib.h>
00049 #include <json-glib/json-glib.h>
00050 #ifdef G_OS_WIN32
00051 #include <windows.h>
00052 #endif
00053 #if HAVE_MPI
00054 #include <mpi.h>
00055 #endif
00056 #if HAVE_GTK
00057 #include <gio/gio.h>
00058 #include <gtk/gtk.h>
00059 #define JBW 2
00060 #else
00061 #define JBW 1
00062 #endif
00063 #include "jb/src/jb_win.h"
00063 #include "jb/src/jb_win.h"
00064 #include "genetic/genetic.h"
00065 #include "tools.h"
00066 #include "experiment.h"
00067 #include "variable.h"
00068 #include "input.h"
00069 #include "optimize.h"
00070 #if HAVE_GTK
00071 #include "interface.h"
00072 #endif
00073 #include "mpcotool.h"
00074
00080 int
00081 main (int argn, char **argc)
00082 {
00083 #if HAVE_GTK
        show_pending = jbw_process_pending;
00085 #endif
00086 jbw_init (&argn, &argc);
00087
        return mpcotool (argn, argc);
00088 }
```

5.17 mpcotool.c File Reference

Main function source file.

```
#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include <math.h>
#include <locale.h>
#include <gsl/gsl_rng.h>
#include <libxml/parser.h>
#include <libintl.h>
#include <glib.h>
#include <json-glib/json-glib.h>
#include <mpi.h>
#include <gio/gio.h>
#include <gtk/gtk.h>
#include "jb/src/jb_win.h"
#include "genetic/genetic.h"
#include "tools.h"
#include "experiment.h"
#include "variable.h"
#include "input.h"
#include "optimize.h"
#include "interface.h"
#include "mpcotool.h"
Include dependency graph for mpcotool.c:
```



Macros

#define DEBUG_MPCOTOOL 1
 Macro to debug main functions.

Functions

• int mpcotool (int argn, char **argc)

5.17.1 Detailed Description

Main function source file.

Authors

Javier Burguete and Borja Latorre.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file mpcotool.c.

5.17.2 Macro Definition Documentation

5.17.2.1 DEBUG MPCOTOOL

```
#define DEBUG_MPCOTOOL 1
```

Macro to debug main functions.

Definition at line 73 of file mpcotool.c.

5.17.3 Function Documentation

5.17.3.1 mpcotool()

```
int mpcotool (
          int argn,
          char ** argc )
```

Main function.

Returns

0 on success, >0 on error.

Parameters

argn	Arguments number.
argc	Arguments pointer.

Definition at line 81 of file mpcotool.c.

```
00083 {
00084
        const struct option options[] = {
         ("seed", required_argument, NULL, 's'),
{"nthreads", required_argument, NULL, 't'),
{NULL, 0, NULL, 0}
00085
00086
00087
00088
00089 #if HAVE_GTK
       GtkApplication *application;
00090
00091 #endif
00092
       int o, option_index;
00093
00094 // Starting pseudo-random numbers generator 00095 \#if DEBUG_MPCOTOOL
00096
       00097 #endif
00098
       optimize->rng = gsl_rng_alloc (gsl_rng_taus2);
00099
00100
        // Allowing spaces in the XML data file
00101 #if DEBUG_MPCOTOOL
00102 fprintf (stderr, "mpcotool: allowing spaces in the XML data file\n");
00103 #endif
00104
       xmlKeepBlanksDefault (0);
00105
```

```
00106
        // Starting MPI
00107 #if HAVE_MPI
00108 #if DEBUG_MPCOTOOL
       fprintf (stderr, "mpcotool: starting MPI\n");
00109
00110 #endif
       MPI_Init (&argn, &argc);
00111
        MPI_Comm_size (MPI_COMM_WORLD, &ntasks);
00112
00113
        MPI_Comm_rank (MPI_COMM_WORLD, &optimize->mpi_rank);
00114
        printf ("rank=%d tasks=%d\n", optimize->mpi_rank, ntasks);
00115 #else
00116
       ntasks = 1:
00117 #endif
00118
00119
        // Getting threads number and pseudo-random numbers generator seed
00120
        nthreads_climbing = nthreads = jb_get_ncores ();
        optimize->seed = DEFAULT_RANDOM_SEED;
00121
00122
00123
        // Parsing command line arguments
00124
        while (1)
00125
        {
00126
            o = getopt_long (argn, argc, "s:t:", options, &option_index);
00127
            if (o == -1)
             break;
00128
00129
            switch (o)
00130
              {
              case 's':
00131
               optimize->seed = atol (optarg);
00132
00133
               break;
00134
              case 't':
              nthreads_climbing = nthreads = atoi (optarg);
00135
00136
                break:
00137
              default:
00138
              printf ("%s\n^ss\n", _("ERROR!"), _("Unknown option"));
00139
                return 1;
00140
              }
          }
00141
00142
       argn -= optind;
00143
00144
        // Resetting result and variables file names
00145 #if DEBUG_MPCOTOOL
00146
       fprintf (stderr, "mpcotool: resetting result and variables file names\n");
00147 #endif
00148
       input->result = input->variables = NULL;
00149
00150 #if HAVE_GTK
00151
        // Setting local language and international floating point numbers notation jb_set_locales (PROGRAM_INTERFACE, LOCALE_DIR, "", "C");
00152
00153
00154
00155
       // Initing GTK+
00156
        window->application_directory = g_get_current_dir ();
00157
        gtk_disable_setlocale ();
00158
        application = gtk_application_new ("es.csic.eead.auladei.sprinkler",
00159
                                            G_APPLICATION_DEFAULT_FLAGS);
        g_signal_connect (application, "activate", G_CALLBACK (window_new), NULL);
00160
00161
00162
        // Opening the main window
00163
        g_application_run (G_APPLICATION (application), 0, NULL);
00164
00165
        // Freeing memory
00166
        input free ();
00167
        gtk window destroy (window->window);
00168
        g_object_unref (application);
00169
        g_free (window->application_directory);
00170
00171 #else
00172
        // Checking syntax
if (argn < 1 || argn > 3)
00173
00174
00175
        {
00176
            printf ("The syntax is:\n"
00177
                     "./mpcotoolbin [-nthreads x] [-seed s] data_file [result_file] "
                    "[variables_file]\n");
00178
00179
            return 2;
00180
00181
        if (argn > 1)
00182
          input->result = (char *) xmlStrdup ((xmlChar *) argc[optind + 1]);
00183
        if (argn == 2)
00184
          input->variables = (char *) xmlStrdup ((xmlChar *) argc[optind + 2]);
00185
00186
        // Making optimization
00187 #if DEBUG_MPCOTOOL
00188
       fprintf (stderr, "mpcotool: making optimization\n");
00189 #endif
00190
       if (input_open (argc[optind]))
00191
          optimize_open ();
00192
```

```
// Freeing memory
00194 #if DEBUG_MPCOTOOL
       fprintf (stderr, "mpcotool: freeing memory and closing\n");
00195
00196 #endif
00197
      optimize_free ();
00198
00200
00201
       // Closing MPI
00202 #if HAVE_MP
00203 MPI Finalize ();
00204 #endif
00205
00206
       // Freeing memory
00207
      gsl_rng_free (optimize->rng);
00208
       // Closing
00209
00210
       return 0;
00211 }
```

5.18 mpcotool.c

Go to the documentation of this file.

```
00001 /*
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00009
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are met:
00012
00013 1. Redistributions of source code must retain the above copyright notice,
00014 this list of conditions and the following disclaimer.
          Redistributions in binary form must reproduce the above copyright notice,
00017 this list of conditions and the following disclaimer in the
00018 documentation and/or other materials provided with the distribution.
00019
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED 00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF 00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN 00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
00031
00038 #define _GNU_SOURCE
00039 #include "config.h"
00040 #include <stdio.h>
00041 #include <stdlib.h>
00042 #include <string.h>
00043 #include <getopt.h>
00044 #include <math.h>
00045 #include <locale.h>
00046 #include <gsl/gsl_rng.h>
00047 #include <libxml/parser.h>
00048 #include <libintl.h>
00049 #include <glib.h>
00050 #include <json-glib/json-glib.h>
00051 #ifdef G_OS_WIN32
00052 #include <windows.h>
00053 #endif
00054 #if HAVE_MPI
00055 #include <mpi.h>
00056 #endif
00057 #if HAVE_GTK
00058 #include <gio/gio.h>
00059 #include <gtk/gtk.h>
00060 #endif
00061 #include "jb/src/jb_win.h"
00062 #include "genetic/genetic.h"
00063 #include "tools.h"
00064 #include "experiment.h"
```

5.18 mpcotool.c 247

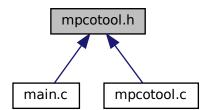
```
00065 #include "variable.h"
00066 #include "input.h"
00067 #include "optimize.h"
00068 #if HAVE_GTK
00069 #include "interface.h"
00070 #endif
00071 #include "mpcotool.h"
00072
00073 #define DEBUG_MPCOTOOL 1
00074
00080 int
00081 mpcotool (int argn,
00082
                 char **argc)
00083 {
00084
        const struct option options[] = {
          {"seed", required_argument, NULL, 's'},
{"nthreads", required_argument, NULL, 't'},
{NULL, 0, NULL, 0}
00085
00086
00087
00088
00089 #if HAVE_GTK
00090
        GtkApplication *application;
00091 #endif
00092
        int o, option_index;
00093
00094
         // Starting pseudo-random numbers generator
00095 #if DEBUG_MPCC
00096
        fprintf (stderr, "mpcotool: starting pseudo-random numbers generator\n");
00097 #endif
00098
        optimize->rng = gsl_rng_alloc (gsl_rng_taus2);
00099
00100
         // Allowing spaces in the XML data file
00101 #if DEBUG_MPCOTOOL
00102
        fprintf (stderr, "mpcotool: allowing spaces in the XML data file\n");
00103 #endif
00104
        xmlKeepBlanksDefault (0);
00105
        // Starting MPI
00106
00107 #if HAVE_MPI
00108 #if DEBUG_MPCOTOOL
00109
        fprintf (stderr, "mpcotool: starting MPI\n");
00110 #endif
00111
        MPI_Init (&argn, &argc);
        MPI_Comm_size (MPI_COMM_WORLD, &ntasks);
MPI_Comm_rank (MPI_COMM_WORLD, &optimize->mpi_rank);
00112
00113
        printf ("rank=%d tasks=%d\n", optimize->mpi_rank, ntasks);
00114
00115 #else
00116
        ntasks = 1;
00117 #endif
00118
        // Getting threads number and pseudo-random numbers generator seed
nthreads_climbing = nthreads = jb_get_ncores ();
00119
00120
00121
        optimize->seed = DEFAULT_RANDOM_SEED;
00122
00123
        // Parsing command line arguments
00124
        while (1)
00125
          {
00126
            o = getopt_long (argn, argc, "s:t:", options, &option_index);
00127
             if (0 == -1)
00128
              break;
00129
             switch (o)
00130
               {
               case 's':
00131
00132
                optimize->seed = atol (optarg);
00133
                 break;
00134
               case 't':
00135
                nthreads_climbing = nthreads = atoi (optarg);
00136
                 break;
00137
               default:
00138
                printf ("%s\n%s\n", _("ERROR!"), _("Unknown option"));
00139
                 return 1;
00140
00141
00142
        argn -= optind;
00143
         // Resetting result and variables file names
00144
00145 #if DEBUG_MPCOTOOL
00146
        fprintf (stderr, "mpcotool: resetting result and variables file names\n");
00147 #endif
00148
        input->result = input->variables = NULL;
00149
00150 #if HAVE_GTK
00151
        // Setting local language and international floating point numbers notation
jb_set_locales (PROGRAM_INTERFACE, LOCALE_DIR, "", "C");
00152
00153
00154
00155
        // Initing GTK+
00156
        window->application directory = g get current dir ();
```

```
gtk_disable_setlocale ();
00158
        application = gtk_application_new ("es.csic.eead.auladei.sprinkler",
00159
                                            G_APPLICATION_DEFAULT_FLAGS);
00160
        g_signal_connect (application, "activate", G_CALLBACK (window_new), NULL);
00161
00162
        // Opening the main window
00163
        g_application_run (G_APPLICATION (application), 0, NULL);
00164
00165
        // Freeing memory
00166
        input_free ();
        gtk_window_destroy (window->window);
00167
00168
        g_object_unref (application);
        g_free (window->application_directory);
00169
00170
00171 #else
00172
00173
        // Checking syntax
00174
        if (argn < 1 || argn > 3)
00176
            printf ("The syntax is:\n"
00177
                     "./mpcotoolbin [-nthreads x] [-seed s] data_file [result_file] "
                    "[variables_file]\n");
00178
00179
           return 2;
00180
00181
        if (argn > 1)
00182
         input->result = (char *) xmlStrdup ((xmlChar *) argc[optind + 1]);
00183
        if (argn == 2)
00184
         input->variables = (char *) xmlStrdup ((xmlChar *) argc[optind + 2]);
00185
       // Making optimization
00186
00187 #if DEBUG_MPCOTOOL
00188
       fprintf (stderr, "mpcotool: making optimization\n");
00189 #endif
00190 if (input_open (argc[optind]))
00191
         optimize_open ();
00192
00193 // Freeing memory
00194 #if DEBUG_MPCOTOOL
00195
       fprintf (stderr, "mpcotool: freeing memory and closing\n");
00196 #endif
00197
       optimize_free ();
00198
00199 #endif
00200
        // Closing MPI
00202 #if HAVE_MP
00203
       MPI_Finalize ();
00204 #endif
00205
00206
       // Freeing memory
       gsl_rng_free (optimize->rng);
00208
00209
        // Closing
00210
       return 0;
00211 }
```

5.19 mpcotool.h File Reference

Main function header file.

This graph shows which files directly or indirectly include this file:



Functions

• int mpcotool (int argn, char **argc)

5.19.1 Detailed Description

Main function header file.

Authors

Javier Burguete and Borja Latorre.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file mpcotool.h.

5.19.2 Function Documentation

5.19.2.1 mpcotool()

```
int mpcotool (
          int argn,
          char ** argc )
```

Main function.

Returns

0 on success, >0 on error.

Parameters

argn	Arguments number.
argc	Arguments pointer.

Definition at line 81 of file mpcotool.c.

```
// Starting pseudo-random numbers generator
00095 #if DEBUG_MPCOTOOL
       fprintf (stderr, "mpcotool: starting pseudo-random numbers generator\n");
00096
00097 #endif
00098
       optimize->rng = gsl_rng_alloc (gsl_rng_taus2);
00099
00100
        // Allowing spaces in the XML data file
00101 #if DEBUG_MPCOTOOL
00102
       fprintf (stderr, "mpcotool: allowing spaces in the XML data file\n");
00103 #endif
00104
       xmlKeepBlanksDefault (0);
00105
00106
        // Starting MPI
00107 #if HAVE_MPI
00108 #if DEBUG_MPCOTOOL
00109
       fprintf (stderr, "mpcotool: starting MPI\n");
00110 #endif
00111
00112
       MPI Init (&argn, &argc);
        MPI_Comm_size (MPI_COMM_WORLD, &ntasks);
        MPI_Comm_rank (MPI_COMM_WORLD, &optimize->mpi_rank);
00113
00114
        printf ("rank=%d tasks=%d\n", optimize->mpi_rank, ntasks);
00115 #else
00116
       ntasks = 1;
00117 #endif
00118
00119
        // Getting threads number and pseudo-random numbers generator seed
       nthreads_climbing = nthreads = jb_get_ncores ();
optimize->seed = DEFAULT_RANDOM_SEED;
00120
00121
00122
00123
        // Parsing command line arguments
00124
       while (1)
00125
         {
00126
            o = getopt_long (argn, argc, "s:t:", options, &option_index);
00127
            if (o == -1)
00128
              break;
00129
            switch (o)
00130
              {
              case 's':
00131
              optimize->seed = atol (optarg);
break;
00132
00133
00134
              case 't':
               nthreads_climbing = nthreads = atoi (optarg);
00135
00136
                break:
00137
              default:
00138
               printf ("%s\n%s\n", _("ERROR!"), _("Unknown option"));
                return 1;
00139
00140
              }
00141
          }
00142
       argn -= optind;
00143
00144
        // Resetting result and variables file names
00145 #if DEBUG_MPCOTOOL
00146
       fprintf (stderr, "mpcotool: resetting result and variables file names\n");
00147 #endif
00148
       input->result = input->variables = NULL;
00149
00150 #if HAVE_GTK
00151
        // Setting local language and international floating point numbers notation jb_set_locales (PROGRAM_INTERFACE, LOCALE_DIR, "", "C");
00152
00153
00154
00155
        // Initing GTK+
00156
        window->application_directory = g_get_current_dir ();
00157
        gtk_disable_setlocale ();
00158
        application = gtk_application_new ("es.csic.eead.auladei.sprinkler",
00159
                                             G_APPLICATION_DEFAULT_FLAGS);
00160
        g_signal_connect (application, "activate", G_CALLBACK (window_new), NULL);
00161
00162
        // Opening the main window
        g_application_run (G_APPLICATION (application), 0, NULL);
00163
00164
00165
        // Freeing memory
00166
        input_free ();
        gtk window destroy (window->window);
00167
00168
        g object unref (application);
00169
        g_free (window->application_directory);
00170
00171 #else
00172
00173
        // Checking syntax
if (argn < 1 || argn > 3)
00174
00175
         {
            printf ("The syntax is:\n"
00176
00177
                     "./mpcotoolbin [-nthreads x] [-seed s] data_file [result_file] "
                    "[variables_file]\n");
00178
00179
            return 2;
00180
```

5.20 mpcotool.h 251

```
if (argn > 1)
         input->result = (char *) xmlStrdup ((xmlChar *) argc[optind + 1]);
00183
       if (argn == 2)
00184
         input->variables = (char *) xmlStrdup ((xmlChar *) argc[optind + 2]);
00185
        // Making optimization
00186
00187 #if DEBUG_MPCOTOOL
00188
       fprintf (stderr, "mpcotool: making optimization\n");
00189 #endif
00190
       if (input_open (argc[optind]))
00191
         optimize_open ();
00192
00193
        // Freeing memory
00194 #if DEBUG_MPCOTOOL
00195
       fprintf (stderr, "mpcotool: freeing memory and closing\n");
00196 #endif
00197
       optimize_free ();
00198
00199 #endif
00200
00201
        // Closing MPI
00202 #if HAVE_MPI
00203
       MPI_Finalize ();
00204 #endif
00205
00206
       // Freeing memory
       gsl_rng_free (optimize->rng);
00207
00208
       // Closing
00209
00210
       return 0;
00211 }
```

5.20 mpcotool.h

```
Go to the documentation of this file.
```

```
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
{\tt 00004} calibrations or optimizations of empirical parameters.
00005
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00009
00010 Redistribution and use in source and binary forms, with or without modification,
\tt 00011 are permitted provided that the following conditions are met:
00013 1. Redistributions of source code must retain the above copyright notice,
00014 this list of conditions and the following disclaimer.
00016 2. Redistributions in binary form must reproduce the above copyright notice,
00017 this list of conditions and the following disclaimer in the
00018 documentation and/or other materials provided with the distribution.
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, 00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING 00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
00031
00038 #ifndef MPCOTOOL__H
00039 #define MPCOTOOL__H 1
00040
00041 extern int mpcotool (int argn, char **argc);
00042
00043 #endif
```

5.21 optimize.c File Reference

Source file to define the optimization functions.

```
#include "config.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <sys/param.h>
#include <qsl/qsl_rnq.h>
#include <libxml/parser.h>
#include <libintl.h>
#include <glib.h>
#include <glib/gstdio.h>
#include <json-glib/json-glib.h>
#include <alloca.h>
#include <mpi.h>
#include "jb/src/jb_win.h"
#include "genetic/genetic.h"
#include "tools.h"
#include "experiment.h"
#include "variable.h"
#include "input.h"
#include "optimize.h"
Include dependency graph for optimize.c:
```



Macros

• #define DEBUG_OPTIMIZE 0

Macro to debug optimize functions.

• #define RM "rm"

Macro to define the shell remove command.

Functions

- static void optimize_input (unsigned int simulation, char *input, GMappedFile *stencil)
- · static double optimize_parse (unsigned int simulation, unsigned int experiment)
- static double optimize norm euclidian (unsigned int simulation)
- static double optimize norm maximum (unsigned int simulation)
- static double optimize_norm_p (unsigned int simulation)
- static double optimize_norm_taxicab (unsigned int simulation)
- static void optimize_print ()
- static void optimize_save_variables (unsigned int simulation, double error)
- static void optimize_best (unsigned int simulation, double value)
- static void optimize_sequential ()
- static void * optimize_thread (ParallelData *data)
- static void optimize_merge (unsigned int nsaveds, unsigned int *simulation_best, double *error_best)
- static void optimize_synchronise ()
- static void optimize sweep ()
- static void optimize MonteCarlo ()
- static void optimize_orthogonal ()
- static void optimize_best_climbing (unsigned int simulation, double value)

- static void optimize_climbing_sequential (unsigned int simulation)
- static void * optimize_climbing_thread (ParallelData *data)
- static double optimize estimate climbing random (unsigned int variable, unsigned int estimate)
- static double optimize_estimate_climbing_coordinates (unsigned int variable, unsigned int estimate)
- static void optimize_step_climbing (unsigned int simulation)
- static void optimize_climbing ()
- static double optimize_genetic_objective (Entity *entity)
- static void optimize_genetic ()
- static void optimize_save_old ()
- static void optimize merge old ()
- static void optimize refine ()
- static void optimize_step ()
- static void optimize_iterate ()
- · void optimize_free ()
- void optimize_open ()

Variables

• Optimize optimize [1]

Optimization data.

· unsigned int nthreads_climbing

Number of threads for the hill climbing method.

static void(* optimize algorithm)()

Pointer to the function to perform a optimization algorithm step.

• static double(* optimize_estimate_climbing)(unsigned int variable, unsigned int estimate)

Pointer to the function to estimate the climbing.

static double(* optimize norm)(unsigned int simulation)

Pointer to the error norm function.

5.21.1 Detailed Description

Source file to define the optimization functions.

Authors

Javier Burguete and Borja Latorre.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file optimize.c.

5.21.2 Macro Definition Documentation

5.21.2.1 DEBUG_OPTIMIZE

```
#define DEBUG_OPTIMIZE 0
```

Macro to debug optimize functions.

Definition at line 67 of file optimize.c.

5.21.2.2 RM

```
#define RM "rm"
```

Macro to define the shell remove command.

Definition at line 76 of file optimize.c.

5.21.3 Function Documentation

5.21.3.1 optimize_best()

Function to save the best simulations.

Parameters

simulation	Simulation number.
value	Objective function value.

Definition at line 450 of file optimize.c.

```
00452 {
00453
        unsigned int i, j;
00454
        double e;
00455 #if DEBUG_OPTIMIZE
00456 fprintf (stderr, "optimize_best: start\n");
00457 fprintf (stderr, "optimize_best: nsaveds=%u nbest=%u\n",
00458 optimize->nsaveds, optimize->nbest);
00459 #endif
00460
        if (optimize->nsaveds < optimize->nbest
00461
            || value < optimize->error_best[optimize->nsaveds - 1])
00462
            if (optimize->nsaveds < optimize->nbest)
00463
00464
              ++optimize->nsaveds;
00465
            optimize->error_best[optimize->nsaveds - 1] = value;
00466
            optimize->simulation_best[optimize->nsaveds - 1] = simulation;
00467
             for (i = optimize->nsaveds; --i;)
00468
00469
                 if (optimize->error_best[i] < optimize->error_best[i - 1])
                   00470
00471
00472
                      e = optimize->error_best[i];
```

```
optimize->simulation_best[i] = optimize->simulation_best[i - 1];
                         optimize->error_best[i] = optimize->error_best[i - 1];
optimize->simulation_best[i - 1] = j;
00474
00475
                         optimize->error_best[i - 1] = e;
00476
00477
00478
                    else
00479
                      break;
00480
                 }
00481
00482 #if DEBUG_OPTIMIZE
00483 fprintf (stderr, "optimize_best: end\n");
00484 #endif
00485 }
```

5.21.3.2 optimize best climbing()

Function to save the best simulation in a hill climbing method.

Parameters

simulation	Simulation number.
value	Objective function value.

Definition at line 812 of file optimize.c.

```
00814 {
00815 #if DEBUG_OPTIMIZE
00816 fprintf (stderr, "optimize_best_climbing: start\n"); 00817 fprintf (stderr,
                  "optimize_best_climbing: simulation=%u value=%.14le best=%.14le\n", simulation, value, optimize->error_best[0]);
00818
00819
00820 #endif
00821 if (value < optimize->error_best[0])
00822
             optimize->error_best[0] = value;
00823
             optimize->simulation_best[0] = simulation;
00824
00825 #if DEBUG_OPTIMIZE
00826 fprintf (stderr,
00827
                       "optimize_best_climbing: BEST simulation=%u value=%.14le\n",
00828
                     simulation, value);
00829 #endif
00830
00831 #if DEBUG_OPTIMIZE
00832
        fprintf (stderr, "optimize_best_climbing: end\n");
00833 #endif
00834 }
```

5.21.3.3 optimize_climbing()

```
static void optimize_climbing ( ) [inline], [static]
```

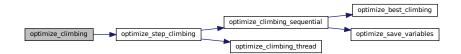
Function to optimize with a hill climbing method.

Definition at line 1040 of file optimize.c.

```
01041 {
01042 unsigned int i, j, k, b, s, adjust;
01043 #if DEBUG_OPTIMIZE
01044 fprintf (stderr, "optimize_climbing: start\n");
01045 #endif
```

```
for (i = 0; i < optimize->nvariables; ++i)
01047
         optimize->climbing[i] = 0.;
01048
       b = optimize->simulation_best[0] * optimize->nvariables;
       s = optimize->nsimulations;
01049
01050
       adjust = 1;
01051
       for (i = 0; i < optimize->nsteps; ++i, s += optimize->nestimates, b = k)
01052
01053 #if DEBUG_OPTIMIZE
01054
           fprintf (stderr, "optimize_climbing: step=%u old_best=%u\n",
01055
                     i, optimize->simulation_best[0]);
01056 #endif
       optimize_step_climbing (s);
01057
01058
            k = optimize->simulation_best[0] * optimize->nvariables;
01059 #if DEBUG_OPTIMIZE
01060
        fprintf (stderr, "optimize_climbing: step=%u best=%u\n",
                    i, optimize->simulation_best[0]);
01061
01062 #endif
01063
           if (k == b)
01064
01065
                if (adjust)
01066
                 for (j = 0; j < optimize->nvariables; ++j)
               optimize->step[j] *= 0.5;
for (j = 0; j < optimize->nvariables; ++j)
01067
01068
                 optimize->climbing[j] = 0.;
01069
01070
                adjust = 1;
01071
01072
            else
01073
             {
01074
               for (j = 0; j < optimize->nvariables; ++j)
01075
01076 #if DEBUG_OPTIMIZE
                    fprintf (stderr,
01078
                             "optimize_climbing: best%u=%.14le old%u=%.14le\n",
01079
                             j, optimize->value[k + j], j, optimize->value[b + j]);
01080 #endif
                    optimize->climbing[j]
01081
01082
                     = (1. - optimize->relaxation) * optimize->climbing[j]
                     + optimize->relaxation
01084
                      * (optimize->value[k + j] - optimize->value[b + j]);
01085 #if DEBUG_OPTIMIZE
                    fprintf (stderr, "optimize_climbing: climbing%u=%.14le\n",
01086
                             j, optimize->climbing[j]);
01087
01088 #endif
01089
01090
               adjust = 0;
01091
             }
01092
01093 #if DEBUG_OPTIMIZE
01094 fprintf (stderr, "optimize_climbing: end\n");
01095 #endif
01096 }
```

Here is the call graph for this function:



5.21.3.4 optimize_climbing_sequential()

Function to estimate the hill climbing sequentially.

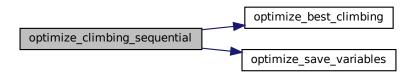
Parameters

simulation | Simulation number.

Definition at line 840 of file optimize.c.

```
00841 {
00842
         double e;
         unsigned int i, j;
00843
00844 #if DEBUG_OPTIMIZE
00845 fprintf (stderr, "optimize_climbing_sequential: start\n");
00846 fprintf (stderr, "optimize_climbing_sequential: nstart_climbing=%u"
00847 "nend_climbing=%u\n",
00848
                   optimize->nstart_climbing, optimize->nend_climbing);
00849 #endif
00850 for (i = optimize->nstart_climbing; i < optimize->nend_climbing; ++i)
00852
             j = simulation + i;
00853
             e = optimize_norm (j);
00854
             optimize_best_climbing (j, e);
             optimize_save_variables (j, e);
00855
             if (e < optimize->threshold)
00856
00857
00858
                  optimize->stop = 1;
00859
                 break;
00860
00861 #if DEBUG_OPTIMIZE
             fprintf (stderr, "optimize_climbing_sequential: i=%u e=%lg\n", i, e);
00862
00863 #endif
00865 #if DEBUG_OPTIMIZE
00866 fprintf (stderr, "optimize_climbing_sequential: end\n"); 00867 #endif
00868 }
```

Here is the call graph for this function:



5.21.3.5 optimize_climbing_thread()

Function to estimate the hill climbing on a thread.

Returns

NULL

Parameters

```
data Function data.
```

Definition at line 876 of file optimize.c.

```
00877
00878
        unsigned int i, thread;
00879
00879 double e;
00880 #if DEBUG_OPTIMIZE
00881
        fprintf (stderr, "optimize_climbing_thread: start\n");
00882 #endif
00883
        thread = data->thread;
00884 #if DEBUG_OPTIMIZE
00885 fprintf (stderr, "optimize_climbing_thread: thread=%u start=%u end=%un",
00886
                  thread,
                  optimize->thread_climbing[thread],
00888
                  optimize->thread_climbing[thread + 1]);
00889 #endif
       for (i = optimize->thread_climbing[thread];
    i < optimize->thread_climbing[thread + 1]; ++i)
00890
00891
00892
            e = optimize_norm (i);
00894
            g_mutex_lock (mutex);
00895
             optimize_best_climbing (i, e);
00896
             optimize_save_variables (i, e);
            if (e < optimize->threshold)
  optimize->stop = 1;
00897
00898
00899
             g_mutex_unlock (mutex);
00900
             if (optimize->stop)
00901
              break;
00902 #if DEBUG_OPTIMIZE 00903 fprintf (stderr, "optimize_climbing_thread: i=%u e=%lg\n", i, e);
00904 #endif
00905
00906 #if DEBUG_OPTIMIZE
00907
        fprintf (stderr, "optimize_climbing_thread: end\n");
00908 #endif
00909 g_thread_exit (NULL);
00910
        return NULL;
00911 }
```

5.21.3.6 optimize estimate climbing coordinates()

Function to estimate a component of the hill climbing vector.

Parameters

variable	Variable number.
estimate	Estimate number.

Definition at line 941 of file optimize.c.

```
00945 {
       double x;
00946
00947 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_estimate_climbing_coordinates: start\n");
00948
00949 #endif
       x = optimize->climbing[variable];
00951
       if (estimate >= (2 * variable) && estimate < (2 * variable + 2))</pre>
00952
        {
00953
           if (estimate & 1)
00954
             x += optimize->step[variable];
           else
00955
00956
             x -= optimize->step[variable];
00957
```

5.21.3.7 optimize_estimate_climbing_random()

Function to estimate a component of the hill climbing vector.

Parameters

variable	Variable number.
estimate	Estimate number.

Definition at line 917 of file optimize.c.

```
00922 {
00923
        double x;
00924 #if DEBUG_OPTIMIZE
00925
        fprintf (stderr, "optimize_estimate_climbing_random: start\n");
00926 #endif
00927
       x = optimize->climbing[variable]
00928 + (1. - 2. * gsl_rng_uniform (optimize->rng)) * optimize->step[variable]; 00929 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_estimate_climbing_random: climbing%u=%lg\n",
        variable, x);
fprintf (stderr, "optimize_estimate_climbing_random: end\n");
00932
00933 #endif
00934
        return x;
00935 }
```

5.21.3.8 optimize_free()

```
void optimize_free ( )
```

Function to free the memory used by the Optimize struct.

Definition at line 1397 of file optimize.c.

```
01398 {
        unsigned int i, j;
01400 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_free: start\n");
01401
01402 #endif
01403
       for (j = 0; j < optimize->ninputs; ++j)
01404
01405
            for (i = 0; i < optimize->nexperiments; ++i)
01406
             g_mapped_file_unref (optimize->file[j][i]);
01407
            g_free (optimize->file[j]);
01408
       g_free (optimize->error_old);
01409
       g_free (optimize->value_old);
01410
01411
       g_free (optimize->value);
        g_free (optimize->genetic_variable);
01413 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_free: end\n");
01414
01415 #endif
01416 }
```

5.21.3.9 optimize_genetic()

```
static void optimize_genetic ( ) [static]
```

Function to optimize with the genetic algorithm.

```
Definition at line 1137 of file optimize.c.
```

```
01138 {
01139
        double *best_variable = NULL;
01140
        char *best_genome = NULL;
01141 double best_objective = 0.;
01142 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_genetic: start\n");
fprintf (stderr, "optimize_genetic: ntasks=%u nthreads=%u\n", ntasks,
01143
01144
01145
                  nthreads);
       fprintf (stderr,
01146
01147
                  "optimize_genetic: nvariables=%u population=%u generations=%u\n",
01148
                 optimize->nvariables, optimize->nsimulations, optimize->niterations);
01149
       fprintf (stderr,
01150
                  "optimize_genetic: mutation=%lg reproduction=%lg adaptation=%lgn",
01151
                  optimize->mutation_ratio, optimize->reproduction_ratio,
01152
                  optimize->adaptation_ratio);
01153 #endif
        genetic_algorithm_default (optimize->nvariables,
01154
01155
                                     optimize->genetic variable.
                                     optimize->nsimulations,
01156
01157
                                     optimize->niterations,
                                     optimize->mutation_ratio,
01158
01159
                                     optimize->reproduction_ratio,
01160
                                     optimize->adaptation_ratio,
01161
                                     optimize->seed,
01162
                                     optimize->threshold,
01163
                                     &optimize_genetic_objective,
01164
                                     &best_genome, &best_variable, &best_objective);
01165 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_genetic: the best\n");
01166
01167 #endif
01168
       optimize->error_old = (double *) g_malloc (sizeof (double));
       optimize->value_old
01170
          = (double *) g_malloc (optimize->nvariables * sizeof (double));
01171
        optimize->error_old[0] = best_objective;
01172
       memcpy (optimize->value_old, best_variable,
                optimize->nvariables * sizeof (double));
01173
01174
       g_free (best_genome);
01175
       g_free (best_variable);
01176
        optimize_print ();
01177 #if DEBUG_OPTIMIZE
01178 fprintf (stderr, "optimize_genetic: end\n");
01179 #endif
01180 }
```

5.21.3.10 optimize genetic objective()

Function to calculate the objective function of an entity.

Returns

objective function value.

Parameters

entity	entity data.
--------	--------------

Definition at line 1104 of file optimize.c.

```
01106
        unsigned int j;
01107
        double objective;
01108    char buffer[64];
01109 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_genetic_objective: start\n");
01110
01111 #endif
01112
        for (j = 0; j < optimize->nvariables; ++j)
01113
             optimize->value[entity->id * optimize->nvariables + j]
01114
               = genetic_get_variable (entity, optimize->genetic_variable + j);
01115
01116
01117
        objective = optimize_norm (entity->id);
01118
        g_mutex_lock (mutex);
01119
        for (j = 0; j < optimize->nvariables; ++j)
01120
             snprintf (buffer, 64, "%s ", format[optimize->precision[j]]);
fprintf (optimize->file_variables, buffer,
01121
01122
                      genetic_get_variable (entity, optimize->genetic_variable + j));
01124
        fprintf (optimize->file_variables, "%.14le\n", objective);
01125
01126
        g_mutex_unlock (mutex);
01127 #if DEBUG OPTIMIZE
01128
        fprintf (stderr, "optimize_genetic_objective: end\n");
01129 #endif
01130
        return objective;
01131 }
```

Here is the call graph for this function:



5.21.3.11 optimize_input()

```
static void optimize_input (
          unsigned int simulation,
          char * input,
          GMappedFile * stencil ) [inline], [static]
```

Function to write the simulation input file.

Parameters

simulation	Simulation number.
input	Input file name.
stencil	Template of the input file name.

Definition at line 95 of file optimize.c.

```
00104
       unsigned int i;
00105
00106 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_input: start\n");
00107
00108 #endif
00109
00110
        // Checking the file
00111
       if (!stencil)
00112
        goto optimize_input_end;
00113
00114
       // Opening stencil
       content = g_mapped_file_get_contents (stencil);
00115
        length = g_mapped_file_get_length (stencil);
00116
00117 #if DEBUG_OPTIMIZE
00118
       fprintf (stderr, "optimize_input: length=%lu\ncontent:\n%s", length, content);
00119 #endif
00120
       file = g_fopen (input, "w");
00121
       // Parsing stencil
00123
       for (i = 0; i < optimize->nvariables; ++i)
00124
00125 #if DEBUG_OPTIMIZE
           fprintf (stderr, "optimize_input: variable=u\n", i);
00126
00127 #endif
00128
           snprintf (buffer, 32, "@variable%u@", i + 1);
           regex = g_regex_new (buffer, (GRegexCompileFlags) 0, (GRegexMatchFlags) 0,
00129
00130
                                 NULL);
00131
            if (i == 0)
00132
               buffer2 = g_regex_replace_literal (regex, content, length, 0,
00133
00134
                                                   optimize->label[i].
00135
                                                   (GRegexMatchFlags) 0, NULL);
00136 #if DEBUG_OPTIMIZE
00137
               fprintf (stderr, "optimize_input: buffer2\n%s", buffer2);
00138 #endif
00139
           else
00140
00142
                length = strlen (buffer3);
00143
               buffer2 = g_regex_replace_literal (regex, buffer3, length, 0,
00144
                                                   optimize->label[i],
                                                   (GRegexMatchFlags) 0, NULL);
             g_free (buffer3);
}
00145
00146
00147
00148
            g_regex_unref (regex);
00149
            length = strlen (buffer2);
00150
           snprintf (buffer, 32, "@value%u@", i + 1);
00151
            regex = g_regex_new (buffer, (GRegexCompileFlags) 0, (GRegexMatchFlags) 0,
                                 NULL);
00152
00153
           snprintf (value, 32, format[optimize->precision[i]],
00154
                     optimize->value[simulation * optimize->nvariables + i]);
00155
00156 #if DEBUG_OPTIMIZE
00157 fprintf (stderr, "optimize_input: value=%s\n", value); 00158 #endif
           buffer3 = g_regex_replace_literal (regex, buffer2, length, 0, value,
00159
                                               (GRegexMatchFlags) 0, NULL);
00161
            g_free (buffer2);
00162
           g_regex_unref (regex);
        }
00163
00164
00165
       // Saving input file
00166
       fwrite (buffer3, strlen (buffer3), sizeof (char), file);
       g_free (buffer3);
fclose (file);
00167
00168
00169
00170 optimize_input_end:
00171 #if DEBUG_OPTIMIZE
00172
       fprintf (stderr, "optimize_input: end\n");
00173 #endif
00174 return;
00175 }
```

5.21.3.12 optimize_iterate()

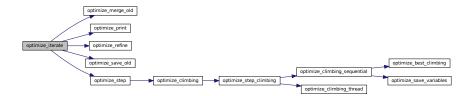
```
static void optimize_iterate ( ) [inline], [static]
```

Function to iterate the algorithm.

Definition at line 1367 of file optimize.c.

```
01369
        unsigned int i;
01370 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_iterate: start\n");
01371
01372 #endif
01373
        optimize->error_old = (double *) g_malloc (optimize->nbest * sizeof (double));
01374
        optimize->value_old =
01375
          (double *) g_malloc (optimize->nbest * optimize->nvariables *
01376
                                 sizeof (double));
01377
        optimize_step ();
01378
        optimize_save_old ();
01379
        optimize_refine ();
01380
        optimize_print ();
01381
        for (i = 1; i < optimize->niterations && !optimize->stop; ++i)
01382
            optimize_step ();
optimize_merge_old ();
optimize_refine ();
01383
01384
01385
01386
            optimize_print ();
01387
01388 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_iterate: end\n");
01389
01390 #endif
01391 }
```

Here is the call graph for this function:



5.21.3.13 optimize merge()

```
static void optimize_merge (
          unsigned int nsaveds,
          unsigned int * simulation_best,
          double * error_best ) [inline], [static]
```

Function to merge the 2 optimization results.

Parameters

nsaveds	Number of saved results.
simulation_best	Array of best simulation numbers.
error_best	Array of best objective function values.

Definition at line 563 of file optimize.c.

```
00568 {
00569    unsigned int i, j, k, s[optimize->nbest];
00570    double e[optimize->nbest];
00571 #if DEBUG_OPTIMIZE
00572    fprintf (stderr, "optimize_merge: start\n");
00573    #endif
00574    i = j = k = 0;
00575    do
00576    {
```

```
if (i == optimize->nsaveds)
00578
00579
                s[k] = simulation_best[j];
00580
                e[k] = error_best[j];
00581
                ++j;
00582
                ++k;
00583
                if (j == nsaveds)
00584
                  break;
00585
00586
            else if (j == nsaveds)
00587
              {
                s[k] = optimize->simulation_best[i];
00588
00589
                e[k] = optimize->error_best[i];
00590
00591
                ++k;
00592
                if (i == optimize->nsaveds)
00593
                  break:
00594
            else if (optimize->error_best[i] > error_best[j])
00596
00597
                s[k] = simulation_best[j];
00598
                e[k] = error_best[j];
00599
                ++ 1;
00600
                ++k;
00601
              }
00602
            else
00603
              {
00604
                s[k] = optimize->simulation_best[i];
                e[k] = optimize->error_best[i];
00605
00606
                ++i;
00607
                ++k;
00608
              }
00609
00610
        while (k < optimize->nbest);
00611
       optimize->nsaveds = k;
       memcpy (optimize->simulation_best, s, k \star sizeof (unsigned int));
00612
00613 memcpy (optimize->error_best, e, k * sizeof (double));
00614 #if DEBUG_OPTIMIZE
00615
       fprintf (stderr, "optimize_merge: end\n");
00616 #endif
00617 }
```

5.21.3.14 optimize_merge_old()

```
static void optimize_merge_old ( ) [inline], [static]
```

Function to merge the best results with the previous step best results on iterative methods.

Definition at line 1218 of file optimize.c.

```
01219 {
       unsigned int i, j, k;
01220
01221
       double v[optimize->nbest * optimize->nvariables], e[optimize->nbest],
01222
        *enew, *eold;
01223 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_merge_old: start\n");
01224
01225 #endif
01226
       enew = optimize->error best;
01227
       eold = optimize->error_old;
       i = j = k = 0;
01228
01229
01230
           if (*enew < *eold)</pre>
01231
01232
01233
              memcpy (v + k * optimize->nvariables,
01234
                      optimize->value
01235
                      + optimize->simulation_best[i] * optimize->nvariables,
01236
                      optimize->nvariables * sizeof (double));
              e[k] = *enew;
01237
01238
              ++k;
01239
              ++enew;
01240
               ++i;
01241
01242
           else
01243
              01244
01245
01246
01247
               e[k] = *eold;
```

```
01248
                 ++k;
01249
                 ++eold;
01250
                 ++j;
              }
01251
01252
        while (k < optimize->nbest);
01253
01254 memcpy (optimize->value_old, v, k * optimize->nvariables * sizeof (double));
01255 memcpy (optimize->error_old, e, k * sizeof (double));
01256 #if DEBUG_OPTIMIZE
01257 fprintf (stderr, "optimize_merge_old: end\n");
01258 #endif
01259 }
```

5.21.3.15 optimize MonteCarlo()

```
static void optimize_MonteCarlo ( ) [static]
```

Function to optimize with the Monte-Carlo algorithm.

Definition at line 721 of file optimize.c.

```
00722
00723
         unsigned int i, j;
         GThread *thread[nthreads];
00725
         ParallelData data[nthreads];
00726 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_MonteCarlo: start\n");
00727
00728 #endif
        for (i = 0; i < optimize->nsimulations; ++i)
  for (j = 0; j < optimize->nvariables; ++j)
    optimize->value[i * optimize->nvariables + j]
00729
00730
00731
                = optimize->rangemin[j] + gsl_rng_uniform (optimize->rng)
* (optimize->rangemax[j] - optimize->rangemin[j]);
00732
00733
00734
         optimize->nsaveds = 0;
00735
         if (nthreads <= 1)
00736
           optimize_sequential ();
00737
         else
00738
00739
              for (i = 0; i < nthreads; ++i)
00740
00741
                  data[i].thread = i;
00742
                  thread[i]
00743
                     = g_thread_new (NULL, (GThreadFunc) optimize_thread, &data[i]);
00744
00745
              for (i = 0; i < nthreads; ++i)</pre>
00746
                g_thread_join (thread[i]);
00747
00748 #if HAVE_MPI
00749 // Communicating tasks results
00750 optimize_synchronise ();
00751 #endif
00752 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_MonteCarlo: end\n");
00753
00754 #endif
00755 }
```

5.21.3.16 optimize_norm_euclidian()

Function to calculate the Euclidian error norm.

Returns

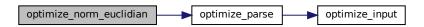
Euclidian error norm.

Parameters

Definition at line 298 of file optimize.c.

```
00299 {
00300 double e, ei;
00301 unsigned int i;
00302 #if DEBUG_OPTIMIZE
00303
         fprintf (stderr, "optimize_norm_euclidian: start\n");
00304 #endif
00307
              ei = optimize_parse (simulation, i);
00308
00309
             e += ei * ei;
00310
00311
        e = sqrt (e);
00312 #if DEBUG_OPTIMIZE
00313 fprintf (stderr, "optimize_norm_euclidian: error=%lg\n", e);
00314 fprintf (stderr, "optimize_norm_euclidian: end\n");
00315 #endif
00316 return e;
00317 }
```

Here is the call graph for this function:



5.21.3.17 optimize_norm_maximum()

```
static double optimize_norm_maximum (
          unsigned int simulation ) [static]
```

Function to calculate the maximum error norm.

Returns

Maximum error norm.

Parameters

```
simulation simulation number.
```

Definition at line 325 of file optimize.c.

```
00326 {
00327    double e, ei;
00328    unsigned int i;
00329 #if DEBUG_OPTIMIZE
00330    fprintf (stderr, "optimize_norm_maximum: start\n");
00331 #endif
00332    e = 0.;
00333    for (i = 0; i < optimize->nexperiments; ++i)
```

```
00334 {
00335          ei = fabs (optimize_parse (simulation, i));
00336          e = fmax (e, ei);
00337    }
00338 #if DEBUG_OPTIMIZE
00339    fprintf (stderr, "optimize_norm_maximum: error=%lg\n", e);
00340    fprintf (stderr, "optimize_norm_maximum: end\n");
00341 #endif
00342    return e;
00343 }
```

Here is the call graph for this function:



5.21.3.18 optimize_norm_p()

```
static double optimize_norm_p (
          unsigned int simulation ) [static]
```

Function to calculate the P error norm.

Returns

P error norm.

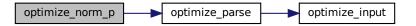
Parameters

simulation simulation number.

Definition at line 351 of file optimize.c.

```
00352 {
00353
         double e, ei;
00354
         unsigned int i;
00355 #if DEBUG_OPTIMIZE
00356 fpri
00357 #endif
        fprintf (stderr, "optimize_norm_p: start\n");
00358
        e = 0.;
00359
         for (i = 0; i < optimize->nexperiments; ++i)
00360
         {
00361
              ei = fabs (optimize_parse (simulation, i));
00362
             e += pow (ei, optimize->p);
00363
00364 e = pow (e, 1. / optimize->p);
00365 #if DEBUG_OPTIMIZE
00366 fprintf (stderr, "optimize_norm_p: error=%lg\n", e);
00367 fprintf (stderr, "optimize_norm_p: end\n");
00368 #endif
00369
        return e;
00370 }
```

Here is the call graph for this function:



5.21.3.19 optimize_norm_taxicab()

```
static double optimize_norm_taxicab (
          unsigned int simulation ) [static]
```

Function to calculate the taxicab error norm.

Returns

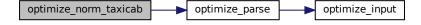
Taxicab error norm.

Parameters

```
simulation simulation number.
```

Definition at line 378 of file optimize.c.

Here is the call graph for this function:



5.21.3.20 optimize_open()

```
void optimize_open ( )
```

Function to open and perform a optimization.

```
Definition at line 1422 of file optimize.c.
```

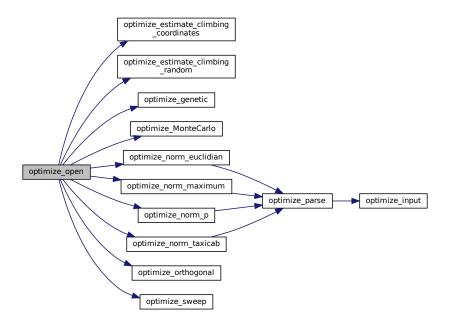
```
01423 {
01424
       GTimeZone *tz;
01425
       GDateTime *t0, *t;
01426
       unsigned int i, j;
01427
01428 #if DEBUG_OPTIMIZE
     char *buffer;
01429
01430
       fprintf (stderr, "optimize_open: start\n");
01431 #endif
01432
01433
        // Getting initial time
01434 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: getting initial time\n");
01435
01436 #endif
01437
       tz = g_time_zone_new_utc ();
01438
      t0 = g_date_time_new_now (tz);
01439
01440
        // Obtaining and initing the pseudo-random numbers generator seed
01441 #if DEBUG_OPTIMIZE
01442
       fprintf (stderr, "optimize_open: getting initial seed\n");
01443 #endif
01444
       if (optimize->seed == DEFAULT_RANDOM_SEED)
       optimize->seed = input->seed;
gsl_rng_set (optimize->rng, optimize->seed);
01445
01446
01447
01448
       // Replacing the working directory
01449 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: replacing the working directory\n");
01450
01451 #endif
       g_chdir (input->directory);
01452
01453
01454
       // Getting results file names
01455
       optimize->result = input->result;
01456
       optimize->variables = input->variables;
01457
01458
       // Obtaining the simulator file
01459
       optimize->simulator = input->simulator;
01460
        // Obtaining the evaluator file
01461
01462
       optimize->evaluator = input->evaluator;
01463
01464
        // Reading the algorithm
       optimize->algorithm = input->algorithm;
01465
01466
       switch (optimize->algorithm)
01467
01468
         case ALGORITHM_MONTE_CARLO:
          optimize_algorithm = optimize_MonteCarlo;
01469
01470
           break:
01471
         case ALGORITHM_SWEEP:
           optimize_algorithm = optimize_sweep;
01473
           break;
01474
         case ALGORITHM_ORTHOGONAL:
01475
          optimize_algorithm = optimize_orthogonal;
01476
           break:
01477
         default:
01478
           optimize_algorithm = optimize_genetic;
01479
            optimize->mutation_ratio = input->mutation_ratio;
01480
            optimize->reproduction_ratio = input->reproduction_ratio;
01481
           optimize->adaptation_ratio = input->adaptation_ratio;
01482
       optimize->nvariables = input->nvariables;
01483
01484
       optimize->nsimulations = input->nsimulations;
       optimize->niterations = input->niterations;
01485
01486
       optimize->nbest = input->nbest;
01487
       optimize->tolerance = input->tolerance;
01488
       optimize->nsteps = input->nsteps;
01489
       optimize->nestimates = 0;
       optimize->threshold = input->threshold;
01490
01491
       optimize->stop = 0;
01492
        if (input->nsteps)
01493
01494
            optimize->relaxation = input->relaxation;
            switch (input->climbing)
01495
01496
01497
              case CLIMBING_METHOD_COORDINATES:
               optimize->nestimates = 2 * optimize->nvariables;
01498
```

```
optimize_estimate_climbing = optimize_estimate_climbing_coordinates;
01500
                break;
01501
              default:
01502
               optimize->nestimates = input->nestimates;
                optimize_estimate_climbing = optimize_estimate_climbing_random;
01503
01504
              }
01505
         }
01506
01507 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: nbest=%u\n", optimize->nbest);
01508
01509 #endif
01510
       optimize->simulation best
          = (unsigned int *) alloca (optimize->nbest * sizeof (unsigned int));
01511
01512
        optimize->error_best = (double *) alloca (optimize->nbest * sizeof (double));
01513
01514
       // Reading the experimental data
01515 #if DEBUG_OPTIMIZE
       buffer = g_get_current_dir ();
01516
        fprintf (stderr, "optimize_open: current directory=%s\n", buffer);
01517
01518
        g_free (buffer);
01519 #endif
01520
       optimize->nexperiments = input->nexperiments;
01521
        optimize->ninputs = input->experiment->ninputs;
01522
        optimize->experiment
01523
          = (char **) alloca (input->nexperiments * sizeof (char *));
        optimize->weight = (double *) alloca (input->nexperiments * sizeof (double));
01524
01525
        for (i = 0; i < input->experiment->ninputs; ++i)
01526
        optimize->file[i] = (GMappedFile **)
01527
            g_malloc (input->nexperiments * sizeof (GMappedFile *));
       for (i = 0; i < input->nexperiments; ++i)
01528
01529
01530 #if DEBUG_OPTIMIZE
01531
            fprintf (stderr, "optimize_open: i=%u\n", i);
01532 #endif
01533
            optimize->experiment[i] = input->experiment[i].name;
            optimize->weight[i] = input->experiment[i].weight;
01534
01535 #if DEBUG_OPTIMIZE
            fprintf (stderr, "optimize_open: experiment=%s weight=%lg\n",
01537
                     optimize->experiment[i], optimize->weight[i]);
01538 #endif
01539
            for (j = 0; j < input->experiment->ninputs; ++j)
01540
01541 #if DEBUG_OPTIMIZE
01542
                fprintf (stderr, "optimize_open: stencil%u\n", j + 1);
01543 #endif
01544
               optimize->file[j][i]
01545
                  = g_mapped_file_new (input->experiment[i].stencil[j], 0, NULL);
01546
             }
         }
01547
01548
        // Reading the variables data
01550 #if DEBUG_OPTIMIZE
01551
       fprintf (stderr, "optimize_open: reading variables\n");
01552 #endif
       optimize->label = (char **) alloca (input->nvariables * sizeof (char *));
01553
        j = input->nvariables * sizeof (double);
01554
        optimize->rangemin = (double *) alloca (j);
        optimize->rangeminabs = (double *) alloca (j);
01556
01557
        optimize->rangemax = (double *) alloca (j);
01558
        optimize->rangemaxabs = (double *) alloca (j);
        optimize->step = (double *) alloca (j);
01559
        j = input->nvariables * sizeof (unsigned int);
01560
        optimize->precision = (unsigned int *) alloca (j);
optimize->nsweeps = (unsigned int *) alloca (j);
01561
01562
01563
        optimize->nbits = (unsigned int *) alloca (j);
01564
        for (i = 0; i < input->nvariables; ++i)
01565
            optimize->label[i] = input->variable[i].name;
01566
01567
            optimize->rangemin[i] = input->variable[i].rangemin;
            optimize->rangeminabs[i] = input->variable[i].rangeminabs;
01569
            optimize->rangemax[i] = input->variable[i].rangemax;
01570
            optimize->rangemaxabs[i] = input->variable[i].rangemaxabs;
            optimize->precision[i] = input->variable[i].precision;
01571
            optimize->step[i] = input->variable[i].step;
optimize->nsweeps[i] = input->variable[i].nsweeps;
01572
01573
01574
            optimize->nbits[i] = input->variable[i].nbits;
01575
01576
        if (input->algorithm == ALGORITHM_SWEEP
            || input->algorithm == ALGORITHM_ORTHOGONAL)
01577
01578
01579
            optimize->nsimulations = 1;
            for (i = 0; i < input->nvariables; ++i)
01581
01582
                optimize->nsimulations *= optimize->nsweeps[i];
01583 #if DEBUG_OPTIMIZE
                fprintf (stderr, "optimize_open: nsweeps=%u nsimulations=%un",
01584
01585
                         optimize->nsweeps[i], optimize->nsimulations);
```

```
01586 #endif
01587
01588
01589
       if (optimize->nsteps)
01590
         optimize->climbing
            = (double *) alloca (optimize->nvariables * sizeof (double));
01591
01593
       // Setting error norm
01594
       switch (input->norm)
01595
01596
         case ERROR NORM EUCLIDIAN:
01597
           optimize_norm = optimize_norm_euclidian;
01598
            break;
01599
         case ERROR_NORM_MAXIMUM:
         optimize_norm = optimize_norm_maximum;
break;
01600
01601
         case ERROR_NORM_P:
01602
          optimize_norm = optimize_norm_p;
optimize->p = input->p;
01603
01604
01605
            break;
01606
          default:
01607
           optimize_norm = optimize_norm_taxicab;
         }
01608
01609
        // Allocating values
01610
01611 #if DEBUG_OPTIMIZE
01612 fprintf (stderr, "optimize_open: allocating variables\n");
01613 fprintf (stderr, "optimize_open: nvariables=%u algorithm=%u\n",
01614
                 optimize->nvariables, optimize->algorithm);
01615 #endif
01616 optimize->genetic variable = NULL:
01617
        if (optimize->algorithm == ALGORITHM_GENETIC)
01618
01619
            optimize->genetic_variable = (GeneticVariable *)
              g_malloc (optimize->nvariables * sizeof (GeneticVariable));
01620
            for (i = 0; i < optimize->nvariables; ++i)
01621
01622
01623 #if DEBUG_OPTIMIZE
01624
         fprintf (stderr, "optimize_open: i=%u min=%lg max=%lg nbits=%u\n",
01625
                         i, optimize->rangemin[i], optimize->rangemax[i],
01626
                          optimize->nbits[i]);
01627 #endif
               optimize->genetic variable[i].minimum = optimize->rangemin[i];
01628
                optimize->genetic_variable[i].maximum = optimize->rangemax[i];
01629
                optimize->genetic_variable[i].nbits = optimize->nbits[i];
01630
01631
01632
01633 #if DEBUG_OPTIMIZE
01634 fprintf (stderr, "optimize_open: nvariables=%u nsimulations=%u\n",
                 optimize->nvariables, optimize->nsimulations);
01635
01636 #endif
01637 optimize->value = (double *)
        g_malloc ((optimize->nsimulations
01638
                    + optimize->nestimates * optimize->nsteps)
* optimize->nvariables * sizeof (double));
01639
01640
01641
01642
       // Calculating simulations to perform for each task
01643 #if HAVE_MPI
01644 #if DEBUG_OPTIMIZE
01645 fprintf (stderr, "optimize_open: rank=%u ntasks=%u\n",
                 optimize->mpi_rank, ntasks);
01646
01647 #endif
01648
       optimize->nstart = optimize->mpi_rank * optimize->nsimulations / ntasks;
        optimize->nend = (1 + optimize->mpi_rank) * optimize->nsimulations / ntasks;
01649
01650
       if (optimize->nsteps)
01651
01652
            optimize->nstart_climbing
01653
              = optimize->mpi_rank * optimize->nestimates / ntasks;
01654
            optimize->nend_climbing
               = (1 + optimize->mpi_rank) * optimize->nestimates / ntasks;
01655
01656
01657 #else
01658
       optimize->nstart = 0;
        optimize->nend = optimize->nsimulations;
01659
       if (optimize->nsteps)
01660
01661
01662
            optimize->nstart_climbing = 0;
01663
            optimize->nend_climbing = optimize->nestimates;
01664
01665 #endif
01666 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: nstart=%u nend=%u\n", optimize->nstart,
01667
                 optimize->nend);
01668
01669 #endif
01670
       // Calculating simulations to perform for each thread
01671
01672
       optimize->thread
```

```
= (unsigned int *) alloca ((1 + nthreads) * sizeof (unsigned int));
01674
         for (i = 0; i <= nthreads; ++i)</pre>
01675
01676
             optimize->thread[i] = optimize->nstart
01677
                + i * (optimize->nend - optimize->nstart) / nthreads;
01678 #if DEBUG_OPTIMIZE
01679
            fprintf (stderr, "optimize_open: i=%u thread=%u\n", i,
01680
                       optimize->thread[i]);
01681 #endif
01682
         if (optimize->nsteps)
01683
          optimize->thread_climbing = (unsigned int *)
01684
             alloca ((1 + nthreads_climbing) * sizeof (unsigned int));
01685
01686
01687
        // Opening result files
01688
        optimize->file_result = g_fopen (optimize->result, "w");
         optimize->file_variables = g_fopen (optimize->variables, "w");
01689
01690
01691
         // Performing the algorithm
01692
         switch (optimize->algorithm)
01693
          // Genetic algorithm
case ALGORITHM_GENETIC:
01694
01695
            optimize_genetic ();
01696
01697
            break;
01698
01699
             // Iterative algorithm
01700
           default:
           optimize_iterate ();
}
01701
01702
01703
01704
        // Getting calculation time
01705
         t = g_date_time_new_now (tz);
01706
         {\tt optimize->calculation\_time} \ = \ 0.000001 \ \star \ {\tt g\_date\_time\_difference} \ \ ({\tt t, t0}) \ ;
        g_date_time_unref (t);
g_date_time_unref (t0);
01707
01708
01709
        g_time_zone_unref (tz);
        printf (optimize->file_result, "%s = %.61g s\n", ("Calculation time"), optimize->calculation_time); fprintf (optimize->file_result, "%s = %.61g s\n",
01710
01711
01712
                  _("Calculation time"), optimize->calculation_time);
01713
        // Closing result files
01714
        fclose (optimize->file_variables);
fclose (optimize->file_result);
01715
01716
01717
01718 #if DEBUG_OPTIMIZE
01719
        fprintf (stderr, "optimize_open: end\n");
01720 #endif
01721 }
```

Here is the call graph for this function:



5.21.3.21 optimize_orthogonal()

```
static void optimize_orthogonal ( ) [static]
```

Function to optimize with the orthogonal sampling algorithm.

Definition at line 761 of file optimize.c.

```
00763
        unsigned int i, j, k, l;
00764
        double e;
00765
        GThread *thread[nthreads];
00766 ParallelData data[nthreads];
00767 #if DEBUG_OPTIMIZE
00768
        fprintf (stderr, "optimize_orthogonal: start\n");
00769 #endif
00770
        for (i = 0; i < optimize->nsimulations; ++i)
00771
00772
            k = i;
             for (j = 0; j < optimize->nvariables; ++j)
00774
00775
                 1 = k % optimize->nsweeps[j];
00776
                k /= optimize->nsweeps[j];
00777
                 e = optimize->rangemin[j];
                 if (optimize->nsweeps[j] > 1)
  e += (1 + gsl_rng_uniform (optimize->rng))
  * (optimize->rangemax[j] - optimize->rangemin[j])
00778
00779
00780
00781
                      / optimize->nsweeps[j];
00782
                 optimize->value[i * optimize->nvariables + j] = e;
              }
00783
00784
00785
        optimize->nsaveds = 0;
        if (nthreads <= 1)
00787
          optimize_sequential ();
00788
00789
00790
            for (i = 0; i < nthreads; ++i)
00791
00792
                 data[i].thread = i;
00793
                 thread[i]
00794
                   = g_thread_new (NULL, (GThreadFunc) optimize_thread, &data[i]);
00795
00796
             for (i = 0; i < nthreads; ++i)
00797
              g_thread_join (thread[i]);
00798
00799 #if HAVE_MPI
00800 // Communicating tasks results
00801
        optimize_synchronise ();
00802 #endif
00803 #if DEBUG_OPTIMIZE
00804
       fprintf (stderr, "optimize_orthogonal: end\n");
00805 #endif
00806 }
```

5.21.3.22 optimize_parse()

```
static double optimize_parse (
          unsigned int simulation,
          unsigned int experiment ) [static]
```

Function to parse input files, simulating and calculating the objective function.

Returns

Objective function value.

Parameters

simulation	Simulation number.
experiment	Experiment number.

Definition at line 184 of file optimize.c.

```
00186 {
00187
        unsigned int i;
00188
        double e;
00189
       char buffer[512], input[MAX_NINPUTS][32], output[32], result[32], *buffer2,
00190
          *buffer3, *buffer4;
00191
       FILE *file result;
00192
00193 #if DEBUG_OPTIMIZE
     fprintf (stderr, "optimize_parse: start\n");
fprintf (stderr, "optimize_parse: simulation=%u experiment=%u\n",
00194
00195
00196
                 simulation, experiment);
00197 #endif
00198
00199
        // Opening input files
       for (i = 0; i < optimize->ninputs; ++i)
00200
00201
            snprintf (&input[i][0], 32, "input-%u-%u-%u", i, simulation, experiment);
00202
00203 #if DEBUG OPTIMIZE
00204
            fprintf (stderr, "optimize_parse: i=%u input=%s\n", i, &input[i][0]);
00205 #endif
00206
           optimize_input (simulation, &input[i][0], optimize->file[i][experiment]);
00207
00211
       fprintf (stderr, "optimize_parse: parsing end\n");
00212 #endif
00213
00214
        \ensuremath{//} Performing the simulation
       snprintf (output, 32, "output-%u-%u", simulation, experiment);
00215
       buffer2 = g_path_get_dirname (optimize->simulator);
00216
00217
       buffer3 = g_path_get_basename (optimize->simulator);
       buffer4 = g_build_filename (buffer2, buffer3, NULL);
00218
00219
       snprintf (buffer, 512, "\"%s\" %s %s",
00220
                  buffer4, input[0], input[1], input[2], input[3], input[4],
00221
                  input[5], input[6], input[7], output);
00222
       g free (buffer4);
00223
       g free (buffer3);
00224
       g_free (buffer2);
00225 #if DEBUG_OPTIMIZE
00226
       fprintf (stderr, "optimize_parse: %s\n", buffer);
00227 #endif
00228
       if (system (buffer) == -1)
00229
         error message = buffer;
00230
00231
        // Checking the objective value function
00232
       if (optimize->evaluator)
00233
00234
            snprintf (result, 32, "result-%u-%u", simulation, experiment);
            buffer2 = g_path_get_dirname (optimize->evaluator);
00235
00236
            buffer3 = g_path_get_basename (optimize->evaluator);
00237
            buffer4 = g_build_filename (buffer2, buffer3, NULL);
00238
            snprintf (buffer, 512, "\"%s\" %s %s %s",
                      buffer4, output, optimize->experiment[experiment], result);
00239
00240
            g_free (buffer4);
00241
            g_free (buffer3);
00242
            g_free (buffer2);
00243 #if DEBUG_OPTIMIZE
            00244
00245
00246 #endif
           if (system (buffer) == -1)
00247
00248
             error message = buffer;
            file_result = g_fopen (result, "r");
e = atof (fgets (buffer, 512, file_result));
00249
00250
00251
            fclose (file_result);
00252
00253
       else
00254
00255 #if DEBUG_OPTIMIZE
00256
           fprintf (stderr, "optimize_parse: output=%s\n", output);
00257 #endif
           strcpy (result, "");
00258
            file_result = g_fopen (output, "r");
e = atof (fgets (buffer, 512, file_result));
00259
00260
00261
            fclose (file_result);
00262
```

```
00263
00264
        // Removing files
00265 #if !DEBUG_OPTIMIZE
00266
       for (i = 0; i < optimize->ninputs; ++i)
00267
00268
           if (optimize->file[i][0])
00269
00270
                snprintf (buffer, 512, RM " %s", &input[i][0]);
00271
                if (system (buffer) == -1)
00272
                 error_message = buffer;
00273
             }
00274
         }
       snprintf (buffer, 512, RM " %s %s", output, result);
00275
00276
       if (system (buffer) == -1)
00277
         error_message = buffer;
00278 #endif
00279
00280
       // Processing pending events
       if (show_pending)
00281
00282
         show_pending ();
00283
00284 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_parse: end\n");
00285
00286 #endif
00287
       // Returning the objective function
00289
       return e * optimize->weight[experiment];
00290 }
```

Here is the call graph for this function:



5.21.3.23 optimize_print()

```
static void optimize_print ( ) [static]
```

Function to print the results.

Definition at line 399 of file optimize.c.

```
00400 {
00401
         unsigned int i;
00402
         char buffer[512];
00403 #if HAVE_MPI
00404 if (optimize->mpi_rank)
00405
           return;
00406 #endif
00407
         printf ("%s\n", _("Best result"));
         fprintf (optimize->file_result, "%s\n", _("Best result"));
printf ("error = %.15le\n", optimize->error_old[0]);
fprintf (optimize->file_result, "error = %.15le\n", optimize->error_old[0]);
00408
00409
00410
         for (i = 0; i < optimize->nvariables; ++i)
00411
00412
00413
              snprintf (buffer, 512, "%s = %s\n",
00414
                           optimize->label[i], format[optimize->precision[i]]);
00415
              printf (buffer, optimize->value_old[i]);
00416
              fprintf (optimize->file_result, buffer, optimize->value_old[i]);
00417
00418
         fflush (optimize->file_result);
00419 }
```

5.21.3.24 optimize_refine()

```
static void optimize_refine ( ) [inline], [static]
```

Function to refine the search ranges of the variables in iterative algorithms.

Definition at line 1266 of file optimize.c.

```
01268
       unsigned int i, j;
01269
       double d;
01270 #if HAVE_MPI
01271
      MPI Status mpi stat;
01272 #endif
01273 #if DEBUG_OPTIMIZE
01274
      fprintf (stderr, "optimize_refine: start\n");
01275 #endif
01276 #if HAVE_MPI
01277 if (!optimize->mpi_rank)
01278
01279 #endif
01280
           for (j = 0; j < optimize->nvariables; ++j)
01281
01282
               optimize->rangemin[j] = optimize->rangemax[j]
01283
                 = optimize->value_old[j];
01284
01285
           for (i = 0; ++i < optimize->nbest;)
01286
01287
               for (j = 0; j < optimize->nvariables; ++j)
01288
01289
                  optimize->rangemin[j]
                    = fmin (optimize->rangemin[j],
01290
01291
                            optimize->value_old[i * optimize->nvariables + j]);
01292
                   optimize->rangemax[j]
                     = fmax (optimize->rangemax[j],
01293
01294
                            optimize->value_old[i * optimize->nvariables + j]);
01295
                 }
01296
           for (j = 0; j < optimize->nvariables; ++j)
01298
01299
               d = optimize->tolerance
01300
                * (optimize->rangemax[j] - optimize->rangemin[j]);
               switch (optimize->algorithm)
01301
01302
01303
                 case ALGORITHM_MONTE_CARLO:
01304
                 d *= 0.5;
01305
                   break;
01306
                 default:
01307
                  if (optimize->nsweeps[j] > 1)
                    d /= optimize->nsweeps[j] - 1;
01308
01309
                   else
                    d = 0.;
01310
01311
01312
               optimize->rangemin[j] -= d;
01313
               optimize->rangemin[j]
01314
                 = fmax (optimize->rangemin[j], optimize->rangeminabs[j]);
01315
               optimize->rangemax[j] += d;
               01316
01317
01318
               printf ("%s min=%lg max=%lg\n", optimize->label[j],
               01319
01320
01321
01322
                       optimize->rangemax[j]);
01323
01324 #if HAVE_MPI
01325
          for (i = 1; (int) i < ntasks; ++i)</pre>
01326
               MPI Send (optimize->rangemin, optimize->nvariables, MPI DOUBLE, i,
01327
01328
                        1, MPI_COMM_WORLD);
01329
               MPI_Send (optimize->rangemax, optimize->nvariables, MPI_DOUBLE, i,
01330
                        1, MPI_COMM_WORLD);
01331
             }
         }
01332
01333
       else
01334
        {
01335
           MPI_Recv (optimize->rangemin, optimize->nvariables, MPI_DOUBLE, 0, 1,
01336
                     MPI_COMM_WORLD, &mpi_stat);
01337
           MPI_Recv (optimize->rangemax, optimize->nvariables, MPI_DOUBLE, 0, 1,
01338
                     MPI_COMM_WORLD, &mpi_stat);
01339
01340 #endif
01341 #if DEBUG_OPTIMIZE
01342 fprintf (stderr, "optimize_refine: end\n");
```

```
01343 #endif
01344 }
```

5.21.3.25 optimize_save_old()

```
static void optimize_save_old ( ) [inline], [static]
```

Function to save the best results on iterative methods.

Definition at line 1186 of file optimize.c.

```
01188
       unsigned int i, j;
01189 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_save_old: start\n");
fprintf (stderr, "optimize_save_old: nsaveds=%u\n", optimize->nsaveds);
01190
01191
01192 #endif
01193 memcpy (optimize->error_old, optimize->error_best,
01194
               optimize->nbest * sizeof (double));
       for (i = 0; i < optimize->nbest; ++i)
01195
       {
01196
            j = optimize->simulation_best[i];
01197
01198 #if DEBUG_OPTIMIZE
01199
            fprintf (stderr, "optimize_save_old: i=%u j=%u\n", i, j);
01200 #endif
01201
           memcpy (optimize->value_old + i * optimize->nvariables,
                    optimize->value + j * optimize->nvariables,
optimize->nvariables * sizeof (double));
01202
01203
01204
01205 #if DEBUG_OPTIMIZE
01206 for (i = 0; i < optimize->nvariables; ++i)
       01207
01208
01209
01210 #endif
01211 }
```

5.21.3.26 optimize_save_variables()

```
static void optimize_save_variables (
          unsigned int simulation,
          double error ) [static]
```

Function to save in a file the variables and the error.

Parameters

simulation	Simulation number.
error	Error value.

Definition at line 425 of file optimize.c.

```
00427 {
00428
        unsigned int i;
00429
        char buffer[64]:
00430 #if DEBUG_OPTIMIZE
00431
        fprintf (stderr, "optimize_save_variables: start\n");
00432 #endif
00433
        for (i = 0; i < optimize->nvariables; ++i)
00434
            snprintf (buffer, 64, "%s ", format[optimize->precision[i]]);
fprintf (optimize->file_variables, buffer,
00435
00436
00437
                       optimize->value[simulation * optimize->nvariables + i]);
00438
```

```
00439 fprintf (optimize->file_variables, "%.14le\n", error);
00440 fflush (optimize->file_variables);
00441 #if DEBUG_OPTIMIZE
00442 fprintf (stderr, "optimize_save_variables: end\n");
00443 #endif
00444 }
```

5.21.3.27 optimize sequential()

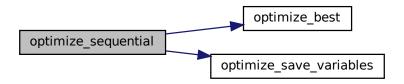
```
static void optimize_sequential ( ) [static]
```

Function to optimize sequentially.

Definition at line 491 of file optimize.c.

```
00492 {
00493
          unsigned int i;
00494
          double e;
00495 #if DEBUG_OPTIMIZE
00495 #IT DEBOG_TIMES
00496 fprintf (stderr, "optimize_sequential: start\n");
00497 fprintf (stderr, "optimize_sequential: nstart=%u nend=%u\n",
00498 optimize->nstart, optimize->nend);
00499 #endif
00500
         for (i = optimize->nstart; i < optimize->nend; ++i)
00501
               e = optimize_norm (i);
00502
              optimize_best (i, e);
optimize_save_variables (i, e);
if (e < optimize->threshold)
00503
00504
00505
00506
00507
                    optimize->stop = 1;
00508
                    break;
00509
00510 #if DEBUG_OPTIMIZE
00511
              fprintf (stderr, "optimize_sequential: i=%u e=%lg\n", i, e);
00512 #endif
00513
00514 #if DEBUG_OPTIMIZE
         fprintf (stderr, "optimize_sequential: end\n");
00515
00516 #endif
00517 }
```

Here is the call graph for this function:



5.21.3.28 optimize_step()

```
static void optimize_step ( ) [static]
```

Function to do a step of the iterative algorithm.

Definition at line 1350 of file optimize.c.

```
01351 {
01352 #if DEBUG_OPTIMIZE
01353 fprintf (stderr, "optimize_step: start\n");
01354 #endif
01355 optimize_algorithm ();
01356 if (optimize->nsteps)
01357 optimize_climbing ();
01358 #if DEBUG_OPTIMIZE
01359 fprintf (stderr, "optimize_step: end\n");
01360 #endif
01361 }
```

Here is the call graph for this function:



5.21.3.29 optimize step climbing()

Function to do a step of the hill climbing method.

Parameters

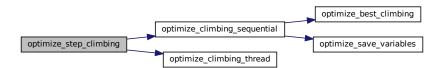
```
simulation Simulation number.
```

Definition at line 971 of file optimize.c.

```
00972 {
00973
        GThread *thread[nthreads_climbing];
00974
        ParallelData data[nthreads_climbing];
00975
        unsigned int i, j, k, b;
00976 #if DEBUG_OPTIMIZE
00977
        fprintf (stderr, "optimize_step_climbing: start\n");
00978 #endif
00979
        for (i = 0; i < optimize->nestimates; ++i)
00980
00981
            k = (simulation + i) * optimize->nvariables;
00982
            b = optimize->simulation_best[0] * optimize->nvariables;
00983 #if DEBUG_OPTIMIZE
            fprintf (stderr, "optimize_step_climbing: simulation=%u best=%u\n", simulation + i, optimize->simulation_best[0]);
00984
00985
00986 #endif
00987
            for (j = 0; j < optimize->nvariables; ++j, ++k, ++b)
00988
00989 #if DEBUG_OPTIMIZE
00990
                 fprintf (stderr,
                           optimize_step_climbing: estimate=%u best%u=%.14le\n",
00991
00992
                          i, j, optimize->value[b]);
00993 #endif
```

```
optimize->value[k]
00995
                  = optimize->value[b] + optimize_estimate_climbing (j, i);
00996
                optimize->value[k] = fmin (fmax (optimize->value[k],
                                                  optimize->rangeminabs[j]),
00997
00998
                                            optimize->rangemaxabs[j]);
00999 #if DEBUG_OPTIMIZE
01000
                fprintf (stderr,
01001
                          "optimize_step_climbing: estimate=%u variable%u=%.14le\n",
01002
                         i, j, optimize->value[k]);
01003 #endif
01004
              }
01005
01006
        if (nthreads_climbing == 1)
01007
         optimize_climbing_sequential (simulation);
01008
        else
01009
            for (i = 0; i <= nthreads_climbing; ++i)</pre>
01010
01011
01012
                optimize->thread_climbing[i]
01013
                 = simulation + optimize->nstart_climbing
01014
                 + i * (optimize->nend_climbing - optimize->nstart_climbing)
01015
                  / nthreads_climbing;
01016 #if DEBUG_OPTIMIZE
                fprintf (stderr,
01017
01018
                          "optimize_step_climbing: i=%u thread_climbing=%u\n",
01019
                         i, optimize->thread_climbing[i]);
01020 #endif
01021
            for (i = 0; i < nthreads_climbing; ++i)</pre>
01022
01023
01024
                data[i].thread = i;
01025
                thread[i] = g_thread_new
01026
                  (NULL, (GThreadFunc) optimize_climbing_thread, &data[i]);
01027
01028
            for (i = 0; i < nthreads_climbing; ++i)</pre>
             g_thread_join (thread[i]);
01029
01030
01031 #if DEBUG_OPTIMIZE
01032
       fprintf (stderr, "optimize_step_climbing: end\n");
01033 #endif
01034 }
```

Here is the call graph for this function:



5.21.3.30 optimize_sweep()

```
static void optimize_sweep ( ) [static]
```

Function to optimize with the sweep algorithm.

Definition at line 671 of file optimize.c.

```
00672 {
00673    unsigned int i, j, k, l;
00674    double e;
00675    GThread *thread[nthreads];
00676    ParallelData data[nthreads];
00677    #if DEBUG_OPTIMIZE
00678    fprintf (stderr, "optimize_sweep: start\n");
00679    #endif
00680    for (i = 0; i < optimize->nsimulations; ++i)
```

```
{
00682
            for (j = 0; j < optimize->nvariables; ++j)
00683
00684
                1 = k % optimize->nsweeps[j];
00685
00686
                k /= optimize->nsweeps[j];
                e = optimize->rangemin[j];
00687
00688
                if (optimize->nsweeps[j] > 1)
00689
                 e += 1 * (optimize->rangemax[j] - optimize->rangemin[j])
00690
                    / (optimize->nsweeps[j] - 1);
00691
                optimize->value[i * optimize->nvariables + j] = e;
00692
00693
00694
        optimize->nsaveds = 0;
00695
        if (nthreads <= 1)</pre>
00696
          optimize_sequential ();
00697
        else
00698
         {
00699
            for (i = 0; i < nthreads; ++i)
00700
              {
00701
                data[i].thread = i;
00702
                thread[i]
00703
                  = g_thread_new (NULL, (GThreadFunc) optimize_thread, &data[i]);
00704
00705
            for (i = 0; i < nthreads; ++i)
00706
              g_thread_join (thread[i]);
00707
00708 #if HAVE_MPI
00709 // Communicating tasks results 00710 optimize_synchronise ();
       optimize_synchronise ();
00711 #endif
00712 #if DEBUG_OPTIMIZE
00713
       fprintf (stderr, "optimize_sweep: end\n");
00714 #endif
00715 }
```

5.21.3.31 optimize synchronise()

static void optimize_synchronise () [static]

Function to synchronise the optimization results of MPI tasks.

Definition at line 624 of file optimize.c.

```
00625
00626
        unsigned int i, nsaveds, simulation best[optimize->nbest], stop;
00627
        double error_best[optimize->nbest];
00628
        MPI_Status mpi_stat;
00629 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_synchronise: start\n");
00630
00631 #endif
       if (optimize->mpi_rank == 0)
00632
00633
          {
            for (i = 1; (int) i < ntasks; ++i)
00634
00635
00636
                MPI_Recv (&nsaveds, 1, MPI_INT, i, 1, MPI_COMM_WORLD, &mpi_stat);
00637
                {\tt MPI\_Recv \ (simulation\_best, nsaveds, MPI\_INT, i, 1,}\\
                           MPI_COMM_WORLD, &mpi_stat);
00638
00639
                 MPI_Recv (error_best, nsaveds, MPI_DOUBLE, i, 1,
00640
                           MPI_COMM_WORLD, &mpi_stat);
00641
                 optimize_merge (nsaveds, simulation_best, error_best);
00642
                 MPI_Recv (&stop, 1, MPI_UNSIGNED, i, 1, MPI_COMM_WORLD, &mpi_stat);
00643
                if (stop)
00644
                   optimize->stop = 1;
00645
00646
            for (i = 1; (int) i < ntasks; ++i)</pre>
00647
              MPI_Send (&optimize->stop, 1, MPI_UNSIGNED, i, 1, MPI_COMM_WORLD);
00648
00649
        else
00650
         {
            MPI_Send (&optimize->nsaveds, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);
00651
00652
            MPI_Send (optimize->simulation_best, optimize->nsaveds, MPI_INT, 0, 1,
00653
                       MPI_COMM_WORLD);
00654
            MPI_Send (optimize->error_best, optimize->nsaveds, MPI_DOUBLE, 0, 1,
                       MPI_COMM_WORLD);
00655
            MPI_Send (&optimize->stop, 1, MPI_UNSIGNED, 0, 1, MPI_COMM_WORLD);
MPI_Recv (&stop, 1, MPI_UNSIGNED, 0, 1, MPI_COMM_WORLD, &mpi_stat);
00656
00657
00658
            if (stop)
              optimize->stop = 1;
```

```
00660  }
00661 #if DEBUG_OPTIMIZE
00662 fprintf (stderr, "optimize_synchronise: end\n");
00663 #endif
00664 }
```

5.21.3.32 optimize_thread()

Function to optimize on a thread.

Returns

NULL.

Parameters

data Function data.

Definition at line 525 of file optimize.c.

```
00526 {
00527
         unsigned int i, thread;
00528 double e;
00529 #if DEBUG_OPTIMIZE
00530 fprintf (stderr, "optimize_thread: start\n");
00531 #endif
00532 thread = data->thread;
00533 #if DEBUG_OPTIMIZE
00534 fprintf (stderr, "optimize_thread: thread=%u start=%u end=%u\n", thread, 00535 optimize->thread[thread], optimize->thread[thread + 1]);
00536 #endif
00537 for (i = optimize->thread[thread]; i < optimize->thread[thread + 1]; ++i)
          e = optimize_norm (i);
g_mutex_lock (muter)
optimic
00538
         {
00539
00540
             optimize_best (i, e);
optimize_save_variables (i, e);
if (e < optimize->threshold)
00541
00542
00543
00544
               optimize->stop = 1;
00545
              g_mutex_unlock (mutex);
00546
              if (optimize->stop)
  break;
00547
00548 #if DEBUG_OPTIMIZE
              fprintf (stderr, "optimize_thread: i=%u e=%lg\n", i, e);
00549
00550 #endif
00551
00552 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_thread: end\n");
00553
00554 #endif
00555 g_thread_exit (NULL);
00556 return NULL;
00557 }
```

5.21.4 Variable Documentation

5.21.4.1 nthreads_climbing

```
unsigned int nthreads_climbing
```

Number of threads for the hill climbing method.

Definition at line 80 of file optimize.c.

5.21.4.2 optimize

```
Optimize optimize[1]
```

Optimization data.

Definition at line 79 of file optimize.c.

5.21.4.3 optimize_algorithm

```
void(* optimize_algorithm) () ( ) [static]
```

Pointer to the function to perform a optimization algorithm step.

Definition at line 83 of file optimize.c.

5.21.4.4 optimize_estimate_climbing

Pointer to the function to estimate the climbing.

Definition at line 85 of file optimize.c.

5.21.4.5 optimize_norm

```
\begin{tabular}{lll} \begin{
```

Pointer to the error norm function.

Definition at line 88 of file optimize.c.

5.22 optimize.c

Go to the documentation of this file.

```
00001 /*
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00009
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are met:
00013 1. Redistributions of source code must retain the above copyright notice,
{\tt 00014} this list of conditions and the following disclaimer.
00015
00016 2. Redistributions in binary form must reproduce the above copyright notice, 00017 this list of conditions and the following disclaimer in the \,
00018 documentation and/or other materials provided with the distribution.
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF 00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING 00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
00031
00038 #define _GNU_SOURCE
00039 #include "config.h"
00040 #include <stdio.h>
00041 #include <stdlib.h>
00042 #include <string.h>
00043 #include <math.h>
00044 #include <sys/param.h>
00045 #include <gsl/gsl_rng.h>
00046 #include <libxml/parser.h>
00047 #include <libintl.h>
00048 #include <glib.h>
00049 #include <glib/gstdio.h>
00050 #include <json-glib/json-glib.h>
00051 #ifdef G_OS_WIN32
00052 #include <windows.h>
00053 #elif !defined(__BSD_VISIBLE) && !defined(NetBSD)
00054 #include <alloca.h>
00055 #endif
00056 #if HAVE_MPI
00057 #include <mpi.h>
00058 #endif
00059 #include "jb/src/jb_win.h"
00060 #include "genetic/genetic.h"
00061 #include "tools.h"
00062 #include "experiment.h"
00063 #include "variable.h"
00064 #include "input.h"
00065 #include "optimize.h"
00067 #define DEBUG_OPTIMIZE 0
00068
00073 #ifdef G_OS_WIN32
00074 #define RM "del"
00075 #else
00076 #define RM "rm"
00077 #endif
00078
00079 Optimize optimize[1];
00080 unsigned int nthreads_climbing;
00082
00083 static void (*optimize algorithm) ();
00085 static double (*optimize_estimate_climbing) (unsigned int variable,
00086
00088 static double (*optimize_norm) (unsigned int simulation);
00090
00094 static inline void
00095 optimize_input (unsigned int simulation,
00096
                         char *input,
00097
                        GMappedFile * stencil)
00098 {
00099
        char buffer[32], value[32];
```

```
00100
        GRegex *regex;
        FILE *file;
00101
00102
        char *buffer2, *buffer3 = NULL, *content;
00103
        gsize length;
00104
       unsigned int i;
00105
00106 #if DEBUG_OPTIMIZE
00107
       fprintf (stderr, "optimize_input: start\n");
00108 #endif
00109
00110
        // Checking the file
00111
       if (!stencil)
00112
        goto optimize_input_end;
00113
00114
       // Opening stencil
       content = g_mapped_file_get_contents (stencil);
length = g_mapped_file_get_length (stencil);
00115
00116
00117 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_input: length=%lu\ncontent:\n%s", length, content);
00118
00119 #endif
00120 file = g_fopen (input, "w");
00121
       // Parsing stencil
00122
00123
       for (i = 0; i < optimize->nvariables; ++i)
00124
00125 #if DEBUG_OPTIMIZE
00126
            fprintf (stderr, "optimize_input: variable=u\n", i);
00127 #endif
           snprintf (buffer, 32, "@variable%u@", i + 1);
regex = g_regex_new (buffer, (GRegexCompileFlags) 0, (GRegexMatchFlags) 0,
00128
00129
00130
                                  NULL);
00131
            if (i == 0)
00132
00133
                buffer2 = g_regex_replace_literal (regex, content, length, 0,
                                                     optimize->label[i],
00134
                                                     (GRegexMatchFlags) 0, NULL);
00135
00136 #if DEBUG_OPTIMIZE
                fprintf (stderr, "optimize_input: buffer2\n%s", buffer2);
00138 #endif
00139
00140
            else
00141
             {
00142
                length = strlen (buffer3):
00143
                buffer2 = g_regex_replace_literal (regex, buffer3, length, 0,
                                                     optimize->label[i],
00144
00145
                                                     (GRegexMatchFlags) 0, NULL);
00146
               g_free (buffer3);
00147
            g_regex_unref (regex);
00148
00149
            length = strlen (buffer2);
            snprintf (buffer, 32, "@value%u@", i + 1);
00150
00151
            regex = g_regex_new (buffer, (GRegexCompileFlags) 0, (GRegexMatchFlags) 0,
00152
                                  NULL);
            snprintf (value, 32, format[optimize->precision[i]],
00153
                      optimize->value[simulation * optimize->nvariables + i]);
00154
00155
00156 #if DEBUG_OPTIMIZE
00157
            fprintf (stderr, "optimize_input: value=%s\n", value);
00158 #endif
00159
            buffer3 = g_regex_replace_literal (regex, buffer2, length, 0, value,
00160
                                                 (GRegexMatchFlags) 0, NULL);
00161
            q free (buffer2);
00162
           g_regex_unref (regex);
00163
00164
00165
        // Saving input file
00166
       fwrite (buffer3, strlen (buffer3), sizeof (char), file);
00167
       g free (buffer3);
00168
       fclose (file);
00169
00170 optimize_input_end:
00171 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_input: end\n");
00172
00173 #endif
00174
       return;
00175 }
00176
00183 static double
00184 optimize_parse (unsigned int simulation,
00185
                      unsigned int experiment)
00186 {
00187
       unsigned int i;
00188
       double e;
00189
        char buffer[512], input[MAX_NINPUTS][32], output[32], result[32], *buffer2,
00190
         *buffer3, *buffer4;
       FILE *file_result;
00191
00192
```

```
00193 #if DEBUG_OPTIMIZE
      fprintf (stderr, "optimize_parse: start\n");
fprintf (stderr, "optimize_parse: simulation
00194
00195
                                              simulation=%u experiment=%u\n",
                 simulation, experiment);
00196
00197 #endif
00198
00199
         // Opening input files
00200
        for (i = 0; i < optimize->ninputs; ++i)
00201
        {
            snprintf (&input[i][0], 32, "input-%u-%u-%u", i, simulation, experiment);
00202
00203 #if DEBUG OPTIMIZE
            fprintf (stderr, "optimize_parse: i=%u input=%s\n", i, &input[i][0]);
00204
00205 #endif
00206
            optimize_input (simulation, &input[i][0], optimize->file[i][experiment]);
00207
fprintf (stderr, "optimize_parse: parsing end\n");
00211
00212 #endif
00213
00214
        \ensuremath{//} Performing the simulation
        snprintf (output, 32, "output-%u-%u", simulation, experiment);
buffer2 = g_path_get_dirname (optimize->simulator);
00215
00216
00217
        buffer3 = g_path_get_basename (optimize->simulator);
        buffer4 = g_build_filename (buffer2, buffer3, NULL);
00218
00219
        snprintf (buffer, 512, "\"%s\" %s %s",
                   buffer4, input[0], input[1], input[2], input[3], input[4],
00220
00221
                   input[5], input[6], input[7], output);
00222
       g_free (buffer4);
00223
       a free (buffer3);
00224
        g_free (buffer2);
00225 #if DEBUG_OPTIMIZE
00226
        fprintf (stderr, "optimize_parse: %s\n", buffer);
00227 #endif
        if (system (buffer) == -1)
00228
00229
          error message = buffer;
00231
        // Checking the objective value function
00232
        if (optimize->evaluator)
00233
00234
             snprintf (result, 32, "result-%u-%u", simulation, experiment);
            buffer2 = g_path_get_dirname (optimize->evaluator);
00235
            buffer3 = g_path_get_basename (optimize->evaluator);
00236
            buffer4 = g_build_filename (buffer2, buffer3, NULL);
00237
00238
            snprintf (buffer, 512, "\"%s\" %s %s %s",
00239
                       buffer4, output, optimize->experiment[experiment], result);
00240
            g_free (buffer4);
            g_free (buffer3);
00241
00242
             a free (buffer2);
00243 #if DEBUG_OPTIMIZE
            fprintf (stderr, "optimize_parse: %s\n", buffer);
fprintf (stderr, "optimize_parse: result=%s\n", result);
00244
00245
00246 #endif
            if (system (buffer) == -1)
00247
00248
               error_message = buffer;
             file_result = g_fopen (result, "r");
e = atof (fgets (buffer, 512, file_result));
00250
00251
             fclose (file_result);
00252
        else
00253
00254
00255 #if DEBUG_OPTIMIZE
            fprintf (stderr, "optimize_parse: output=%s\n", output);
00256
00257 #endif
00258
            strcpy (result, "");
            file_result = g_fopen (output, "r");
e = atof (fgets (buffer, 512, file_result));
00259
00260
00261
            fclose (file_result);
00262
          }
00263
00264
        // Removing files
00265 #if !DEBUG_OPTIMIZE
        for (i = 0; i < optimize->ninputs; ++i)
00266
00267
             if (optimize->file[i][0])
00268
00269
              {
00270
                 snprintf (buffer, 512, RM " %s", &input[i][0]);
00271
                 if (system (buffer) == -1)
00272
                   error_message = buffer;
00273
00274
        snprintf (buffer, 512, RM " %s %s", output, result); if (system (buffer) == -1)
00275
00276
00277
          error_message = buffer;
00278 #endif
00279
```

```
// Processing pending events
00281
        if (show_pending)
00282
          show_pending ();
00283
00284 #if DEBUG OPTIMIZE
        fprintf (stderr, "optimize_parse: end\n");
00285
00286 #endif
00287
00288
        // Returning the objective function
00289
        return e * optimize->weight[experiment];
00290 }
00291
00297 static double
00298 optimize_norm_euclidian (unsigned int simulation)
00299 {
        double e, ei;
00300
00301
        unsigned int i;
00302 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_norm_euclidian: start\n");
00303
00304 #endif
00305 e = 0.;
        for (i = 0; i < optimize->nexperiments; ++i)
00306
00307
            ei = optimize_parse (simulation, i);
00308
00309
            e += ei * ei;
00310
        e = sqrt (e);
00311
00312 #if DEBUG_OPTIMIZE
00313 #II DEBOG_OFIMIZE
00313 fprintf (stderr, "optimize_norm_euclidian: error=%lg\n", e);
00314 fprintf (stderr, "optimize_norm_euclidian: end\n");
00315 #endif
00316
        return e;
00317 }
00318
00324 static double
00325 optimize_norm_maximum (unsigned int simulation)
00326 {
        double e, ei;
00328
        unsigned int i;
00329 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_norm_maximum: start\n");
00330
00331 #endif
00332 e = 0.;
00333
        for (i = 0; i < optimize->nexperiments; ++i)
00334
00335
             ei = fabs (optimize_parse (simulation, i));
00336
            e = fmax (e, ei);
00337
00338 #if DEBUG_OPTIMIZE
00339 fprintf (stderr, "optimize_norm_maximum: error=%lg\n", e);
00340 fprintf (stderr, "optimize_norm_maximum: end\n");
00341 #endif
00342
        return e;
00343 }
00344
00350 static double
00351 optimize_norm_p (unsigned int simulation)
00352 {
00353 double e, ei;
00354 unsigned int i; 00355 #if DEBUG_OPTIMIZE
00356 fprintf (stderr, "optimize_norm_p: start\n");
00357 #endif
00358
      e = 0.;
00359
        for (i = 0; i < optimize->nexperiments; ++i)
00360
00361
            ei = fabs (optimize_parse (simulation, i));
00362
            e += pow (ei, optimize->p);
00363
00364
        e = pow (e, 1.
                          / optimize->p);
00365 #if DEBUG_OPTIMIZE
00366 fprintf (stderr, "optimize_norm_p: error=%lg\n", e);
00367 fprintf (stderr, "optimize_norm_p: end\n");
00368 #endif
00369
        return e;
00370 }
00371
00377 static double
00378 optimize_norm_taxicab (unsigned int simulation)
00379 {
00380
        double e;
00381
        unsigned int i;
00382 #if DEBUG_OPTIMIZE
00383
        fprintf (stderr, "optimize_norm_taxicab: start\n");
00384 #endif
       e = 0.;
00385
00386
        for (i = 0; i < optimize->nexperiments; ++i)
```

```
e += fabs (optimize_parse (simulation, i));
00388 #if DEBUG_OPTIMIZE
00389 fprintf (stderr, "optimize_norm_taxicab: error=%lg\n", e);
        fprintf (stderr, "optimize_norm_taxicab: end\n");
00390
00391 #endif
00392
        return e:
00393 }
00394
00398 static void
00399 optimize_print ()
00400 {
        unsigned int i;
00401
00402
        char buffer[512];
00403 #if HAVE_MPI
00404 if (optimize->mpi_rank)
00405
          return;
00406 #endif
        printf ("%s\n", _("Best result"));
fprintf (optimize->file_result, "%s\n", _("Best result"));
00407
        printf ("error = %.15le\n", optimize->error_old[0]);
00409
00410
         fprintf (optimize->file_result, "error = %.15le\n", optimize->error_old[0]);
00411
         for (i = 0; i < optimize->nvariables; ++i)
00412
             snprintf (buffer, 512, "%s = %s\n",
00413
             optimize->label[i], format[optimize->precision[i]]);
printf (buffer, optimize->value_old[i]);
00414
00415
00416
             fprintf (optimize->file_result, buffer, optimize->value_old[i]);
00417
00418
        fflush (optimize->file_result);
00419 }
00420
00424 static void
00425 optimize_save_variables (unsigned int simulation,
00426
                                  double error)
00427 {
        unsigned int i;
00428
        char buffer[64];
00429
00430 #if DEBUG_OPTIMIZE
00431
        fprintf (stderr, "optimize_save_variables: start\n");
00432 #endif
00433
        for (i = 0; i < optimize->nvariables; ++i)
00434
             snprintf (buffer, 64, "%s ", format[optimize->precision[i]]);
fprintf (optimize->file_variables, buffer,
00435
00436
                       optimize->value[simulation * optimize->nvariables + i]);
00437
00438
00439
        fprintf (optimize->file_variables, "%.14le\n", error);
00440 fflush (optimize->file_variables);
00441 #if DEBUG_OPTIMIZE
00442 fprintf (stderr, "optimize_save_variables: end\n");
00443 #endif
00444 }
00445
00449 static void
00450 optimize_best (unsigned int simulation,
00451
                       double value)
00452 {
00453 unsigned int i, j;
00454 double e;
00455 #if DEBUG_OPTIMIZE
00456 fprintf (stderr, "optimize_best: start\n");
00457 fprintf (stderr, "optimize_best: nsaveds=%u nbest=%u\n",
00458
                  optimize->nsaveds, optimize->nbest);
00459 #endif
00460
        if (optimize->nsaveds < optimize->nbest
00461
            || value < optimize->error_best[optimize->nsaveds - 1])
00462
             if (optimize->nsaveds < optimize->nbest)
00463
00464
              ++optimize->nsaveds;
             optimize->error_best[optimize->nsaveds - 1] = value;
optimize->simulation_best[optimize->nsaveds - 1] = simulation;
00465
00466
00467
             for (i = optimize->nsaveds; --i;)
00468
                 if (optimize->error_best[i] < optimize->error_best[i - 1])
00469
00470
                   {
00471
                      j = optimize->simulation_best[i];
00472
                      e = optimize->error_best[i];
00473
                      optimize->simulation_best[i] = optimize->simulation_best[i - 1];
00474
                      optimize->error_best[i] = optimize->error_best[i - 1];
00475
                      optimize->simulation_best[i - 1] = j;
00476
                      optimize->error_best[i - 1] = e;
                   }
00478
                 else
00479
                   break;
00480
               }
00481
00482 #if DEBUG_OPTIMIZE
```

```
fprintf (stderr, "optimize_best: end\n");
00484 #endif
00485 }
00486
00490 static void
00491 optimize seguential ()
00492 {
00493
        unsigned int i;
       double e;
00494
00495 #if DEBUG_OPTIMIZE
00496 fprintf (stderr, "optimize_sequential: start\n");
00497 fprintf (stderr, "optimize_sequential: nstart=%u nend=%u\n",
                 optimize->nstart, optimize->nend);
00498
00499 #endif
00500
       for (i = optimize->nstart; i < optimize->nend; ++i)
00501
00502
            e = optimize_norm (i);
            optimize_best (i, e);
optimize_save_variables (i, e);
00503
00505
            if (e < optimize->threshold)
00506
00507
                optimize->stop = 1;
00508
                break;
00509
00510 #if DEBUG_OPTIMIZE
            fprintf (stderr, "optimize_sequential: i=%u e=%lg\n", i, e);
00511
00512 #endif
00513
00514 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_sequential: end\n");
00515
00516 #endif
00517 }
00518
00524 static void \star
00525 optimize_thread (ParallelData * data)
00526 {
00527
       unsigned int i, thread;
        double e;
00529 #if DEBUG_OPTIMIZE
00530
       fprintf (stderr, "optimize_thread: start\n");
00531 #endif
       thread = data->thread;
00532
00533 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_thread: thread=%u start=%u end=%u\n", thread,
00534
                 optimize->thread[thread], optimize->thread[thread + 1]);
00535
00536 #endif
00537
       for (i = optimize->thread[thread]; i < optimize->thread[thread + 1]; ++i)
00538
00539
            e = optimize norm (i);
00540
            g_mutex_lock (mutex);
00541
            optimize_best (i, e);
00542
            optimize_save_variables (i, e);
00543
            if (e < optimize->threshold)
00544
             optimize->stop = 1;
00545
            g_mutex_unlock (mutex);
00546
            if (optimize->stop)
00547
              break;
00548 #if DEBUG_OPTIMIZE
00549
            fprintf (stderr, "optimize_thread: i=%u e=%lg\n", i, e);
00550 #endif
00551
00552 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_thread: end\n");
00554 #endif
00555
       g_thread_exit (NULL);
00556
       return NULL;
00557 }
00558
00562 static inline void
00563 optimize_merge (unsigned int nsaveds,
00564
                      unsigned int *simulation_best,
00566
                       double *error_best)
00568 {
       unsigned int i, j, k, s[optimize->nbest]; double e[optimize->nbest];
00569
00570
00571 #if DEBUG_OPTIMIZE
00572
       fprintf (stderr, "optimize_merge: start\n");
00573 #endif
00574 i = j = k = 0;
00575
       do
00576
00577
            if (i == optimize->nsaveds)
00578
00579
                s[k] = simulation_best[j];
00580
                e[k] = error_best[j];
00581
                ++ 1;
00582
                ++k;
```

```
if (j == nsaveds)
00584
                 break;
00585
00586
            else if (j == nsaveds)
00587
              {
00588
                s[k] = optimize->simulation_best[i];
                e[k] = optimize->error_best[i];
00590
                ++i;
00591
                ++k;
00592
                if (i == optimize->nsaveds)
00593
                  break:
00594
00595
            else if (optimize->error_best[i] > error_best[j])
00596
00597
                s[k] = simulation_best[j];
00598
                e[k] = error_best[j];
00599
                ++i;
00600
                ++k;
00601
00602
            else
00603
              {
00604
                s[k] = optimize->simulation_best[i];
                e[k] = optimize->error_best[i];
00605
00606
                ++i:
00607
                ++k;
00608
00609
00610
       while (k < optimize->nbest);
00611
       optimize->nsaveds = k;
        memcpy (optimize->simulation_best, s, k * sizeof (unsigned int));
00612
00613 memcpy (optimize->error_best, e, k * sizeof (double));
00614 #if DEBUG_OPTIMIZE
00615
       fprintf (stderr, "optimize_merge: end\n");
00616 #endif
00617 }
00618
00622 #if HAVE_MPI
00623 static void
00624 optimize_synchronise ()
00625 {
00626
       unsigned int i, nsaveds, simulation_best[optimize->nbest], stop;
00627
       double error best[optimize->nbest];
00628
       MPI_Status mpi_stat;
00629 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_synchronise: start\n");
00631 #endif
00632
       if (optimize->mpi_rank == 0)
00633
00634
            for (i = 1; (int) i < ntasks; ++i)</pre>
00635
00636
                MPI_Recv (&nsaveds, 1, MPI_INT, i, 1, MPI_COMM_WORLD, &mpi_stat);
00637
                MPI_Recv (simulation_best, nsaveds, MPI_INT, i, 1,
00638
                           MPI_COMM_WORLD, &mpi_stat);
00639
                MPI_Recv (error_best, nsaveds, MPI_DOUBLE, i, 1,
00640
                           MPI_COMM_WORLD, &mpi_stat);
                optimize_merge (nsaveds, simulation_best, error_best);
MPI_Recv (&stop, 1, MPI_UNSIGNED, i, 1, MPI_COMM_WORLD, &mpi_stat);
00641
00643
                if (stop)
00644
                  optimize->stop = 1;
00645
            for (i = 1; (int) i < ntasks; ++i)</pre>
00646
              MPI_Send (&optimize->stop, 1, MPI_UNSIGNED, i, 1, MPI_COMM_WORLD);
00647
00648
00649
        else
00650
00651
            MPI_Send (&optimize->nsaveds, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);
00652
            MPI_Send (optimize->simulation_best, optimize->nsaveds, MPI_INT, 0, 1,
                      MPI_COMM_WORLD);
00653
00654
            MPI_Send (optimize->error_best, optimize->nsaveds, MPI_DOUBLE, 0, 1,
                      MPI_COMM_WORLD);
00656
            MPI_Send (&optimize->stop, 1, MPI_UNSIGNED, 0, 1, MPI_COMM_WORLD);
00657
            MPI_Recv (&stop, 1, MPI_UNSIGNED, 0, 1, MPI_COMM_WORLD, &mpi_stat);
00658
            if (stop)
00659
              optimize->stop = 1;
00660
00661 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_synchronise: end\n");
00662
00663 #endif
00664 3
00665 #endif
00666
00670 static void
00671 optimize_sweep ()
00672 {
00673
       unsigned int i, j, k, l;
00674
        double e;
00675
       GThread *thread[nthreads];
```

```
00676 ParallelData data[nthreads];
00677 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_sweep: start\n");
00678
00679 #endif
00680
       for (i = 0; i < optimize->nsimulations; ++i)
00681
            k = i;
00683
            for (j = 0; j < optimize->nvariables; ++j)
00684
00685
                1 = k % optimize->nsweeps[j];
                k /= optimize->nsweeps[j];
00686
                e = optimize->rangemin[j];
00687
                if (optimize->nsweeps[j] > 1)
  e += 1 * (optimize->rangemax[j] - optimize->rangemin[j])
00688
00689
00690
                   / (optimize->nsweeps[j] - 1);
00691
                optimize->value[i * optimize->nvariables + j] = e;
00692
00693
00694
        optimize->nsaveds = 0;
00695
        if (nthreads <= 1)</pre>
00696
          optimize_sequential ();
00697
        else
00698
         {
00699
            for (i = 0; i < nthreads; ++i)
00700
              {
00701
                data[i].thread = i;
00702
                thread[i]
00703
                  = g_thread_new (NULL, (GThreadFunc) optimize_thread, &data[i]);
00704
00705
            for (i = 0; i < nthreads; ++i)
00706
             g_thread_join (thread[i]);
00707
00708 #if HAVE_MPI
00709
       // Communicating tasks results
00710
       optimize_synchronise ();
00711 #endif
00712 #if DEBUG_OPTIMIZE
00713 fprintf (stderr, "optimize_sweep: end\n");
00714 #endif
00715 }
00716
00720 static void
00721 optimize MonteCarlo ()
00722 {
00723
        unsigned int i, j;
00724
        GThread *thread[nthreads];
00725
       ParallelData data[nthreads];
00726 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_MonteCarlo: start\n");
00727
00728 #endif
00729
       for (i = 0; i < optimize->nsimulations; ++i)
00730
         for (j = 0; j < optimize->nvariables; ++j)
00731
            optimize->value[i \star optimize->nvariables + j]
00732
             = optimize->rangemin[j] + gsl_rng_uniform (optimize->rng)
              * (optimize->rangemax[j] - optimize->rangemin[j]);
00733
00734
        optimize->nsaveds = 0;
00735
        if (nthreads <= 1)</pre>
00736
          optimize_sequential ();
00737
00738
         {
00739
            for (i = 0; i < nthreads; ++i)
00740
              {
00741
                data[i].thread = i;
00742
                thread[i]
00743
                  = g_thread_new (NULL, (GThreadFunc) optimize_thread, &data[i]);
00744
            for (i = 0; i < nthreads; ++i)
00745
00746
              g_thread_join (thread[i]);
00747
00748 #if HAVE_MPI
00749 // Communicating tasks results
00750
       optimize_synchronise ();
00751 #endif
00752 #if DEBUG_OPTIMIZE
00753
       fprintf (stderr, "optimize_MonteCarlo: end\n");
00754 #endif
00755 }
00756
00760 static void
00761 optimize_orthogonal ()
00762 {
       unsigned int i, j, k, l;
00764
        double e;
00765
       GThread *thread[nthreads];
00766
       ParallelData data[nthreads];
00767 #if DEBUG_OPTIMIZE
00768
       fprintf (stderr, "optimize_orthogonal: start\n");
```

```
00769 #endif
00770
       for (i = 0; i < optimize->nsimulations; ++i)
00771
00772
            k = i:
             for (j = 0; j < optimize->nvariables; ++j)
00773
00774
00775
                 1 = k % optimize->nsweeps[j];
00776
                 k /= optimize->nsweeps[j];
                 e = optimize->rangemin[j];
00777
00778
                 if (optimize->nsweeps[j] > 1)
00779
                  e += (1 + gsl_rng_uniform (optimize->rng))
00780
                    * (optimize->rangemax[j] - optimize->rangemin[j])
                      / optimize->nsweeps[j];
00781
00782
                 optimize->value[i * optimize->nvariables + j] = e;
00783
00784
        optimize->nsaveds = 0;
00785
00786
        if (nthreads <= 1)</pre>
          optimize_sequential ();
00788
        else
00789
00790
             for (i = 0; i < nthreads; ++i)
00791
              {
00792
                data[i].thread = i;
00793
                thread[i]
00794
                  = g_thread_new (NULL, (GThreadFunc) optimize_thread, &data[i]);
00795
00796
             for (i = 0; i < nthreads; ++i)</pre>
00797
              g_thread_join (thread[i]);
00798
00799 #if HAVE_MPI
00800 // Communicating tasks results
00801 optimize_synchronise ();
00802 #endif
00803 #if DEBUG_OPTIMIZE 00804 fprintf (stderr, "optimize_orthogonal: end\n");
00805 #endif
00807
00811 static void
00812 optimize_best_climbing (unsigned int simulation, 00813 double value)
00814 {
00815 #if DEBUG_OPTIMIZE
00816 fprintf (stderr, "optimize_best_climbing: start\n");
        fprintf (stderr,
00818
                  "optimize_best_climbing: simulation=%u value=%.14le best=%.14le\n",
00819
                  simulation, value, optimize->error_best[0]);
00820 #endif
00821 if (value < optimize->error_best[0])
        {
00823
            optimize->error_best[0] = value;
00824
            optimize->simulation_best[0] = simulation;
00825 #if DEBUG_OPTIMIZE
00826
            fprintf (stderr,
00827
                       "optimize_best_climbing: BEST simulation=%u value=%.14le\n",
00828
                      simulation, value);
00829 #endif
00830 }
00831 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_best_climbing: end\n");
00832
00833 #endif
00834 }
00835
00839 static inline void
00840 optimize_climbing_sequential (unsigned int simulation)
00841 {
00842
        double e;
        unsigned int i, j;
00843
00844 #if DEBUG_OPTIMIZE
00845 fprintf (stderr, "optimize_climbing_sequential: start\n");
00846 fprintf (stderr, "optimize_climbing_sequential: nstart_climbing=%u "
                  "nend_climbing=%u\n",
00847
00848
                  optimize->nstart_climbing, optimize->nend_climbing);
00849 #endif
00850 for (i = optimize->nstart_climbing; i < optimize->nend_climbing; ++i)
00851
         {
00852
            j = simulation + i;
00853
             e = optimize_norm (j);
00854
            optimize_best_climbing (j, e);
            optimize_save_variables (j, e);
00855
00856
            if (e < optimize->threshold)
00857
00858
                 optimize->stop = 1;
00859
                break;
00860
00861 #if DEBUG_OPTIMIZE
```

```
fprintf (stderr, "optimize_climbing_sequential: i=%u e=%lg\n", i, e);
00863 #endif
00864
00865 #if DEBUG OPTIMIZE
       fprintf (stderr, "optimize_climbing_sequential: end\n");
00866
00867 #endif
00868 }
00869
00875 static void *
00876 optimize_climbing_thread (ParallelData * data)
00877 {
       unsigned int i, thread;
00878
00879
        double e;
00880 #if DEBUG_OPTIMIZE
00881
       fprintf (stderr, "optimize_climbing_thread: start\n");
00882 #endif
00883
       thread = data->thread;
00884 #if DEBUG_OPTIMIZE
00885 fprintf (stderr, "optimize_climbing_thread: thread=%u start=%u end=%u\n",
                 thread,
00886
00887
                  optimize->thread_climbing[thread],
                 optimize->thread_climbing[thread + 1]);
00888
00889 #endif
       for (i = optimize->thread_climbing[thread];
00890
00891
             i < optimize->thread_climbing[thread + 1]; ++i)
00892
00893
            e = optimize_norm (i);
00894
            g_mutex_lock (mutex);
00895
            optimize_best_climbing (i, e);
00896
            optimize_save_variables (i, e);
            if (e < optimize->threshold)
  optimize->stop = 1;
00897
00898
00899
            g_mutex_unlock (mutex);
            if (optimize->stop)
00900
00901
              break;
00902 #if DEBUG_OPTIMIZE
00903
            fprintf (stderr, "optimize_climbing_thread: i=%u e=%lg\n", i, e);
00904 #endif
00905
00906 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_climbing_thread: end\n");
00907
00908 #endif
00909 g thread exit (NULL);
00910
        return NULL;
00911 }
00912
00916 static double
{\tt 00917~optimize\_estimate\_climbing\_random~(unsigned~int~variable,}
00919
                                          unsigned int estimate
00920
                                           attribute ((unused)))
00922 {
00923
        double x;
00924 #if DEBUG_OPTIMIZE
00925
       fprintf (stderr, "optimize_estimate_climbing_random: start\n");
00926 #endif
00927 x = optimize->climbing[variable]
00928 + (1. - 2. * gsl_rng_uniform
          + (1. - 2. * gsl_rng_uniform (optimize->rng)) * optimize->step[variable];
00929 #if DEBUG_OPTIMIZE
00930 fprintf (stderr, "optimize_estimate_climbing_random: climbing%u=%lg\n",
       variable, x);
fprintf (stderr, "optimize_estimate_climbing_random: end\n");
00931
00932
00933 #endif
00934
       return x;
00935 }
00936
00940 static double
{\tt 00941\ optimize\_estimate\_climbing\_coordinates\ (unsigned\ int\ variable,}
00943
                                                unsigned int estimate)
00945 {
00946
        double x;
00947 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_estimate_climbing_coordinates: start\n");
00948
00949 #endif
00950
       x = optimize->climbing[variable];
00951
        if (estimate >= (2 * variable) && estimate < (2 * variable + 2))</pre>
00952
00953
            if (estimate & 1)
00954
              x += optimize->step[variable];
00955
            else
00956
              x -= optimize->step[variable];
00957
00958 #if DEBUG_OPTIMIZE
00959 fprintf (stderr,
00960
                  "optimize_estimate_climbing_coordinates: climbing%u=%lg\n",
       variable, x);
fprintf (stderr, "optimize_estimate_climbing_coordinates: end\n");
00961
00962
00963 #endif
```

```
return x;
00965 }
00966
00970 static inline void
00971 optimize_step_climbing (unsigned int simulation)
00972 {
00973
        GThread *thread[nthreads_climbing];
00974
       ParallelData data[nthreads_climbing];
00975 unsigned int i, j, k, b;
00976 #if DEBUG OPTIMIZE
       fprintf (stderr, "optimize_step_climbing: start\n");
00977
00978 #endif
00979
       for (i = 0; i < optimize->nestimates; ++i)
00980
00981
           k = (simulation + i) * optimize->nvariables;
00982
           b = optimize->simulation_best[0] * optimize->nvariables;
00983 #if DEBUG_OPTIMIZE
        00984
00986 #endif
00987
       for (j = 0; j < optimize->nvariables; ++j, ++k, ++b)
00988
00989 #if DEBUG_OPTIMIZE
00990
               fprintf (stderr,
00991
                         optimize_step_climbing: estimate=%u best%u=%.14le\n",
                        i, j, optimize->value[b]);
00992
00993 #endif
00994
              optimize->value[k]
                 = optimize->value[b] + optimize_estimate_climbing (j, i);
00995
              optimize->value[k] = fmin (fmax (optimize->value[k],
00996
                                                optimize->rangeminabs[j]),
00997
00998
                                          optimize->rangemaxabs[j]);
00999 #if DEBUG_OPTIMIZE
        fprintf (stderr,
01000
01001
                         "optimize_step_climbing: estimate=%u variable%u=%.14le\n",
01002
                        i, j, optimize->value[k]);
01003 #endif
01005
01006
       if (nthreads_climbing == 1)
01007
         optimize_climbing_sequential (simulation);
       else
01008
01009
        {
01010
           for (i = 0; i <= nthreads_climbing; ++i)</pre>
01011
01012
               optimize->thread_climbing[i]
01013
                 = simulation + optimize->nstart_climbing
                 + i * (optimize->nend_climbing - optimize->nstart_climbing)
01014
                 / nthreads_climbing;
01015
01016 #if DEBUG_OPTIMIZE
            fprintf (stderr,
01018
                         "optimize_step_climbing: i=%u thread_climbing=%u\n",
01019
                        i, optimize->thread_climbing[i]);
01020 #endif
01021
            for (i = 0; i < nthreads_climbing; ++i)</pre>
01022
01024
                data[i].thread = i;
01025
                thread[i] = g_thread_new
01026
                 (NULL, (GThreadFunc) optimize_climbing_thread, &data[i]);
01027
           for (i = 0; i < nthreads_climbing; ++i)</pre>
01028
01029
             g_thread_join (thread[i]);
01030
01031 #if DEBUG_OPTIMIZE
01032 fprintf (stderr, "optimize_step_climbing: end\n"); 01033 \#endif
01034 }
01035
01039 static inline void
01040 optimize_climbing ()
01041 {
01042 unsigned int i, j, k, b, s, adjust; 01043 #if DEBUG_OPTIMIZE
01044
       fprintf (stderr, "optimize_climbing: start\n");
01045 #endif
01046 for (i = 0; i < optimize->nvariables; ++i)
01047
         optimize->climbing[i] = 0.;
      b = optimize->simulation_best[0] * optimize->nvariables;
01048
       s = optimize->nsimulations;
01049
01050
       adjust = 1;
       for (i = 0; i < optimize->nsteps; ++i, s += optimize->nestimates, b = k)
01052
01053 #if DEBUG_OPTIMIZE
01054
           fprintf (stderr, "optimize_climbing: step=%u old_best=%u\n",
01055
                    i, optimize->simulation_best[0]);
01056 #endif
```

```
optimize_step_climbing (s);
            k = optimize->simulation_best[0] * optimize->nvariables;
01058
01059 #if DEBUG_OPTIMIZE
01060
           fprintf (stderr, "optimize_climbing: step=%u best=%u\n",
01061
                     i, optimize->simulation_best[0]);
01062 #endif
            if (k == b)
01063
01064
              {
01065
                if (adjust)
01066
                  for (j = 0; j < optimize->nvariables; ++j)
                optimize->step[j] *= 0.5;
for (j = 0; j < optimize->nvariables; ++j)
01067
01068
01069
                  optimize->climbing[j] = 0.;
01070
                adjust = 1;
01071
01072
            else
01073
01074
                for (j = 0; j < optimize->nvariables; ++j)
01076 #if DEBUG_OPTIMIZE
                    fprintf (stderr,
01077
01078
                               "optimize_climbing: best%u=%.14le old%u=%.14le\n",
                              j, optimize->value[k + j], j, optimize->value[b + j]);
01079
01080 #endif
01081
                    optimize->climbing[j]
01082
                      = (1. - optimize->relaxation) * optimize->climbing[j]
01083
                       + optimize->relaxation
01084
                       * (optimize->value[k + j] - optimize->value[b + j]);
01085 #if DEBUG_OPTIMIZE
                  fprintf (stderr, "optimize_climbing: climbing%u=%.14le\n",
01086
01087
                              j, optimize->climbing[j]);
01088 #endif
01089
01090
               adjust = 0;
              }
01091
01092
01093 #if DEBUG_OPTIMIZE
01094 fprintf (stderr, "optimize_climbing: end\n");
01095 #endif
01096 }
01097
01103 static double
01104 optimize_genetic_objective (Entity * entity)
01105 {
01106
        unsigned int j;
01107
        double objective;
01108
       char buffer[64];
01109 #if DEBUG OPTIMIZE
       fprintf (stderr, "optimize_genetic_objective: start\n");
01110
01111 #endif
01112
        for (j = 0; j < optimize->nvariables; ++j)
01113
01114
            optimize->value[entity->id * optimize->nvariables + j]
01115
              = genetic_get_variable (entity, optimize->genetic_variable + j);
01116
01117
        objective = optimize norm (entity->id);
        g_mutex_lock (mutex);
01118
01119
        for (j = 0; j < optimize->nvariables; ++j)
01120
            snprintf (buffer, 64, "%s ", format[optimize->precision[j]]);
fprintf (optimize->file_variables, buffer,
01121
01122
01123
                      genetic_get_variable (entity, optimize->genetic_variable + j));
01124
01125
       fprintf (optimize->file_variables, "%.14le\n", objective);
01126
        g_mutex_unlock (mutex);
01127 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_genetic_objective: end\n");
01128
01129 #endif
01130 return objective;
01131 }
01132
01136 static void
01137 optimize_genetic ()
01138 {
01139
        double *best variable = NULL;
01140
        char *best_genome = NULL;
01141
        double best_objective = 0.;
01142 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_genetic: start\n");
fprintf (stderr, "optimize_genetic: ntasks=%u nthreads=%u\n", ntasks,
01143
01144
01145
                 nthreads);
01146
       fprintf (stderr,
                  "optimize_genetic: nvariables=%u population=%u generations=%u\n",
01147
01148
                  optimize->nvariables, optimize->nsimulations, optimize->niterations);
01149
        fprintf (stderr,
01150
                  "optimize_genetic: mutation=%lg reproduction=%lg adaptation=%lg\n",
01151
                  optimize->mutation_ratio, optimize->reproduction_ratio,
```

```
optimize->adaptation_ratio);
01153 #endif
01154
        genetic_algorithm_default (optimize->nvariables,
01155
                                    optimize->genetic_variable,
01156
                                    optimize->nsimulations,
                                    optimize->niterations,
01157
01158
                                    optimize->mutation_ratio,
01159
                                    optimize->reproduction_ratio,
01160
                                    optimize->adaptation_ratio,
01161
                                    optimize->seed,
                                    optimize->threshold.
01162
                                    &optimize genetic objective.
01163
                                    &best_genome, &best_variable, &best_objective);
01164
01165 #if DEBUG_OPTIMIZE
01166
       fprintf (stderr, "optimize_genetic: the best\n");
01167 #endif
       optimize->error_old = (double *) g_malloc (sizeof (double));
01168
       optimize->value old
01169
         = (double *) g_malloc (optimize->nvariables * sizeof (double));
01170
01171
        optimize->error_old[0] = best_objective;
01172
       memcpy (optimize->value_old, best_variable,
01173
                optimize->nvariables * sizeof (double));
       g_free (best_genome);
01174
01175
       g_free (best_variable);
01176
        optimize_print ();
01177 #if DEBUG_OPTIMIZE
01178
       fprintf (stderr, "optimize_genetic: end\n");
01179 #endif
01180 }
01181
01185 static inline void
01186 optimize_save_old ()
01187 {
01188
       unsigned int i, j;
Oli89 #if DEBUG_OPTIMIZE

Oli90 fprintf (stderr, "optimize_save_old: start\n");

Oli91 fprintf (stderr, "optimize_save_old: nsaveds=%u\n", optimize->nsaveds);
01192 #endif
01193
      memcpy (optimize->error_old, optimize->error_best,
01194
                optimize->nbest * sizeof (double));
01195
        for (i = 0; i < optimize->nbest; ++i)
       j = optimize->simulation_best[i];
01196
01197
01198 #if DEBUG_OPTIMIZE
01199
            fprintf (stderr, "optimize_save_old: i=%u j=%u\n", i, j);
01200 #endif
01201 memcpy (optimize->value_old + i * optimize->nvariables,
                    optimize->value + j * optimize->nvariables,
optimize->nvariables * sizeof (double));
01202
01203
01204
01205 #if DEBUG_OPTIMIZE
01207
        fprintf (stderr, "optimize_save_old: best variable %u=%lg\n",
       i, optimize->value_old[i]);
fprintf (stderr, "optimize_save_old: end\n");
01208
01209
01210 #endif
01211 }
01212
01217 static inline void
01218 optimize_merge_old ()
01219 {
       unsigned int i, j, k;
01220
01221
       double v[optimize->nbest * optimize->nvariables], e[optimize->nbest],
01222
         *enew, *eold;
01223 #if DEBUG_OPTIMIZE
01224
       fprintf (stderr, "optimize_merge_old: start\n");
01225 #endif
01226
       enew = optimize->error best;
01227
        eold = optimize->error_old;
        i = j = k = 0;
01228
01229
       do
01230
            if (*enew < *eold)</pre>
01231
01232
               memcpy (v + k * optimize->nvariables,
01233
                        optimize->value
01234
01235
                         + optimize->simulation_best[i] * optimize->nvariables,
01236
                         optimize->nvariables * sizeof (double));
01237
               e[k] = *enew;
01238
                ++k:
01239
                ++enew;
                ++i;
01241
01242
            else
01243
             {
                memcpy (v + k * optimize->nvariables,
01244
01245
                         optimize->value_old + j * optimize->nvariables,
```

```
01246
                        optimize->nvariables * sizeof (double));
01247
                e[k] = *eold;
01248
                ++k;
01249
                ++eold;
01250
                ++j;
              }
01251
01252
        while (k < optimize->nbest);
01253
01254
       memcpy (optimize->value_old, v, k * optimize->nvariables * sizeof (double));
01255 memcpy (optimize->error_old, e, k * sizeof (double));
01256 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_merge_old: end\n");
01257
01258 #endif
01259 }
01260
01265 static inline void
01266 optimize_refine ()
01267 {
01268
       unsigned int i, j;
01269
        double d;
01270 #if HAVE_MPI
01271
       MPI_Status mpi_stat;
01272 #endif
01273 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_refine: start\n");
01275 #endif
01276 #if HAVE_MPI
01277 if (!optimize->mpi_rank)
01278
01279 #endif
01280
            for (j = 0; j < optimize->nvariables; ++j)
01281
01282
                optimize->rangemin[j] = optimize->rangemax[j]
01283
                  = optimize->value_old[j];
01284
            for (i = 0; ++i < optimize->nbest;)
01285
01286
01287
                for (j = 0; j < optimize->nvariables; ++j)
01288
01289
                    optimize->rangemin[j]
                      01290
01291
                    optimize->rangemax[j]
01292
01293
                      = fmax (optimize->rangemax[j],
01294
                              optimize->value_old[i * optimize->nvariables + j]);
01295
01296
01297
            for (j = 0; j < optimize->nvariables; ++j)
01298
01299
                d = optimize->tolerance
01300
                  * (optimize->rangemax[j] - optimize->rangemin[j]);
01301
                switch (optimize->algorithm)
01302
01303
                  case ALGORITHM_MONTE_CARLO:
01304
                   d *= 0.5;
01305
                    break;
01306
                  default:
01307
                   if (optimize->nsweeps[j] > 1)
01308
                      d /= optimize->nsweeps[j] - 1;
01309
                    else
01310
                      d = 0.:
01311
01312
                optimize->rangemin[j] -= d;
01313
                optimize->rangemin[j]
01314
                  = fmax (optimize->rangemin[j], optimize->rangeminabs[j]);
01315
                optimize->rangemax[j] += d;
01316
                optimize->rangemax[j]
                = fmin (optimize->rangemax[j], optimize->rangemaxabs[j]);
printf ("%s min=%lg max=%lg\n", optimize->label[j],
01317
01318
                        optimize->rangemin[j], optimize->rangemax[j]);
01320
                fprintf (optimize->file_result, "%s min=%lg max=%lg\n",
01321
                         optimize->label[j], optimize->rangemin[j],
01322
                         optimize->rangemax[j]);
01323
01324 #if HAVE_MPI
01325
            for (i = 1; (int) i < ntasks; ++i)</pre>
01326
01327
                MPI_Send (optimize->rangemin, optimize->nvariables, MPI_DOUBLE, i,
01328
                          1, MPI_COMM_WORLD);
                MPI_Send (optimize->rangemax, optimize->nvariables, MPI_DOUBLE, i,
01329
                          1, MPI COMM WORLD);
01330
01331
              }
01332
01333
        else
01334
            MPI_Recv (optimize->rangemin, optimize->nvariables, MPI_DOUBLE, 0, 1,
01335
01336
                      MPI COMM WORLD, &mpi stat);
```

```
MPI_Recv (optimize->rangemax, optimize->nvariables, MPI_DOUBLE, 0, 1,
01338
                      MPI_COMM_WORLD, &mpi_stat);
01339
01340 #endif
01341 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_refine: end\n");
01342
01343 #endif
01344 }
01345
01349 static void
01350 optimize_step ()
01351 {
01352 #if DEBUG_OPTIMIZE
01353
       fprintf (stderr, "optimize_step: start\n");
01354 #endif
01355    optimize_algorithm ();
01356    if (optimize->nsteps)
         optimize_climbing ();
01357
01358 #if DEBUG_OPTIMIZE
01359
       fprintf (stderr, "optimize_step: end\n");
01360 #endif
01361 }
01362
01366 static inline void
01367 optimize_iterate ()
01368 {
01369
        unsigned int i;
01370 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_iterate: start\n");
01371
01372 #endif
01373
       optimize->error_old = (double *) g_malloc (optimize->nbest * sizeof (double));
01374
        optimize->value_old =
01375
          (double *) g_malloc (optimize->nbest * optimize->nvariables *
01376
                                sizeof (double));
01377
        optimize_step ();
01378
        optimize_save_old ();
01379
        optimize_refine ();
01380
        optimize_print ();
01381
        for (i = 1; i < optimize->niterations && !optimize->stop; ++i)
01382
01383
            optimize_step ();
            optimize_merge_old ();
01384
            optimize_refine ();
01385
01386
            optimize_print ();
01387
01388 #if DEBUG_OPTIMIZE
01389 fprintf (stderr, "optimize_iterate: end\n");
01390 #endif
01391 }
01392
01396 void
01397 optimize_free ()
01398 {
01399
        unsigned int i, j;
01400 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_free: start\n");
01401
01402 #endif
01403
       for (j = 0; j < optimize->ninputs; ++j)
01404
01405
            for (i = 0; i < optimize->nexperiments; ++i)
            g_mapped_file_unref (optimize->file[j][i]);
g_free (optimize->file[j]);
01406
01407
01408
01409
       g_free (optimize->error_old);
       g_free (optimize->value_old);
01410
01411
       g_free (optimize->value);
01412    g_free (optimize->genetic_variable);
01413 #if DEBUG_OPTIMIZE
01414 fprintf (stderr, "optimize_free: end\n");
01415 #endif
01416 }
01417
01421 void
01422 optimize_open ()
01423 {
01424 GTimeZone *tz;
01425
        GDateTime *t0, *t;
01426
       unsigned int i, j;
01427
01428 #if DEBUG OPTIMIZE
01429 char *buffer;
01430 fprintf (stden
        fprintf (stderr, "optimize_open: start\n");
01431 #endif
01432
01433
        // Getting initial time
01434 #if DEBUG OPTIMIZE
01435
       fprintf (stderr, "optimize_open: getting initial time\n");
```

```
01436 #endif
       tz = g_time_zone_new_utc ();
01437
01438
       t0 = g_date_time_new_now (tz);
01439
01440
       \ensuremath{//} Obtaining and initing the pseudo-random numbers generator seed
01441 #if DEBUG_OPTIMIZE
01442
       fprintf (stderr, "optimize_open: getting initial seed\n");
01443 #endif
01444
       if (optimize->seed == DEFAULT_RANDOM_SEED)
01445
         optimize->seed = input->seed;
       gsl_rng_set (optimize->rng, optimize->seed);
01446
01447
01448
        // Replacing the working directory
01449 #if DEBUG_OPTIMIZE
01450
       fprintf (stderr, "optimize_open: replacing the working directory\n");
01451 #endif
       q_chdir (input->directory);
01452
01453
01454
       // Getting results file names
01455
       optimize->result = input->result;
01456
       optimize->variables = input->variables;
01457
01458
       // Obtaining the simulator file
01459
       optimize->simulator = input->simulator;
01460
01461
       // Obtaining the evaluator file
01462
       optimize->evaluator = input->evaluator;
01463
01464
        // Reading the algorithm
01465
       optimize->algorithm = input->algorithm;
01466
       switch (optimize->algorithm)
01467
01468
         case ALGORITHM_MONTE_CARLO:
01469
           optimize_algorithm = optimize_MonteCarlo;
01470
           break;
          case ALGORITHM SWEEP:
01471
          optimize_algorithm = optimize_sweep;
break;
01472
         case ALGORITHM_ORTHOGONAL:
01474
01475
          optimize_algorithm = optimize_orthogonal;
01476
           break;
01477
         default:
01478
           optimize algorithm = optimize genetic;
01479
            optimize->mutation_ratio = input->mutation_ratio;
            optimize->reproduction_ratio = input->reproduction_ratio;
01480
01481
            optimize->adaptation_ratio = input->adaptation_ratio;
01482
01483
       optimize->nvariables = input->nvariables;
       optimize->nsimulations = input->nsimulations;
01484
       optimize->niterations = input->niterations;
01485
       optimize->nbest = input->nbest;
01486
01487
       optimize->tolerance = input->tolerance;
01488
       optimize->nsteps = input->nsteps;
       optimize > nestimates = 0;
optimize -> threshold = input -> threshold;
01489
01490
01491
       optimize->stop = 0;
01492
       if (input->nsteps)
01493
         {
01494
            optimize->relaxation = input->relaxation;
01495
            switch (input->climbing)
01496
             {
01497
              case CLIMBING METHOD COORDINATES:
01498
               optimize->nestimates = 2 * optimize->nvariables;
                optimize_estimate_climbing = optimize_estimate_climbing_coordinates;
01499
01500
                break;
              default:
01501
01502
               optimize->nestimates = input->nestimates;
01503
               optimize_estimate_climbing = optimize_estimate_climbing_random;
01504
         }
01506
01507 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: nbest=%u\n", optimize->nbest);
01508
01509 #endif
01510
       optimize->simulation best
01511
          = (unsigned int *) alloca (optimize->nbest * sizeof (unsigned int));
       optimize->error_best = (double *) alloca (optimize->nbest * sizeof (double));
01512
01513
01514
       // Reading the experimental data
01515 #if DEBUG_OPTIMIZE
01516
       buffer = g_get_current_dir ();
       fprintf (stderr, "optimize_open: current directory=%s\n", buffer);
01517
01518
        g_free (buffer);
01519 #endif
01520
       optimize->nexperiments = input->nexperiments;
01521
       optimize->ninputs = input->experiment->ninputs;
01522
       optimize->experiment
```

```
= (char **) alloca (input->nexperiments * sizeof (char *));
        optimize->weight = (double *) alloca (input->nexperiments * sizeof (double));
01524
01525
        for (i = 0; i < input->experiment->ninputs; ++i)
         optimize->file[i] = (GMappedFile **)
01526
        g_malloc (input->nexperiments * sizeof (GMappedFile *));
for (i = 0; i < input->nexperiments; ++i)
01527
01528
01529
01530 #if DEBUG_OPTIMIZE
            fprintf (stderr, "optimize_open: i=%u\n", i);
01531
01532 #endif
            optimize->experiment[i] = input->experiment[i].name;
01533
01534
            optimize->weight[i] = input->experiment[i].weight;
01535 #if DEBUG_OPTIMIZE
01536
            fprintf (stderr, "optimize_open: experiment=%s weight=%lg\n",
01537
                     optimize->experiment[i], optimize->weight[i]);
01538 #endif
01539
            for (j = 0; j < input->experiment->ninputs; ++j)
01540
01541 #if DEBUG_OPTIMIZE
                fprintf (stderr, "optimize_open: stencil%u\n", j + 1);
01543 #endif
01544
                optimize->file[j][i]
01545
                  = g_mapped_file_new (input->experiment[i].stencil[j], 0, NULL);
01546
             }
01547
         }
01548
01549
        // Reading the variables data
01550 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: reading variables\n");
01551
01552 #endif
01553
       optimize->label = (char **) alloca (input->nvariables * sizeof (char *));
01554
        j = input->nvariables * sizeof (double);
01555
        optimize->rangemin = (double *) alloca (j);
01556
        optimize->rangeminabs = (double *) alloca (j);
01557
        optimize->rangemax = (double *) alloca (j);
        optimize->rangemaxabs = (double *) alloca (j);
01558
        optimize->step = (double *) alloca (j);
01559
        j = input->nvariables * sizeof (unsigned int);
01560
01561
        optimize->precision = (unsigned int *) alloca (j);
01562
        optimize->nsweeps = (unsigned int *) alloca (j);
01563
        optimize->nbits = (unsigned int *) alloca (j);
        for (i = 0; i < input->nvariables; ++i)
01564
01565
01566
            optimize->label[i] = input->variable[i].name;
            optimize->rangemin[i] = input->variable[i].rangemin;
01567
01568
            optimize->rangeminabs[i] = input->variable[i].rangeminabs;
01569
            optimize->rangemax[i] = input->variable[i].rangemax;
01570
            optimize->rangemaxabs[i] = input->variable[i].rangemaxabs;
            optimize->precision[i] = input->variable[i].precision;
01571
01572
            optimize->step[i] = input->variable[i].step;
            optimize->nsweeps[i] = input->variable[i].nsweeps;
01574
            optimize->nbits[i] = input->variable[i].nbits;
01575
01576
        if (input->algorithm == ALGORITHM_SWEEP
01577
            || input->algorithm == ALGORITHM_ORTHOGONAL)
01578
         {
            optimize->nsimulations = 1;
            for (i = 0; i < input->nvariables; ++i)
01580
01581
01582
                optimize->nsimulations *= optimize->nsweeps[i];
01583 #if DEBUG_OPTIMIZE
                fprintf (stderr, "optimize_open: nsweeps=%u nsimulations=%u\n",
01584
01585
                         optimize->nsweeps[i], optimize->nsimulations);
01586 #endif
01587
01588
01589
        if (optimize->nsteps)
         optimize->climbing
01590
01591
            = (double *) alloca (optimize->nvariables * sizeof (double));
01593
01594
        switch (input->norm)
01595
          case ERROR NORM EUCLIDIAN:
01596
01597
           optimize norm = optimize norm euclidian;
01598
           break;
01599
          case ERROR_NORM_MAXIMUM:
01600
          optimize_norm = optimize_norm_maximum;
01601
           break:
          case ERROR NORM P:
01602
           optimize_norm = optimize_norm_p;
01603
            optimize->p = input->p;
01604
01605
01606
          default:
01607
           optimize_norm = optimize_norm_taxicab;
01608
01609
```

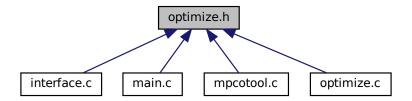
```
// Allocating values
01611 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: allocating variables\n");
fprintf (stderr, "optimize_open: nvariables=%u algorithm=%u\n",
01612
01613
01614
                optimize->nvariables, optimize->algorithm);
01615 #endif
01616
       optimize->genetic_variable = NULL;
          (optimize->algorithm == ALGORITHM_GENETIC)
01617
01618
           optimize->genetic_variable = (GeneticVariable *)
01619
             g_malloc (optimize->nvariables * sizeof (GeneticVariable));
01620
            for (i = 0; i < optimize->nvariables; ++i)
01621
01622
01623 #if DEBUG_OPTIMIZE
01624
               fprintf (stderr, "optimize_open: i=%u min=%lg max=%lg nbits=%u\n",
                         i, optimize->rangemin[i], optimize->rangemax[i],
01625
                         optimize->nbits[i]);
01626
01627 #endif
01628
               optimize->genetic_variable[i].minimum = optimize->rangemin[i];
               optimize->genetic_variable[i].maximum = optimize->rangemax[i];
01629
               optimize->genetic_variable[i].nbits = optimize->nbits[i];
01630
01631
01632
01633 #if DEBUG_OPTIMIZE
01634 fprintf (stderr, "optimize_open: nvariables=%u nsimulations=%u\n",
                optimize->nvariables, optimize->nsimulations);
01635
01636 #endif
01637 optimize->value = (double *)
01638
         g_malloc ((optimize->nsimulations
01639
                     + optimize->nestimates * optimize->nsteps)
                    * optimize->nvariables * sizeof (double));
01640
01641
01642
       // Calculating simulations to perform for each task
01643 #if HAVE_MPI
01644 #if DEBUG_OPTIMIZE
01645 fprintf (stderr, "optimize_open: rank=%u ntasks=%u\n",
                optimize->mpi_rank, ntasks);
01646
01647 #endif
01648
       optimize->nstart = optimize->mpi_rank * optimize->nsimulations / ntasks;
01649
        optimize->nend = (1 + optimize->mpi_rank) * optimize->nsimulations / ntasks;
01650
       if (optimize->nsteps)
01651
            optimize->nstart climbing
01652
01653
              = optimize->mpi_rank * optimize->nestimates / ntasks;
            optimize->nend_climbing
01654
01655
              = (1 + optimize->mpi_rank) * optimize->nestimates / ntasks;
01656
01657 #else
01658
       optimize->nstart = 0;
01659
       optimize->nend = optimize->nsimulations;
01660
       if (optimize->nsteps)
01661
01662
            optimize->nstart_climbing = 0;
01663
           optimize->nend_climbing = optimize->nestimates;
01664
01665 #endif
01666 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: nstart=%u nend=%u\n", optimize->nstart,
01667
01668
                optimize->nend);
01669 #endif
01670
01671
        // Calculating simulations to perform for each thread
01672
       optimize->thread
         = (unsigned int *) alloca ((1 + nthreads) * sizeof (unsigned int));
01673
01674
       for (i = 0; i <= nthreads; ++i)</pre>
01675
01676
           optimize->thread[i] = optimize->nstart
              + i * (optimize->nend - optimize->nstart) / nthreads;
01677
01678 #if DEBUG_OPTIMIZE
           01680
01681 #endif
01682
       if (optimize->nsteps)
01683
         optimize->thread_climbing = (unsigned int *)
01684
            alloca ((1 + nthreads_climbing) * sizeof (unsigned int));
01685
01686
01687
       // Opening result files
       optimize->file_result = g_fopen (optimize->result, "w");
01688
       optimize->file_variables = g_fopen (optimize->variables, "w");
01689
01690
01691
        // Performing the algorithm
01692
       switch (optimize->algorithm)
01693
01694
           // Genetic algorithm
01695
         case ALGORITHM GENETIC:
01696
           optimize genetic ():
```

```
break;
01698
01699
             // Iterative algorithm
          default:
01700
01701
            optimize_iterate ();
01702
01703
01704
        // Getting calculation time
01705
        t = g_date_time_new_now (tz);
        optimize->calculation_time = 0.000001 * g_date_time_difference (t, t0);
01706
01707
        g_date_time_unref (t);
01708
        g_date_time_unref (t0);
01709
        g_time_zone_unref (tz);
        printf ("%s = %.61g s\n", _("Calculation time"), optimize->calculation_time); fprintf (optimize->file_result, "%s = %.61g s\n",
01710
01711
01712
01713
                  _("Calculation time"), optimize->calculation_time);
01714
        // Closing result files
fclose (optimize->file_variables);
01716
        fclose (optimize->file_result);
01717
01718 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: end\n");
01719
01720 #endif
01721 }
```

5.23 optimize.h File Reference

Header file to define the optimization functions.

This graph shows which files directly or indirectly include this file:



Data Structures

• struct Optimize

Struct to define the optimization ation data.

struct ParallelData

Struct to pass to the GThreads parallelized function.

Functions

- · void optimize_free ()
- void optimize_open ()

Variables

- · int ntasks
- · unsigned int nthreads
- · unsigned int nthreads_climbing

Number of threads for the hill climbing method.

- GMutex mutex [1]
- Optimize optimize [1]

Optimization data.

5.23.1 Detailed Description

Header file to define the optimization functions.

Authors

Javier Burguete.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file optimize.h.

5.23.2 Function Documentation

5.23.2.1 optimize_free()

```
void optimize_free ( )
```

Function to free the memory used by the Optimize struct.

Definition at line 1397 of file optimize.c.

```
01398 {
        unsigned int i, j;
01400 #if DEBUG_OPTIMIZE
01401
        fprintf (stderr, "optimize_free: start\n");
01402 #endif
01403
        for (j = 0; j < optimize->ninputs; ++j)
01404
01405
             for (i = 0; i < optimize->nexperiments; ++i)
01406
              g_mapped_file_unref (optimize->file[j][i]);
01407
             g_free (optimize->file[j]);
01408
01409 g_free (optimize->error_old);
01410 g_free (optimize->value_old);
01411 g_free (optimize->value);
01412 g_free (optimize->genetic_variable);
01413 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_free: end\n");
01414
01415 #endif
01416 }
```

5.23.2.2 optimize_open()

```
void optimize_open ( )
```

Function to open and perform a optimization.

Definition at line 1422 of file optimize.c.

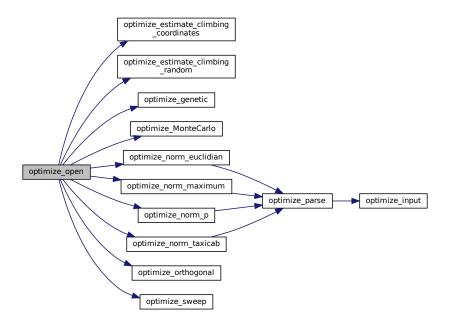
```
01423 {
01424
        GTimeZone *tz;
01425
        GDateTime *t0, *t;
       unsigned int i, j;
01426
01427
01428 #if DEBUG_OPTIMIZE
      char *buffer;
01429
01430
       fprintf (stderr, "optimize_open: start\n");
01431 #endif
01432
01433
        // Getting initial time
01434 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: getting initial time\n");
01435
01436 #endif
01437
       tz = g_time_zone_new_utc ();
01438
       t0 = g_date_time_new_now (tz);
01439
01440
        // Obtaining and initing the pseudo-random numbers generator seed
01441 #if DEBUG_OPTIMIZE
01442
       fprintf (stderr, "optimize_open: getting initial seed\n");
01443 #endif
01444
       if (optimize->seed == DEFAULT_RANDOM_SEED)
       optimize->seed = input->seed;
gsl_rng_set (optimize->rng, optimize->seed);
01445
01446
01447
01448
        // Replacing the working directory
01449 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: replacing the working directory\n");
01450
01451 #endif
        g_chdir (input->directory);
01452
01453
01454
        // Getting results file names
01455
        optimize->result = input->result;
01456
       optimize->variables = input->variables;
01457
01458
       // Obtaining the simulator file
01459
        optimize->simulator = input->simulator;
01460
01461
        // Obtaining the evaluator file
01462
        optimize->evaluator = input->evaluator;
01463
01464
        // Reading the algorithm
        optimize->algorithm = input->algorithm;
01465
01466
        switch (optimize->algorithm)
01467
01468
          case ALGORITHM_MONTE_CARLO:
           optimize_algorithm = optimize_MonteCarlo;
01469
01470
           break:
01471
         case ALGORITHM_SWEEP:
           optimize_algorithm = optimize_sweep;
01473
           break;
01474
          case ALGORITHM_ORTHOGONAL:
01475
          optimize_algorithm = optimize_orthogonal;
01476
           break:
01477
          default:
01478
           optimize_algorithm = optimize_genetic;
01479
            optimize->mutation_ratio = input->mutation_ratio;
01480
            optimize->reproduction_ratio = input->reproduction_ratio;
01481
            optimize->adaptation_ratio = input->adaptation_ratio;
01482
        optimize->nvariables = input->nvariables;
01483
01484
        optimize->nsimulations = input->nsimulations;
        optimize->niterations = input->niterations;
01485
01486
        optimize->nbest = input->nbest;
01487
        optimize->tolerance = input->tolerance;
        optimize->nsteps = input->nsteps;
01488
01489
        optimize->nestimates = 0;
        optimize->threshold = input->threshold;
01490
01491
        optimize->stop = 0;
01492
        if (input->nsteps)
01493
01494
            optimize->relaxation = input->relaxation;
            switch (input->climbing)
01495
01496
01497
              case CLIMBING_METHOD_COORDINATES:
                optimize->nestimates = 2 * optimize->nvariables;
01498
```

```
optimize_estimate_climbing = optimize_estimate_climbing_coordinates;
01500
                break;
01501
              default:
01502
               optimize->nestimates = input->nestimates;
01503
                optimize_estimate_climbing = optimize_estimate_climbing_random;
              }
01504
01505
          }
01506
01507 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: nbest=%u\n", optimize->nbest);
01508
01509 #endif
       optimize->simulation_best
01510
          = (unsigned int *) alloca (optimize->nbest * sizeof (unsigned int));
01511
01512
        optimize->error_best = (double *) alloca (optimize->nbest * sizeof (double));
01513
01514
       // Reading the experimental data
01515 #if DEBUG_OPTIMIZE
       buffer = g_get_current_dir ();
01516
       fprintf (stderr, "optimize_open: current directory=%s\n", buffer);
01517
01518
        g_free (buffer);
01519 #endif
01520
       optimize->nexperiments = input->nexperiments;
01521
        optimize->ninputs = input->experiment->ninputs;
        optimize->experiment
01522
01523
          = (char **) alloca (input->nexperiments * sizeof (char *));
        optimize->weight = (double *) alloca (input->nexperiments * sizeof (double));
01524
01525
        for (i = 0; i < input->experiment->ninputs; ++i)
01526
        optimize->file[i] = (GMappedFile **)
01527
            g_malloc (input->nexperiments * sizeof (GMappedFile *));
01528
       for (i = 0; i < input->nexperiments; ++i)
01529
01530 #if DEBUG_OPTIMIZE
01531
            fprintf (stderr, "optimize_open: i=%u\n", i);
01532 #endif
01533
            optimize->experiment[i] = input->experiment[i].name;
01534
            optimize->weight[i] = input->experiment[i].weight;
01535 #if DEBUG_OPTIMIZE
            fprintf (stderr, "optimize_open: experiment=%s weight=%lg\n",
01537
                     optimize->experiment[i], optimize->weight[i]);
01538 #endif
01539
            for (j = 0; j < input->experiment->ninputs; ++j)
01540
01541 #if DEBUG_OPTIMIZE
01542
                fprintf (stderr, "optimize_open: stencil%u\n", j + 1);
01543 #endif
01544
               optimize->file[j][i]
01545
                  = g_mapped_file_new (input->experiment[i].stencil[j], 0, NULL);
01546
              }
         }
01547
01548
        // Reading the variables data
01550 #if DEBUG_OPTIMIZE
01551
       fprintf (stderr, "optimize_open: reading variables\n");
01552 #endif
       optimize->label = (char **) alloca (input->nvariables * sizeof (char *));
01553
01554
        j = input->nvariables * sizeof (double);
        optimize->rangemin = (double *) alloca (j);
        optimize->rangeminabs = (double *) alloca (j);
01556
01557
        optimize->rangemax = (double *) alloca (j);
01558
        optimize->rangemaxabs = (double *) alloca (j);
        optimize->step = (double *) alloca (j);
j = input->nvariables * sizeof (unsigned int);
01559
01560
        optimize->precision = (unsigned int *) alloca (j);
optimize->nsweeps = (unsigned int *) alloca (j);
01561
01562
01563
        optimize->nbits = (unsigned int *) alloca (j);
01564
        for (i = 0; i < input->nvariables; ++i)
01565
01566
            optimize->label[i] = input->variable[i].name;
01567
            optimize->rangemin[i] = input->variable[i].rangemin;
01568
            optimize->rangeminabs[i] = input->variable[i].rangeminabs;
01569
            optimize->rangemax[i] = input->variable[i].rangemax;
01570
            optimize->rangemaxabs[i] = input->variable[i].rangemaxabs;
            optimize->precision[i] = input->variable[i].precision;
01571
            optimize->step[i] = input->variable[i].step;
optimize->nsweeps[i] = input->variable[i].nsweeps;
01572
01573
01574
            optimize->nbits[i] = input->variable[i].nbits;
01575
01576
        if (input->algorithm == ALGORITHM_SWEEP
01577
            || input->algorithm == ALGORITHM_ORTHOGONAL)
01578
          {
01579
            optimize->nsimulations = 1;
            for (i = 0; i < input->nvariables; ++i)
01581
01582
                optimize->nsimulations *= optimize->nsweeps[i];
01583 #if DEBUG_OPTIMIZE
                fprintf (stderr, "optimize_open: nsweeps=%u nsimulations=%un",
01584
01585
                          optimize->nsweeps[i], optimize->nsimulations);
```

```
01586 #endif
01587
01588
01589
        if (optimize->nsteps)
01590
         optimize->climbing
01591
            = (double *) alloca (optimize->nvariables * sizeof (double));
01592
01593
        // Setting error norm
01594
        switch (input->norm)
01595
         case ERROR NORM EUCLIDIAN:
01596
01597
           optimize norm = optimize norm euclidian;
01598
            break;
01599
          case ERROR_NORM_MAXIMUM:
          optimize_norm = optimize_norm_maximum;
01600
01601
           break;
          case ERROR_NORM_P:
01602
          optimize_norm = optimize_norm_p;
optimize->p = input->p;
01603
01604
01605
            break;
01606
          default:
01607
            optimize_norm = optimize_norm_taxicab;
         }
01608
01609
        // Allocating values
01610
01611 #if DEBUG_OPTIMIZE
01612 fprintf (stderr, "optimize_open: allocating variables\n"); 01613 fprintf (stderr, "optimize_open: nvariables=\n"u algorithm=\n"u,",
01614
                 optimize->nvariables, optimize->algorithm);
01615 #endif
01616    optimize->genetic_variable = NULL;
01617
        if (optimize->algorithm == ALGORITHM_GENETIC)
01618
01619
            optimize->genetic_variable = (GeneticVariable *)
            g_malloc (optimize->nvariables * sizeof (GeneticVariable));
for (i = 0; i < optimize->nvariables; ++i)
01620
01621
01622
01623 #if DEBUG_OPTIMIZE
01624
         fprintf (stderr, "optimize_open: i=%u min=%lg max=%lg nbits=%u\n",
01625
                          i, optimize->rangemin[i], optimize->rangemax[i],
01626
                          optimize->nbits[i]);
01627 #endif
01628
                optimize->genetic variable[i].minimum = optimize->rangemin[i];
                optimize->genetic_variable[i].maximum = optimize->rangemax[i];
01629
                optimize->genetic_variable[i].nbits = optimize->nbits[i];
01630
01631
01632
01633 #if DEBUG_OPTIMIZE
01634 fprintf (stderr, "optimize_open: nvariables=%u nsimulations=%u\n",
                 optimize->nvariables, optimize->nsimulations);
01635
01636 #endif
01637 optimize->value = (double *)
        g_malloc ((optimize->nsimulations
01638
                     + optimize->nestimates * optimize->nsteps)
* optimize->nvariables * sizeof (double));
01639
01640
01641
01642
        // Calculating simulations to perform for each task
01643 #if HAVE_MPI
01644 #if DEBUG_OPTIMIZE
01645 fprintf (stderr, "optimize_open: rank=%u ntasks=%u\n",
                 optimize->mpi_rank, ntasks);
01646
01647 #endif
01648
       optimize->nstart = optimize->mpi_rank * optimize->nsimulations / ntasks;
        optimize->nend = (1 + optimize->mpi_rank) * optimize->nsimulations / ntasks;
01649
01650
        if (optimize->nsteps)
01651
01652
            optimize->nstart_climbing
01653
              = optimize->mpi_rank * optimize->nestimates / ntasks;
01654
            optimize->nend_climbing
               = (1 + optimize->mpi_rank) * optimize->nestimates / ntasks;
01655
01656
01657 #else
01658
       optimize->nstart = 0;
        optimize->nend = optimize->nsimulations;
01659
        if (optimize->nsteps)
01660
01661
01662
            optimize->nstart_climbing = 0;
01663
            optimize->nend_climbing = optimize->nestimates;
01664
01665 #endif
01666 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: nstart=%u nend=%u\n", optimize->nstart,
01667
                 optimize->nend);
01668
01669 #endif
01670
        // Calculating simulations to perform for each thread
01671
01672
       optimize->thread
```

```
= (unsigned int *) alloca ((1 + nthreads) * sizeof (unsigned int));
01674
        for (i = 0; i <= nthreads; ++i)</pre>
01675
01676
            optimize->thread[i] = optimize->nstart
01677
               + i * (optimize->nend - optimize->nstart) / nthreads;
01678 #if DEBUG_OPTIMIZE
01679
            fprintf (stderr, "optimize_open: i=%u thread=%u\n", i,
01680
                      optimize->thread[i]);
01681 #endif
01682
        if (optimize->nsteps)
01683
         optimize->thread_climbing = (unsigned int *)
01684
01685
            alloca ((1 + nthreads_climbing) * sizeof (unsigned int));
01686
01687
        // Opening result files
01688
        optimize->file_result = g_fopen (optimize->result, "w");
        optimize->file_variables = g_fopen (optimize->variables, "w");
01689
01690
01691
        // Performing the algorithm
01692
        switch (optimize->algorithm)
01693
01694
            // Genetic algorithm
          case ALGORITHM_GENETIC:
01695
01696
           optimize_genetic ();
01697
           break;
01698
            // Iterative algorithm
01699
01700
          optimize_iterate ();
}
          default:
01701
01702
01703
01704
        // Getting calculation time
01705
        t = g_date_time_new_now (tz);
01706
        {\tt optimize->calculation\_time} \ = \ 0.000001 \ \star \ {\tt g\_date\_time\_difference} \ \ ({\tt t, t0}) \ ;
        g_date_time_unref (t);
g_date_time_unref (t0);
01707
01708
01709
        g_time_zone_unref (tz);
01710
        printf ("%s = %.61g s\n", _("Calculation time"), optimize->calculation_time);
01711
        fprintf (optimize->file_result, "%s = %.6lg s\n",
01712
                 _("Calculation time"), optimize->calculation_time);
01713
        // Closing result files
01714
       fclose (optimize->file_variables);
fclose (optimize->file_result);
01715
01716
01717
01718 #if DEBUG_OPTIMIZE
01719
       fprintf (stderr, "optimize_open: end\n");
01720 #endif
01721 }
```

Here is the call graph for this function:



5.23.3 Variable Documentation

5.23.3.1 nthreads_climbing

```
unsigned int nthreads_climbing [extern]
```

Number of threads for the hill climbing method.

Definition at line 80 of file optimize.c.

5.23.3.2 optimize

```
Optimize optimize[1] [extern]
```

Optimization data.

Definition at line 79 of file optimize.c.

5.24 optimize.h

Go to the documentation of this file.

```
00001 /*
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00005
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are \text{met}:
00012
00013 1. Redistributions of source code must retain the above copyright notice,
00014 this list of conditions and the following disclaimer.
          Redistributions in binary form must reproduce the above copyright notice,
00017 this list of conditions and the following disclaimer in the
00018 documentation and/or other materials provided with the distribution.
00019
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, 00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
00031
00038 #ifndef OPTIMIZE__H
00039 #define OPTIMIZE__H 1
00040
00045 typedef struct
00046 {
        GMappedFile **file[MAX_NINPUTS];
00047
       char **experiment;
char **label;
00048
00049
00050
       gsl_rng *rng;
```

5.25 tools.c File Reference 309

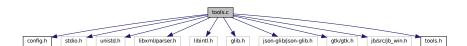
```
GeneticVariable *genetic_variable;
00053
        FILE *file_result;
00054
        FILE *file_variables;
00055
        char *result;
00056
        char *variables:
00057
        char *simulator:
00058
        char *evaluator;
00060
        double *value;
00061
        double *rangemin;
00062
        double *rangemax;
00063
        double *rangeminabs;
       double *rangemaxabs;
00064
00065
        double *error_best;
00066
        double *weight;
00067
        double *step;
00068
        double *climbing;
       double *value_old;
00069
00071
        double *error old;
00073
       unsigned int *precision;
00074
        unsigned int *nsweeps;
00075
        unsigned int *nbits;
00077
        unsigned int *thread;
       unsigned int *thread_climbing;
unsigned int *simulation_best;
00079
00082
00083
       double tolerance;
00084
       double mutation_ratio;
00085
        double reproduction_ratio;
00086
        double adaptation_ratio;
00087
        double relaxation;
       double calculation_time;
00088
       double p;
double threshold;
00089
00090
00091
        unsigned long int seed;
00093
        unsigned int nvariables;
00094
       unsigned int nexperiments;
00095
       unsigned int ninputs;
00096
       unsigned int nsimulations;
00097
       unsigned int nsteps;
00099
       unsigned int nestimates;
00101
       unsigned int algorithm;
00102
       unsigned int nstart;
00103
       unsigned int nend;
00104
       unsigned int nstart climbing;
       unsigned int nend_climbing;
00106
00108
       unsigned int niterations;
00109
       unsigned int nbest;
00110
       unsigned int nsaveds;
00111
        unsigned int stop;
00112 #if HAVE_MPI
00113 int mpi_rank;
00114 #endif
00115 } Optimize;
00116
00121 typedef struct
00122 {
00123
       unsigned int thread;
00124 } ParallelData;
00125
00126 // Global variables
00127 extern int ntasks;
00128 extern unsigned int nthreads;
00129 extern unsigned int nthreads_climbing;
00130 extern GMutex mutex[1];
00131 extern Optimize optimize[1];
00132
00133 // Public functions
00134 void optimize_free ();
00135 void optimize_open ();
00136
00137 #endif
```

5.25 tools.c File Reference

Source file to define some useful functions.

```
#include "config.h"
#include <stdio.h>
#include <unistd.h>
#include <libxml/parser.h>
```

```
#include <libintl.h>
#include <glib.h>
#include <json-glib/json-glib.h>
#include <gtk/gtk.h>
#include "jb/src/jb_win.h"
#include "tools.h"
Include dependency graph for tools.c:
```



Variables

• GtkWindow * main_window

Main GtkWindow.

• char * error_message

Error message.

void(* show_pending)() = NULL

Pointer to the function to show pending events.

5.25.1 Detailed Description

Source file to define some useful functions.

Authors

Javier Burguete and Borja Latorre.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file tools.c.

5.25.2 Variable Documentation

5.25.2.1 error_message

char* error_message

Error message.

Definition at line 59 of file tools.c.

5.26 tools.c 311

5.25.2.2 main_window

```
GtkWindow* main window
```

Main GtkWindow.

Definition at line 56 of file tools.c.

5.25.2.3 show_pending

```
void(* show_pending) () ( ) = NULL
```

Pointer to the function to show pending events.

Definition at line 60 of file tools.c.

5.26 tools.c

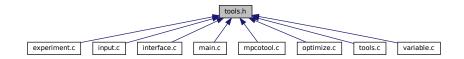
Go to the documentation of this file.

```
00001 /
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00005
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00009
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are met:
00012
00013 1. Redistributions of source code must retain the above copyright notice,
00014 this list of conditions and the following disclaimer.
00015
          Redistributions in binary form must reproduce the above copyright notice,
00017 this list of conditions and the following disclaimer in the
00018 documentation and/or other materials provided with the distribution.
00019
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF 00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN 00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
00031
00038 #define _GNU_SOURCE
00039 #include "config.h"
00040 #include <stdio.h>
00041 #include <unistd.h>
00042 #include <libxml/parser.h>
00043 #include <libintl.h>
00044 #include <glib.h>
00045 #include <json-glib/json-glib.h>
00046 #ifdef G_OS_WIN32
00047 #include <windows.h>
00048 #endif
00049 #if HAVE_GTK
00050 #include <gtk/gtk.h>
00051 #endif
00052 #include "jb/src/jb_win.h"
00053 #include "tools.h
00054
00055 #if HAVE_GTK
00056 GtkWindow *main_window;
00057 #endif
00058
00059 char *error_message;
00060 void (*show_pending) () = NULL;
```

5.27 tools.h File Reference

Header file to define some useful functions.

This graph shows which files directly or indirectly include this file:



Macros

- #define ERROR_TYPE GTK_MESSAGE_ERROR
 - Macro to define the error message type.
- #define INFO_TYPE GTK_MESSAGE_INFO

Macro to define the information message type.

Variables

- GtkWindow * main_window
 - Main GtkWindow.
- GtkWindow * window_parent
- char * error_message

Error message.

void(* show_pending)()

Pointer to the function to show pending events.

5.27.1 Detailed Description

Header file to define some useful functions.

Authors

Javier Burguete.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file tools.h.

5.27.2 Macro Definition Documentation

5.27 tools.h File Reference 313

5.27.2.1 ERROR_TYPE

```
#define ERROR_TYPE GTK_MESSAGE_ERROR
```

Macro to define the error message type.

Definition at line 48 of file tools.h.

5.27.2.2 INFO_TYPE

```
#define INFO_TYPE GTK_MESSAGE_INFO
```

Macro to define the information message type.

Definition at line 49 of file tools.h.

5.27.3 Variable Documentation

5.27.3.1 error_message

```
char* error_message [extern]
```

Error message.

Definition at line 59 of file tools.c.

5.27.3.2 main_window

```
GtkWindow* main_window [extern]
```

Main GtkWindow.

Definition at line 56 of file tools.c.

5.27.3.3 show_pending

```
void(* show_pending) () ( ) [extern]
```

Pointer to the function to show pending events.

Definition at line 60 of file tools.c.

5.28 tools.h

Go to the documentation of this file.

```
00001 /3
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00009
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are met:
00013 1. Redistributions of source code must retain the above copyright notice,
00014 this list of conditions and the following disclaimer.
00015
00016 2. Redistributions in binary form must reproduce the above copyright notice, 00017 this list of conditions and the following disclaimer in the \,
00018 documentation and/or other materials provided with the distribution.
00019
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF 00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN 00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING 00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
00031
00038 #ifndef TOOLS__H
00039 #define TOOLS__H 1
00040
00047 #if HAVE_GTK
00048 #define ERROR_TYPE GTK_MESSAGE_ERROR
00049 #define INFO_TYPE GTK_MESSAGE_INFO
00050 extern GtkWindow *main_window;
00051 extern GtkWindow *window_parent;
00052 #else
00053 #define ERROR_TYPE 0
00054 #define INFO_TYPE 0
00055 #endif
00056
00057 // Public functions
00058
00059 extern char *error_message;
00060 extern void (*show_pending) ();
00061
00062 #endif
```

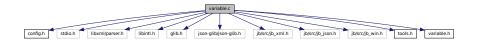
5.29 variable.c File Reference

Source file to define the variable data.

```
#include "config.h"
#include <stdio.h>
#include <libxml/parser.h>
#include <libintl.h>
#include <glib.h>
#include <json-glib/json-glib.h>
#include "jb/src/jb_xml.h"
#include "jb/src/jb_json.h"
#include "jb/src/jb_win.h"
#include "tools.h"
```

#include "variable.h"

Include dependency graph for variable.c:



Macros

• #define DEBUG_VARIABLE 0

Macro to debug variable functions.

Functions

- void variable_free (Variable *variable, unsigned int type)
- void variable_error (Variable *variable, char *message)
- int variable_open_xml (Variable *variable, xmlNode *node, unsigned int algorithm, unsigned int nsteps)
- int variable_open_json (Variable *variable, JsonNode *node, unsigned int algorithm, unsigned int nsteps)

Variables

• const char * format [NPRECISIONS]

Array of C-strings with variable formats.

const double precision [NPRECISIONS]

Array of variable precisions.

5.29.1 Detailed Description

Source file to define the variable data.

Authors

Javier Burguete and Borja Latorre.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file variable.c.

5.29.2 Macro Definition Documentation

5.29.2.1 DEBUG_VARIABLE

```
#define DEBUG_VARIABLE 0
```

Macro to debug variable functions.

Definition at line 51 of file variable.c.

5.29.3 Function Documentation

5.29.3.1 variable_error()

Function to print a message error opening an Variable struct.

Parameters

variable	Variable struct.
message	Error message.

Definition at line 88 of file variable.c.

```
00092 {
00093     char buffer[64];
00094     if (!variable->name)
00095          snprintf (buffer, 64, "%s: %s", _("Variable"), message);
00096     else
00097          snprintf (buffer, 64, "%s %s: %s", _("Variable"), variable->name, message);
00098     error_message = g_strdup (buffer);
00099 }
```

5.29.3.2 variable_free()

Function to free the memory of a Variable struct.

Parameters

variable	Variable struct.
type	Type of input file.

Definition at line 67 of file variable.c.

5.29.3.3 variable open json()

Function to open the variable file.

Returns

1 on success, 0 on error.

Parameters

variable	Variable struct.
node	XML node.
algorithm	Algorithm type.
nsteps	Number of steps to do the hill climbing method.

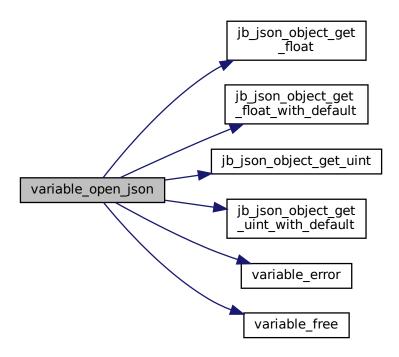
Definition at line 270 of file variable.c.

```
00275 {
00276
        JsonObject *object;
00277
        const char *label;
00278
00278 int error_code;
00279 #if DEBUG_VARIABLE
00280
        fprintf (stderr, "variable_open_json: start\n");
00281 #endif
00282
        object = json_node_get_object (node);
        label = json_object_get_string_member (object, LABEL_NAME);
00283
00284
        if (!label)
00285
00286
            variable_error (variable, _("no name"));
00287
            goto exit_on_error;
00288
00289
        variable -> name = g_strdup (label);
00290
        if (json_object_get_member (object, LABEL_MINIMUM))
00291
         {
00292
            variable->rangemin
00293
               = jb_json_object_get_float (object, LABEL_MINIMUM, &error_code);
00294
            if (!error_code)
00295
                variable_error (variable, _("bad minimum"));
goto exit_on_error;
00296
00297
00298
00299
            variable->rangeminabs
00300
              = jb_json_object_get_float_with_default (object, LABEL_ABSOLUTE_MINIMUM,
00301
                                                          &error_code, -G_MAXDOUBLE);
00302
            if (!error_code)
00303
00304
                variable_error (variable, _("bad absolute minimum"));
00305
                goto exit_on_error;
```

```
00307
            if (variable->rangemin < variable->rangeminabs)
00308
00309
               variable_error (variable, _("minimum range not allowed"));
00310
               goto exit_on_error;
00311
00312
00313
        else
00314
00315
            variable_error (variable, _("no minimum range"));
00316
           goto exit_on_error;
00317
00318
        if (json_object_get_member (object, LABEL_MAXIMUM))
00319
00320
            variable->rangemax
00321
              = jb_json_object_get_float (object, LABEL_MAXIMUM, &error_code);
            if (!error_code)
00322
00323
            {
00324
               variable_error (variable, _("bad maximum"));
00325
               goto exit_on_error;
00326
00327
            variable->rangemaxabs
              = jb_json_object_get_float_with_default (object, LABEL_ABSOLUTE_MAXIMUM,
00328
00329
                                                       &error code, G MAXDOUBLE);
00330
            if (!error_code)
00331
00332
                variable_error (variable, _("bad absolute maximum"));
00333
                goto exit_on_error;
00334
00335
            if (variable->rangemax > variable->rangemaxabs)
00336
00337
                variable_error (variable, _("maximum range not allowed"));
00338
               goto exit_on_error;
00339
00340
            if (variable->rangemax < variable->rangemin)
00341
00342
               variable_error (variable, _("bad range"));
00343
               goto exit_on_error;
00344
00345
00346
       else
        {
00347
           variable_error (variable, _("no maximum range"));
00348
00349
           goto exit_on_error;
00350
00351
        variable->precision
00352
         = jb_json_object_get_uint_with_default (object, LABEL_PRECISION,
00353
                                                   &error_code, DEFAULT_PRECISION);
        if (!error code || variable->precision >= NPRECISIONS)
00354
00355
00356
            variable_error (variable, _("bad precision"));
00357
            goto exit_on_error;
00358
00359
        if (algorithm == ALGORITHM_SWEEP || algorithm == ALGORITHM_ORTHOGONAL)
00360
00361
            if (json_object_get_member (object, LABEL_NSWEEPS))
00363
00364
                  = jb_json_object_get_uint (object, LABEL_NSWEEPS, &error_code);
00365
                if (!error_code || !variable->nsweeps)
00366
00367
                   variable_error (variable, _("bad sweeps"));
00368
                   goto exit_on_error;
00369
00370
             }
00371
            else
00372
             {
00373
               variable_error (variable, _("no sweeps number"));
00374
               goto exit_on_error;
00375
00376 #if DEBUG_VARIABLE
00377
            fprintf (stderr, "variable_open_json: nsweeps=%u\n", variable->nsweeps);
00378 #endif
00379
00380
        if (algorithm == ALGORITHM_GENETIC)
00381
00382
            // Obtaining bits representing each variable
00383
            if (json_object_get_member (object, LABEL_NBITS))
00384
00385
               variable->nbits
00386
                 = jb_json_object_get_uint (object, LABEL_NBITS, &error_code);
00387
                if (!error_code || !variable->nbits)
00388
00389
                    variable_error (variable, _("invalid bits number"));
00390
                    goto exit_on_error;
00391
00392
              }
```

```
00393
              else
00394
               {
                   variable_error (variable, _("no bits number"));
00395
00396
                   goto exit_on_error;
00397
00398
00399
         else if (nsteps)
00400
               variable->step
00401
               = jb_json_object_get_float (object, LABEL_STEP, &error_code);
if (!error_code || variable->step < 0.)</pre>
00402
00403
00404
                   variable_error (variable, _("bad step size"));
00405
00406
                   goto exit_on_error;
00407
00408
         }
00409
00410 #if DEBUG_VARIABLE
00411 fprintf (stderr, "variable_open_json: end\n");
00412 #endif
00413
         return 1;
00414 exit_on_error:
00415 variable_free (variable, INPUT_TYPE_JSON);
00416 #if DEBUG_VARIABLE
00417 fprintf (stderr, "variable_open_json: end\n");
00418 #endif
00419
         return 0;
00420 }
```

Here is the call graph for this function:



5.29.3.4 variable_open_xml()

```
xmlNode * node,
unsigned int algorithm,
unsigned int nsteps )
```

Function to open the variable file.

Returns

1 on success, 0 on error.

Parameters

variable	Variable struct.
node	XML node.
algorithm	Algorithm type.
nsteps	Number of steps to do the hill climbing method.

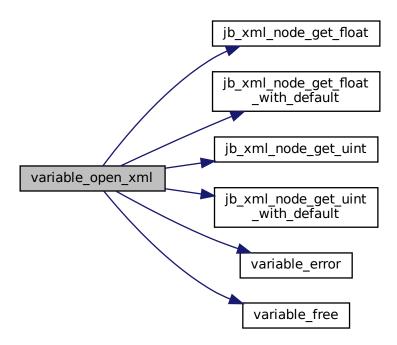
Definition at line 107 of file variable.c.

```
00113
        int error_code;
00114
00115 #if DEBUG_VARIABLE
        fprintf (stderr, "variable_open_xml: start\n");
00116
00117 #endif
00118
00119
        variable->name = (char *) xmlGetProp (node, (const xmlChar *) LABEL_NAME);
00120
        if (!variable->name)
00121
          {
             variable_error (variable, _("no name"));
00122
00123
            goto exit_on_error;
00124
00125
        if (xmlHasProp (node, (const xmlChar *) LABEL_MINIMUM))
00126
00127
            variable->rangemin
               = jb_xml_node_get_float (node, (const xmlChar *) LABEL_MINIMUM,
00128
00129
                                          &error code):
00130
            if (!error_code)
00131
00132
                 variable_error (variable, _("bad minimum"));
00133
                 goto exit_on_error;
00134
00135
            variable->rangeminabs = jb_xml_node_get_float_with_default
  (node, (const xmlChar *) LABEL_ABSOLUTE_MINIMUM, &error_code,
00136
00137
                -G_MAXDOUBLE);
00138
00139
            if (!error_code)
00140
              {
00141
                 variable_error (variable, _("bad absolute minimum"));
00142
                 goto exit_on_error;
00143
00144
             if (variable->rangemin < variable->rangeminabs)
00145
                variable_error (variable, _("minimum range not allowed"));
00146
00147
                goto exit_on_error;
00148
00149
00150
00151
00152
            variable_error (variable, _("no minimum range"));
00153
            goto exit_on_error;
00154
00155
        if (xmlHasProp (node, (const xmlChar *) LABEL_MAXIMUM))
00156
00157
            variable->rangemax
00158
               = jb_xml_node_get_float (node, (const xmlChar *) LABEL_MAXIMUM,
00159
                                          &error_code);
00160
            if (!error_code)
00161
              {
00162
                 variable_error (variable, _("bad maximum"));
00163
                 goto exit_on_error;
00164
            variable->rangemaxabs = jb_xml_node_get_float_with_default
  (node, (const xmlChar *) LABEL_ABSOLUTE_MAXIMUM, &error_code,
00165
00166
00167
                G_MAXDOUBLE);
00168
             if (!error_code)
```

```
00170
                variable_error (variable, _("bad absolute maximum"));
00171
                goto exit_on_error;
00172
00173
            if (variable->rangemax > variable->rangemaxabs)
00174
             {
00175
                variable_error (variable, _("maximum range not allowed"));
00176
                goto exit_on_error;
00177
00178
            if (variable->rangemax < variable->rangemin)
00179
             {
               variable_error (variable, _("bad range"));
00180
00181
               goto exit_on_error;
00182
00183
00184
        else
00185
00186
            variable_error (variable, _("no maximum range"));
00187
           goto exit_on_error;
00188
00189
       variable->precision
00190
          = jb_xml_node_get_uint_with_default (node,
                                                (const xmlChar *) LABEL PRECISION.
00191
00192
                                                &error code, DEFAULT PRECISION);
00193
        if (!error_code || variable->precision >= NPRECISIONS)
00194
00195
            variable_error (variable, _("bad precision"));
00196
            goto exit_on_error;
00197
00198
        if (algorithm == ALGORITHM SWEEP || algorithm == ALGORITHM ORTHOGONAL)
00199
00200
            if (xmlHasProp (node, (const xmlChar *) LABEL_NSWEEPS))
00201
00202
               variable->nsweeps
00203
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSWEEPS,
00204
                                           &error_code);
               if (!error_code || !variable->nsweeps)
00205
00207
                    variable_error (variable, _("bad sweeps"));
00208
                    goto exit_on_error;
00209
00210
00211
           else
00212
             {
00213
               variable_error (variable, _("no sweeps number"));
00214
               goto exit_on_error;
00215
00216 #if DEBUG_VARIABLE
            fprintf (stderr, "variable_open_xml: nsweeps=%u\n", variable->nsweeps);
00217
00218 #endif
00219
00220
           (algorithm == ALGORITHM_GENETIC)
00221
         {
00222
            \ensuremath{//} Obtaining bits representing each variable
            if (xmlHasProp (node, (const xmlChar *) LABEL_NBITS))
00223
00224
             {
                variable->nbits
00226
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NBITS,
00227
                                           &error_code);
00228
               if (!error_code || !variable->nbits)
00229
                 {
00230
                   variable_error (variable, _("invalid bits number"));
00231
                    goto exit_on_error;
00232
00233
             }
00234
            else
00235
             {
               variable_error (variable, _("no bits number"));
00236
00237
               goto exit on error;
             }
00238
00239
00240
       else if (nsteps)
00241
00242
           variable->step
              = jb_xml_node_get_float (node, (const xmlChar *) LABEL_STEP,
00243
00244
                                        &error_code);
00245
            if (!error_code || variable->step < 0.)</pre>
00246
00247
               variable_error (variable, _("bad step size"));
               goto exit_on_error;
00248
00249
00250
         }
00251
00252 #if DEBUG_VARIABLE
       fprintf (stderr, "variable_open_xml: end\n");
00253
00254 #endif
00255
       return 1:
```

```
00256 exit_on_error:
00257    variable_free (variable, INPUT_TYPE_XML);
00258 #if DEBUG_VARIABLE
00259    fprintf (stderr, "variable_open_xml: end\n");
00260 #endif
00261    return 0;
00262 }
```

Here is the call graph for this function:



5.29.4 Variable Documentation

5.29.4.1 format

```
const char* format[NPRECISIONS]
```

Initial value:

```
= {
  "%.01f", "%.11f", "%.21f", "%.31f", "%.41f", "%.51f", "%.61f", "%.71f",
  "%.81f", "%.91f", "%.101f", "%.111f", "%.121f", "%.131f", "%.141f"
}
```

Array of C-strings with variable formats.

Definition at line 53 of file variable.c.

5.30 variable.c 323

5.29.4.2 precision

```
const double precision[NPRECISIONS]
```

Initial value:

```
= {
    1., 0.1, 0.01, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7, 1e-8, 1e-9, 1e-10, 1e-11,
    1e-12, 1e-13, 1e-14
}
```

Array of variable precisions.

Definition at line 58 of file variable.c.

5.30 variable.c

Go to the documentation of this file.

```
00001 /*
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00005
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00009
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are met:
00012
00013 1.
          Redistributions of source code must retain the above copyright notice.
00014 this list of conditions and the following disclaimer.
           Redistributions in binary form must reproduce the above copyright notice,
00017 this list of conditions and the following disclaimer in the
00018 documentation and/or other materials provided with the distribution.
00019
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, 00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
00031
00038 #define _GNU_SOURCE
00039 #include "config.h"
00040 #include <stdio.h>
00041 #include <libxml/parser.h>
00042 #include <libintl.h>
00043 #include <glib.h>
00044 #include <json-glib/json-glib.h>
00045 #include "jb/src/jb_xml.h
00046 #include "jb/src/jb_json.h"
00047 #include "jb/src/jb_win.h"
00048 #include "tools.h"
00049 #include "variable.h"
00050
00051 #define DEBUG_VARIABLE 0
00052
00053 const char *format[NPRECISIONS] = {
00054    "%.01f", "%.11f", "%.21f", "%.31f", "%.41f", "%.51f", "%.61f", "%.71f",
00055    "%.81f", "%.91f", "%.101f", "%.111f", "%.121f", "%.131f", "%.141f"
00056 };
00057
00058 const double precision[NPRECISIONS] = {
00059 1., 0.1, 0.01, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7, 1e-8, 1e-9, 1e-10, 1e-11,
00060
        1e-12, 1e-13, 1e-14
00061 };
00062
00066 void
00067 variable_free (Variable * variable,
00069
                       unsigned int type)
```

```
00071 {
00072 #if DEBUG_VARIABLE
        fprintf (stderr, "variable_free: start\n");
00073
00074 #endif
00075 if (type == INPUT_TYPE_XML)
00076
         xmlFree (variable->name);
00077
       else
00078 g_free (variable->name);
00079 #if DEBUG_VARIABLE
00080 fprintf (stderr, "variable_free: end\n");
00081 #endif
00082 }
00083
00087 void
00088 variable_error (Variable * variable,
00090
                      char *message)
00092 {
00093
        char buffer[64];
       if (!variable->name)
         snprintf (buffer, 64, "%s: %s", _("Variable"), message);
00095
00096
         snprintf (buffer, 64, "%s %s: %s", _("Variable"), variable->name, message);
00097
00098 error_message = g_strdup (buffer);
00099 }
00100
00106 int
00107 variable_open_xml (Variable * variable,
00108
                          xmlNode * node,
00109
                          unsigned int algorithm,
00110
                          unsigned int nsteps)
00112 {
00113
        int error_code;
00114
00115 #if DEBUG_VARIABLE
        fprintf (stderr, "variable_open_xml: start\n");
00116
00117 #endif
00118
00119
        variable->name = (char *) xmlGetProp (node, (const xmlChar *) LABEL_NAME);
00120
        if (!variable->name)
00121
00122
            variable_error (variable, _("no name"));
00123
            goto exit_on_error;
00124
00125
        if (xmlHasProp (node, (const xmlChar *) LABEL_MINIMUM))
00126
            variable->rangemin
00127
00128
              = jb_xml_node_get_float (node, (const xmlChar *) LABEL_MINIMUM,
00129
                                         &error_code);
00130
            if (!error code)
00131
00132
                variable_error (variable, _("bad minimum"));
00133
                goto exit_on_error;
00134
            variable->rangeminabs = jb_xml_node_get_float_with_default
  (node, (const xmlChar *) LABEL_ABSOLUTE_MINIMUM, &error_code,
00135
00136
               -G_MAXDOUBLE);
00137
00139
            if (!error_code)
00140
00141
                variable_error (variable, _("bad absolute minimum"));
00142
                goto exit_on_error;
00143
00144
            if (variable->rangemin < variable->rangeminabs)
00145
00146
                variable_error (variable, _("minimum range not allowed"));
00147
                goto exit_on_error;
00148
00149
          }
00150
       else
00151
00152
            variable_error (variable, _("no minimum range"));
00153
            goto exit_on_error;
00154
        if (xmlHasProp (node, (const xmlChar *) LABEL MAXIMUM))
00155
00156
00157
            variable->rangemax
00158
              = jb_xml_node_get_float (node, (const xmlChar *) LABEL_MAXIMUM,
00159
                                        &error_code);
00160
            if (!error_code)
00161
             {
                variable_error (variable, _("bad maximum"));
00162
00163
                goto exit_on_error;
00164
00165
            variable->rangemaxabs = jb_xml_node_get_float_with_default
00166
             (node, (const xmlChar *) LABEL_ABSOLUTE_MAXIMUM, &error_code,
               G MAXDOUBLE):
00167
            if (!error_code)
00168
```

5.30 variable.c 325

```
{
00170
                variable_error (variable, _("bad absolute maximum"));
00171
                goto exit_on_error;
00172
00173
            if (variable->rangemax > variable->rangemaxabs)
00174
              {
00175
                variable_error (variable, _("maximum range not allowed"));
00176
                goto exit_on_error;
00177
00178
            if (variable->rangemax < variable->rangemin)
00179
             {
               variable_error (variable, _("bad range"));
00180
00181
               goto exit_on_error;
00182
00183
00184
        else
00185
00186
            variable_error (variable, _("no maximum range"));
00187
           goto exit_on_error;
00188
00189
        variable->precision
00190
          = jb_xml_node_get_uint_with_default (node,
                                                (const xmlChar *) LABEL PRECISION.
00191
00192
                                                &error code, DEFAULT PRECISION);
00193
        if (!error_code || variable->precision >= NPRECISIONS)
00194
00195
            variable_error (variable, _("bad precision"));
00196
            goto exit_on_error;
00197
00198
        if (algorithm == ALGORITHM SWEEP || algorithm == ALGORITHM ORTHOGONAL)
00199
00200
            if (xmlHasProp (node, (const xmlChar *) LABEL_NSWEEPS))
00201
00202
                variable->nsweeps
00203
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSWEEPS,
00204
                                           &error_code);
                if (!error_code || !variable->nsweeps)
00205
00207
                    variable_error (variable, _("bad sweeps"));
00208
                    goto exit_on_error;
00209
00210
00211
           else
00212
              {
00213
                variable_error (variable, _("no sweeps number"));
00214
                goto exit_on_error;
00215
00216 #if DEBUG_VARIABLE
            fprintf (stderr, "variable_open_xml: nsweeps=%u\n", variable->nsweeps);
00217
00218 #endif
00219
00220
           (algorithm == ALGORITHM_GENETIC)
00221
         {
00222
            \ensuremath{//} Obtaining bits representing each variable
            if (xmlHasProp (node, (const xmlChar *) LABEL_NBITS))
00223
00224
              {
00225
                variable->nbits
00226
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NBITS,
00227
                                           &error_code);
00228
                if (!error_code || !variable->nbits)
00229
                 {
00230
                   variable_error (variable, _("invalid bits number"));
00231
                    goto exit_on_error;
00232
00233
              }
00234
            else
00235
             {
                variable_error (variable, _("no bits number"));
00236
00237
               goto exit_on_error;
              }
00238
00239
00240
        else if (nsteps)
00241
00242
           variable->step
              = jb_xml_node_get_float (node, (const xmlChar *) LABEL_STEP,
00243
00244
                                        &error_code);
00245
            if (!error_code || variable->step < 0.)</pre>
00246
00247
                variable_error (variable, _("bad step size"));
               goto exit_on_error;
00248
00249
00250
         }
00251
00252 #if DEBUG_VARIABLE
       fprintf (stderr, "variable_open_xml: end\n");
00253
00254 #endif
00255
       return 1:
```

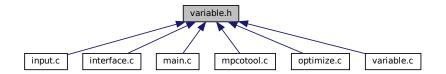
```
00256 exit_on_error:
       variable_free (variable, INPUT_TYPE_XML);
00258 #if DEBUG_VARIABLE
       fprintf (stderr, "variable_open_xml: end\n");
00259
00260 #endif
00261
       return 0:
00262 }
00263
00269 int
00270 variable_open_json (Variable * variable,
00271
                          JsonNode * node,
00272
                          unsigned int algorithm,
00273
                          unsigned int nsteps)
00275 {
00276
       JsonObject *object;
00277
       const char *label;
00278 int error_code;
00279 #if DEBUG_VARIABLE
       fprintf (stderr, "variable_open_json: start\n");
00281 #endif
00282
       object = json_node_get_object (node);
00283
        label = json_object_get_string_member (object, LABEL_NAME);
00284
       if (!label)
00285
00286
            variable_error (variable, _("no name"));
00287
           goto exit_on_error;
00288
00289
        variable->name = g_strdup (label);
00290
        if (json_object_get_member (object, LABEL_MINIMUM))
00291
        {
00292
           variable->rangemin
00293
              = jb_json_object_get_float (object, LABEL_MINIMUM, &error_code);
00294
            if (!error_code)
00295
             {
00296
               variable_error (variable, _("bad minimum"));
00297
               goto exit_on_error;
00298
00299
            variable->rangeminabs
00300
             = jb_json_object_get_float_with_default (object, LABEL_ABSOLUTE_MINIMUM,
00301
                                                        &error_code, -G_MAXDOUBLE);
00302
            if (!error_code)
00303
             {
00304
               variable_error (variable, _("bad absolute minimum"));
00305
                goto exit_on_error;
00306
00307
            if (variable->rangemin < variable->rangeminabs)
00308
             {
00309
               variable_error (variable, _("minimum range not allowed"));
               goto exit_on_error;
00310
00311
00312
          }
00313
00314
00315
            variable_error (variable, _("no minimum range"));
00316
            goto exit_on_error;
00317
00318
       if (json_object_get_member (object, LABEL_MAXIMUM))
00319
        {
00320
           variable->rangemax
00321
             = jb_json_object_get_float (object, LABEL_MAXIMUM, &error_code);
00322
            if (!error_code)
00323
             {
00324
               variable_error (variable, _("bad maximum"));
00325
               goto exit_on_error;
00326
00327
           variable->rangemaxabs
              = jb_json_object_get_float_with_default (object, LABEL_ABSOLUTE_MAXIMUM,
00328
00329
                                                        &error code, G MAXDOUBLE);
00330
            if (!error code)
00331
             {
00332
                variable_error (variable, _("bad absolute maximum"));
00333
                goto exit_on_error;
00334
00335
            if (variable->rangemax > variable->rangemaxabs)
00336
             {
00337
               variable_error (variable, _("maximum range not allowed"));
00338
               goto exit_on_error;
00339
00340
            if (variable->rangemax < variable->rangemin)
00341
             {
00342
               variable_error (variable, _("bad range"));
00343
               goto exit_on_error;
00344
00345
         }
00346
       else
00347
00348
            variable error (variable, ("no maximum range"));
```

```
goto exit_on_error;
00350
00351
        variable->precision
          = jb_json_object_get_uint_with_default (object, LABEL_PRECISION,
00352
00353
                                                   &error_code, DEFAULT_PRECISION);
        if (!error_code || variable->precision >= NPRECISIONS)
00354
00356
            variable_error (variable, _("bad precision"));
00357
            goto exit_on_error;
00358
00359
        if (algorithm == ALGORITHM_SWEEP || algorithm == ALGORITHM_ORTHOGONAL)
00360
00361
            if (json_object_get_member (object, LABEL_NSWEEPS))
00362
00363
                variable->nsweeps
00364
                  = jb_json_object_get_uint (object, LABEL_NSWEEPS, &error_code);
                if (!error_code || !variable->nsweeps)
00365
00366
                 {
00367
                   variable_error (variable, _("bad sweeps"));
00368
                    goto exit_on_error;
00369
00370
           else
00371
00372
             {
00373
                variable_error (variable, _("no sweeps number"));
00374
                goto exit_on_error;
00375
00376 #if DEBUG_VARIABLE
            fprintf (stderr, "variable_open_json: nsweeps=%u\n", variable->nsweeps);
00377
00378 #endif
00379
00380
        if
           (algorithm == ALGORITHM_GENETIC)
00381
00382
            \ensuremath{//} Obtaining bits representing each variable
00383
            if (json_object_get_member (object, LABEL_NBITS))
00384
00385
                variable->nbits
00386
                  = jb_json_object_get_uint (object, LABEL_NBITS, &error_code);
00387
                if (!error_code || !variable->nbits)
00388
00389
                    variable_error (variable, _("invalid bits number"));
00390
                   goto exit_on_error;
00391
00392
              }
00393
            else
00394
00395
                variable_error (variable, _("no bits number"));
00396
                goto exit_on_error;
              }
00397
00398
          }
00399
       else if (nsteps)
00400
00401
            variable->step
00402
              = jb_json_object_get_float (object, LABEL_STEP, &error_code);
            if (!error_code || variable->step < 0.)</pre>
00403
             {
00404
               variable_error (variable, _("bad step size"));
00406
                goto exit_on_error;
00407
00408
         }
00409
00410 #if DEBUG_VARIABLE
00411
       fprintf (stderr, "variable_open_json: end\n");
00412 #endif
00413
       return 1;
00414 exit_on_error:
00415 variable_free (variable, INPUT_TYPE_JSON); 00416 #if DEBUG_VARIABLE
00417 fprintf (stderr, "variable_open_json: end\n");
00418 #endif
00419
       return 0;
00420 }
```

5.31 variable.h File Reference

Header file to define the variable data.

This graph shows which files directly or indirectly include this file:



Data Structures

· struct Variable

Struct to define the variable data.

Enumerations

enum Algorithm { ALGORITHM_MONTE_CARLO = 0 , ALGORITHM_SWEEP = 1 , ALGORITHM_GENETIC = 2 , ALGORITHM ORTHOGONAL = 3 }

Enum to define the algorithms.

Functions

- void variable_free (Variable *variable, unsigned int type)
- void variable_error (Variable *variable, char *message)
- int variable_open_xml (Variable *variable, xmlNode *node, unsigned int algorithm, unsigned int nsteps)
- int variable_open_json (Variable *variable, JsonNode *node, unsigned int algorithm, unsigned int nsteps)

Variables

const char * format [NPRECISIONS]

Array of C-strings with variable formats.

• const double precision [NPRECISIONS]

Array of variable precisions.

5.31.1 Detailed Description

Header file to define the variable data.

Authors

Javier Burguete.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file variable.h.

5.31.2 Enumeration Type Documentation

5.31.2.1 Algorithm

```
enum Algorithm
```

Enum to define the algorithms.

Enumerator

ALGORITHM_MONTE_CARLO	Monte-Carlo algorithm.
ALGORITHM_SWEEP	Sweep algorithm.
ALGORITHM_GENETIC	Genetic algorithm.
ALGORITHM_ORTHOGONAL	Orthogonal sampling algorithm.

Definition at line 42 of file variable.h.

```
00043 {
00044 ALGORITHM_MONTE_CARLO = 0,
00045 ALGORITHM_SWEEP = 1,
00046 ALGORITHM_GENETIC = 2,
00047 ALGORITHM_ORTHOGONAL = 3
00048 };
```

5.31.3 Function Documentation

5.31.3.1 variable error()

Function to print a message error opening an Variable struct.

Parameters

variable	Variable struct.
message	Error message.

Definition at line 88 of file variable.c.

5.31.3.2 variable free()

Function to free the memory of a Variable struct.

Parameters

variable	Variable struct.
type	Type of input file.

Definition at line 67 of file variable.c.

```
00071 {
00072 #if DEBUG_VARIABLE
00073     fprintf (stderr, "variable_free: start\n");
00074 #endif
00075     if (type == INPUT_TYPE_XML)
00076          xmlFree (variable->name);
00077     else
00078          g_free (variable->name);
00079 #if DEBUG_VARIABLE
00080     fprintf (stderr, "variable_free: end\n");
00081 #endif
00082 }
```

5.31.3.3 variable_open_json()

Function to open the variable file.

Returns

1 on success, 0 on error.

Parameters

variable	Variable struct.
node	XML node.
algorithm	Algorithm type.
nsteps	Number of steps to do the hill climbing method.

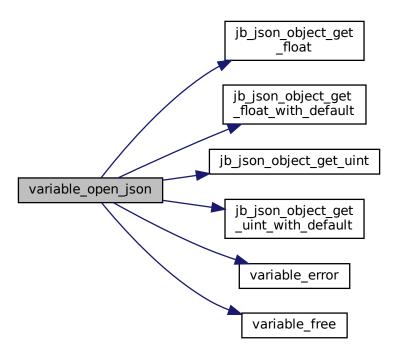
Definition at line 270 of file variable.c.

```
00275 {
00276  JsonObject *object;
```

```
00277
       const char *label;
00278
       int error_code;
00279 #if DEBUG_VARIABLE
00280
       fprintf (stderr, "variable_open_json: start\n");
00281 #endif
00282
        object = json_node_get_object (node);
        label = json_object_get_string_member (object, LABEL_NAME);
00284
        if (!label)
00285
00286
            variable_error (variable, _("no name"));
00287
           goto exit_on_error;
00288
00289
        variable->name = g strdup (label);
00290
       if (json_object_get_member (object, LABEL_MINIMUM))
00291
00292
            variable->rangemin
00293
             = jb_json_object_get_float (object, LABEL_MINIMUM, &error_code);
00294
            if (!error_code)
00295
00296
               variable_error (variable, _("bad minimum"));
00297
               goto exit_on_error;
00298
00299
            variable->rangeminabs
              = jb_json_object_get_float_with_default (object, LABEL_ABSOLUTE_MINIMUM,
00300
00301
                                                       &error_code, -G_MAXDOUBLE);
00302
            if (!error_code)
00303
00304
               variable_error (variable, _("bad absolute minimum"));
00305
                goto exit_on_error;
00306
00307
            if (variable->rangemin < variable->rangeminabs)
00308
             {
00309
               variable_error (variable, _("minimum range not allowed"));
00310
               goto exit_on_error;
00311
00312
         }
00313
       else
00314
00315
            variable_error (variable, _("no minimum range"));
00316
            goto exit_on_error;
00317
       if (json_object_get_member (object, LABEL_MAXIMUM))
00318
00319
00320
            variable->rangemax
00321
              = jb_json_object_get_float (object, LABEL_MAXIMUM, &error_code);
            if (!error_code)
00322
00323
00324
               variable_error (variable, _("bad maximum"));
00325
               goto exit_on_error;
00326
00327
            variable->rangemaxabs
00328
             = jb_json_object_get_float_with_default (object, LABEL_ABSOLUTE_MAXIMUM,
00329
                                                        &error_code, G_MAXDOUBLE);
00330
            if (!error_code)
00331
               variable_error (variable, _("bad absolute maximum"));
00332
               goto exit_on_error;
00334
00335
              (variable->rangemax > variable->rangemaxabs)
00336
00337
               variable_error (variable, _("maximum range not allowed"));
00338
               goto exit_on_error;
00339
00340
            if (variable->rangemax < variable->rangemin)
00341
00342
               variable_error (variable, _("bad range"));
00343
               goto exit_on_error;
              }
00344
00345
          }
00346
       else
00347
00348
            variable_error (variable, _("no maximum range"));
00349
           goto exit_on_error;
00350
00351
       variable->precision
         = jb_json_object_get_uint_with_default (object, LABEL_PRECISION,
00352
00353
                                                   &error_code, DEFAULT_PRECISION);
00354
        if (!error_code || variable->precision >= NPRECISIONS)
00355
00356
            variable_error (variable, _("bad precision"));
00357
            goto exit_on_error;
00358
00359
           (algorithm == ALGORITHM_SWEEP || algorithm == ALGORITHM_ORTHOGONAL)
00360
00361
            if (json_object_get_member (object, LABEL_NSWEEPS))
00362
00363
                variable->nsweeps
```

```
= jb_json_object_get_uint (object, LABEL_NSWEEPS, &error_code);
00365
                if (!error_code || !variable->nsweeps)
00366
                    variable_error (variable, _("bad sweeps"));
00367
00368
                    goto exit_on_error;
                  }
00369
00370
00371
            else
00372
00373
                variable_error (variable, _("no sweeps number"));
00374
                goto exit_on_error;
00375
00376 #if DEBUG_VARIABLE
00377
            fprintf (stderr, "variable_open_json: nsweeps=%u\n", variable->nsweeps);
00378 #endif
00379
           (algorithm == ALGORITHM_GENETIC)
00380
00381
         {
            // Obtaining bits representing each variable
00382
00383
            if (json_object_get_member (object, LABEL_NBITS))
00384
00385
                variable->nbits
                  = jb_json_object_get_uint (object, LABEL_NBITS, &error_code);
00386
00387
                if (!error_code || !variable->nbits)
00388
                  {
00389
                   variable_error (variable, _("invalid bits number"));
00390
                    goto exit_on_error;
                  }
00391
00392
              }
00393
            else
00394
              {
00395
                variable_error (variable, _("no bits number"));
00396
                goto exit_on_error;
00397
00398
        else if (nsteps)
00399
00400
        {
            variable->step
00402
              = jb_json_object_get_float (object, LABEL_STEP, &error_code);
00403
            if (!error_code || variable->step < 0.)</pre>
00404
                variable_error (variable, _("bad step size"));
goto exit_on_error;
00405
00406
              }
00407
00408
         }
00409
00410 #if DEBUG_VARIABLE 00411 fprintf (stderr, "variable_open_json: end\n");
00412 #endif
00413
       return 1;
00414 exit_on_error:
00415 variable_free (variable, INPUT_TYPE_JSON);
00416 #if DEBUG_VARIABLE
00417
       fprintf (stderr, "variable_open_json: end\n");
00418 #endif
00419
       return 0;
```

Here is the call graph for this function:



5.31.3.4 variable_open_xml()

Function to open the variable file.

Returns

1 on success, 0 on error.

Parameters

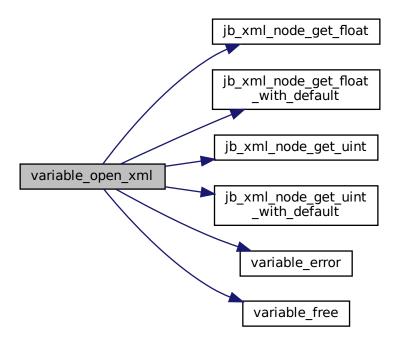
variable	Variable struct.
node	XML node.
algorithm	Algorithm type.
nsteps	Number of steps to do the hill climbing method.

Definition at line 107 of file variable.c.

```
00113
        int error_code;
00114
00115 #if DEBUG VARIABLE
       fprintf (stderr, "variable_open_xml: start\n");
00116
00117 #endif
00118
00119
        variable->name = (char *) xmlGetProp (node, (const xmlChar *) LABEL_NAME);
00120
        if (!variable->name)
00121
00122
            variable_error (variable, _("no name"));
00123
            goto exit on error;
00124
00125
        if (xmlHasProp (node, (const xmlChar *) LABEL_MINIMUM))
00126
00127
            variable->rangemin
              = jb_xml_node_get_float (node, (const xmlChar *) LABEL_MINIMUM,
00128
                                       &error_code);
00130
            if (!error_code)
00131
00132
               variable_error (variable, _("bad minimum"));
00133
               goto exit_on_error;
00134
00135
            variable->rangeminabs = jb_xml_node_get_float_with_default
00136
             (node, (const xmlChar *) LABEL_ABSOLUTE_MINIMUM, &error_code,
00137
               -G_MAXDOUBLE);
00138
00139
            if (!error_code)
00140
            {
00141
               variable_error (variable, _("bad absolute minimum"));
00142
               goto exit_on_error;
00143
00144
            if (variable->rangemin < variable->rangeminabs)
00145
               variable_error (variable, _("minimum range not allowed"));
00146
               goto exit_on_error;
00147
00149
00150
       else
00151
            variable_error (variable, _("no minimum range"));
00152
00153
            goto exit_on_error;
00154
00155
        if (xmlHasProp (node, (const xmlChar *) LABEL_MAXIMUM))
00156
00157
            variable->rangemax
00158
              = jb_xml_node_get_float (node, (const xmlChar *) LABEL_MAXIMUM,
00159
                                       &error_code);
00160
            if (!error code)
00161
             {
00162
                variable_error (variable, _("bad maximum"));
00163
               goto exit_on_error;
00164
            variable->rangemaxabs = jb_xml_node_get_float_with_default
00165
             (node, (const xmlChar *) LABEL_ABSOLUTE_MAXIMUM, &error_code,
00166
               G_MAXDOUBLE);
00168
            if (!error_code)
00169
00170
                variable_error (variable, _("bad absolute maximum"));
00171
               goto exit_on_error;
00172
00173
            if (variable->rangemax > variable->rangemaxabs)
00174
00175
                variable_error (variable, _("maximum range not allowed"));
00176
               goto exit_on_error;
00177
00178
            if (variable->rangemax < variable->rangemin)
00179
00180
               variable_error (variable, _("bad range"));
00181
                goto exit_on_error;
00182
00183
00184
        else
00185
00186
            variable_error (variable, _("no maximum range"));
00187
           goto exit_on_error;
00188
00189
        variable->precision
          = jb_xml_node_get_uint_with_default (node,
00190
                                                (const xmlChar *) LABEL_PRECISION,
00191
00192
                                                &error_code, DEFAULT_PRECISION);
00193
        if (!error_code || variable->precision >= NPRECISIONS)
00194
00195
            variable_error (variable, _("bad precision"));
00196
            goto exit_on_error;
00197
          }
```

```
if (algorithm == ALGORITHM_SWEEP || algorithm == ALGORITHM_ORTHOGONAL)
00199
00200
            if (xmlHasProp (node, (const xmlChar *) LABEL_NSWEEPS))
00201
                variable->nsweeps
00202
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSWEEPS, &error_code);
00203
00205
                if (!error_code || !variable->nsweeps)
00206
00207
                    variable_error (variable, _("bad sweeps"));
00208
                   goto exit_on_error;
00209
00210
00211
00212
             {
00213
                variable_error (variable, _("no sweeps number"));
00214
                goto exit_on_error;
00215
00216 #if DEBUG_VARIABLE
00217
            fprintf (stderr, "variable_open_xml: nsweeps=%u\n", variable->nsweeps);
00218 #endif
00219
        if (algorithm == ALGORITHM_GENETIC)
00220
00221
00222
            // Obtaining bits representing each variable
00223
            if (xmlHasProp (node, (const xmlChar *) LABEL_NBITS))
00224
00225
                variable->nbits
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NBITS,
00226
00227
                                           &error_code);
                if (!error_code || !variable->nbits)
00228
00229
                 {
00230
                   variable_error (variable, _("invalid bits number"));
00231
                    goto exit_on_error;
                  }
00232
00233
00234
            else
00235
00236
                variable_error (variable, _("no bits number"));
00237
                goto exit_on_error;
00238
00239
       else if (nsteps)
00240
00241
00242
            variable->step
00243
              = jb_xml_node_get_float (node, (const xmlChar *) LABEL_STEP,
00244
                                        &error_code);
            if (!error_code || variable->step < 0.)</pre>
00245
00246
             {
               variable_error (variable, _("bad step size"));
00247
00248
                goto exit_on_error;
00249
00250
         }
00251
00252 #if DEBUG_VARIABLE
00253
       fprintf (stderr, "variable_open_xml: end\n");
00254 #endif
       return 1;
00255
00256 exit_on_error:
00257 variable_free (variable, INPUT_TYPE_XML); 00258 #if DEBUG_VARIABLE
00259 fprintf (stderr, "variable_open_xml: end\n");
00260 #endif
00261
       return 0;
00262 }
```

Here is the call graph for this function:



5.31.4 Variable Documentation

5.31.4.1 format

const char* format[NPRECISIONS] [extern]

Array of C-strings with variable formats.

Definition at line 53 of file variable.c.

5.31.4.2 precision

const double precision[NPRECISIONS] [extern]

Array of variable precisions.

Definition at line 58 of file variable.c.

5.32 variable.h

5.32 variable.h

Go to the documentation of this file.

```
00001 /*
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
00004 calibrations or optimizations of empirical parameters.
00006 AUTHORS: Javier Burguete and Borja Latorre.
00007
00008 Copyright 2012-2023, AUTHORS.
00009
00010 Redistribution and use in source and binary forms, with or without modification,
00011 are permitted provided that the following conditions are met:
00012
00013 1. Redistributions of source code must retain the above copyright notice,
00014 this list of conditions and the following disclaimer.
00015
00016 2. Redistributions in binary form must reproduce the above copyright notice, 00017 this list of conditions and the following disclaimer in the \,
00018 documentation and/or other materials provided with the distribution.
00019
00020 THIS SOFTWARE IS PROVIDED BY AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED
00021 WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF 00022 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
00023 SHALL AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
00024 SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
00025 PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
00026 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
00027 CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING 00028 IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
00029 OF SUCH DAMAGE.
00030 */
00031
00038 #ifndef VARIABLE__H
00039 #define VARIABLE__H 1
00040
00042 enum Algorithm
00043 {
00044
         ALGORITHM_MONTE_CARLO = 0,
         ALGORITHM_SWEEP = 1,
ALGORITHM_GENETIC = 2,
00045
00046
        ALGORITHM_ORTHOGONAL = 3
00047
00049
00054 typedef struct
00055 {
00056
        char *name;
00057
         double rangemin;
00058
         double rangemax;
00059
        double rangeminabs;
00060
         double rangemaxabs;
00061
        double step;
00062
        unsigned int precision;
00063
        unsigned int nsweeps;
00064
        unsigned int nbits:
00065 } Variable;
00066
00067 extern const char *format[NPRECISIONS];
00068 extern const double precision[NPRECISIONS];
00069
00070 // Public functions
00071 void variable_free (Variable * variable, unsigned int type);
00072 void variable_error (Variable * variable, char *message);
00073 int variable_open_xml (Variable * variable, xmlNode * node,
00074
                                  unsigned int algorithm, unsigned int nsteps);
00075 int variable_open_json (Variable * variable, JsonNode * node, 00076 unsigned int algorithm, unsigned int nsteps);
00077
00078 #endif
```

Index

adaptation_ratio	Window, 48
Input, 14	button_save
Optimize, 22	Window, 48
Algorithm	button_simulator
variable.h, 329	Window, 49
algorithm	button_template
Input, 14	Window, 49
Optimize, 22	
ALGORITHM_GENETIC	calculation_time
variable.h, 329	Optimize, 23
ALGORITHM_MONTE_CARLO	check_climbing
variable.h, 329	Window, 49
ALGORITHM_ORTHOGONAL	check_evaluator
variable.h, 329	Window, 49
ALGORITHM SWEEP	check_maxabs
variable.h, 329	Window, 49
application_directory	check_minabs
Window, 45	Window, 50
	check_template
box_buttons	Window, 50
Window, 45	climbing
button_about	Input, 14
Window, 45	Optimize, 23
button_add_experiment	CLIMBING_METHOD_COORDINATES
Window, 46	input.h, 135
button_add_variable	CLIMBING METHOD RANDOM
Window, 46	input.h, 135
button_algorithm	ClimbingMethod
Window, 46	input.h, 134
button_climbing	combo_experiment
Window, 46	Window, 50
button evaluator	combo variable
Window, 46	Window, 50
button exit	config.h, 69
Window, 47	DEFAULT PRECISION, 72
button experiment	DEFAULT_RANDOM_SEED, 72
Window, 47	DEFAULT RELAXATION, 72
button help	INPUT_TYPE, 85
Window, 47	INPUT_TYPE_JSON, 85
button_norm	INPUT TYPE XML, 85
Window, 47	LABEL_ABSOLUTE_MAXIMUM, 73
button_open	LABEL_ABSOLUTE_MINIMUM, 73
Window, 47	LABEL ADAPTATION, 73
button options	LABEL ALGORITHM, 73
Window, 48	LABEL CLIMBING, 73
button remove experiment	LABEL COORDINATES, 74
Window, 48	LABEL EUCLIDIAN, 74
button_remove_variable	LABEL_EVALUATOR, 74
Window, 48	LABEL_EXPERIMENT, 74
button_run	LABEL_EXPERIMENTS, 74

LABEL_GENETIC, 75	DEBUG_OPTIMIZE
LABEL_MAXIMUM, 75	optimize.c, 253
LABEL_MINIMUM, 75	DEBUG_VARIABLE
LABEL_MONTE_CARLO, 75	variable.c, 315
LABEL_MUTATION, 75	DEFAULT_PRECISION
LABEL NAME, 76	config.h, 72
LABEL NBEST, 76	DEFAULT RANDOM SEED
LABEL NBITS, 76	config.h, 72
LABEL NESTIMATES, 76	DEFAULT RELAXATION
LABEL NGENERATIONS, 76	config.h, 72
LABEL NITERATIONS, 77	dialog
LABEL NORM, 77	Options, 34
LABEL NPOPULATION, 77	Running, 37
LABEL NSIMULATIONS, 77	dialog_evaluator
LABEL NSTEPS, 77	interface.c, 143
LABEL NSWEEPS, 78	dialog_evaluator_close
LABEL OPTIMIZE, 78	interface.c, 143
LABEL ORTHOGONAL, 78	dialog name experiment close
LABEL P, 78	interface.c, 144
LABEL_PRECISION, 78	dialog_open_close
LABEL_RANDOM, 79	interface.c, 144
LABEL RELAXATION, 79	dialog_options_close
LABEL REPRODUCTION, 79	·
	interface.c, 146
LABEL_RESULT_FILE, 79	dialog_save_close
LABEL_SEED, 79	interface.c, 147
LABEL_SIMULATOR, 80	dialog_simulator
LABEL_STEP, 80	interface.c, 148
LABEL_SWEEP, 80	dialog_simulator_close
LABEL_TAXICAB, 80	interface.c, 149
LABEL_TEMPLATE1, 80	directory
LABEL_TEMPLATE2, 81	Input, 14
LABEL_TEMPLATE3, 81	anders and a sub-
LABEL_TEMPLATE4, 81	entry_result
LABEL_TEMPLATE5, 81	Window, 50
LABEL_TEMPLATE6, 81	entry_variable
LABEL_TEMPLATE7, 82	Window, 51
LABEL_TEMPLATE8, 82	entry_variables
LABEL_THRESHOLD, 82	Window, 51
LABEL_TOLERANCE, 82	error_best
LABEL_VARIABLE, 82	Optimize, 23
LABEL_VARIABLES, 83	error_message
LABEL_VARIABLES_FILE, 83	tools.c, 310
LABEL_WEIGHT, 83	tools.h, 313
LOCALE_DIR, 83	ERROR_NORM_EUCLIDIAN
MAX_NINPUTS, 83	input.h, 135
NALGORITHMS, 84	ERROR_NORM_MAXIMUM
NCLIMBINGS, 84	input.h, 135
NNORMS, 84	ERROR_NORM_P
NPRECISIONS, 84	input.h, 135
PROGRAM_INTERFACE, 84	ERROR_NORM_TAXICAB
	input.h, 135
DEBUG_EXPERIMENT	error_old
experiment.c, 88	Optimize, 23
DEBUG_INPUT	ERROR_TYPE
input.c, 106	tools.h, 312
DEBUG_INTERFACE	ErrorNorm
interface.c, 142	input.h, 135
DEBUG_MPCOTOOL	evaluator
mpcotool.c, 244	Input, 15

Optimize, 23	grid
Experiment, 11	Options, 34
name, 11	Running, 37
ninputs, 11	Window, 52
stencil, 12	grid_algorithm
weight, 12	Window, 52
experiment	grid_climbing
Input, 15	Window, 52
Optimize, 24	grid_experiment
Window, 51	Window, 53
experiment.c, 87	grid_files
DEBUG_EXPERIMENT, 88	Window, 53
experiment_error, 88	grid_norm
experiment_free, 89	Window, 53
experiment_new, 89	grid_variable
experiment_open_json, 90	Window, 53
experiment_open_xml, 92	
stencil, 94	id_experiment
experiment.h, 98	Window, 53
experiment_error, 99	id_experiment_name
experiment_free, 99	Window, 54
experiment_open_json, 100	id_input
experiment_open_xml, 102	Window, 54
stencil, 104	id_template
experiment_error	Window, 54
experiment.c, 88	id_variable
experiment.h, 99	Window, 54
experiment_free	id_variable_label
experiment.c, 89	Window, 54
experiment.h, 99	INFO_TYPE
experiment_new	tools.h, 313
experiment.c, 89	Input, 12
experiment_open_json	adaptation_ratio, 14
experiment.c, 90	algorithm, 14
experiment.h, 100	climbing, 14
experiment_open_xml	directory, 14
experiment.c, 92	evaluator, 15
experiment.h, 102	experiment, 15
oxportmentall, roz	mutation_ratio, 15
file	name, 15
Optimize, 24	nbest, 15
file result	nestimates, 16
Optimize, 24	nexperiments, 16
file_variables	niterations, 16
Optimize, 24	norm, 16
format	nsimulations, 16
variable.c, 322	nsteps, 17
variable.h, 336	nvariables, 17
frame_algorithm	p, 17
Window, 51	relaxation, 17
frame_experiment	reproduction_ratio, 17
Window, 51	result, 18
frame norm	seed, 18
Window, 52	simulator, 18
frame_variable	threshold, 18
Window, 52	tolerance, 18
,-	type, 19
genetic_variable	variable, 19
Optimize, 24	variables, 19
•	, -

input	INPUT_TYPE_JSON
input.c, 121	config.h, 85
input.h, 138	INPUT_TYPE_XML
input.c, 105	config.h, 85
DEBUG_INPUT, 106	interface.c, 140
input, 121	DEBUG_INTERFACE, 142
input_error, 106	dialog_evaluator, 143
input_free, 107	dialog_evaluator_close, 143
input_new, 107	dialog_name_experiment_close, 144
input_open, 108	dialog_open_close, 144
input_open_json, 109	dialog_options_close, 146
input_open_xml, 115	dialog_save_close, 147
result_name, 121	dialog_simulator, 148
variables_name, 121	dialog_simulator_close, 149
input.h, 133	INPUT_FILE, 142
CLIMBING_METHOD_COORDINATES, 135	input_save, 150
CLIMBING_METHOD_RANDOM, 135	input_save_climbing_json, 151
ClimbingMethod, 134	input_save_climbing_xml, 152
ERROR_NORM_EUCLIDIAN, 135	input_save_json, 153
ERROR_NORM_MAXIMUM, 135	input_save_xml, 155
ERROR_NORM_P, 135	logo, 192
ERROR_NORM_TAXICAB, 135	options, 192
ErrorNorm, 135	options_new, 158
input, 138	running, 192
input_free, 135	running_new, 159
input_new, 136	window, 192
input_open, 136	window_about, 159
result_name, 138	window_add_experiment, 160
variables_name, 138	window_add_variable, 161
input_error	window_get_algorithm, 161
input.c, 106	window_get_climbing, 162
INPUT_FILE	window_get_norm, 162
interface.c, 142	window_help, 162
input_free	window_inputs_experiment, 163
input.c, 107	window_label_variable, 163
input.h, 135	window_name_experiment, 164
input_new	window_new, 165
input.c, 107	window_open, 174
input.h, 136	window_precision_variable, 175
input_open	window_rangemax_variable, 175
input.c, 108	window_rangemaxabs_variable, 176
input.h, 136	window_rangemin_variable, 176
input_open_json	window_rangeminabs_variable, 176
input.c, 109	window_read, 177
input_open_xml	window_remove_experiment, 179
input.c, 115	window_remove_variable, 180
input_save	window_run, 180
interface.c, 150	window_save, 181
input_save_climbing_json	window_save_climbing, 182
interface.c, 151	window_set_algorithm, 183
input_save_climbing_xml	window_set_experiment, 184
interface.c, 152	window_set_variable, 184
input_save_json	window_step_variable, 186
interface.c, 153	window_template_experiment, 186
input_save_xml	window_template_experiment_close, 187
interface.c, 155	window_update, 188
INPUT_TYPE	window_update_climbing, 190
config.h, 85	window_update_variable, 190

window_weight_experiment, 191	config.h, 75
interface.h, 226	LABEL_MUTATION
MAX_LENGTH, 227	config.h, 75
window, 237	label_mutation
window_new, 227	Window, 56
	LABEL_NAME
JBW	config.h, 76
main.c, 240	LABEL_NBEST
	config.h, 76
label	LABEL_NBITS
Optimize, 25	config.h, 76
Running, 37	LABEL NESTIMATES
LABEL_ABSOLUTE_MAXIMUM	config.h, 76
config.h, 73	LABEL NGENERATIONS
LABEL_ABSOLUTE_MINIMUM	config.h, 76
config.h, 73	LABEL NITERATIONS
LABEL_ADAPTATION	config.h, 77
config.h, 73	LABEL_NORM
label_adaptation	config.h, 77
Window, 55	LABEL NPOPULATION
LABEL_ALGORITHM	config.h, 77
config.h, 73	LABEL NSIMULATIONS
label bests	config.h, 77
Window, 55	LABEL NSTEPS
label bits	-
Window, 55	config.h, 77
LABEL CLIMBING	LABEL_NSWEEPS
config.h, 73	config.h, 78
label climbing	LABEL_OPTIMIZE
Options, 34	config.h, 78
LABEL COORDINATES	LABEL_ORTHOGONAL
config.h, 74	config.h, 78
label estimates	LABEL_P
Window, 55	config.h, 78
LABEL EUCLIDIAN	label_p
config.h, 74	Window, 57
LABEL EVALUATOR	label_population
config.h, 74	Window, 57
LABEL EXPERIMENT	LABEL_PRECISION
config.h, 74	config.h, 78
label_experiment	label_precision
Window, 55	Window, 57
LABEL EXPERIMENTS	LABEL_RANDOM
config.h, 74	config.h, 79
label generations	LABEL_RELAXATION
	config.h, 79
Window, 56	label_relaxation
LABEL_GENETIC	Window, 57
config.h, 75	LABEL_REPRODUCTION
label_iterations	config.h, 79
Window, 56	label_reproduction
label_max	Window, 57
Window, 56	label_result
LABEL_MAXIMUM	Window, 58
config.h, 75	LABEL_RESULT_FILE
label_min	config.h, 79
Window, 56	LABEL_SEED
LABEL_MINIMUM	config.h, 79
config.h, 75	label_seed
LABEL MONTE CARLO	

Options, 34	Window, 60
label_simulations	LOCALE_DIR
Window, 58	config.h, 83
LABEL_SIMULATOR	logo
config.h, 80	interface.c, 192
label_simulator	Window, 60
Window, 58	
LABEL_STEP	main
config.h, 80	main.c, <mark>241</mark>
label_step	main.c, 239
Window, 58	JBW, 240
label_steps	main, <mark>241</mark>
Window, 58	main_window
LABEL SWEEP	tools.c, 310
config.h, 80	tools.h, 313
label_sweeps	MAX LENGTH
Window, 59	interface.h, 227
LABEL TAXICAB	MAX NINPUTS
config.h, 80	config.h, 83
LABEL TEMPLATE1	mpcotool
_	mpcotool.c, 244
config.h, 80	mpcotool.h, 249
LABEL_TEMPLATE2	mpcotool.c, 243
config.h, 81	DEBUG MPCOTOOL, 244
LABEL_TEMPLATE3	-
config.h, 81	mpcotool, 244
LABEL_TEMPLATE4	mpcotool.h, 248
config.h, 81	mpcotool, 249
LABEL_TEMPLATE5	mpi_rank
config.h, 81	Optimize, 25
LABEL_TEMPLATE6	mutation_ratio
config.h, 81	Input, 15
LABEL_TEMPLATE7	Optimize, 25
config.h, 82	NIAL CODITUMO
LABEL_TEMPLATE8	NALGORITHMS
config.h, 82	config.h, 84
label_threads	name
Options, 34	Experiment, 11
LABEL_THRESHOLD	Input, 15
config.h, 82	Variable, 38
label_threshold	nbest
Window, 59	Input, 15
LABEL_TOLERANCE	Optimize, 25
config.h, 82	nbits
label tolerance	Optimize, 25
Window, 59	Variable, 38
LABEL VARIABLE	NCLIMBINGS
config.h, 82	config.h, 84
label_variable	nend
Window, 59	Optimize, 26
LABEL_VARIABLES	nend_climbing
config.h, 83	Optimize, 26
label_variables	nestimates
Window, 59	Input, 16
LABEL_VARIABLES_FILE	Optimize, 26
config.h, 83	nexperiments
LABEL_WEIGHT	Input, 16
	Optimize, 26
config.h, 83	Window, 60
label_weight	ninputs

Experiment, 11	nsimulations, 27
Optimize, 26	nstart, 27
niterations	nstart_climbing, 27
Input, 16	nsteps, 28
Optimize, 27	nsweeps, 28
NNORMS	•
	nvariables, 28
config.h, 84	p, 28
norm	precision, 28
Input, 16	rangemax, 29
NPRECISIONS	rangemaxabs, 29
config.h, 84	rangemin, 29
nsaveds	rangeminabs, 29
Optimize, 27	relaxation, 29
nsimulations	reproduction_ratio, 30
Input, 16	result, 30
Optimize, 27	rng, 30
nstart	seed, 30
Optimize, 27	simulation_best, 30
nstart_climbing	simulator, 31
Optimize, 27	step, 31
nsteps	stop, 31
Input, 17	thread, 31
Optimize, 28	thread_climbing, 31
nsweeps	threshold, 32
Optimize, 28	tolerance, 32
Variable, 39	value, 32
nthreads_climbing	value_old, 32
optimize.c, 282	variables, 32
optimize.h, 308	weight, 33
nvariables	optimize
11141145166	Optimi20
Input 17	ontimize c. 283
Input, 17 Ontimize, 28	optimize.c, 283
Optimize, 28	optimize.h, 308
•	optimize.h, 308 optimize.c, 251
Optimize, 28 Window, 60	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253
Optimize, 28 Window, 60 Optimize, 20	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 experiment, 24	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 experiment, 24 file, 24	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 experiment, 24 file, 24 file_result, 24	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 experiment, 24 file, 24 file_result, 24 file_variables, 24	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 experiment, 24 file, 24 file_result, 24 file_variables, 24 genetic_variable, 24	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259 optimize_genetic, 259
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 evaluator, 23 experiment, 24 file_result, 24 file_result, 24 genetic_variable, 24 label, 25	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259 optimize_genetic, 259 optimize_genetic_objective, 260
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 evaluator, 23 experiment, 24 file_result, 24 file_result, 24 file_variables, 24 genetic_variable, 24 label, 25 mpi_rank, 25	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259 optimize_genetic, 259 optimize_genetic_objective, 260 optimize_input, 261
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 evaluator, 23 experiment, 24 file, 24 file_result, 24 file_variables, 24 genetic_variable, 24 label, 25 mpi_rank, 25 mutation_ratio, 25	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259 optimize_genetic, 259 optimize_genetic_objective, 260 optimize_input, 261 optimize_iterate, 262
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 evaluator, 23 experiment, 24 file, 24 file_result, 24 file_variables, 24 genetic_variable, 24 label, 25 mpi_rank, 25 mutation_ratio, 25 nbest, 25	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259 optimize_genetic, 259 optimize_genetic_objective, 260 optimize_input, 261 optimize_iterate, 262 optimize_merge, 263
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 evaluator, 23 experiment, 24 file, 24 file_result, 24 file_variables, 24 genetic_variable, 24 label, 25 mpi_rank, 25 mutation_ratio, 25 nbest, 25 nbits, 25	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259 optimize_genetic, 259 optimize_genetic_objective, 260 optimize_input, 261 optimize_iterate, 262 optimize_merge, 263 optimize_merge_old, 264
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 experiment, 24 file, 24 file_result, 24 file_variables, 24 genetic_variable, 24 label, 25 mpi_rank, 25 mutation_ratio, 25 nbest, 25 nend, 26	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259 optimize_genetic, 259 optimize_genetic_objective, 260 optimize_input, 261 optimize_iterate, 262 optimize_merge, 263 optimize_merge_old, 264 optimize_MonteCarlo, 265
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 experiment, 24 file_result, 24 file_result, 24 file_variables, 24 genetic_variable, 24 label, 25 mpi_rank, 25 mutation_ratio, 25 nbest, 25 nbits, 25 nend, 26 nend_climbing, 26	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259 optimize_genetic, 259 optimize_genetic_objective, 260 optimize_input, 261 optimize_iterate, 262 optimize_merge, 263 optimize_merge_old, 264 optimize_MonteCarlo, 265 optimize_norm, 283
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 experiment, 24 file_result, 24 file_result, 24 file_variables, 24 genetic_variable, 24 label, 25 mpi_rank, 25 mutation_ratio, 25 nbest, 25 nbits, 25 nend, 26 nend_climbing, 26 nestimates, 26	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259 optimize_genetic, 259 optimize_genetic_objective, 260 optimize_input, 261 optimize_iterate, 262 optimize_merge, 263 optimize_merge_old, 264 optimize_MonteCarlo, 265
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 experiment, 24 file_result, 24 file_result, 24 file_variables, 24 genetic_variable, 24 label, 25 mpi_rank, 25 mutation_ratio, 25 nbest, 25 nbits, 25 nend, 26 nend_climbing, 26 nestimates, 26 nexperiments, 26	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259 optimize_genetic, 259 optimize_genetic_objective, 260 optimize_input, 261 optimize_iterate, 262 optimize_merge, 263 optimize_merge_old, 264 optimize_MonteCarlo, 265 optimize_norm, 283
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 evaluator, 23 experiment, 24 file, 24 file_result, 24 file_variables, 24 genetic_variable, 24 label, 25 mpi_rank, 25 mutation_ratio, 25 nbest, 25 nbits, 25 nend, 26 nend_climbing, 26 nestimates, 26 nexperiments, 26 ninputs, 26	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259 optimize_genetic, 259 optimize_genetic_objective, 260 optimize_input, 261 optimize_iterate, 262 optimize_merge, 263 optimize_merge_old, 264 optimize_MonteCarlo, 265 optimize_norm, 283 optimize_norm_euclidian, 265
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 evaluator, 23 experiment, 24 file, 24 file_result, 24 file_variables, 24 genetic_variable, 24 label, 25 mpi_rank, 25 mutation_ratio, 25 nbest, 25 nbits, 25 nend, 26 nend_climbing, 26 nestimates, 26 ninputs, 26 niterations, 27	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259 optimize_genetic, 259 optimize_genetic_objective, 260 optimize_input, 261 optimize_iterate, 262 optimize_merge, 263 optimize_merge_old, 264 optimize_norm, 283 optimize_norm_euclidian, 265 optimize_norm_maximum, 266
Optimize, 28 Window, 60 Optimize, 20 adaptation_ratio, 22 algorithm, 22 calculation_time, 23 climbing, 23 error_best, 23 error_old, 23 evaluator, 23 evaluator, 23 experiment, 24 file, 24 file_result, 24 file_variables, 24 genetic_variable, 24 label, 25 mpi_rank, 25 mutation_ratio, 25 nbest, 25 nbits, 25 nend, 26 nend_climbing, 26 nestimates, 26 nexperiments, 26 ninputs, 26	optimize.h, 308 optimize.c, 251 DEBUG_OPTIMIZE, 253 nthreads_climbing, 282 optimize, 283 optimize_algorithm, 283 optimize_best, 254 optimize_best_climbing, 255 optimize_climbing, 255 optimize_climbing_sequential, 256 optimize_climbing_thread, 257 optimize_estimate_climbing, 283 optimize_estimate_climbing_coordinates, 258 optimize_estimate_climbing_random, 259 optimize_free, 259 optimize_genetic, 259 optimize_genetic_objective, 260 optimize_input, 261 optimize_iterate, 262 optimize_merge, 263 optimize_merge_old, 264 optimize_MonteCarlo, 265 optimize_norm, 283 optimize_norm_euclidian, 265 optimize_norm_maximum, 266 optimize_norm_p, 267

optimize_open, 268	optimize_norm_maximum
optimize_orthogonal, 273	optimize.c, 266
optimize_parse, 273	optimize_norm_p
optimize_print, 275	optimize.c, 267
optimize_refine, 275	optimize_norm_taxicab
optimize_save_old, 277	optimize.c, 268
optimize_save_variables, 277	optimize_open
optimize_sequential, 278	optimize.c, 268
optimize_step, 278	optimize.h, 303
optimize_step_climbing, 279	optimize_orthogonal
optimize sweep, 280	optimize.c, 273
optimize_synchronise, 281	optimize_parse
optimize_thread, 282	optimize.c, 273
RM, 254	optimize_print
optimize.h, 302	optimize.c, 275
nthreads_climbing, 308	optimize_refine
optimize, 308	optimize.c, 275
optimize_free, 303	optimize_save_old
optimize_open, 303	optimize.c, 277
optimize_algorithm	optimize_save_variables
optimize_aigonitim	optimize_save_variables
optimize_best	optimize_sequential
•	
optimize.c, 254	optimize.c, 278
optimize_best_climbing	optimize_step
optimize.c, 255	optimize.c, 278
optimize_climbing	optimize_step_climbing
optimize.c, 255	optimize.c, 279
optimize_climbing_sequential	optimize_sweep
optimize.c, 256	optimize.c, 280
optimize_climbing_thread	optimize_synchronise
optimize.c, 257	optimize.c, 281
optimize_estimate_climbing	optimize_thread
optimize.c, 283	optimize.c, 282
optimize_estimate_climbing_coordinates	Options, 33
optimize.c, 258	dialog, 34
optimize_estimate_climbing_random	grid, <mark>34</mark>
optimize.c, 259	label_climbing, 34
optimize_free	label_seed, 34
optimize.c, 259	label_threads, 34
optimize.h, 303	spin_climbing, 35
optimize_genetic	spin_seed, 35
optimize.c, 259	spin_threads, 35
optimize_genetic_objective	options
optimize.c, 260	interface.c, 192
optimize_input	options_new
optimize.c, 261	interface.c, 158
optimize_iterate	
optimize.c, 262	p
optimize_merge	Input, 17
optimize.c, 263	Optimize, 28
optimize_merge_old	ParallelData, 35
optimize.c, 264	thread, 36
optimize_MonteCarlo	precision
optimize.c, 265	Optimize, 28
optimize_norm	Variable, 39
optimize_norm	variable.c, 322
optimize_norm_euclidian	variable.h, 336
optimize_norm_edicidari	PROGRAM_INTERFACE
opumize.c, 200	config.h, 84
	5 /

rangemax	Optimize, 30
Optimize, 29	simulator
Variable, 39	Input, 18
rangemaxabs	Optimize, 31
Optimize, 29	spin_adaptation
Variable, 39	Window, 62
rangemin	spin_bests
Optimize, 29	Window, 62
Variable, 39	spin_bits
rangeminabs	Window, 62
Optimize, 29	spin_climbing
Variable, 40	Options, 35
relaxation	spin_estimates
Input, 17	Window, 62
Optimize, 29	spin_generations
reproduction_ratio	Window, 63
Input, 17	spin_iterations
Optimize, 30	Window, 63
result	spin_max
Input, 18	Window, 63
Optimize, 30	spin_maxabs
result_name	Window, 63
input.c, 121	spin_min
input.h, 138	Window, 63
RM	spin_minabs
optimize.c, 254	Window, 64
rng	spin_mutation
Optimize, 30	Window, 64
Running, 36	spin_p
dialog, 37	Window, 64
grid, 37	spin_population
label, 37	Window, 64
spinner, 37	spin_precision
running	Window, 64
interface.c, 192	spin_relaxation
running_new	Window, 65
interface.c, 159	spin_reproduction
savelled may	Window, 65
scrolled_max	spin_seed
Window, 60	Options, 35
scrolled_maxabs	spin_simulations
Window, 61	Window, 65
scrolled_min	spin_step
Window, 61	Window, 65
scrolled_minabs	spin_steps
Window, 61	Window, 65
scrolled_p	spin_sweeps
Window, 61	Window, 66
scrolled_step	spin_threads
Window, 61	Options, 35
scrolled_threshold	spin_threshold
Window, 62	Window, 66
seed	spin_tolerance
Input, 18 Optimize, 30	Window, 66
show_pending	spin_weight
tools.c, 311	Window, 66
	spinner
tools.h, 313 simulation best	Running, 37
Simulation_Dest	

Experiment, 12		
experiment., 94 experiment.h, 104 step Optimize, 31 Variable, 40 Stop Optimize, 31 Variable, 40 Stop Optimize, 31 ParallelData, 36 United Climbing Optimize, 31 Optimize, 31 Optimize, 31 Optimize, 31 Variable, 40 Optimize, 31 ParallelData, 36 Unitered Climbing Optimize, 31 Optimize, 31 Optimize, 31 Optimize, 32 Variable, 329 Variable, pen _xmi, 333 Variable _open _sion, 330 Variable, open _xmi, 333 Variable _open _sion Variable, open _xmi, 333 Variable, n, 329 Variable, n, 320 Variable, n, 320 Variable, n, 330 Variable, n, 330 Variable, pen _xmi, 333 Variable, n, 330 Variable, open _sion Variable, n, 330 Variable, nervi, 330 Variable, n, 330 Variable, n, 330 Variable, n, 330 Variable, nervi, 340 Variable, n, 330 Variable, n, 330 Variable, nervi, 340 Variable, nervi, 141 Variable, nervi, 141 Variable, nervi, 141 Variable, nervi, 141 V	stencil	variable.h, 327
experiment.h, 104 step Optimize, 31 Variable, 40 Stop Optimize, 31 Optimize, 32 Variable, 33 Show_pending, 313 Show_pending, 313 Show_pending, 313 Show_pending, 313 Show_pending, 313 Optimize, 32 Value_old Optimize, 32 Variable, 38 Optimize, 39 rangemaxabs, 39 rangemaxabs, 39 rangemaxabs, 39 rangemin, 39 rangeminab, 40 Stop, 40 Variable Input, 19 Window, 66 Variable, C, 314 DEBUG_VARIABLE, 315 Optimize, 32 Variable, open_json Optimize, 32 Variable, open_json Optimize, 32 Optimize,	Experiment, 12	Algorithm, 329
Step	experiment.c, 94	ALGORITHM_GENETIC, 329
Optimize, 31 Variable, 40 stop Optimize, 31 Variable, 40 stop Optimize, 31 thread Optimize, 31 Parallel(Data, 36 Variable, 216 Variable, 38 ERROR_TYPE, 312 INFO_TYPE, 312 INFO_TYPE, 313 main_window, 313 show_pending, 313 type Input, 19 Optimize, 32 value Optimize, 32 value Optimize, 31 variable, 333 variables, 343 variable, 353 variables, 363 variable, 363 variable, 363 variable, 363 variable, 364 variable, 374 variable, 384 variable, 385 variable, 387 variable, 397 variable, 497 variable, 497	experiment.h, 104	ALGORITHM_MONTE_CARLO, 329
Variable, 40 stop Optimize, 31 Optimize, 31 thread Optimize, 31 ParallelData, 36 Variable, error, 329 variable free, 330 variable error, 329 variable open_ison, 330 variable open_ison, 330 variable open_ison, 330 variable open_ison, 330 variable, 330 variable, 330 variable, open_ison, 330 variable, 330 variable, 339 variable, 339 variable, 329 threshold Input, 18 Optimize, 32 tolerance Input, 18 Optimize, 32 tolerance Input, 18 Optimize, 32 variable, 330 variable, 330 variable, 330 variable, 330 variable, 317 variable, 331 variable, 331 variable, 331 variable, 331 variable, 333 variable, 334 variable, 334 variable, 335 variable, 336 variable, 337 variable, 338 variable, 338 variable, 340 variable, 640 varia	step	ALGORITHM ORTHOGONAL, 329
Variable, 40 stop Optimize, 31 Optimize, 31 thread Optimize, 31 ParallelData, 36 Variable, error, 329 variable free, 330 variable error, 329 variable open_ison, 330 variable open_ison, 330 variable open_ison, 330 variable open_ison, 330 variable, 330 variable, 330 variable, open_ison, 330 variable, 330 variable, 339 variable, 339 variable, 329 threshold Input, 18 Optimize, 32 tolerance Input, 18 Optimize, 32 tolerance Input, 18 Optimize, 32 variable, 330 variable, 330 variable, 330 variable, 330 variable, 317 variable, 331 variable, 331 variable, 331 variable, 331 variable, 333 variable, 334 variable, 334 variable, 335 variable, 336 variable, 337 variable, 338 variable, 338 variable, 340 variable, 640 varia	Optimize, 31	ALGORITHM SWEEP, 329
Stop	•	-
Optimize, 31 thread		
thread		
thread variable_open_json, 330 Optimize, 31 variable_open_json, 330 ParallelData, 36 variable_open_xml, 333 thread_climbing variable.c, 316 Optimize, 31 variable.c, 316 Optimize, 32 variable.c, 316 Optimize, 32 variable.c, 317 tolerance variable.c, 317 Input, 18 variable.c, 317 Optimize, 32 variable.c, 317 tools.c, 309 variable.c, 317 error_message, 310 variable.c, 319 main_window, 310 variable.c, 319 show_pending, 311 variable.c, 319 tools.h, 312 Input, 19 error_message, 313 Optimize, 32 INFO_TYPE, 313 input. 19 weight Experiment, 12 optimize, 32 variable.name value Optimize, 32 Value_old Optimize, 32 Variable, 38 button_dimition value_old postimize, 32 Variable, 38 button_dimition name, 38 nabts, 36		
Optimize, 31 ParallelData, 36 thread_climbing Optimize, 31 threshold Input, 18 Optimize, 32 tolerance Input, 18 Optimize, 32 tolerance Input, 18 Optimize, 32 tolescoil input, 30 orariable, 317 variable, 330 orariable, 330 orariable, 331 tolescoil input, 311 tolescoil input, 311 tolescoil input, 312 error_message, 313 ERROR_TYPE, 312 INFO_TYPE, 312 INFO_TYPE, 313 main_window, 313 show_pending, 313 type Input, 19	thread	- :
ParallelData, 36 thread_climbing	Optimize, 31	
thread_climbing	•	— · —
Optimize, 31 threshold Input, 18 Optimize, 32 tolerance Input, 18 Optimize, 32 tolerance Input, 18 Optimize, 32 tools.c, 309 error_message, 310 main_window, 310 show_pending, 311 tools.h, 312 error_message, 313 ERROR TYPE, 312 INFO_TYPE, 313 main_window, 313 show_pending, 313 type Input, 19 Variable, 33 variables_name Input, 19 Input, 19 Very and the		
threshold	_ · · · · ·	
Input, 18 Optimize, 32 Variable, 330 Variable, 330 Variable, 330 Variable, 330 Variable, 330 Variable, 330 Variable, 331 Variable, 330 Variable, 331 Variable, 311 variable, 311 variable, 311 variable, 311 variable, 312 Input, 19 Optimize, 32 variables, 313 Anain, window, 313 show pending, 313 value Input, 19 Value Optimize, 32 value Optimize, 32 value, 38 name, 39 precision, 39 pragemax, 39 pragemax, 39 rangemin, 39 rangemax, 39 rangemin, 39 rangemin, 39 rangeminabs, 40 step, 40 variable, 46 variable, 47 button_evaluator, 46 button_evaluator, 46 button_evaluator, 47 button_evaluator, 48 button_ever, 47 button_ever, 48 button_remove_experiment, 48 button_remove_experiment, 48 button_remove_experiment, 48 button_remove_experiment, 48 button_save, 48 button_evaluator, 49 check_elimbing, 49 check_elimbing, 49 check_evaluator, 49 variable, 49	•	
Optimize, 32 tolerance		
tolerance	•	
Input, 18 Optimize, 32 Optimize, 32 tools.c, 309 error_message, 310 main_window, 310 show_pending, 311 tools.h, 312 error_message, 313 ERROR_TYPE, 312 INFO_TYPE, 313 main_window, 313 show_pending, 313 type Input, 19 Variable, 32 Variables Input, 19 Variables Input, 19 Variables, 32 Variables, name input, 1, 121 input, 1, 138 weight Experiment, 12 Optimize, 32 Variable, 38 Variable, 38 Variable, 38 Variable, 38 Variable, 38 Input, 32 Variable, 38 Input, 32 Variable, 38 Input, 32 Variable, 38 Input, 39 Variable, 38 Input, 39 Variable, 39 Input, 46 Dotton_add_experiment, 46 Dutton_add_variable, 46 Dutton_add_variable, 46 Dutton_exit, 47 Dutton_experiment, 47 Dutton_experiment, 47 Dutton_open, 47 Dutton_open, 47 Dutton_open, 47 Dutton_open, 47 Dutton_remove_experiment, 48 Dutton_remove_variable, 48 Dutton_simulator, 49 Dutton_template, 49 Check_climbing, 49 Check_climbing, 49 Check_evaluator, 49 Che	•	
Optimize, 32 tools.c., 309 error_message, 310 main_window, 310 show_pending, 311 tools.h, 312 error_message, 313 ERROR_TYPE, 312 INFO_TYPE, 312 INFO_TYPE, 313 main_window, 313 show_pending, 313 type Input, 19 Inpu		_ ·
tools.c; 309 error_message, 310 main_window, 310 show_pending, 311 tools.h, 312 error_message, 313 ERROR_TYPE, 312 INFO_TYPE, 313 main_window, 313 show_pending, 313 type Input, 19 Variable.s, 33 window, 40 Optimize, 32 value Optimize, 32 Value Optimize, 32 Value old Optimize, 32 Value, 38 name, 38 noits, 38 name, 38 noits, 38 nsweeps, 39 precision, 39 rangemax, 39 rangeminabs, 40 step, 40 variable Input, 19 window, 66 variable, 314 DEBUG_VARIABLE, 315 format, 322 precision, 322 variable_error, 316 variable_error, 316 variable_error, 316 variable_error, 316 variable_error, 316 variable_poen_json, 317 variables, 313 variables, 313 nput, 19 variable, 0, 33 variable	•	
error_message, 310 main_window, 310 show_pending, 311 tools.h, 312 error_message, 313 ERROR_TYPE, 312 INFO_TYPE, 313 main_window, 313 show_pending, 313 type Input, 19 Input, 19 Input, 19 Input, 19 Weight Input, 19 Input, 19 Input, 19 Weight Experiment, 12 Optimize, 32 value Optimize, 32 value Optimize, 32 value, 0d Optimize, 33 variable, 38 Input, 19 Input, 19 Vindow, 46 Variable, 39 rangemaxabs, 39 rangeminabs, 40 variable Input, 19 Window, 66 variable, 314 DEBUG_VARIABLE, 315 format, 322 precision, 322 variable_error, 316 variable_error, 316 variable_error, 316 variable_popen_json, 317 variable_maxabs, 49	•	
main_window, 310 show_pending, 311 tools.h, 312 error_message, 313 ERROR_TYPE, 312 INFO_TYPE, 313 main_window, 313 show_pending, 313 type Input, 19 Input, 1	•	-· -
show_pending, 311 tools.h, 312 error_message, 313 ERROR_TYPE, 312 INFO_TYPE, 312 INFO_TYPE, 313 main_window, 313 show_pending, 313 type Input, 19 Optimize, 32 value Optimize, 32 value_old Optimize, 32 button_about, 45 button_add_experiment, 46 button_add_experiment, 46 button_adl_experiment, 46 button_adl_experiment, 46 button_adl_experiment, 46 button_experiment, 46 button_experiment, 47 button_experiment, 47 button_experiment, 47 button_experiment, 47 button_open, 48 button_remove_experiment, 48 button_remove_experiment, 48 button_remove_variable, 48 button_save, 48 button_		
tools.h, 312 error_message, 313 ERROR_TYPE, 312 INFO_TYPE, 313 main_window, 313 show_pending, 313 type Input, 19		variable.h, 333
error_message, 313 ERROR_TYPE, 312 INFO_TYPE, 313 main_window, 313 show_pending, 313 type Input, 19 Value Optimize, 32 Variables_name input.c, 121 input.h, 138 weight Experiment, 12 Optimize, 33 Window, 40 Application_directory, 45 box_buttons, 45 box_buttons, 45 box_button_about, 45 Variable, 38 Noits, 38 Noite, 39 Precision, 39 Precision, 39 Prangemaxabs, 39 Prangemin, 39 Prangemin, 39 Prangemin, 39 Prangeminabs, 40 Step, 40 Variable Input, 19 Window, 66 Variable.c, 314 DEBUG_VARIABLE, 315 format, 322 precision, 322 variable_error, 316 variable_free, 316 variable_free, 316 variable_free, 316 variable_free, 316 variable_free, 316 variable_open_json, 317 Variable_maxabs, 49 Variable_maxabs, 49 Variable_maxabs, 49		variables
ERROR_TYPE, 312 INFO_TYPE, 313 main_window, 313 show_pending, 313 type Input, 19 Optimize, 32 value Optimize, 32 value_old Optimize, 32 Variable, 38 name, 12 name, 12 name, 12 optimize, 32 Variable, 38 name, 18 name, 19 weight Experiment, 12 optimize, 33 Window, 40 Optimize, 32 Variable, 38 name, 19 box_button, 40 Application_directory, 45 box_buttons, 45 box_buttons, 45 box_button_about, 45 box_button_about, 45 button_add_experiment, 46 button_add_experiment, 46 button_add_experiment, 46 button_adgorithm, 46 button_climbing, 46 button_climbing, 46 button_evaluator, 46 button_evaluator, 46 button_experiment, 47 button_experiment, 47 button_experiment, 47 button_norm, 47 step, 40 variable Input, 19 Window, 66 variable, 314 DEBUG_VARIABLE, 315 format, 322 precision, 322 variable_error, 316 variable_free, 316 variable_free, 316 variable_open_json, 317 variable_maxabs, 49		Input, 19
INFO_TYPE, 313 main_window, 313 show_pending, 313 type Input, 19 Optimize, 32 value Optimize, 32 value_old Optimize, 32 Variable, 38 name, 38 nsweeps, 39 precision, 39 rangemaxabs, 39 rangemaxabs, 39 rangeminabs, 40 step, 40 variable Input, 19 INFO_TYPE, 313 input.c, 121 input.h, 138 Experiment, 12 Optimize, 33 Window, 40 Application_directory, 45 box_buttons, 45 box_buttons, 45 box_button, 45 box_button_about, 45 button_add_experiment, 46 button_add_variable, 46 button_add_variable, 46 button_climbing, 46 button_climbing, 46 button_evaluator, 46 button_evaluator, 46 button_exit, 47 cangemaxabs, 39 cangeminabs, 40 step, 40 button_norm, 47 button_open, 47 variable Input, 19 button_open, 47 button_open, 47 button_open, 47 button_open, 47 button_open, 48 button_remove_experiment, 48 button_remove_variable, 48 variable.c, 314 DEBUG_VARIABLE, 315 format, 322 precision, 322 variable_error, 316 variable_free, 316 variable_open_json, 317 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49		Optimize, 32
main_window, 313 show_pending, 313 type Input, 19 Coptimize, 32 value Optimize, 32 Value_old Optimize, 32 Variable, 38 name, 38 name, 38 nobits, 38 nsweeps, 39 precision, 39 rangemaxabs, 39 rangeminabs, 40 variable Input, 19 Vari		variables_name
show_pending, 313 type Input, 19 Experiment, 12 Optimize, 33 value Optimize, 32 value_old Optimize, 32 Variable, 38 name, 38 name, 38 nbits, 38 nsweeps, 39 precision, 39 rangemaxabs, 39 rangemin, 39 rangemin, 39 rangeminabs, 40 step, 40 variable Input, 19 Window, 66 variable_c, 314 DEBUG_VARIABLE, 315 format, 322 variable_error, 316 variable_free, 316 variable_free, 316 variable_free, 316 variable_free, 316 variable_pother variable_pother variable_cone, 33 veight Experiment, 12 Optimize, 33 Window, 40 application_de poptimize, 33 weight Experiment, 12 Optimize, 33 Window, 40 application_directory, 45 button_about, 45 button_add_experiment, 46 button_add_experiment, 46 button_add_experiment, 46 button_climbing, 46 button_climbing, 46 button_evaluator, 46 button_evaluator, 47 button_experiment, 47 button_norm, 47 button_norm, 47 button_options, 48 button_remove_experiment, 48 button_remove_experiment, 48 button_remove_experiment, 48 button_save, 48 button_simulator, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49		input.c, 121
type Input, 19 Experiment, 12 Optimize, 33 value Optimize, 32 application_directory, 45 value_old box_buttons, 45 Optimize, 32 button_about, 45 Variable, 38 button_add_experiment, 46 name, 38 button_add_experiment, 46 nibits, 38 button_add_variable, 46 nibits, 38 button_algorithm, 46 nsweeps, 39 button_evaluator, 46 rangemax, 39 rangemaxabs, 39 rangeminabs, 40 step, 40 variable Input, 19 button_options, 48 Variable.c, 314 button_remove_experiment, 48 variable_cror, 316 variable_free, 316 variable_free, 316 variable_open_json, 317 value Experiment, 12 optimize, 33 Experiment, 12 optimize, 33 Experiment, 12 optimize, 33 button_application_directory, 45 button_apout, 46 button_exit, 47 button_experiment, 47 button_norm, 47 button_options, 48 button_remove_experiment, 48 button_remove_experiment, 48 button_remove_variable, 48 button_save, 48 button_simulator, 49 button_template, 49 check_climbing, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49	main_window, 313	input.h, 138
Input, 19 Experiment, 12 Optimize, 33 Value Optimize, 32 Value_old Optimize, 32 Variable, 38 name, 38 nbits, 38 nsweeps, 39 precision, 39 rangemax, 39 rangemaxabs, 39 rangemin, 39 rangemin, 39 rangeminabs, 40 Step, 40 Variable Input, 19 Window, 66 Variable. C, 314 DEBUG_VARIABLE, 315 format, 322 value_old Window, 40 Optimize, 32 Vindow, 40 Application_directory, 45 box_button_about, 45 button_add_experiment, 46 button_add_experiment, 46 button_adg_orithm, 46 button_algorithm, 46 button_climbing, 46 button_exit, 47 rangeming, 46 button_exit, 47 button_experiment, 47 button_norm, 47 button_norm, 47 button_norm, 47 button_open, 47 button_open, 47 button_open, 47 button_open, 47 button_remove_experiment, 48 button_remove_experiment, 48 button_remove_experiment, 48 button_remove_variable, 48 button_run, 48 button_run, 48 button_save, 48 button_simulator, 49 button_save, 48 button_simulator, 49 button_template, 49 check_climbing, 49 check_evaluator, 49	show_pending, 313	
value Optimize, 32 Value_old Optimize, 32 Variable, 38 name, 38 nbits, 38 nsweeps, 39 precision, 39 rangemaxabs, 39 rangeminabs, 40 step, 40 Variable Input, 19 Window, 66 Variable.c, 314 DEBUG_VARIABLE, 315 format, 322 precision, 322 value_old Optimize, 32 Window, 40 application_directory, 45 box_buttons, 45 box_button_about, 45 box_button_about, 45 button_add_experiment, 46 button_experiment, 46 button_evaluator, 46 button_experiment, 47 button_experiment, 47 button_open, 47 button_open, 47 button_open, 47 button_open, 47 button_open, 48 button_remove_experiment, 48 button_run, 48 button_run, 48 button_run, 48 button_save, 48 button_simulator, 49 button_template, 49 check_climbing, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49 check_maxabs, 49	type	-
value Optimize, 32 application_directory, 45 box_buttons, 45 Optimize, 32 button_about, 45 button_add_experiment, 46 name, 38 answeeps, 39 button_exit, 47 sangemax, 39 rangeminabs, 40 step, 40 button_norm, 47 step, 40 button_open, 47 button_open, 47 wariable Input, 19 Window, 66 button_experiment, 48 button_experiment, 49 button_experiment, 49 check_climbing, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49 check_experiment, 49	Input, 19	·
Optimize, 32 value_old box_buttons, 45 Optimize, 32 Variable, 38 button_about, 45 Name, 38 button_add_experiment, 46 Nobits, 38 button_add_variable, 46 Nobits, 38 button_algorithm, 46 Nobits, 38 button_climbing, 46 Nobits, 39 button_evaluator, 46 Nobits, 39 button_evaluator, 46 Nobits, 39 precision, 39 button_exit, 47 Nobits, 39 button_exit, 47 Nobits, 47 Nobits, 47 Nobits, 48 button_exit, 47 Nobits, 49 button_open, 47 Nobits, 40 button_open, 47 Variable Nobits, 40 button_open, 47 Nobits, 40 button_open, 47 Nobits, 314 Nobits, 314 Nobits, 315 Nobits, 322 Nobits, 314 Nobits, 315 Nobits, 322 Nobits, 316 Nobits, 48 Nobits, 316 Nobits, 49 Nobits, 45 Nobits, 45 Nobits, 46 Nobits, 47 Nobits, 47 Nobits, 47 Nobits, 48 Nobits, 49 Nobits, 45 Nobits, 46 Nobits, 47 Nobits, 48 Nobits, 49 Nobits, 45 Nobits, 46 Nobits, 46 Nobits, 47 Nobits, 47 Nobits, 48 Nobits, 49 Nobits, 45 Nobits, 46 Nobits, 46 Nobits, 46 Nobits, 47 Nobits, 48 No		•
value_old Optimize, 32 Variable, 38 name, 38 name, 38 nbits, 38 nsweeps, 39 precision, 39 rangemax, 39 rangeminabs, 40 step, 40 variable Input, 19 Window, 66 variable.c, 314 DEBUG_VARIABLE, 315 format, 322 precision, 322 variable_error, 316 variable_error, 316 variable_giant variable_giant variable_giant variable_giant variable_error, 316 variable_giant variable_giant variable_giant variable_gree, 316 variable_open_json, 317 button_about, 45 button_add_experiment, 46 button_algorithm, 46 button_experiment, 46 button_evaluator, 46 button_experiment, 47 button_experiment, 47 button_norm, 47 button_open, 47 button_remove_experiment, 48 button_remove_experiment, 48 button_run, 48 button_save, 48 button_simulator, 49 button_template, 49 check_evaluator, 49 variable_open_json, 317 check_maxabs, 49	value	Window, 40
Optimize, 32 Variable, 38 name, 38 name, 38 nbits, 38 nsweeps, 39 precision, 39 precision, 39 rangemaxabs, 39 rangemin, 39 rangeminabs, 40 step, 40 Variable Input, 19 Window, 66 Variable.c, 314 DEBUG_VARIABLE, 315 format, 322 precision, 322 variable_error, 316 variable_free, 316 variable_open_json, 317 button_about, 45 button_add_variable, 46 button_add_variable, 46 button_add_variable, 46 button_add_variable, 46 button_add_variable, 46 button_algorithm, 46 button_evaluator, 46 button_experiment, 47 button_experiment, 47 button_norm, 47 button_open, 47 button_open, 47 button_remove_experiment, 48 button_remove_variable, 48 button_run, 48 button_save, 48 button_simulator, 49 button_template, 49 check_evaluator, 49 variable_open_json, 317 check_maxabs, 49	Optimize, 32	application_directory, 45
Variable, 38 name, 38 nbits, 38 nbits, 38 nbits, 38 nsweeps, 39 precision, 39 precision, 39 trangemax, 39 trangemaxabs, 39 trangemin, 39 trangeminabs, 40 trangeminabs, 40 trangeminable linput, 19 Window, 66 Variable.c, 314 DEBUG_VARIABLE, 315 format, 322 precision, 322 variable_error, 316 variable_copen_json, 317 button_add_experiment, 46 button_add_variable, 46 button_algorithm, 46 button_experiment, 46 button_evaluator, 46 button_experiment, 47 button_experiment, 47 button_norm, 47 button_open, 47 button_open, 47 button_remove_experiment, 48 button_remove_variable, 48 button_run, 48 button_save, 48 button_save, 48 button_simulator, 49 button_template, 49 check_evaluator, 49 check_evaluator, 49 check_evaluator, 49 check_maxabs, 49	value_old	box_buttons, 45
name, 38	Optimize, 32	button_about, 45
nbits, 38	Variable, 38	button_add_experiment, 46
nsweeps, 39 precision, 322 precision, 322 precision, 322 precision, 322 precision, 316 precision, 317 putton_evaluator, 46 putton_exit, 47 putton_experiment, 47 putton_experiment, 47 putton_open, 47 putton_open, 47 putton_open, 47 putton_remove_experiment, 48 putton_remove_variable, 48 putton_remove_variable, 48 putton_save, 48 putton_simulator, 49 putton_template, 40 putton_template, 40 putton_template, 40 putton_template, 40 putton_te	name, 38	button_add_variable, 46
precision, 39	nbits, 38	button_algorithm, 46
precision, 39	nsweeps, 39	button climbing, 46
rangemaxabs, 39	precision, 39	button evaluator, 46
rangemaxabs, 39	rangemax, 39	button exit, 47
rangemin, 39	rangemaxabs, 39	- :
rangeminabs, 40 button_norm, 47 step, 40 button_open, 47 variable button_options, 48 Input, 19 button_remove_experiment, 48 Window, 66 button_remove_variable, 48 variable.c, 314 button_run, 48 DEBUG_VARIABLE, 315 button_save, 48 format, 322 button_simulator, 49 precision, 322 button_template, 49 variable_error, 316 check_climbing, 49 variable_free, 316 check_evaluator, 49 variable_open_json, 317 check_maxabs, 49		_ ·
step, 40 variable lnput, 19 Window, 66 variable.c, 314 DEBUG_VARIABLE, 315 format, 322 precision, 322 variable_error, 316 variable_free, 316 variable_open_json, 317 button_open, 47 button_open, 47 button_remove_experiment, 48 button_remove_variable, 48 button_run, 48 button_save, 48 button_save, 48 button_simulator, 49 button_template, 49 check_climbing, 49 check_evaluator, 49 check_maxabs, 49	rangeminabs, 40	_ ·
variable button_options, 48 Input, 19 button_remove_experiment, 48 Window, 66 button_remove_variable, 48 variable.c, 314 button_run, 48 DEBUG_VARIABLE, 315 button_save, 48 format, 322 button_simulator, 49 precision, 322 button_template, 49 variable_error, 316 check_climbing, 49 variable_free, 316 check_evaluator, 49 variable_open_json, 317 check_maxabs, 49		
Input, 19 Window, 66 button_remove_experiment, 48 variable.c, 314 DEBUG_VARIABLE, 315 format, 322 precision, 322 variable_error, 316 variable_free, 316 variable_open_json, 317 button_remove_experiment, 48 button_save, 48 button_save, 49 button_sav	•	_ ·
Window, 66 variable.c, 314 DEBUG_VARIABLE, 315 format, 322 precision, 322 variable_error, 316 variable_free, 316 variable_open_json, 317 button_remove_variable, 48 button_run, 48 button_save, 48 button_simulator, 49 button_template, 49 check_climbing, 49 check_evaluator, 49 check_maxabs, 49		_ ·
variable.c, 314 DEBUG_VARIABLE, 315 format, 322 precision, 322 variable_error, 316 variable_free, 316 variable_open_json, 317 button_run, 48 button_save, 48 button_simulator, 49 button_template, 49 check_climbing, 49 check_evaluator, 49 check_maxabs, 49		·
DEBUG_VARIABLE, 315 button_save, 48 format, 322 button_simulator, 49 precision, 322 button_template, 49 variable_error, 316 check_climbing, 49 variable_free, 316 check_evaluator, 49 variable_open_json, 317 check_maxabs, 49		
format, 322 button_simulator, 49 precision, 322 button_template, 49 variable_error, 316 check_climbing, 49 variable_free, 316 check_evaluator, 49 variable_open_json, 317 check_maxabs, 49		
precision, 322 button_template, 49 variable_error, 316 check_climbing, 49 variable_free, 316 check_evaluator, 49 variable_open_json, 317 check_maxabs, 49	— · · · · · · · · · · · · · · · · · · ·	- · · ·
variable_error, 316 check_climbing, 49 variable_free, 316 check_evaluator, 49 variable_open_json, 317 check_maxabs, 49		
variable_free, 316 check_evaluator, 49 variable_open_json, 317 check_maxabs, 49	•	_ ·
variable_open_json, 317 check_maxabs, 49		— -
_ · _ ,	- · · ·	-
variable_operi_xiiii, 319 cneck_minabs, 50	— · —	
	variable_operi_xirii, 313	check_minabs, 50

check_template, 50	scrolled_step, 61
combo_experiment, 50	scrolled_threshold, 62
combo_variable, 50	spin_adaptation, 62
entry_result, 50	spin_bests, 62
entry_variable, 51	spin_bits, 62
entry_variables, 51	spin_estimates, 62
experiment, 51	spin_generations, 63
frame_algorithm, 51	spin_iterations, 63
frame_experiment, 51	spin_max, 63
frame norm, 52	spin_maxabs, 63
frame_variable, 52	spin_min, 63
grid, 52	spin_minabs, 64
grid_algorithm, 52	spin_mutation, 64
grid_climbing, 52	spin_p, 64
grid_experiment, 53	spin_population, 64
grid_files, 53	spin_precision, 64
grid norm, 53	spin relaxation, 65
grid_variable, 53	spin_reproduction, 65
id_experiment, 53	
_ ·	spin_simulations, 65
id_experiment_name, 54	spin_step, 65
id_input, 54	spin_steps, 65
id_template, 54	spin_sweeps, 66
id_variable, 54	spin_threshold, 66
id_variable_label, 54	spin_tolerance, 66
label_adaptation, 55	spin_weight, 66
label_bests, 55	variable, 66
label_bits, 55	window, 67
label_estimates, 55	window
label_experiment, 55	interface.c, 192
label_generations, 56	interface.h, 237
label_iterations, 56	Window, 67
label_max, 56	window_about
label_min, 56	interface.c, 159
label_mutation, 56	window_add_experiment
label_p, 57	interface.c, 160
label_population, 57	window_add_variable
label_precision, 57	interface.c, 161
label_relaxation, 57	window_get_algorithm
label_reproduction, 57	interface.c, 161
label_result, 58	window_get_climbing
label_simulations, 58	interface.c, 162
label_simulator, 58	window_get_norm
label_step, 58	interface.c, 162
label_steps, 58	window help
label_sweeps, 59	interface.c, 162
label threshold, 59	window_inputs_experiment
label tolerance, 59	interface.c, 163
label variable, 59	window_label_variable
label variables, 59	interface.c, 163
label_weight, 60	window_name_experiment
_ •	
logo, 60	interface.c, 164
nexperiments, 60	window_new
nvariables, 60	interface.c, 165
scrolled_max, 60	interface.h, 227
scrolled_maxabs, 61	window_open
scrolled_min, 61	interface.c, 174
scrolled_minabs, 61 scrolled_p, 61	window_precision_variable interface.c, 175

```
window_rangemax_variable
    interface.c, 175
window_rangemaxabs_variable
    interface.c, 176
window_rangemin_variable
    interface.c, 176
window_rangeminabs_variable
    interface.c, 176
window read
    interface.c, 177
window_remove_experiment
    interface.c, 179
window_remove_variable
    interface.c, 180
window_run
    interface.c, 180
window save
    interface.c, 181
window_save_climbing
    interface.c, 182
window_set_algorithm
    interface.c, 183
window_set_experiment
    interface.c, 184
window_set_variable
    interface.c, 184
window_step_variable
    interface.c, 186
window_template_experiment
    interface.c, 186
window_template_experiment_close
    interface.c, 187
window_update
    interface.c, 188
window_update_climbing
    interface.c, 190
window_update_variable
    interface.c, 190
window_weight_experiment
    interface.c, 191
```