MPCOTool

Generated by Doxygen 1.9.4

1 Data Structure Index	1
1.1 Data Structures	 1
2 File Index	3
2.1 File List	 3
3 Data Structure Documentation	5
3.1 Experiment Struct Reference	 5
3.1.1 Detailed Description	 5
3.1.2 Field Documentation	 5
3.1.2.1 name	 6
3.1.2.2 ninputs	 6
3.1.2.3 stencil	 6
3.1.2.4 template_flags	 6
3.1.2.5 weight	 6
3.2 Input Struct Reference	 7
3.2.1 Detailed Description	 8
3.2.2 Field Documentation	 8
3.2.2.1 adaptation_ratio	 8
3.2.2.2 algorithm	 8
3.2.2.3 cleaner	 9
3.2.2.4 climbing	 9
3.2.2.5 directory	 9
3.2.2.6 evaluator	 9
3.2.2.7 experiment	 9
3.2.2.8 mutation_ratio	 10
3.2.2.9 name	10
3.2.2.10 nbest	 10
3.2.2.11 nestimates	 10
3.2.2.12 nexperiments	10
3.2.2.13 nfinal_steps	11
3.2.2.14 niterations	11
3.2.2.15 norm	11
3.2.2.16 nsimulations	11
3.2.2.17 nsteps	11
3.2.2.18 nvariables	12
3.2.2.19 p	12
3.2.2.20 relaxation	12
3.2.2.21 reproduction_ratio	12
3.2.2.22 result	12
3.2.2.23 seed	13
3.2.2.24 simulator	13
3.2.2.25 template_flags	13
0.2.2.20 template_nags	 13

3.2.2.26 threshold	. 13
3.2.2.27 tolerance	. 13
3.2.2.28 type	. 14
3.2.2.29 variable	. 14
3.2.2.30 variables	. 14
3.3 Optimize Struct Reference	. 14
3.3.1 Detailed Description	. 17
3.3.2 Field Documentation	. 17
3.3.2.1 adaptation_ratio	. 17
3.3.2.2 algorithm	. 17
3.3.2.3 calculation_time	. 17
3.3.2.4 cleaner	. 18
3.3.2.5 climbing	. 18
3.3.2.6 error_best	. 18
3.3.2.7 error_old	. 18
3.3.2.8 evaluator	. 18
3.3.2.9 experiment	. 19
3.3.2.10 file	. 19
3.3.2.11 file_result	. 19
3.3.2.12 file_variables	. 19
3.3.2.13 genetic_variable	. 19
3.3.2.14 label	. 20
3.3.2.15 mpi_rank	. 20
3.3.2.16 mutation_ratio	. 20
3.3.2.17 nbest	. 20
3.3.2.18 nbits	. 20
3.3.2.19 nend	. 21
3.3.2.20 nend_climbing	. 21
3.3.2.21 nestimates	. 21
3.3.2.22 nexperiments	. 21
3.3.2.23 nfinal_steps	. 21
3.3.2.24 ninputs	. 22
3.3.2.25 niterations	. 22
3.3.2.26 nsaveds	. 22
3.3.2.27 nsimulations	. 22
3.3.2.28 nstart	. 22
3.3.2.29 nstart_climbing	. 23
3.3.2.30 nsteps	. 23
3.3.2.31 nsweeps	. 23
3.3.2.32 nvariables	. 23
3.3.2.33 p	. 23
3.3.2.34 precision	. 24

3.3.2.35 rangemax	24
3.3.2.36 rangemaxabs	24
3.3.2.37 rangemin	24
3.3.2.38 rangeminabs	24
3.3.2.39 relaxation	25
3.3.2.40 reproduction_ratio	25
3.3.2.41 result	25
3.3.2.42 rng	25
3.3.2.43 seed	25
3.3.2.44 simulation_best	26
3.3.2.45 simulator	26
3.3.2.46 step	26
3.3.2.47 stop	26
3.3.2.48 template_flags	26
3.3.2.49 thread	27
3.3.2.50 thread_climbing	27
3.3.2.51 threshold	27
3.3.2.52 tolerance	27
3.3.2.53 value	27
3.3.2.54 value_old	28
3.3.2.55 variables	28
3.3.2.56 weight	28
3.4 Options Struct Reference	28
3.4.1 Detailed Description	29
3.4.2 Field Documentation	29
3.4.2.1 dialog	29
3.4.2.2 grid	29
3.4.2.3 label_climbing	29
3.4.2.4 label_seed	29
3.4.2.5 label_threads	30
3.4.2.6 spin_climbing	30
3.4.2.7 spin_seed	30
3.4.2.8 spin_threads	30
3.5 ParallelData Struct Reference	30
3.5.1 Detailed Description	31
3.5.2 Field Documentation	31
3.5.2.1 thread	31
3.6 Running Struct Reference	31
3.6.1 Detailed Description	32
3.6.2 Field Documentation	32
3.6.2.1 dialog	32
3.6.2.2 grid	32

3.6.2.3 label	. 32
3.6.2.4 spinner	. 32
3.7 Variable Struct Reference	. 33
3.7.1 Detailed Description	. 33
3.7.2 Field Documentation	. 33
3.7.2.1 name	. 33
3.7.2.2 nbits	. 34
3.7.2.3 nsweeps	. 34
3.7.2.4 precision	. 34
3.7.2.5 rangemax	. 34
3.7.2.6 rangemaxabs	. 34
3.7.2.7 rangemin	. 35
3.7.2.8 rangeminabs	. 35
3.7.2.9 step	. 35
3.8 Window Struct Reference	. 35
3.8.1 Detailed Description	. 40
3.8.2 Field Documentation	. 40
3.8.2.1 application_directory	. 40
3.8.2.2 box_buttons	. 41
3.8.2.3 button_about	. 41
3.8.2.4 button_add_experiment	. 41
3.8.2.5 button_add_variable	. 41
3.8.2.6 button_algorithm	. 41
3.8.2.7 button_cleaner	. 42
3.8.2.8 button_climbing	. 42
3.8.2.9 button_evaluator	. 42
3.8.2.10 button_exit	. 42
3.8.2.11 button_experiment	. 42
3.8.2.12 button_help	. 43
3.8.2.13 button_norm	. 43
3.8.2.14 button_open	. 43
3.8.2.15 button_options	. 43
3.8.2.16 button_remove_experiment	. 43
3.8.2.17 button_remove_variable	. 44
3.8.2.18 button_run	. 44
3.8.2.19 button_save	. 44
3.8.2.20 button_simulator	. 44
3.8.2.21 button_template	. 44
3.8.2.22 check_cleaner	. 45
3.8.2.23 check_climbing	. 45
3.8.2.24 check_evaluator	. 45
3.8.2.25 check_maxabs	. 45

3.8.2.26 check_minabs
3.8.2.27 check_template
3.8.2.28 combo_experiment
3.8.2.29 combo_variable
3.8.2.30 entry_result
3.8.2.31 entry_variable
3.8.2.32 entry_variables
3.8.2.33 experiment
3.8.2.34 frame_algorithm
3.8.2.35 frame_experiment
3.8.2.36 frame_norm
3.8.2.37 frame_variable
3.8.2.38 grid
3.8.2.39 grid_algorithm
3.8.2.40 grid_climbing
3.8.2.41 grid_experiment
3.8.2.42 grid_files
3.8.2.43 grid_norm
3.8.2.44 grid_variable
3.8.2.45 id_experiment
3.8.2.46 id_experiment_name
3.8.2.47 id_input
3.8.2.48 id_template
3.8.2.49 id_variable
3.8.2.50 id_variable_label
3.8.2.51 label_adaptation
3.8.2.52 label_bests
3.8.2.53 label_bits
3.8.2.54 label_estimates
3.8.2.55 label_experiment
3.8.2.56 label_final_steps
3.8.2.57 label_generations
3.8.2.58 label_iterations
3.8.2.59 label_max
3.8.2.60 label_min
3.8.2.61 label_mutation
3.8.2.62 label_p
3.8.2.63 label_population
3.8.2.64 label_precision
3.8.2.65 label_relaxation
3.8.2.66 label_reproduction
3.8.2.67 label_result

3.8.2.68 label_simulations
3.8.2.69 label_simulator
3.8.2.70 label_step
3.8.2.71 label_steps
3.8.2.72 label_sweeps
3.8.2.73 label_threshold
3.8.2.74 label_tolerance
3.8.2.75 label_variable
3.8.2.76 label_variables
3.8.2.77 label_weight
3.8.2.78 logo
3.8.2.79 nexperiments
3.8.2.80 nvariables
3.8.2.81 scrolled_max
3.8.2.82 scrolled_maxabs
3.8.2.83 scrolled_min
3.8.2.84 scrolled_minabs
3.8.2.85 scrolled_p
3.8.2.86 scrolled_step
3.8.2.87 scrolled_threshold
3.8.2.88 spin_adaptation
3.8.2.89 spin_bests
3.8.2.90 spin_bits
3.8.2.91 spin_estimates
3.8.2.92 spin_final_steps
3.8.2.93 spin_generations
3.8.2.94 spin_iterations
3.8.2.95 spin_max
3.8.2.96 spin_maxabs
3.8.2.97 spin_min
3.8.2.98 spin_minabs
3.8.2.99 spin_mutation
3.8.2.100 spin_p
3.8.2.101 spin_population
3.8.2.102 spin_precision
3.8.2.103 spin_relaxation
3.8.2.104 spin_reproduction
3.8.2.105 spin_simulations
3.8.2.106 spin_step
3.8.2.107 spin_steps
3.8.2.108 spin_sweeps
3.8.2.109 spin_threshold

3.8.2.110 spin_tolerance	 62
3.8.2.111 spin_weight	 62
3.8.2.112 variable	 63
3.8.2.113 window	 63
4 File Documentation	65
4.1 config.h File Reference	 65
4.2 config.h	
4.3 experiment.c File Reference	
4.3.1 Detailed Description	
4.3.2 Macro Definition Documentation	
4.3.2.1 DEBUG_EXPERIMENT	
4.3.3 Function Documentation	 68
4.3.3.1 experiment_error()	 68
4.3.3.2 experiment_free()	 68
4.3.3.3 experiment_new()	 69
4.3.3.4 experiment_open_json()	 69
4.3.3.5 experiment_open_xml()	 71
4.3.4 Variable Documentation	 73
4.3.4.1 stencil	 73
4.3.4.2 stencilbin	 74
4.4 experiment.c	 74
4.5 experiment.h File Reference	 78
4.5.1 Detailed Description	 79
4.5.2 Function Documentation	 79
4.5.2.1 experiment_error()	 79
4.5.2.2 experiment_free()	 80
4.5.2.3 experiment_open_json()	 80
4.5.2.4 experiment_open_xml()	 82
4.5.3 Variable Documentation	 84
4.5.3.1 stencil	 84
4.5.3.2 stencilbin	 84
4.6 experiment.h	 85
4.7 input.c File Reference	 85
4.7.1 Detailed Description	 86
4.7.2 Macro Definition Documentation	 86
4.7.2.1 DEBUG_INPUT	 87
4.7.3 Function Documentation	 87
4.7.3.1 input_error()	 87
4.7.3.2 input_free()	 87
4.7.3.3 input_new()	 88
4.7.3.4 input_open()	 88

4.7.3.5 input_open_json()	89
4.7.3.6 input_open_xml()	94
4.7.4 Variable Documentation	100
4.7.4.1 input	100
4.7.4.2 result_name	100
4.7.4.3 variables_name	100
4.8 input.c	101
4.9 input.h File Reference	113
4.9.1 Detailed Description	113
4.9.2 Enumeration Type Documentation	114
4.9.2.1 ClimbingMethod	114
4.9.2.2 ErrorNorm	114
4.9.3 Function Documentation	114
4.9.3.1 input_free()	115
4.9.3.2 input_new()	115
4.9.3.3 input_open()	116
4.9.4 Variable Documentation	117
4.9.4.1 input	117
4.9.4.2 result_name	117
4.9.4.3 variables_name	117
4.10 input.h	117
4.11 interface.c File Reference	118
4.11.1 Detailed Description	120
4.11.2 Macro Definition Documentation	121
4.11.2.1 DEBUG_INTERFACE	121
4.11.2.2 INPUT_FILE	121
4.11.3 Function Documentation	121
4.11.3.1 dialog_cleaner()	121
4.11.3.2 dialog_cleaner_close()	121
4.11.3.3 dialog_evaluator()	122
4.11.3.4 dialog_evaluator_close()	122
4.11.3.5 dialog_name_experiment_close()	123
4.11.3.6 dialog_open_close()	124
4.11.3.7 dialog_options_close()	125
4.11.3.8 dialog_save_close()	125
4.11.3.9 dialog_simulator()	127
4.11.3.10 dialog_simulator_close()	127
4.11.3.11 input_save()	128
4.11.3.12 input_save_climbing_json()	129
4.11.3.13 input_save_climbing_xml()	129
4.11.3.14 input_save_json()	130
4.11.3.15 input_save_xml()	132

4.11.3.16 options_new()	135
4.11.3.17 running_new()	136
4.11.3.18 window_about()	136
4.11.3.19 window_add_experiment()	137
4.11.3.20 window_add_variable()	137
4.11.3.21 window_get_algorithm()	138
4.11.3.22 window_get_climbing()	138
4.11.3.23 window_get_norm()	139
4.11.3.24 window_help()	139
4.11.3.25 window_inputs_experiment()	140
4.11.3.26 window_label_variable()	140
4.11.3.27 window_name_experiment()	140
4.11.3.28 window_new()	141
4.11.3.29 window_open()	151
4.11.3.30 window_precision_variable()	151
4.11.3.31 window_rangemax_variable()	152
4.11.3.32 window_rangemaxabs_variable()	152
4.11.3.33 window_rangemin_variable()	152
4.11.3.34 window_rangeminabs_variable()	153
4.11.3.35 window_read()	153
4.11.3.36 window_remove_experiment()	154
4.11.3.37 window_remove_variable()	155
4.11.3.38 window_run()	155
4.11.3.39 window_save()	156
4.11.3.40 window_save_climbing()	157
4.11.3.41 window_set_algorithm()	157
4.11.3.42 window_set_experiment()	158
4.11.3.43 window_set_variable()	158
4.11.3.44 window_step_variable()	159
4.11.3.45 window_template_experiment()	160
4.11.3.46 window_template_experiment_close()	160
4.11.3.47 window_update()	161
4.11.3.48 window_update_climbing()	163
4.11.3.49 window_update_variable()	164
4.11.3.50 window_weight_experiment()	164
4.11.4 Variable Documentation	164
4.11.4.1 logo	165
4.11.4.2 options	165
4.11.4.3 running	165
4.11.4.4 window	165
4.12 interface.c	166
4.13 interface h File Reference	วกก

4.13.1 Detailed Description
4.13.2 Macro Definition Documentation
4.13.2.1 MAX_LENGTH
4.13.3 Function Documentation
4.13.3.1 window_new()
4.13.4 Variable Documentation
4.13.4.1 window
4.14 interface.h
4.15 main.c File Reference
4.15.1 Detailed Description
4.15.2 Macro Definition Documentation
4.15.2.1 JBW
4.15.3 Function Documentation
4.15.3.1 main()
4.16 main.c
4.17 mpcotool.c File Reference
4.17.1 Detailed Description
4.17.2 Macro Definition Documentation
4.17.2.1 DEBUG_MPCOTOOL
4.17.3 Function Documentation
4.17.3.1 mpcotool()
4.18 mpcotool.c
4.19 mpcotool.h File Reference
4.19.1 Detailed Description
4.19.2 Function Documentation
4.19.2.1 mpcotool()
4.20 mpcotool.h
4.21 optimize.c File Reference
4.21.1 Detailed Description
4.21.2 Macro Definition Documentation
4.21.2.1 CP
4.21.2.2 DEBUG_OPTIMIZE
4.21.2.3 RM
4.21.3 Function Documentation
4.21.3.1 optimize_best()
4.21.3.2 optimize_best_climbing()
4.21.3.3 optimize_climbing()
4.21.3.4 optimize_climbing_best()
4.21.3.5 optimize_climbing_sequential()
4.21.3.6 optimize_climbing_thread()
4.21.3.7 optimize_estimate_climbing_coordinates()
4.21.3.8 optimize_estimate_climbing_random()

4.21.3.9 optimize_free()	3
4.21.3.10 optimize_genetic()	3
4.21.3.11 optimize_genetic_objective()	4
4.21.3.12 optimize_input()	5
4.21.3.13 optimize_iterate()	6
4.21.3.14 optimize_merge()	7
4.21.3.15 optimize_merge_old()	7
4.21.3.16 optimize_MonteCarlo()	8
4.21.3.17 optimize_norm_euclidian()	9
4.21.3.18 optimize_norm_maximum()	9
4.21.3.19 optimize_norm_p()	0
4.21.3.20 optimize_norm_taxicab()	1
4.21.3.21 optimize_open()	1
4.21.3.22 optimize_orthogonal()	5
4.21.3.23 optimize_parse()	6
4.21.3.24 optimize_print()	7
4.21.3.25 optimize_refine()	8
4.21.3.26 optimize_save_old()	9
4.21.3.27 optimize_save_optimal()	9
4.21.3.28 optimize_save_variables()	0
4.21.3.29 optimize_sequential()	0
4.21.3.30 optimize_step()	1
4.21.3.31 optimize_step_climbing()	1
4.21.3.32 optimize_sweep()	2
4.21.3.33 optimize_synchronise()	3
4.21.3.34 optimize_thread()	4
4.21.4 Variable Documentation	4
4.21.4.1 nthreads_climbing	5
4.21.4.2 optimize	
4.21.4.3 optimize_algorithm	5
4.21.4.4 optimize_estimate_climbing	5
4.21.4.5 optimize_norm	5
4.22 optimize.c	
4.23 optimize.h File Reference	
4.23.1 Detailed Description	6
4.23.2 Function Documentation	6
4.23.2.1 optimize_free()	6
4.23.2.2 optimize_open()	
4.23.3 Variable Documentation	
4.23.3.1 nthreads_climbing	
4.23.3.2 optimize	
4.24 optimize.h	1

4.25 tools.c File Reference
4.25.1 Detailed Description
4.25.2 Variable Documentation
4.25.2.1 error_message
4.25.2.2 main_window
4.25.2.3 show_pending
4.26 tools.c
4.27 tools.h File Reference
4.27.1 Detailed Description
4.27.2 Macro Definition Documentation
4.27.2.1 ERROR_TYPE
4.27.2.2 INFO_TYPE
4.27.3 Variable Documentation
4.27.3.1 error_message
4.27.3.2 main_window
4.27.3.3 show_pending
4.28 tools.h
4.29 variable.c File Reference
4.29.1 Detailed Description
4.29.2 Macro Definition Documentation
4.29.2.1 DEBUG_VARIABLE
4.29.3 Function Documentation
4.29.3.1 variable_error()
4.29.3.2 variable_free()
4.29.3.3 variable_open_json()
4.29.3.4 variable_open_xml()
4.29.4 Variable Documentation
4.29.4.1 format
4.29.4.2 precision
4.30 variable.c
4.31 variable.h File Reference
4.31.1 Detailed Description
4.31.2 Enumeration Type Documentation
4.31.2.1 Algorithm
4.31.3 Function Documentation
4.31.3.1 variable_error()
4.31.3.2 variable_free()
4.31.3.3 variable_open_json()
4.31.3.4 variable_open_xml()
4.31.4 Variable Documentation
4.31.4.1 format
4.31.4.2 precision

	xiii
4.32 variable.h	306
Index	309

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Experime	ant	
	Struct to define the experiment data	5
Input		
	Struct to define the optimization input file	7
Optimize		
	Struct to define the optimization ation data	14
Options		
	Struct to define the options dialog	28
ParallelD	Pata Pata	
	Struct to pass to the GThreads parallelized function	30
Running		
	Struct to define the running dialog	31
Variable		
	Struct to define the variable data	33
Window		
	Struct to define the main window	35

2 Data Structure Index

Chapter 2

File Index

2.1 File List

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

config.h		
	Configuration header file	65
experim	ent.c	
	Source file to define the experiment data	67
experim	ent.h	
	Header file to define the experiment data	78
input.c		
	Source file to define the input functions	85
input.h		
	Header file to define the input functions	113
interface		
	Source file to define the graphical interface functions	118
interface		
	Header file to define the graphical interface functions	200
main.c		
	Main source file	214
mpcotoc		
		217
mpcotoc		
	Main function header file	222
optimize		
	Source file to define the optimization functions	225
optimize		
	Header file to define the optimization functions	275
tools.c		
	Source file to define some useful functions	282
tools.h		
		284
variable.		
		287
variable.		
	Header file to define the variable data	299

File Index

Chapter 3

Data Structure Documentation

3.1 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.

• unsigned int template_flags

Flags of template files.

3.1.1 Detailed Description

Struct to define the experiment data.

Definition at line 45 of file experiment.h.

3.1.2 Field Documentation

3.1.2.1 name

char* Experiment::name

File name.

Definition at line 47 of file experiment.h.

3.1.2.2 ninputs

unsigned int Experiment::ninputs

Number of input files to the simulator.

Definition at line 50 of file experiment.h.

3.1.2.3 stencil

```
char* Experiment::stencil[MAX_NINPUTS]
```

Array of template names of input files.

Definition at line 48 of file experiment.h.

3.1.2.4 template_flags

```
{\tt unsigned\ int\ Experiment::template\_flags}
```

Flags of template files.

Definition at line 51 of file experiment.h.

3.1.2.5 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

3.2 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 * cleaner

Name of the cleaner program.

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 nfinal_steps

Number of steps to do the hill climbing method at the final pass.

· 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.

• unsigned int template_flags

Flags of template files.

3.2.1 Detailed Description

Struct to define the optimization input file.

Definition at line 65 of file input.h.

3.2.2 Field Documentation

3.2.2.1 adaptation_ratio

double Input::adaptation_ratio

Adaptation probability.

Definition at line 80 of file input.h.

3.2.2.2 algorithm

unsigned int Input::algorithm

Algorithm type.

Definition at line 89 of file input.h.

3.2.2.3 cleaner

char* Input::cleaner

Name of the cleaner program.

Definition at line 74 of file input.h.

3.2.2.4 climbing

unsigned int Input::climbing

Method to estimate the hill climbing.

Definition at line 94 of file input.h.

3.2.2.5 directory

char* Input::directory

Working directory.

Definition at line 75 of file input.h.

3.2.2.6 evaluator

char* Input::evaluator

Name of the program to evaluate the objective function.

Definition at line 72 of file input.h.

3.2.2.7 experiment

Experiment* Input::experiment

Array or experiments.

Definition at line 67 of file input.h.

3.2.2.8 mutation_ratio

double Input::mutation_ratio

Mutation probability.

Definition at line 78 of file input.h.

3.2.2.9 name

char* Input::name

Input data file name.

Definition at line 76 of file input.h.

3.2.2.10 nbest

unsigned int Input::nbest

Number of best simulations.

Definition at line 98 of file input.h.

3.2.2.11 nestimates

unsigned int Input::nestimates

Number of simulations to estimate the hill climbing.

Definition at line 95 of file input.h.

3.2.2.12 nexperiments

unsigned int Input::nexperiments

Experiments number.

Definition at line 87 of file input.h.

3.2.2.13 nfinal_steps

```
unsigned int Input::nfinal_steps
```

Number of steps to do the hill climbing method at the final pass.

Definition at line 92 of file input.h.

3.2.2.14 niterations

```
unsigned int Input::niterations
```

Number of algorithm iterations.

Definition at line 97 of file input.h.

3.2.2.15 norm

unsigned int Input::norm

Error norm type.

Definition at line 99 of file input.h.

3.2.2.16 nsimulations

unsigned int Input::nsimulations

Simulations number per experiment.

Definition at line 88 of file input.h.

3.2.2.17 nsteps

unsigned int Input::nsteps

Number of steps to do the hill climbing method.

Definition at line 90 of file input.h.

3.2.2.18 nvariables

unsigned int Input::nvariables

Variables number.

Definition at line 86 of file input.h.

3.2.2.19 p

double Input::p

Exponent of the P error norm.

Definition at line 82 of file input.h.

3.2.2.20 relaxation

double Input::relaxation

Relaxation parameter.

Definition at line 81 of file input.h.

3.2.2.21 reproduction_ratio

double Input::reproduction_ratio

Reproduction probability.

Definition at line 79 of file input.h.

3.2.2.22 result

char* Input::result

Name of the result file.

Definition at line 69 of file input.h.

3.2.2.23 seed

unsigned long int Input::seed

Seed of the pseudo-random numbers generator.

Definition at line 84 of file input.h.

3.2.2.24 simulator

char* Input::simulator

Name of the simulator program.

Definition at line 71 of file input.h.

3.2.2.25 template_flags

unsigned int Input::template_flags

Flags of template files.

Definition at line 101 of file input.h.

3.2.2.26 threshold

double Input::threshold

Threshold to finish the optimization.

Definition at line 83 of file input.h.

3.2.2.27 tolerance

double Input::tolerance

Algorithm tolerance.

Definition at line 77 of file input.h.

3.2.2.28 type

unsigned int Input::type

Type of input file.

Definition at line 100 of file input.h.

3.2.2.29 variable

```
Variable* Input::variable
```

Array of variables.

Definition at line 68 of file input.h.

3.2.2.30 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

3.3 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.

· char * cleaner

Name of the cleaner program.

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 nfinal_steps

Number of steps to do the hill climbing method at the final pass.

• 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.

• unsigned int template_flags

Flags of template files.

• int mpi_rank

Number of MPI task.

3.3.1 Detailed Description

Struct to define the optimization ation data.

Definition at line 45 of file optimize.h.

3.3.2 Field Documentation

3.3.2.1 adaptation_ratio

double Optimize::adaptation_ratio

Adaptation probability.

Definition at line 87 of file optimize.h.

3.3.2.2 algorithm

 ${\tt unsigned\ int\ Optimize::} {\tt algorithm}$

Algorithm type.

Definition at line 104 of file optimize.h.

3.3.2.3 calculation_time

 $\verb|double Optimize::calculation_time|\\$

Calculation time.

Definition at line 89 of file optimize.h.

3.3.2.4 cleaner

char* Optimize::cleaner

Name of the cleaner program.

Definition at line 60 of file optimize.h.

3.3.2.5 climbing

double* Optimize::climbing

Vector of hill climbing estimation.

Definition at line 69 of file optimize.h.

3.3.2.6 error_best

double* Optimize::error_best

Array of the best minimum errors.

Definition at line 66 of file optimize.h.

3.3.2.7 error_old

double* Optimize::error_old

Array of the best minimum errors on the previous step.

Definition at line 72 of file optimize.h.

3.3.2.8 evaluator

char* Optimize::evaluator

Name of the program to evaluate the objective function.

Definition at line 58 of file optimize.h.

3.3.2.9 experiment

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

Array of experimental data file names.

Definition at line 48 of file optimize.h.

3.3.2.10 file

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

Matrix of input template files.

Definition at line 47 of file optimize.h.

3.3.2.11 file_result

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

Result file.

Definition at line 53 of file optimize.h.

3.3.2.12 file_variables

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

Variables file.

Definition at line 54 of file optimize.h.

3.3.2.13 genetic_variable

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

Array of variables for the genetic algorithm.

Definition at line 51 of file optimize.h.

3.3.2.14 label

char** Optimize::label

Array of variable names.

Definition at line 49 of file optimize.h.

3.3.2.15 mpi_rank

int Optimize::mpi_rank

Number of MPI task.

Definition at line 117 of file optimize.h.

3.3.2.16 mutation_ratio

double Optimize::mutation_ratio

Mutation probability.

Definition at line 85 of file optimize.h.

3.3.2.17 nbest

unsigned int Optimize::nbest

Number of best simulations.

Definition at line 112 of file optimize.h.

3.3.2.18 nbits

unsigned int* Optimize::nbits

Array of bits number of the genetic algorithm.

Definition at line 76 of file optimize.h.

3.3.2.19 nend

unsigned int Optimize::nend

Ending simulation number of the task.

Definition at line 106 of file optimize.h.

3.3.2.20 nend_climbing

unsigned int Optimize::nend_climbing

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

Definition at line 109 of file optimize.h.

3.3.2.21 nestimates

unsigned int Optimize::nestimates

Number of simulations to estimate the climbing.

Definition at line 102 of file optimize.h.

3.3.2.22 nexperiments

unsigned int Optimize::nexperiments

Experiments number.

Definition at line 95 of file optimize.h.

3.3.2.23 nfinal_steps

unsigned int Optimize::nfinal_steps

Number of steps to do the hill climbing method at the final pass.

Definition at line 100 of file optimize.h.

3.3.2.24 ninputs

unsigned int Optimize::ninputs

Number of input files to the simulator.

Definition at line 96 of file optimize.h.

3.3.2.25 niterations

unsigned int Optimize::niterations

Number of algorithm iterations.

Definition at line 111 of file optimize.h.

3.3.2.26 nsaveds

unsigned int Optimize::nsaveds

Number of saved simulations.

Definition at line 113 of file optimize.h.

3.3.2.27 nsimulations

unsigned int Optimize::nsimulations

Simulations number per experiment.

Definition at line 97 of file optimize.h.

3.3.2.28 nstart

unsigned int Optimize::nstart

Beginning simulation number of the task.

Definition at line 105 of file optimize.h.

3.3.2.29 nstart_climbing

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

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

Definition at line 107 of file optimize.h.

3.3.2.30 nsteps

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

Number of steps for the hill climbing method.

Definition at line 98 of file optimize.h.

3.3.2.31 nsweeps

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

Array of sweeps of the sweep algorithm.

Definition at line 75 of file optimize.h.

3.3.2.32 nvariables

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

Variables number.

Definition at line 94 of file optimize.h.

3.3.2.33 p

double Optimize::p

Exponent of the P error norm.

Definition at line 90 of file optimize.h.

3.3.2.34 precision

unsigned int* Optimize::precision

Array of variable precisions.

Definition at line 74 of file optimize.h.

3.3.2.35 rangemax

double* Optimize::rangemax

Array of maximum variable values.

Definition at line 63 of file optimize.h.

3.3.2.36 rangemaxabs

double* Optimize::rangemaxabs

Array of absolute maximum variable values.

Definition at line 65 of file optimize.h.

3.3.2.37 rangemin

double* Optimize::rangemin

Array of minimum variable values.

Definition at line 62 of file optimize.h.

3.3.2.38 rangeminabs

double* Optimize::rangeminabs

Array of absolute minimum variable values.

Definition at line 64 of file optimize.h.

3.3.2.39 relaxation

double Optimize::relaxation

Relaxation parameter.

Definition at line 88 of file optimize.h.

3.3.2.40 reproduction_ratio

double Optimize::reproduction_ratio

Reproduction probability.

Definition at line 86 of file optimize.h.

3.3.2.41 result

char* Optimize::result

Name of the result file.

Definition at line 55 of file optimize.h.

3.3.2.42 rng

gsl_rng* Optimize::rng

GSL random number generator.

Definition at line 50 of file optimize.h.

3.3.2.43 seed

unsigned long int Optimize::seed

Seed of the pseudo-random numbers generator.

Definition at line 92 of file optimize.h.

3.3.2.44 simulation_best

unsigned int* Optimize::simulation_best

Array of best simulation numbers.

Definition at line 83 of file optimize.h.

3.3.2.45 simulator

char* Optimize::simulator

Name of the simulator program.

Definition at line 57 of file optimize.h.

3.3.2.46 step

double* Optimize::step

Array of hill climbing method step sizes.

Definition at line 68 of file optimize.h.

3.3.2.47 stop

unsigned int Optimize::stop

To stop the simulations.

Definition at line 114 of file optimize.h.

3.3.2.48 template_flags

unsigned int Optimize::template_flags

Flags of template files.

Definition at line 115 of file optimize.h.

3.3.2.49 thread

```
unsigned int* Optimize::thread
```

Array of simulation numbers to calculate on the thread.

Definition at line 78 of file optimize.h.

3.3.2.50 thread_climbing

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

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

Definition at line 80 of file optimize.h.

3.3.2.51 threshold

double Optimize::threshold

Threshold to finish the optimization.

Definition at line 91 of file optimize.h.

3.3.2.52 tolerance

double Optimize::tolerance

Algorithm tolerance.

Definition at line 84 of file optimize.h.

3.3.2.53 value

double* Optimize::value

Array of variable values.

Definition at line 61 of file optimize.h.

3.3.2.54 value_old

```
double* Optimize::value_old
```

Array of the best variable values on the previous step.

Definition at line 70 of file optimize.h.

3.3.2.55 variables

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

Name of the variables file.

Definition at line 56 of file optimize.h.

3.3.2.56 weight

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

Array of the experiment weights.

Definition at line 67 of file optimize.h.

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

· optimize.h

3.4 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

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

Climbing threads number GtkSpinButton.

3.4.1 Detailed Description

Struct to define the options dialog.

Definition at line 48 of file interface.h.

3.4.2 Field Documentation

3.4.2.1 dialog

GtkDialog* Options::dialog

Main GtkDialog.

Definition at line 50 of file interface.h.

3.4.2.2 grid

GtkGrid* Options::grid

Main GtkGrid.

Definition at line 51 of file interface.h.

3.4.2.3 label_climbing

GtkLabel* Options::label_climbing

Climbing threads number GtkLabel.

Definition at line 58 of file interface.h.

3.4.2.4 label_seed

GtkLabel* Options::label_seed

Pseudo-random numbers generator seed GtkLabel.

Definition at line 52 of file interface.h.

3.4.2.5 label_threads

GtkLabel* Options::label_threads

Threads number GtkLabel.

Definition at line 56 of file interface.h.

3.4.2.6 spin_climbing

GtkSpinButton* Options::spin_climbing

Climbing threads number GtkSpinButton.

Definition at line 59 of file interface.h.

3.4.2.7 spin_seed

GtkSpinButton* Options::spin_seed

Pseudo-random numbers generator seed GtkSpinButton.

Definition at line 54 of file interface.h.

3.4.2.8 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

3.5 ParallelData Struct Reference

Struct to pass to the GThreads parallelized function.

#include <optimize.h>

Data Fields

· unsigned int thread

Thread number.

3.5.1 Detailed Description

Struct to pass to the GThreads parallelized function.

Definition at line 125 of file optimize.h.

3.5.2 Field Documentation

3.5.2.1 thread

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

Thread number.

Definition at line 127 of file optimize.h.

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

• optimize.h

3.6 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.

3.6.1 Detailed Description

Struct to define the running dialog.

Definition at line 66 of file interface.h.

3.6.2 Field Documentation

3.6.2.1 dialog

GtkDialog* Running::dialog

Main GtkDialog.

Definition at line 68 of file interface.h.

3.6.2.2 grid

GtkGrid* Running::grid

Grid GtkGrid.

Definition at line 71 of file interface.h.

3.6.2.3 label

GtkLabel* Running::label

Label GtkLabel.

Definition at line 69 of file interface.h.

3.6.2.4 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

3.7 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.

3.7.1 Detailed Description

Struct to define the variable data.

Definition at line 54 of file variable.h.

3.7.2 Field Documentation

3.7.2.1 name

char* Variable::name

Variable name.

Definition at line 56 of file variable.h.

3.7.2.2 nbits

unsigned int Variable::nbits

Bits number of the genetic algorithm.

Definition at line 64 of file variable.h.

3.7.2.3 nsweeps

unsigned int Variable::nsweeps

Sweeps of the sweep algorithm.

Definition at line 63 of file variable.h.

3.7.2.4 precision

unsigned int Variable::precision

Variable precision.

Definition at line 62 of file variable.h.

3.7.2.5 rangemax

double Variable::rangemax

Maximum variable value.

Definition at line 58 of file variable.h.

3.7.2.6 rangemaxabs

double Variable::rangemaxabs

Absolute maximum variable value.

Definition at line 60 of file variable.h.

3.7.2.7 rangemin

double Variable::rangemin

Minimum variable value.

Definition at line 57 of file variable.h.

3.7.2.8 rangeminabs

double Variable::rangeminabs

Absolute minimum variable value.

Definition at line 59 of file variable.h.

3.7.2.9 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

3.8 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.

• GtkCheckButton * check_cleaner

Cleaner program GtkCheckButton.

• GtkButton * button_cleaner

Cleaner 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.

GtkSpinButton * spin population

GtkSpinButton to set the population number.

GtkLabel * label_generations

GtkLabel to set the generations number.

• GtkSpinButton * spin_generations

 ${\it GtkSpinButton\ to\ set\ the\ generations\ number.}$

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_final_steps

GtkLabel to set the final steps number.

GtkSpinButton * spin_final_steps

GtkSpinButton to set the final 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.

GtkComboBoxText * 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.

3.8.1 Detailed Description

Struct to define the main window.

Definition at line 78 of file interface.h.

3.8.2 Field Documentation

3.8.2.1 application_directory

char* Window::application_directory

Application directory.

Definition at line 220 of file interface.h.

3.8.2.2 box_buttons

GtkBox* Window::box_buttons

GtkBox to store the main buttons.

Definition at line 82 of file interface.h.

3.8.2.3 button_about

GtkButton* Window::button_about

Help GtkButton.

Definition at line 88 of file interface.h.

3.8.2.4 button_add_experiment

GtkButton* Window::button_add_experiment

GtkButton to add a experiment.

Definition at line 206 of file interface.h.

3.8.2.5 button_add_variable

GtkButton* Window::button_add_variable

GtkButton to add a variable.

Definition at line 178 of file interface.h.

3.8.2.6 button_algorithm

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

Array of GtkRadioButtons to set the algorithm.

Definition at line 117 of file interface.h.

3.8.2.7 button_cleaner

GtkButton* Window::button_cleaner

Cleaner program GtkButton.

Definition at line 96 of file interface.h.

3.8.2.8 button_climbing

GtkRadioButton* Window::button_climbing[NCLIMBINGS]

Array of GtkRadioButtons array to set the hill climbing method.

Definition at line 152 of file interface.h.

3.8.2.9 button evaluator

GtkButton* Window::button_evaluator

Evaluator program GtkButton.

Definition at line 94 of file interface.h.

3.8.2.10 button_exit

GtkButton* Window::button_exit

Exit GtkButton.

Definition at line 89 of file interface.h.

3.8.2.11 button_experiment

GtkButton* Window::button_experiment

GtkButton to set the experimental data file.

Definition at line 209 of file interface.h.

3.8.2.12 button_help

GtkButton* Window::button_help

Help GtkButton.

Definition at line 87 of file interface.h.

3.8.2.13 button_norm

GtkRadioButton* Window::button_norm[NNORMS]

Array of GtkRadioButtons to set the error norm.

Definition at line 104 of file interface.h.

3.8.2.14 button_open

GtkButton* Window::button_open

Open GtkButton.

Definition at line 83 of file interface.h.

3.8.2.15 button_options

GtkButton* Window::button_options

Options GtkButton.

Definition at line 86 of file interface.h.

3.8.2.16 button_remove_experiment

GtkButton* Window::button_remove_experiment

GtkButton to remove a experiment.

Definition at line 207 of file interface.h.

3.8.2.17 button_remove_variable

GtkButton* Window::button_remove_variable

GtkButton to remove a variable.

Definition at line 179 of file interface.h.

3.8.2.18 button_run

GtkButton* Window::button_run

Run GtkButton.

Definition at line 85 of file interface.h.

3.8.2.19 button save

GtkButton* Window::button_save

Save GtkButton.

Definition at line 84 of file interface.h.

3.8.2.20 button_simulator

GtkButton* Window::button_simulator

Simulator program GtkButton.

Definition at line 92 of file interface.h.

3.8.2.21 button_template

GtkButton* Window::button_template[MAX_NINPUTS]

Array of GtkButtons to set the input templates.

Definition at line 215 of file interface.h.

3.8.2.22 check_cleaner

GtkCheckButton* Window::check_cleaner

Cleaner program GtkCheckButton.

Definition at line 95 of file interface.h.

3.8.2.23 check_climbing

GtkCheckButton* Window::check_climbing

GtkCheckButton to check running the hill climbing method.

Definition at line 147 of file interface.h.

3.8.2.24 check evaluator

GtkCheckButton* Window::check_evaluator

Evaluator program GtkCheckButton.

Definition at line 93 of file interface.h.

3.8.2.25 check_maxabs

GtkCheckButton* Window::check_maxabs

Absolute maximum GtkCheckButton.

Definition at line 191 of file interface.h.

3.8.2.26 check_minabs

GtkCheckButton* Window::check_minabs

Absolute minimum GtkCheckButton.

Definition at line 188 of file interface.h.

3.8.2.27 check_template

GtkCheckButton* Window::check_template[MAX_NINPUTS]

Array of GtkCheckButtons to set the input templates.

Definition at line 213 of file interface.h.

3.8.2.28 combo_experiment

GtkComboBoxText* Window::combo_experiment

Experiment GtkComboBoxEntry.

Definition at line 205 of file interface.h.

3.8.2.29 combo variable

GtkComboBoxText* Window::combo_variable

GtkComboBoxEntry to select a variable.

Definition at line 176 of file interface.h.

3.8.2.30 entry_result

GtkEntry* Window::entry_result

Result file GtkEntry.

Definition at line 98 of file interface.h.

3.8.2.31 entry_variable

GtkEntry* Window::entry_variable

GtkEntry to set the variable name.

Definition at line 181 of file interface.h.

3.8.2.32 entry_variables

GtkEntry* Window::entry_variables

Variables file GtkEntry.

Definition at line 100 of file interface.h.

3.8.2.33 experiment

Experiment* Window::experiment

Array of experiments data.

Definition at line 218 of file interface.h.

3.8.2.34 frame_algorithm

GtkFrame* Window::frame_algorithm

GtkFrame to set the algorithm.

Definition at line 114 of file interface.h.

3.8.2.35 frame_experiment

GtkFrame* Window::frame_experiment

Experiment GtkFrame.

Definition at line 203 of file interface.h.

3.8.2.36 frame_norm

GtkFrame* Window::frame_norm

GtkFrame to set the error norm.

Definition at line 101 of file interface.h.

3.8.2.37 frame_variable

GtkFrame* Window::frame_variable

Variable GtkFrame.

Definition at line 174 of file interface.h.

3.8.2.38 grid

GtkGrid* Window::grid

Main GtkGrid.

Definition at line 81 of file interface.h.

3.8.2.39 grid_algorithm

GtkGrid* Window::grid_algorithm

GtkGrid to set the algorithm.

Definition at line 115 of file interface.h.

3.8.2.40 grid_climbing

GtkGrid* Window::grid_climbing

GtkGrid to pack the hill climbing method widgets.

Definition at line 149 of file interface.h.

3.8.2.41 grid_experiment

GtkGrid* Window::grid_experiment

Experiment GtkGrid.

Definition at line 204 of file interface.h.

3.8.2.42 grid_files

GtkGrid* Window::grid_files

Files GtkGrid.

Definition at line 90 of file interface.h.

3.8.2.43 grid_norm

GtkGrid* Window::grid_norm

GtkGrid to set the error norm.

Definition at line 102 of file interface.h.

3.8.2.44 grid_variable

GtkGrid* Window::grid_variable

Variable GtkGrid.

Definition at line 175 of file interface.h.

3.8.2.45 id_experiment

gulong Window::id_experiment

Identifier of the combo_experiment signal.

Definition at line 221 of file interface.h.

3.8.2.46 id_experiment_name

gulong Window::id_experiment_name

Identifier of the button_experiment signal.

Definition at line 222 of file interface.h.

3.8.2.47 id_input

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

Array of identifiers of the button_template signal.

Definition at line 227 of file interface.h.

3.8.2.48 id_template

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

Array of identifiers of the check_template signal.

Definition at line 225 of file interface.h.

3.8.2.49 id variable

```
gulong Window::id_variable
```

Identifier of the combo_variable signal.

Definition at line 223 of file interface.h.

3.8.2.50 id_variable_label

```
gulong Window::id_variable_label
```

Identifier of the entry_variable signal.

Definition at line 224 of file interface.h.

3.8.2.51 label_adaptation

```
GtkLabel* Window::label_adaptation
```

GtkLabel to set the adaptation ratio.

Definition at line 144 of file interface.h.

3.8.2.52 label_bests

GtkLabel* Window::label_bests

GtkLabel to set the best number.

Definition at line 131 of file interface.h.

3.8.2.53 label_bits

GtkLabel* Window::label_bits

Bits number GtkLabel.

Definition at line 198 of file interface.h.

3.8.2.54 label estimates

GtkLabel* Window::label_estimates

GtkLabel to set the estimates number.

Definition at line 163 of file interface.h.

3.8.2.55 label_experiment

GtkLabel* Window::label_experiment

Experiment GtkLabel.

Definition at line 208 of file interface.h.

3.8.2.56 label_final_steps

GtkLabel* Window::label_final_steps

GtkLabel to set the final steps number.

Definition at line 160 of file interface.h.

3.8.2.57 label_generations

GtkLabel* Window::label_generations

GtkLabel to set the generations number.

Definition at line 136 of file interface.h.

3.8.2.58 label_iterations

GtkLabel* Window::label_iterations

GtkLabel to set the iterations number.

Definition at line 126 of file interface.h.

3.8.2.59 label max

GtkLabel* Window::label_max

Maximum GtkLabel.

Definition at line 185 of file interface.h.

3.8.2.60 label_min

GtkLabel* Window::label_min

Minimum GtkLabel.

Definition at line 182 of file interface.h.

3.8.2.61 label_mutation

GtkLabel* Window::label_mutation

GtkLabel to set the mutation ratio.

Definition at line 139 of file interface.h.

3.8.2.62 label_p

GtkLabel* Window::label_p

GtkLabel to set the p parameter.

Definition at line 110 of file interface.h.

3.8.2.63 label_population

GtkLabel* Window::label_population

GtkLabel to set the population number.

Definition at line 133 of file interface.h.

3.8.2.64 label_precision

GtkLabel* Window::label_precision

Precision GtkLabel.

Definition at line 194 of file interface.h.

3.8.2.65 label_relaxation

GtkLabel* Window::label_relaxation

GtkLabel to set the relaxation parameter.

Definition at line 166 of file interface.h.

3.8.2.66 label_reproduction

GtkLabel* Window::label_reproduction

GtkLabel to set the reproduction ratio.

Definition at line 141 of file interface.h.

3.8.2.67 label_result

GtkLabel* Window::label_result

Result file GtkLabel.

Definition at line 97 of file interface.h.

3.8.2.68 label_simulations

GtkLabel* Window::label_simulations

GtkLabel to set the simulations number.

Definition at line 123 of file interface.h.

3.8.2.69 label simulator

GtkLabel* Window::label_simulator

Simulator program GtkLabel.

Definition at line 91 of file interface.h.

3.8.2.70 label_step

GtkLabel* Window::label_step

GtkLabel to set the step.

Definition at line 200 of file interface.h.

3.8.2.71 label_steps

GtkLabel* Window::label_steps

GtkLabel to set the steps number.

Definition at line 158 of file interface.h.

3.8.2.72 label_sweeps

GtkLabel* Window::label_sweeps

Sweeps number GtkLabel.

Definition at line 196 of file interface.h.

3.8.2.73 label_threshold

GtkLabel* Window::label_threshold

GtkLabel to set the threshold.

Definition at line 170 of file interface.h.

3.8.2.74 label_tolerance

GtkLabel* Window::label_tolerance

GtkLabel to set the tolerance.

Definition at line 129 of file interface.h.

3.8.2.75 label_variable

GtkLabel* Window::label_variable

Variable GtkLabel.

Definition at line 180 of file interface.h.

3.8.2.76 label_variables

GtkLabel* Window::label_variables

Variables file GtkLabel.

Definition at line 99 of file interface.h.

3.8.2.77 label_weight

GtkLabel* Window::label_weight

Weight GtkLabel.

Definition at line 211 of file interface.h.

3.8.2.78 logo

GdkPixbuf* Window::logo

Logo GdkPixbuf.

Definition at line 217 of file interface.h.

3.8.2.79 nexperiments

unsigned int Window::nexperiments

Number of experiments.

Definition at line 229 of file interface.h.

3.8.2.80 nvariables

unsigned int Window::nvariables

Number of variables.

Definition at line 230 of file interface.h.

3.8.2.81 scrolled_max

GtkScrolledWindow* Window::scrolled_max

Maximum GtkScrolledWindow.

Definition at line 187 of file interface.h.

3.8.2.82 scrolled_maxabs

GtkScrolledWindow* Window::scrolled_maxabs

Absolute maximum GtkScrolledWindow.

Definition at line 193 of file interface.h.

3.8.2.83 scrolled_min

GtkScrolledWindow* Window::scrolled_min

Minimum GtkScrolledWindow.

Definition at line 184 of file interface.h.

3.8.2.84 scrolled minabs

GtkScrolledWindow* Window::scrolled_minabs

Absolute minimum GtkScrolledWindow.

Definition at line 190 of file interface.h.

3.8.2.85 scrolled_p

GtkScrolledWindow* Window::scrolled_p

GtkScrolledWindow to set the p parameter.

Definition at line 112 of file interface.h.

3.8.2.86 scrolled_step

GtkScrolledWindow* Window::scrolled_step

step GtkScrolledWindow.

Definition at line 202 of file interface.h.

3.8.2.87 scrolled_threshold

GtkScrolledWindow* Window::scrolled_threshold

GtkScrolledWindow to set the threshold.

Definition at line 172 of file interface.h.

3.8.2.88 spin_adaptation

 ${\tt GtkSpinButton*\ Window::spin_adaptation}$

GtkSpinButton to set the adaptation ratio.

Definition at line 145 of file interface.h.

3.8.2.89 spin bests

GtkSpinButton* Window::spin_bests

GtkSpinButton to set the best number.

Definition at line 132 of file interface.h.

3.8.2.90 spin_bits

GtkSpinButton* Window::spin_bits

Bits number GtkSpinButton.

Definition at line 199 of file interface.h.

3.8.2.91 spin_estimates

GtkSpinButton* Window::spin_estimates

GtkSpinButton to set the estimates number.

Definition at line 164 of file interface.h.

3.8.2.92 spin_final_steps

GtkSpinButton* Window::spin_final_steps

GtkSpinButton to set the final steps number.

Definition at line 161 of file interface.h.

3.8.2.93 spin_generations

GtkSpinButton* Window::spin_generations

GtkSpinButton to set the generations number.

Definition at line 137 of file interface.h.

3.8.2.94 spin_iterations

GtkSpinButton* Window::spin_iterations

GtkSpinButton to set the iterations number.

Definition at line 127 of file interface.h.

3.8.2.95 spin_max

GtkSpinButton* Window::spin_max

Maximum GtkSpinButton.

Definition at line 186 of file interface.h.

3.8.2.96 spin_maxabs

GtkSpinButton* Window::spin_maxabs

Absolute maximum GtkSpinButton.

Definition at line 192 of file interface.h.

3.8.2.97 spin_min

GtkSpinButton* Window::spin_min

Minimum GtkSpinButton.

Definition at line 183 of file interface.h.

3.8.2.98 spin_minabs

GtkSpinButton* Window::spin_minabs

Absolute minimum GtkSpinButton.

Definition at line 189 of file interface.h.

3.8.2.99 spin_mutation

GtkSpinButton* Window::spin_mutation

GtkSpinButton to set the mutation ratio.

Definition at line 140 of file interface.h.

3.8.2.100 spin_p

GtkSpinButton* Window::spin_p

GtkSpinButton to set the p parameter.

Definition at line 111 of file interface.h.

3.8.2.101 spin_population

 ${\tt GtkSpinButton*\ Window::spin_population}$

GtkSpinButton to set the population number.

Definition at line 134 of file interface.h.

3.8.2.102 spin_precision

GtkSpinButton* Window::spin_precision

Precision digits GtkSpinButton.

Definition at line 195 of file interface.h.

3.8.2.103 spin_relaxation

GtkSpinButton* Window::spin_relaxation

GtkSpinButton to set the relaxation parameter.

Definition at line 168 of file interface.h.

3.8.2.104 spin reproduction

GtkSpinButton* Window::spin_reproduction

GtkSpinButton to set the reproduction ratio.

Definition at line 142 of file interface.h.

3.8.2.105 spin_simulations

GtkSpinButton* Window::spin_simulations

GtkSpinButton to set the simulations number.

Definition at line 124 of file interface.h.

3.8.2.106 spin_step

GtkSpinButton* Window::spin_step

GtkSpinButton to set the step.

Definition at line 201 of file interface.h.

3.8.2.107 spin_steps

GtkSpinButton* Window::spin_steps

GtkSpinButton to set the steps number.

Definition at line 159 of file interface.h.

3.8.2.108 spin_sweeps

GtkSpinButton* Window::spin_sweeps

Sweeps number GtkSpinButton.

Definition at line 197 of file interface.h.

3.8.2.109 spin_threshold

GtkSpinButton* Window::spin_threshold

GtkSpinButton to set the threshold.

Definition at line 171 of file interface.h.

3.8.2.110 spin_tolerance

GtkSpinButton* Window::spin_tolerance

GtkSpinButton to set the tolerance.

Definition at line 130 of file interface.h.

3.8.2.111 spin_weight

GtkSpinButton* Window::spin_weight

 $Weight\ GtkSpinButton.$

Definition at line 212 of file interface.h.

3.8.2.112 variable

Variable* Window::variable

Array of variables data.

Definition at line 219 of file interface.h.

3.8.2.113 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 4

File Documentation

4.1 config.h File Reference

Configuration header file.

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

4.2 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.
80000
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.
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_
00040 #define CONFIG__H 1
00042 #define HAVE_MPI 1
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_CLEANER "cleaner"
00073 #define LABEL_CLIMBING "climbing"
00074 #define LABEL_COORDINATES "coordinates"
00075 #define LABEL_EUCLIDIAN "euclidian"
00076 #define LABEL_EVALUATOR "evaluator"
00077 #define LABEL_EXPERIMENT "experiment"
00078 #define LABEL_EXPERIMENTS "experiments"
00079 #define LABEL_GENETIC "genetic" 00080 #define LABEL_INPUT1 "input1"
00081 #define LABEL_INPUT2 "input2"
00082 #define LABEL_INPUT3 "input3"
00083 #define LABEL_INPUT4 "input4"
00084 #define LABEL_INPUT5 "input5"
00085 #define LABEL_INPUT6 "input6"
00086 #define LABEL_INPUT7 "input7"
00087 #define LABEL_INPUT8 "input8"
00088 #define LABEL_MINIMUM "minimum"
00089 #define LABEL_MAXIMUM "maximum"
00090 #define LABEL_MONTE_CARLO "Monte-Carlo"
00091 #define LABEL_MUTATION "mutation"
00092 #define LABEL_NAME "name"
00093 #define LABEL_NBEST "nbest"
00094 #define LABEL_NBITS "nbits"
00095 #define LABEL_NESTIMATES "nestimates"
00096 #define LABEL_NFINAL_STEPS "nfinal_steps"
00097 #define LABEL_NGENERATIONS "ngenerations"
00098 #define LABEL_NITERATIONS "niterations" 00099 #define LABEL_NORM "norm"
00100 #define LABEL_NPOPULATION "npopulation"
00101 #define LABEL_NSIMULATIONS "nsimulations"
00102 #define LABEL_NSTEPS "nsteps"
00103 #define LABEL_NSWEEPS "nsweeps"
00104 #define LABEL_OPTIMIZE "optimize"
00105 #define LABEL_ORTHOGONAL "orthogonal"
00106 #define LABEL_P "p"
00107 #define LABEL_PRECISION "precision"
00108 #define LABEL_RANDOM "random"
00109 #define LABEL_RELAXATION "relaxation"
00110 #define LABEL_REPRODUCTION "reproduction"
00111 #define LABEL_RESULT_FILE "result_file"
00112 #define LABEL_SIMULATOR "simulator"
00113 #define LABEL_SEED "seed"
00114 #define LABEL_STEP "step"
00115 #define LABEL_SWEEP "sweep"
00116 #define LABEL_TAXICAB "taxicab"
00117 #define LABEL_TEMPLATE1 "template1" 00118 #define LABEL_TEMPLATE2 "template2"
00119 #define LABEL_TEMPLATE3 "template3"
00120 #define LABEL_TEMPLATE4 "template4"
00121 #define LABEL_TEMPLATE5 "template5"
00122 #define LABEL_TEMPLATE6 "template6"
00123 #define LABEL_TEMPLATE7 "template7"
00124 #define LABEL_TEMPLATE8 "template8"
00125 #define LABEL_THRESHOLD "threshold"
00126 #define LABEL_TOLERANCE "tolerance"
00127 #define LABEL_VARIABLE "variable"
00128 #define LABEL_VARIABLES "variables"
00129 #define LABEL_VARIABLES_FILE "variables_file" 00130 #define LABEL_WEIGHT "weight"
00131
00132 // Enumerations
00133
00135 enum INPUT_TYPE
00136 {
        INPUT_TYPE_XML = 0,
00137
00138
        INPUT_TYPE_JSON = 1
00139 };
```

```
00140
00141 #endif
```

4.3 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/xml.h"
#include "jb/src/json.h"
#include "jb/src/win.h"
#include "tools.h"
#include "experiment.h"
```

Include dependency graph for 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 strings with stencil labels.

const char * stencilbin [MAX NINPUTS]

Array of strings with binary stencil labels.

4.3.1 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.

4.3.2 Macro Definition Documentation

4.3.2.1 DEBUG_EXPERIMENT

```
#define DEBUG_EXPERIMENT 0
```

Macro to debug experiment functions.

Definition at line 51 of file experiment.c.

4.3.3 Function Documentation

4.3.3.1 experiment_error()

Function to print a message error opening an Experiment struct.

Parameters

experiment	Experiment struct.
message	Error message.

Definition at line 117 of file experiment.c.

4.3.3.2 experiment_free()

Function to free the memory of an Experiment struct.

Parameters

experiment	Experiment struct.
type	Type of input file.

Definition at line 88 of file experiment.c.

```
00091
        unsigned int i;
00092 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_free: start\n");
00093
00094 #endif
      if (type == INPUT_TYPE_XML)
00096
00097
           for (i = 0; i < experiment->ninputs; ++i)
00098
             xmlFree (experiment->stencil[i]);
00099
            xmlFree (experiment->name);
00100
00101
       else
00102
        {
00103
            for (i = 0; i < experiment->ninputs; ++i)
         g_free (experiment->sten
g_free (experiment->name);
}
             g_free (experiment->stencil[i]);
00104
00105
00106
00107 experiment->ninputs = experiment->template_flags = 0;
00108 #if DEBUG_EXPERIMENT
00109 fprintf (stderr, "experiment_free: end\n");
00110 #endif
00111 }
```

4.3.3.3 experiment_new()

Function to create a new Experiment struct.

Parameters

```
experiment | Experiment struct.
```

Definition at line 69 of file experiment.c.

```
00070 {
00071    unsigned int i;
00072    #if DEBUG_EXPERIMENT
00073    fprintf (stderr, "experiment_new: start\n");
00074    #endif
00075    experiment->name = NULL;
00076    experiment->ninputs = experiment->template_flags = 0;
00077    for (i = 0; i < MAX_NINPUTS; ++i)
00078         experiment->stencil[i] = NULL;
00079    #if DEBUG_EXPERIMENT
00080    fprintf (stderr, "input_new: end\n");
00081    #endif
00082 }
```

4.3.3.4 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 270 of file experiment.c.

```
00274 {
00275
        char buffer[64];
00276
       JsonObject *object;
const char *name;
00277
00278
       int error_code;
       unsigned int i;
       unsigned int flags = 1;
00280
00281
00282 #if DEBUG_EXPERIMENT
00283 fprintf (stderr, "experiment_open_json: start\n");
00284 #endif
00285
00286
       // Resetting experiment data
00287
       experiment_new (experiment);
00288
00289
       // Getting JSON object
00290
       object = json_node_get_object (node);
00291
00292
        // Reading the experimental data
00293
       name = json_object_get_string_member (object, LABEL_NAME);
00294
       if (!name)
00295
00296
           experiment_error (experiment, _("no data file name"));
00297
           goto exit_on_error;
00298
00299
       experiment->name = g_strdup (name);
00300 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_json: name=%s\n", experiment->name);
00301
00302 #endif
00303
       experiment->weight
00304
         = jb_json_object_get_float_with_default (object, LABEL_WEIGHT, &error_code,
00305
00306
       if (!error_code)
00307
00308
           experiment_error (experiment, _("bad weight"));
00309
           goto exit_on_error;
00310
00311 #if DEBUG_EXPERIMENT
00312
       fprintf (stderr, "experiment_open_json: weight=%lg\n", experiment->weight);
00313 #endif
00314 name = json_object_get_string_member (object, stencil[0]);
00315
       if (name)
00316
00317 #if DEBUG_EXPERIMENT
00318
           fprintf (stderr, "experiment_open_json: experiment=%s template1=%s\n",
00319
                    name, stencil[0]);
00320 #endif
00321
           ++experiment->ninputs;
00322
           experiment->template_flags |= flags;
00323
00324
       else
00325
           name = json_object_get_string_member (object, stencilbin[0]);
00326
00327
           if (name)
00328
00329 #if DEBUG_EXPERIMENT
00330
             fprintf (stderr, "experiment_open_json: experiment=%s template1=%s\n",
00331
                         name, stencilbin[0]);
00332 #endif
00333
               ++experiment->ninputs;
00334
             }
00335
           else
00336
            {
00337
               experiment_error (experiment, _("no template"));
00338
               goto exit_on_error;
             }
00339
       }
00340
00341
       experiment->stencil[0] = g_strdup (name);
00342
       for (i = 1; i < MAX_NINPUTS; ++i)</pre>
00343
00344 #if DEBUG_EXPERIMENT
            fprintf (stderr, "experiment_open_json: stencil%u\n", i + 1);
00345
00346 #endif
00347
          flags «= 1;
            if (json_object_get_member (object, stencil[i]))
```

```
00349
00350
                if (ninputs && ninputs <= i)</pre>
00351
                    experiment_error (experiment, _("bad templates number"));
00352
00353
                    goto exit_on_error;
00354
00355
                name = json_object_get_string_member (object, stencil[i]);
00356 #if DEBUG_EXPERIMENT
00357
               fprintf (stderr,
                          "experiment_open_json: experiment=%s stencil%u=%s\n",
00358
                          experiment->nexperiments, name, stencil[i]);
00359
00360 #endif
                experiment->stencil[i] = g_strdup (name);
00361
00362
                ++experiment->ninputs;
00363
                experiment->template_flags |= flags;
00364
            else if (json_object_get_member (object, stencilbin[i]))
00365
00366
00367
                if (ninputs && ninputs <= i)</pre>
00368
                  {
00369
                     experiment_error (experiment, _("bad templates number"));
00370
                     goto exit_on_error;
                  }
00371
                name = json_object_get_string_member (object, stencilbin[i]);
00372
00373 #if DEBUG_EXPERIMENT
00374
               fprintf (stderr,
00375
                          "experiment_open_json: experiment=%s stencil%u=%s\n",
00376
                          experiment->nexperiments, name, stencilbin[i]);
00377 #endif
00378
                experiment->stencil[i] = g_strdup (name);
00379
                ++experiment->ninputs;
00380
00381
            else if (ninputs && ninputs > i)
00382
                snprintf (buffer, 64, "%s%u", _("no template"), i + 1);
experiment_error (experiment, buffer);
00383
00384
00385
                goto exit_on_error;
00386
00387
            else
00388
              break;
00389
          }
00390
00391 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_json: end\n");
00392
00393 #endif
00394
       return 1;
00395
00396 exit_on_error:
00397 experiment_free (experiment, INPUT_TYPE_JSON); 00398 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_json: end\n");
00400 #endif
00401
       return 0;
00402 }
```

Here is the call graph for this function:

4.3.3.5 experiment_open_xml()

```
int experiment_open_xml (
            Experiment * experiment,
            xmlNode * node,
             unsigned int ninputs )
```

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.

enerated by Doxygen

Definition at line 133 of file experiment.c.

```
00138
        char buffer[64];
00139
        int error_code;
00140
       unsigned int i;
unsigned int flags = 1;
00141
00143 #if DEBUG_EXPERIMENT
00144
       fprintf (stderr, "experiment_open_xml: start\n");
00145 #endif
00146
00147
        // Resetting experiment data
00148
       experiment new (experiment);
00149
00150
        // Reading the experimental data
00151
        experiment->name = (char *) xmlGetProp (node, (const xmlChar *) LABEL_NAME);
        if (!experiment->name)
00152
00153
        {
00154
            experiment_error (experiment, _("no data file name"));
00155
            goto exit_on_error;
00156
00157 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_xml: name=%s\n", experiment->name);
00158
00159 #endif
00160
       experiment->weight
00161
          = jb_xml_node_get_float_with_default (node, (const xmlChar *) LABEL_WEIGHT,
00162
                                                 &error_code, 1.);
00163
00164
00165
            experiment_error (experiment, _("bad weight"));
00166
           goto exit_on_error;
00167
00168 #if DEBUG_EXPERIMENT
00169
       fprintf (stderr, "experiment_open_xml: weight=%lg\n", experiment->weight);
00170 #endif
00171 experiment->stencil[0]
00172
         = (char *) xmlGetProp (node, (const xmlChar *) stencil[0]);
       if (experiment->stencil[0])
00174
00175 #if DEBUG_EXPERIMENT
00176
            fprintf (stderr, "experiment_open_xml: experiment=%s stencil1=%s\n",
00177
                     experiment->name, stencil[0]);
00178 #endif
00179
            ++experiment->ninputs;
00180
            experiment->template_flags |= flags;
00181
00182
       else
00183
00184
           experiment->stencil[0]
00185
             = (char *) xmlGetProp (node, (const xmlChar *) stencilbin[0]);
            if (experiment->stencil[0])
00187
00188 #if DEBUG_EXPERIMENT
00189
              fprintf (stderr, "experiment_open_xml: experiment=%s stencil1=%s\n",
                         experiment->name, stencilbin[0]);
00190
00191 #endif
               ++experiment->ninputs;
00193
00194
            else
00195
             {
00196
               experiment_error (experiment, _("no template"));
00197
               goto exit_on_error;
00198
00199
00200
       for (i = 1; i < MAX_NINPUTS; ++i)</pre>
00201
00202 #if DEBUG_EXPERIMENT
            fprintf (stderr, "experiment_open_xml: stencil%u\n", i + 1);
00203
00204 #endif
            flags «= 1;
00206
            if (xmlHasProp (node, (const xmlChar *) stencil[i]))
00207
00208
                if (ninputs && ninputs <= i)</pre>
00209
                   experiment_error (experiment, _("bad templates number"));
00210
00211
                   goto exit_on_error;
00212
00213
                experiment->stencil[i]
00214
                  = (char *) xmlGetProp (node, (const xmlChar *) stencil[i]);
00215 #if DEBUG EXPERIMENT
00216
                fprintf (stderr,
                          experiment_open_xml: experiment=%s stencil%u=%s\n",
00218
                         experiment->nexperiments, experiment->name,
00219
                         experiment->stencil[i]);
00220 #endif
00221
               ++experiment->ninputs;
               experiment->template_flags |= flags;
00222
```

```
00223
00224
            else if (xmlHasProp (node, (const xmlChar *) stencilbin[i]))
00225
                if (ninputs && ninputs <= i)</pre>
00226
00227
00228
                    experiment_error (experiment, _("bad templates number"));
                    goto exit_on_error;
00230
00231
                experiment->stencil[i]
00232 = (char *) xmlGetProp (node, (const xmlChar *) stencilbin[i]);
00233 #if DEBUG_EXPERIMENT
00234
               fprintf (stderr,
00235
                           "experiment_open_xml: experiment=%s stencil%u=%s\n",
00236
                          experiment->nexperiments, experiment->name,
                          experiment->stencil[i]);
00237
00238 #endif
00239
                ++experiment->ninputs;
00240
            else if (ninputs && ninputs > i)
00242
             {
                snprintf (buffer, 64, "%s%u", _("no template"), i + 1);
experiment_error (experiment, buffer);
00243
00244
00245
                goto exit_on_error;
00246
00247
            else
00248
              break;
00249
00250
00251 #if DEBUG_EXPERIMENT
00252 fprintf (stderr, "experiment_open_xml: end\n");
00253 #endif
00254
       return 1;
00255
00256 exit_on_error:
00257
        experiment_free (experiment, INPUT_TYPE_XML);
00258 #if DEBUG_EXPERIMENT
00259
       fprintf (stderr, "experiment_open_xml: end\n");
00260 #endif
00261
       return 0;
00262 }
```

Here is the call graph for this function:

4.3.4 Variable Documentation

4.3.4.1 stencil

```
Initial value:
= {
   LABEL_TEMPLATE1, LABEL_TEMPLATE2, LABEL_TEMPLATE3, LABEL_TEMPLATE4,
   LABEL_TEMPLATE5, LABEL_TEMPLATE6, LABEL_TEMPLATE7, LABEL_TEMPLATE8
}
```

Array of strings with stencil labels.

Definition at line 54 of file experiment.c.

4.3.4.2 stencilbin

```
const char* stencilbin[MAX_NINPUTS]

Initial value:
= {
    LABEL_INPUT1, LABEL_INPUT2, LABEL_INPUT3, LABEL_INPUT4,
    LABEL_INPUT5, LABEL_INPUT6, LABEL_INPUT7, LABEL_INPUT8
}
```

Array of strings with binary stencil labels.

Definition at line 60 of file experiment.c.

4.4 experiment.c

Go to the documentation of this file.

```
00001 /*
00002 MPCOTool:
00003 The Multi-Purposes Calibration and Optimization Tool. A software to perform
\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,
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.
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/xml.h"
00046 #include "jb/src/json.h"
00046 #Include "jb/src/json.n
00047 #include "jb/src/win.h"
00048 #include "tools.h"
00049 #include "experiment.h"
00050
00051 #define DEBUG_EXPERIMENT 0
00052
00054 const char *stencil[MAX_NINPUTS] = {
         LABEL_TEMPLATE1, LABEL_TEMPLATE2, LABEL_TEMPLATE3, LABEL_TEMPLATE4, LABEL_TEMPLATE5, LABEL_TEMPLATE6, LABEL_TEMPLATE7, LABEL_TEMPLATE8
00055
00056
00057 };
00058
00060 const char *stencilbin[MAX_NINPUTS] = {
00061 LABEL_INPUT1, LABEL_INPUT2, LABEL_INPUT3, LABEL_INPUT4,
00062
         LABEL_INPUT5, LABEL_INPUT6, LABEL_INPUT7, LABEL_INPUT8
00063 };
00064
00068 static void
```

4.4 experiment.c 75

```
00069 experiment_new (Experiment * experiment)
00070 {
00071
       unsigned int i;
00072 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_new: start\n");
00073
00074 #endif
00075 experiment->name = NULL;
00076
       experiment->ninputs = experiment->template_flags = 0;
00077 for (i = 0; i < MAX_NINPUTS; ++i)
00078
         experiment->stencil[i] = NULL;
00079 #if DEBUG_EXPERIMENT
00080 fprintf (stderr, "input_new: end\n");
00081 #endif
00082 }
00083
00087 void
00088 experiment_free (Experiment * experiment,
00089
                      unsigned int type)
00091
       unsigned int i;
00092 #if DEBUG_EXPERIMENT
00093
       fprintf (stderr, "experiment_free: start\n");
00094 #endif
       if (type == INPUT_TYPE_XML)
00095
00096
       {
           for (i = 0; i < experiment->ninputs; ++i)
00097
00098
              xmlFree (experiment->stencil[i]);
00099
            xmlFree (experiment->name);
00100
00101
       else
00102
        {
00103
           for (i = 0; i < experiment->ninputs; ++i)
00104
             g_free (experiment->stencil[i]);
00105
           g_free (experiment->name);
         1
00106
       experiment->ninputs = experiment->template_flags = 0;
00107
00108 #if DEBUG_EXPERIMENT
00109 fprintf (stderr, "experiment_free: end\n");
00110 #endif
00111 }
00112
00116 void
00117 experiment_error (Experiment * experiment,
00118
                       char *message)
00119 {
00120
       if (!experiment->name)
00121
         error_message = g_strconcat (_("Experiment"), ": ", message, NULL);
00122
       else
         error_message = g_strconcat (_("Experiment"), " ", experiment->name, ": ",
00123
00124
                                       message, NULL);
00125 }
00126
00132 int
00133 experiment_open_xml (Experiment * experiment,
00134
                           xmlNode * node.
                          unsigned int ninputs)
00135
00137 {
00138
       char buffer[64];
00139
       int error_code;
00140
       unsigned int i;
00141
       unsigned int flags = 1;
00142
00143 #if DEBUG_EXPERIMENT
00144
       fprintf (stderr, "experiment_open_xml: start\n");
00145 #endif
00146
00147
       // Resetting experiment data
00148
       experiment_new (experiment);
00149
00150
        // Reading the experimental data
00151
       experiment->name = (char *) xmlGetProp (node, (const xmlChar *) LABEL_NAME);
00152
          (!experiment->name)
00153
        {
00154
            experiment_error (experiment, _("no data file name"));
00155
           goto exit_on_error;
00156
00157 #if DEBUG_EXPERIMENT
00158
       fprintf (stderr, "experiment_open_xml: name=%s\n", experiment->name);
00159 #endif
00160
       experiment->weight
00161
         = jb_xml_node_get_float_with_default (node, (const xmlChar *) LABEL_WEIGHT,
00162
                                                &error_code, 1.);
00163
        if (!error_code)
00164
00165
           experiment_error (experiment, _("bad weight"));
00166
           goto exit_on_error;
00167
```

```
00168 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_xml: weight=%lg\n", experiment->weight);
00170 #endif
00171 experiment->stencil[0]
00172
         = (char *) xmlGetProp (node, (const xmlChar *) stencil[0]);
       if (experiment->stencil[0])
00173
00174
00175 #if DEBUG_EXPERIMENT
00176
         fprintf (stderr, "experiment_open_xml: experiment=%s stencil1=%s\n",
00177
                     experiment->name, stencil[0]);
00178 #endif
00179
         ++experiment->ninputs;
00180
           experiment->template_flags |= flags;
00181
00182
       else
00183
       {
           experiment->stencil[0]
00184
00185
             = (char *) xmlGetProp (node, (const xmlChar *) stencilbin[0]);
            if (experiment->stencil[0])
00186
00187
00188 #if DEBUG_EXPERIMENT
         fprintf (stderr, "experiment_open_xml: experiment=%s stencil1=%s\n",
00189
                        experiment->name, stencilbin[0]);
00190
00191 #endif
00192
               ++experiment->ninputs;
             }
00193
00194
            else
            {
00195
00196
               experiment_error (experiment, _("no template"));
00197
               goto exit_on_error;
00198
00199
00200 for (i = 1; i < MAX_NINPUTS; ++i)
00201
00202 #if DEBUG EXPERIMENT
            fprintf (stderr, "experiment_open_xml: stencil%u\n", i + 1);
00203
00204 #endif
           flags «= 1;
00206
            if (xmlHasProp (node, (const xmlChar *) stencil[i]))
00207
00208
                if (ninputs && ninputs <= i)</pre>
00209
                 {
                   experiment_error (experiment, _("bad templates number"));
00210
00211
                   goto exit_on_error;
00212
00213
               experiment->stencil[i]
00214
                  = (char *) xmlGetProp (node, (const xmlChar *) stencil[i]);
00215 #if DEBUG_EXPERIMENT
00216
               fprintf (stderr,
                         "experiment_open_xml: experiment=%s stencil%u=%s\n",
00217
                         experiment->nexperiments, experiment->name,
00218
00219
                         experiment->stencil[i]);
00220 #endif
00221
               ++experiment->ninputs;
00222
                experiment->template_flags |= flags;
00223
            else if (xmlHasProp (node, (const xmlChar *) stencilbin[i]))
00225
             {
00226
                if (ninputs && ninputs <= i)</pre>
00227
                   experiment_error (experiment, _("bad templates number"));
00228
00229
                   goto exit_on_error;
00230
00231
                experiment->stencil[i]
00232
                  = (char *) xmlGetProp (node, (const xmlChar *) stencilbin[i]);
00233 #if DEBUG_EXPERIMENT
00234
               fprintf (stderr,
00235
                         "experiment open xml: experiment=%s stencil%u=%s\n",
00236
                         experiment->nexperiments, experiment->name,
                         experiment->stencil[i]);
00238 #endif
00239
               ++experiment->ninputs;
00240
00241
            else if (ninputs && ninputs > i)
00242
            {
               snprintf (buffer, 64, "%s%u", _("no template"), i + 1);
experiment_error (experiment, buffer);
00243
00244
00245
                goto exit_on_error;
00246
           else
00247
00248
             break;
00249
         }
00250
00251 #if DEBUG_EXPERIMENT
00252
       fprintf (stderr, "experiment_open_xml: end\n");
00253 #endif
00254
       return 1:
```

4.4 experiment.c 77

```
00255
00256 exit_on_error:
00257
       experiment_free (experiment, INPUT_TYPE_XML);
00258 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_xml: end\n");
00259
00260 #endif
00261 return 0;
00262 }
00263
00269 int
00270 experiment_open_json (Experiment * experiment,
00271
                            JsonNode * node,
00272
                            unsigned int ninputs)
00274 {
00275
       char buffer[64];
       JsonObject *object;
const char *name;
00276
00277
00278
       int error_code;
00279
       unsigned int i;
00280
       unsigned int flags = 1;
00281
00282 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_json: start\n");
00283
00284 #endif
00285
00286
        // Resetting experiment data
00287
       experiment_new (experiment);
00288
00289
       // Getting JSON object
00290
       object = json_node_get_object (node);
00291
00292
        // Reading the experimental data
00293
        name = json_object_get_string_member (object, LABEL_NAME);
00294
        if (!name)
00295
00296
           experiment_error (experiment, _("no data file name"));
00297
           goto exit_on_error;
00298
00299
        experiment->name = g_strdup (name);
00300 #if DEBUG_EXPERIMENT
00301 fprintf (stderr, "experiment_open_json: name=%s\n", experiment->name);
00302 #endif
00303
       experiment->weight
00304
          = jb_json_object_get_float_with_default (object, LABEL_WEIGHT, &error_code,
00305
00306
00307
00308
            experiment_error (experiment, _("bad weight"));
00309
            goto exit_on_error;
00310
00311 #if DEBUG_EXPERIMENT
00312
       fprintf (stderr, "experiment_open_json: weight=%lg\n", experiment->weight);
00313 #endif
00314
      name = json_object_get_string_member (object, stencil[0]);
00315
       if (name)
00316
00317 #if DEBUG_EXPERIMENT
00318
           fprintf (stderr, "experiment_open_json: experiment=%s template1=%s\n",
00319
                    name, stencil[0]);
00320 #endif
           ++experiment->ninputs;
00321
00322
           experiment->template_flags |= flags;
00323
00324
00325
00326
           name = json_object_get_string_member (object, stencilbin[0]);
00327
           if (name)
00328
00329 #if DEBUG_EXPERIMENT
         fprintf (stderr, "experiment_open_json: experiment=%s template1=%s\n",  
00331
                         name, stencilbin[0]);
00332 #endif
00333
               ++experiment->ninputs;
              }
00334
00335
            else
00336
00337
                experiment_error (experiment, _("no template"));
00338
                goto exit_on_error;
00339
00340
         }
       experiment->stencil[0] = g_strdup (name);
00341
       for (i = 1; i < MAX_NINPUTS; ++i)</pre>
00342
00343
00344 #if DEBUG_EXPERIMENT
00345
            fprintf (stderr, "experiment_open_json: stencil%u\n", i + 1);
00346 #endif
00347
            flags «= 1;
```

```
(json_object_get_member (object, stencil[i]))
00350
                if (ninputs && ninputs <= i)</pre>
00351
                  {
00352
                    experiment_error (experiment, _("bad templates number"));
00353
                    goto exit on error:
00355
                name = json_object_get_string_member (object, stencil[i]);
00356 #if DEBUG_EXPERIMENT
00357
                fprintf (stderr,
                         "experiment_open_json: experiment=%s stencil%u=%s\n",
00358
00359
                         experiment->nexperiments, name, stencil[i]);
00360 #endif
00361
               experiment->stencil[i] = g_strdup (name);
00362
                ++experiment->ninputs;
00363
                experiment->template_flags |= flags;
00364
00365
            else if (json_object_get_member (object, stencilbin[i]))
00366
00367
                if (ninputs && ninputs <= i)</pre>
00368
00369
                    experiment_error (experiment, _("bad templates number"));
00370
                    goto exit_on_error;
00371
00372
               name = json_object_get_string_member (object, stencilbin[i]);
00373 #if DEBUG_EXPERIMENT
00374
                fprintf (stderr,
00375
                         "experiment_open_json: experiment=%s stencil%u=%sn",
00376
                         experiment->nexperiments, name, stencilbin[i]);
00377 #endif
00378
                experiment->stencil[i] = g_strdup (name);
00379
               ++experiment->ninputs;
00380
00381
            else if (ninputs && ninputs > i)
00382
            {
                snprintf (buffer, 64, "%s%u", _("no template"), i + 1);
00383
00384
                experiment_error (experiment, buffer);
00385
               goto exit_on_error;
00386
00387
           else
00388
             break;
         }
00389
00390
00391 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_json: end\n");
00392
00393 #endif
00394
       return 1;
00395
00396 exit on error:
00397
       experiment_free (experiment, INPUT_TYPE_JSON);
00398 #if DEBUG_EXPERIMENT
00399
       fprintf (stderr, "experiment_open_json: end\n");
00400 #endif
00401
       return 0;
00402 }
```

4.5 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 strings with stencil labels.

• const char * stencilbin [MAX_NINPUTS]

Array of strings with binary stencil labels.

4.5.1 Detailed Description

Header file to define the experiment data.

Authors

Javier Burguete.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file experiment.h.

4.5.2 Function Documentation

4.5.2.1 experiment_error()

Function to print a message error opening an Experiment struct.

Parameters

experiment	Experiment struct.
message	Error message.

Definition at line 117 of file experiment.c.

4.5.2.2 experiment_free()

Function to free the memory of an Experiment struct.

Parameters

experiment	Experiment struct.
type	Type of input file.

Definition at line 88 of file experiment.c.

```
00090 {
00091 unsigned int i;
00092 #if DEBUG_EXPERIMENT
00091
00093
       fprintf (stderr, "experiment_free: start\n");
00094 #endif
00095
00096
       if (type == INPUT_TYPE_XML)
00097
            for (i = 0; i < experiment->ninputs; ++i)
00098
             xmlFree (experiment->stencil[i]);
00099
           xmlFree (experiment->name);
00100
00101
        else
00102
        {
00103
           for (i = 0; i < experiment->ninputs; ++i)
00104
             g_free (experiment->stencil[i]);
00105
           g_free (experiment->name);
00106
00107
       experiment->ninputs = experiment->template_flags = 0;
00108 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_free: end\n");
00109
00110 #endif
00111 }
```

4.5.2.3 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 270 of file experiment.c.

```
00274 {
00275
       char buffer[64];
00276
        JsonObject *object;
00277
       const char *name;
00278
       int error_code;
00279
       unsigned int i:
00280
       unsigned int flags = 1;
00281
00282 #if DEBUG_EXPERIMENT 00283 fprintf (stderr, "experiment_open_json: start\n");
00284 #endif
00285
00286
       // Resetting experiment data
00287
       experiment_new (experiment);
00288
00289
       // Getting JSON object
00290
       object = json_node_get_object (node);
00291
00292
       // Reading the experimental data
00293
        name = json_object_get_string_member (object, LABEL_NAME);
00294
        if (!name)
00295
00296
           experiment_error (experiment, _("no data file name"));
00297
           goto exit_on_error;
00298
00299
       experiment->name = g_strdup (name);
00300 #if DEBUG_EXPERIMENT
00301
       fprintf (stderr, "experiment_open_json: name=%s\n", experiment->name);
00302 #endif
00303
       experiment->weight
         = jb_json_object_get_float_with_default (object, LABEL_WEIGHT, &error_code,
00304
00305
00306
       {
00307
00308
           experiment_error (experiment, _("bad weight"));
00309
            goto exit_on_error;
00310
00311 #if DEBUG_EXPERIMENT
00312
       fprintf (stderr, "experiment_open_json: weight=%lg\n", experiment->weight);
00313 #endif
00314   name = json_object_get_string_member (object, stencil[0]);
00315 if (name)
00316
00317 #if DEBUG_EXPERIMENT
      fprintf (stderr, "experiment_open_json: experiment=%s template1=%s\n",
00318
00319
                    name, stencil[0]);
00320 #endif
00321
           ++experiment->ninputs;
           experiment->template_flags |= flags;
00322
00323
00324
       else
00325
        {
00326
           name = json_object_get_string_member (object, stencilbin[0]);
00327
          if (name)
00328
00329 #if DEBUG_EXPERIMENT
              fprintf (stderr, "experiment_open_json: experiment=%s template1=%s\n",
00331
                        name, stencilbin[0]);
00332 #endif
00333
               ++experiment->ninputs;
             }
00334
00335
           else
00336
            {
00337
              experiment_error (experiment, _("no template"));
00338
               goto exit_on_error;
             }
00339
00340
         }
       experiment->stencil[0] = g_strdup (name);
00341
       for (i = 1; i < MAX_NINPUTS; ++i)</pre>
00342
00343
00344 #if DEBUG_EXPERIMENT
00345
            fprintf (stderr, "experiment_open_json: stencil%u\n", i + 1);
00346 #endif
           flags «= 1;
00347
00348
            if (json_object_get_member (object, stencil[i]))
00349
00350
                if (ninputs && ninputs <= i)</pre>
00351
00352
                    experiment_error (experiment, _("bad templates number"));
00353
                    goto exit_on_error;
00354
00355
               name = json_object_get_string_member (object, stencil[i]);
00356 #if DEBUG_EXPERIMENT
00357
                fprintf (stderr,
00358
                         "experiment_open_json: experiment=%s stencil%u=%sn",
00359
                         experiment->nexperiments, name, stencil[i]);
00360 #endif
```

```
experiment->stencil[i] = g_strdup (name);
00362
                 ++experiment->ninputs;
00363
                 experiment->template_flags |= flags;
00364
            else if (json_object_get_member (object, stencilbin[i]))
00365
00366
00367
                 if (ninputs && ninputs <= i)</pre>
00368
00369
                     experiment_error (experiment, _("bad templates number"));
00370
                     goto exit_on_error;
00371
                  }
00372
                name = json_object_get_string_member (object, stencilbin[i]);
00373 #if DEBUG_EXPERIMENT
00374
               fprintf (stderr,
00375
                          "experiment_open_json: experiment=%s stencil%u=%s\n",
00376
00377 #endif
                          experiment->nexperiments, name, stencilbin[i]);
00378
                experiment->stencil[i] = g_strdup (name);
                ++experiment->ninputs;
00380
00381
            else if (ninputs && ninputs > i)
00382
                snprintf (buffer, 64, "%s%u", _("no template"), i + 1);
experiment_error (experiment, buffer);
00383
00384
00385
                goto exit_on_error;
00386
00387
            else
00388
              break;
        }
00389
00390
00391 #if DEBUG_EXPERIMENT
00392 fprintf (stderr, "c
       fprintf (stderr, "experiment_open_json: end\n");
00393 #endif
00394
        return 1;
00395
00396 exit_on_error:
00397
       experiment_free (experiment, INPUT_TYPE_JSON);
00398 #if DEBUG_EXPERIMENT
00399
        fprintf (stderr, "experiment_open_json: end\n");
00400 #endif
00401
       return 0;
00402 }
```

Here is the call graph for this function:

4.5.2.4 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 133 of file experiment.c.

```
00137 {

00138 char buffer[64];

00139 int error_code;

00140 unsigned int i;
```

```
00141
       unsigned int flags = 1;
00142
00143 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_xml: start\n");
00144
00145 #endif
00146
00147
        // Resetting experiment data
00148
       experiment_new (experiment);
00149
00150
       // Reading the experimental data
        experiment->name = (char *) xmlGetProp (node, (const xmlChar *) LABEL_NAME);
00151
        if (!experiment->name)
00152
00153
         {
00154
           experiment_error (experiment, _("no data file name"));
00155
            goto exit_on_error;
00156
00157 #if DEBUG_EXPERIMENT
       fprintf (stderr, "experiment_open_xml: name=%s\n", experiment->name);
00158
00159 #endif
00160
       experiment->weight
00161
          = jb_xml_node_get_float_with_default (node, (const xmlChar *) LABEL_WEIGHT,
00162
                                                 &error_code, 1.);
00163
       if (!error_code)
00164
00165
           experiment_error (experiment, _("bad weight"));
00166
           goto exit_on_error;
00167
00168 #if DEBUG_EXPERIMENT
00169 fprintf (stderr, "experiment_open_xml: weight=%lg\n", experiment->weight);
00170 #endif
00171 experiment->stencil[0]
00172
          = (char *) xmlGetProp (node, (const xmlChar *) stencil[0]);
00173
       if (experiment->stencil[0])
00174
00175 #if DEBUG EXPERIMENT
       fprintf (stderr, "experiment_open_xml: experiment=%s stencill=%s\n",
00176
00177
                    experiment->name, stencil[0]);
00178 #endif
00179
           ++experiment->ninputs;
00180
            experiment->template_flags |= flags;
00181
00182
       else
00183
        {
00184
           experiment->stencil[0]
00185
             = (char *) xmlGetProp (node, (const xmlChar *) stencilbin[0]);
00186
            if (experiment->stencil[0])
00187
00188 #if DEBUG_EXPERIMENT
               fprintf (stderr, "experiment_open_xml: experiment=%s stencil1=%s\n",
00189
00190
                        experiment->name, stencilbin[0]);
00191 #endif
00192
               ++experiment->ninputs;
00193
             }
00194
           else
00195
             {
00196
               experiment_error (experiment, _("no template"));
               goto exit_on_error;
00198
00199
      for (i = 1; i < MAX_NINPUTS; ++i)</pre>
00200
00201
00202 #if DEBUG_EXPERIMENT
00203
           fprintf (stderr, "experiment_open_xml: stencil%u\n", i + 1);
00204 #endif
00205
            flags «= 1;
00206
            if (xmlHasProp (node, (const xmlChar *) stencil[i]))
00207
00208
                if (ninputs && ninputs <= i)</pre>
00209
                 {
00210
                   experiment_error (experiment, _("bad templates number"));
00211
                   goto exit_on_error;
00212
00213
                experiment->stencil[i]
                  = (char *) xmlGetProp (node, (const xmlChar *) stencil[i]);
00214
00215 #if DEBUG_EXPERIMENT
00216
                fprintf (stderr,
00217
                         "experiment_open_xml: experiment=%s stencil%u=%sn",
00218
                         experiment->nexperiments, experiment->name,
00219
                         experiment->stencil[i]);
00220 #endif
               ++experiment->ninputs;
00221
               experiment->template_flags |= flags;
00222
00223
00224
            else if (xmlHasProp (node, (const xmlChar *) stencilbin[i]))
00225
00226
                if (ninputs && ninputs <= i)
00227
```

```
experiment_error (experiment, _("bad templates number"));
00229
                     goto exit_on_error;
00230
00231
                experiment->stencil[i]
00232 = (char *) xmlGetProp (node, (const xmlChar *) stencilbin[i]);
00233 #if DEBUG_EXPERIMENT
             fprintf (stderr,
00235
                            "experiment_open_xml: experiment=%s stencil%u=%s\n",
00236
                           experiment->nexperiments, experiment->name,
00237
                           experiment->stencil[i]);
00238 #endif
                 ++experiment->ninputs;
00239
00240
00241
             else if (ninputs && ninputs > i)
            {
00242
               snprintf (buffer, 64, "%s%u", _("no template"), i + 1);
experiment_error (experiment, buffer);
00243
00244
00245
                 goto exit_on_error;
00246
00247
            else
       }
00248
              break;
00249
00250
00251 #if DEBUG_EXPERIMENT
00252 fprintf (stderr, "experiment_open_xml: end\n");
00253 #endif
00254
00255
00256 exit_on_error:
00257 experiment_free (experiment, INPUT_TYPE_XML); 00258 #if DEBUG_EXPERIMENT
        fprintf (stderr, "experiment_open_xml: end\n");
00260 #endif
00261
        return 0;
00262 }
```

Here is the call graph for this function:

4.5.3 Variable Documentation

4.5.3.1 stencil

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

Array of strings with stencil labels.

Definition at line 54 of file experiment.c.

4.5.3.2 stencilbin

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

Array of strings with binary stencil labels.

Definition at line 60 of file experiment.c.

4.6 experiment.h

4.6 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.
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.
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 EXPERIMENT__H
00039 #define EXPERIMENT__H 1
00040
00045 typedef struct
00046 {
00047
         char *name;
00048
         char *stencil[MAX_NINPUTS];
00049
        double weight;
        unsigned int ninputs;
unsigned int template_flags;
00050
00052 } Experiment;
00053
00054 extern const char *stencil[MAX_NINPUTS];
00055 extern const char *stencilbin[MAX_NINPUTS];
00056
00057 // Public functions
00058 void experiment_free (Experiment * experiment, unsigned int type);
00059 void experiment_error (Experiment * experiment, char *message);
00060 int experiment_open_xml (Experiment * experiment, xmlNode * node,
00061
                                    unsigned int ninputs);
00062 int experiment_open_json (Experiment * experiment, JsonNode * node,
00063
                                     unsigned int ninputs);
00064
00065 #endif
```

4.7 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/xml.h"
#include "jb/src/json.h"
```

```
#include "jb/src/win.h"
#include "tools.h"
#include "experiment.h"
#include "variable.h"
#include "input.h"
Include dependency graph for input.c:
```

Macros

#define DEBUG_INPUT 0
 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.

4.7.1 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.

4.7.2 Macro Definition Documentation

4.7.2.1 DEBUG_INPUT

```
#define DEBUG_INPUT 0
```

Macro to debug input functions.

Definition at line 55 of file input.c.

4.7.3 Function Documentation

4.7.3.1 input_error()

Function to print an error message opening an Input struct.

Parameters

```
message Error message.
```

Definition at line 127 of file input.c.

4.7.3.2 input_free()

```
void input_free ( )
```

Function to free the memory of the input file data.

Definition at line 86 of file input.c.

```
00087 {
00088
        unsigned int i;
00089 #if DEBUG_INPUT
00090
       fprintf (stderr, "input_free: start\n");
00091 #endif
00092
       g_free (input->name);
00093
        g_free (input->directory);
       for (i = 0; i < input->nexperiments; ++i)
00094
         experiment_free (input->experiment + i, input->type);
00095
00096
        for (i = 0; i < input->nvariables; ++i)
00097
         variable_free (input->variable + i, input->type);
00098
        g_free (input->experiment);
       g_free (input->variable);
if (input->type == INPUT_TYPE_XML)
00099
00100
00101
00102
            xmlFree (input->cleaner);
00103
            xmlFree (input->evaluator);
00104
            xmlFree (input->simulator);
00105
            xmlFree (input->result);
00106
            xmlFree (input->variables);
00107
00108
       else
```

```
{
          g_free (input->cleaner);
g_free (input->evaluator);
g_free (input->simulator);
00110
00111
00112
          g_free (input->variables);
}
            g_free (input->result);
00113
00114
00115
       input->nexperiments = input->nvariables = input->nsteps
00116
00117
           = input->nfinal_steps = 0;
00118 #if DEBUG_INPUT
00119 fprintf (stderr, "input_free: end\n");
00120 #endif
00121 }
```

Here is the call graph for this function:

4.7.3.3 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");
0070 #endif
0071 input->nvariables = input->nexperiments = input->nsteps = input->nfinal_steps
0072 = 0;
0073 input->simulator = input->evaluator = input->cleaner = input->directory
0074 = input->name = NULL;
0075 input->experiment = NULL;
0076 input->experiment = NULL;
0077 #if DEBUG_INPUT
0078 fprintf (stderr, "input_new: end\n");
00079 #endif
00080 }
```

4.7.3.4 input_open()

Function to open the input file.

Returns

1_on_success, 0_on_error.

Parameters

```
filename Input data file name.
```

Definition at line 1021 of file input.c.

```
01029
01030
         // Resetting input data
01031
         input_new ();
01032
01033 // Opening input file
01034 #if DEBUG_INPUT
01035 fprintf (stderr, "input_open: opening the input file %s\n", filename);
01036 fprintf (stderr, "input_open: trying XML format\n");
01037 #endif
01038 doc = xmlParseFile (filename);
        if (!doc)
01039
01040
01041 #if DEBUG_INPUT
01042
             fprintf (stderr, "input_open: trying JSON format\n");
01043 #endif
01044
            parser = json_parser_new ();
             if (!json_parser_load_from_file (parser, filename, NULL))
01045
01046
              {
01047
                  input_error (_("Unable to parse the input file"));
01048
                 goto exit_on_error;
01049
01050
             if (!input_open_json (parser))
01051
               goto exit_on_error;
01052
01053
        else if (!input_open_xml (doc))
01054
          goto exit_on_error;
01055
01056
        // Getting the working directory
01057
        input->directory = g_path_get_dirname (filename);
01058
        input->name = g_path_get_basename (filename);
01059
01060 #if DEBUG_INPUT
01061
        fprintf (stderr, "input_open: end\n");
01062 #endif
01063
        return 1;
01064
01065 exit_on_error:
01066 jb_show_error (error_message);
01067 g_free (error_message);
01068 input_free ();
01069 #if DEBUG_INPUT
01070
        fprintf (stderr, "input_open: end\n");
01071 #endif
01072
        return 0;
01073 }
```

Here is the call graph for this function:

4.7.3.5 input_open_json()

Function to open the input file in JSON format.

Returns

```
1 on success, 0 on error.
```

Parameters

parser	JsonParser struct.

Definition at line 603 of file input.c.

```
00604 {
00605 Experiment *experiment;
00606 JsonNode *node, *child;
00607 JsonObject *object;
00608 JsonArray *array;
00609 const char *buffer;
00610 int error_code;
```

```
unsigned int i, n;
00612
00613 #if DEBUG_INPUT
       fprintf (stderr, "input_open_json: start\n");
00614
00615 #endif
00616
00617
        // Resetting input data
00618
       input->type = INPUT_TYPE_JSON;
00619
00620
       // Getting the root node
00621 #if DEBUG_INPUT
       fprintf (stderr, "input_open_json: getting the root node\n");
00622
00623 #endif
00624
       node = json_parser_get_root (parser);
00625
       object = json_node_get_object (node);
00626
        // Getting result and variables file names
00627
00628
       if (!input->result)
00629
00630
            buffer = json_object_get_string_member (object, LABEL_RESULT_FILE);
00631
            if (!buffer)
00632
             buffer = result_name;
           input->result = g_strdup (buffer);
00633
00634
00635
       else
         input->result = g_strdup (result_name);
00636
00637
        if (!input->variables)
00638
00639
           buffer = json_object_get_string_member (object, LABEL_VARIABLES_FILE);
00640
           if (!buffer)
00641
             buffer = variables name:
00642
            input->variables = g_strdup (buffer);
00643
00644
       else
00645
         input->variables = g_strdup (variables_name);
00646
00647
        // Opening simulator program name
       buffer = json_object_get_string_member (object, LABEL_SIMULATOR);
00648
00649
       if (!buffer)
00650
00651
            input_error (_("Bad simulator program"));
00652
           goto exit_on_error;
00653
00654
       input->simulator = g_strdup (buffer);
00655
00656
        // Opening evaluator program name
00657
       buffer = json_object_get_string_member (object, LABEL_EVALUATOR);
00658
       if (buffer)
00659
         input->evaluator = q_strdup (buffer);
00660
00661
        // Opening cleaner program name
00662
       buffer = json_object_get_string_member (object, LABEL_CLEANER);
       if (buffer)
00663
00664
         input->cleaner = g_strdup (buffer);
00665
00666
        // Obtaining pseudo-random numbers generator seed
00667
00668
          = jb_json_object_get_uint_with_default (object, LABEL_SEED,
00669
                                                  &error_code, DEFAULT_RANDOM_SEED);
00670
        if (!error_code)
00671
        {
           input_error (_("Bad pseudo-random numbers generator seed"));
00672
00673
           goto exit_on_error;
00674
00675
00676
        // Opening algorithm
00677
       buffer = json_object_get_string_member (object, LABEL_ALGORITHM);
00678
        if (!strcmp (buffer, LABEL_MONTE_CARLO))
00679
00680
            input->algorithm = ALGORITHM_MONTE_CARLO;
00681
00682
            // Obtaining simulations number
00683
           input->nsimulations
             = jb_json_object_get_uint (object, LABEL_NSIMULATIONS, &error_code);
00684
            if (!error_code)
00685
00686
00687
               input_error (_("Bad simulations number"));
00688
               goto exit_on_error;
00689
00690
       else if (!strcmp (buffer, LABEL_SWEEP))
00691
         input->algorithm = ALGORITHM_SWEEP;
00692
00693
        else if (!strcmp (buffer, LABEL_ORTHOGONAL))
00694
         input->algorithm = ALGORITHM_ORTHOGONAL;
00695
       else if (!strcmp (buffer, LABEL_GENETIC))
00696
00697
            input->algorithm = ALGORITHM_GENETIC;
```

```
00698
            // Obtaining population
00699
00700
            if (json_object_get_member (object, LABEL_NPOPULATION))
00701
00702
                input->nsimulations
                = jb_json_object_get_uint (object, LABEL_NPOPULATION, &error_code);
if (!error_code || input->nsimulations < 3)</pre>
00703
00704
00705
00706
                     input_error (_("Invalid population number"));
00707
                     goto exit_on_error;
                   }
00708
00709
              }
00710
            else
00711
              {
00712
                input_error (_("No population number"));
00713
                goto exit_on_error;
00714
00715
            // Obtaining generations
00717
            if (json_object_get_member (object, LABEL_NGENERATIONS))
00718
00719
                input->niterations
00720
                   = jb_json_object_get_uint_with_default (object, LABEL_NGENERATIONS,
00721
                                                             &error_code, 1);
00722
                if (!error_code || !input->niterations)
00723
00724
                     input_error (_("Invalid generations number"));
00725
                     goto exit_on_error;
00726
00727
              }
00728
            else
00729
              {
00730
                input_error (_("No generations number"));
00731
                goto exit_on_error;
00732
00733
00734
            // Obtaining mutation probability
            if (json_object_get_member (object, LABEL_MUTATION))
00736
              {
00737
                input->mutation_ratio
00738
                   = jb_json_object_get_float (object, LABEL_MUTATION, &error_code);
00739
                if (!error_code || input->mutation_ratio < 0.</pre>
00740
                     || input->mutation_ratio >= 1.)
00741
00742
                    input_error (_("Invalid mutation probability"));
00743
                     goto exit_on_error;
00744
00745
              }
00746
            else
00747
              {
00748
                input_error (_("No mutation probability"));
00749
                goto exit_on_error;
00750
00751
00752
            // Obtaining reproduction probability
00753
            if (json_object_get_member (object, LABEL_REPRODUCTION))
00754
                input->reproduction_ratio
00755
00756
                   = jb_json_object_get_float (object, LABEL_REPRODUCTION,
00757
                                                &error_code);
                if (!error_code || input->reproduction_ratio < 0.</pre>
00758
00759
                     || input->reproduction_ratio >= 1.0)
00760
                   {
00761
                     input_error (_("Invalid reproduction probability"));
00762
                     goto exit_on_error;
00763
                   }
00764
00765
            else
00766
              {
00767
                input_error (_("No reproduction probability"));
00768
                goto exit_on_error;
00769
00770
00771
            // Obtaining adaptation probability
            if (json_object_get_member (object, LABEL_ADAPTATION))
00772
00773
00774
                input->adaptation_ratio
00775
                   = jb_json_object_get_float (object, LABEL_ADAPTATION, &error_code);
00776
                if (!error_code || input->adaptation_ratio < 0.</pre>
00777
                     || input->adaptation_ratio >= 1.)
00778
00779
                     input_error (_("Invalid adaptation probability"));
00780
                     goto exit_on_error;
00781
00782
00783
            else
00784
```

```
input_error (_("No adaptation probability"));
00786
                goto exit_on_error;
00787
00788
00789
             // Checking survivals
00790
            i = input->mutation_ratio * input->nsimulations;
00791
            i += input->reproduction_ratio * input->nsimulations;
00792
             i += input->adaptation_ratio * input->nsimulations;
00793
             if (i > input->nsimulations - 2)
00794
              {
00795
                input error
                   (_("No enough survival entities to reproduce the population"));
00796
00797
                goto exit on error;
00798
00799
00800
        else
00801
             input_error (_("Unknown algorithm"));
00802
00803
            goto exit_on_error;
00804
00805
00806
        if (input->algorithm == ALGORITHM_MONTE_CARLO
            || input->algorithm == ALGORITHM_SWEEP
|| input->algorithm == ALGORITHM_ORTHOGONAL)
00807
00808
00809
00811
             // Obtaining iterations number
00812
             input->niterations
00813
               = jb_json_object_get_uint (object, LABEL_NITERATIONS, &error_code);
             if (!error_code || !input->niterations)
00814
00815
              {
00816
                 input_error (_("Bad iterations number"));
00817
                goto exit_on_error;
00818
00819
             // Obtaining best number
00820
00821
             input->nbest
               = jb_json_object_get_uint_with_default (object, LABEL_NBEST,
00823
                                                          &error_code, 1);
00824
             if (!error_code || !input->nbest)
00825
                 input_error (_("Invalid best number"));
00826
00827
                 goto exit_on_error;
00828
00829
00830
             // Obtaining tolerance
00831
             input->tolerance
               = jb_json_object_get_float_with_default (object, LABEL_TOLERANCE,
00832
00833
                                                           &error_code, 0.);
00834
             if (!error code || input->tolerance < 0.)
00835
              {
00836
                 input_error (_("Invalid tolerance"));
00837
                 goto exit_on_error;
00838
00839
00840
             // Getting hill climbing method parameters
00841
               (json_object_get_member (object, LABEL_NSTEPS))
00842
               {
00843
                 input->nsteps
00844
                   = jb_json_object_get_uint (object, LABEL_NSTEPS, &error_code);
00845
                 if (!error_code)
00846
00847
                     input_error (_("Invalid steps number"));
00848
                     goto exit_on_error;
00849
00850
                 if (json_object_has_member (object, LABEL_NFINAL_STEPS))
00851
00852
                     input->nfinal steps
                       = jb_json_object_get_uint (object, LABEL_NFINAL_STEPS,
00853
00854
                                                    &error_code);
00855
00856
00857
                         input_error (_("Invalid final steps number"));
00858
                         goto exit_on_error;
00859
00860
00861
00862
                   input->nfinal_steps = input->nsteps;
                 buffer = json_object_get_string_member (object, LABEL_CLIMBING);
if (!strcmp (buffer, LABEL_COORDINATES))
  input->climbing = CLIMBING_METHOD_COORDINATES;
00863
00864
00865
00866
                 else if (!strcmp (buffer, LABEL_RANDOM))
00867
00868
                     input->climbing = CLIMBING_METHOD_RANDOM;
00869
                     input->nestimates
                        = jb_json_object_get_uint (object, LABEL_NESTIMATES,
00870
00871
                                                     &error code):
```

```
if (!error_code || !input->nestimates)
00873
00874
                        input_error (_("Invalid estimates number"));
00875
                        goto exit_on_error;
00876
00877
                  }
00878
                else
00879
                  {
00880
                    input_error (_("Unknown method to estimate the hill climbing"));
00881
                    goto exit_on_error;
                  }
00882
00883
                input->relaxation
00884
                  = jb_json_object_get_float_with_default (object, LABEL_RELAXATION,
00885
                                                             &error_code,
00886
                                                            DEFAULT_RELAXATION);
00887
                if (!error_code || input->relaxation < 0. || input->relaxation > 2.)
00888
                    input_error (_("Invalid relaxation parameter"));
00889
00890
                    goto exit_on_error;
00891
00892
00893
            else
00894
              input->nsteps = input->nfinal_steps = 0;
00895
00896
        // Obtaining the threshold
00897
        input->threshold
00898
           jb_json_object_get_float_with_default (object, LABEL_THRESHOLD,
00899
                                                    &error_code, 0.);
00900
00901
        if (!error_code)
00902
         {
00903
            input_error (_("Invalid threshold"));
00904
            goto exit_on_error;
00905
00906
       // Reading the experimental data
00907
       array = json_object_get_array_member (object, LABEL_EXPERIMENTS);
n = json_array_get_length (array);
00908
00910
        input->experiment = experiment = (Experiment *)
00911
          g_malloc (n * sizeof (Experiment));
00912
        for (i = 0; i < n; ++i)
00913
00914 #if DEBUG INPUT
00915
            fprintf (stderr, "input_open_json: nexperiments=%u\n",
00916
                     input->nexperiments);
00917 #endif
00918
           child = json_array_get_element (array, i);
00919
            if (!input->nexperiments)
00920
              {
00921
                if (!experiment_open_json (experiment, child, 0))
00922
                  goto exit_on_error;
00923
00924
            else
00925
             {
00926
                if (!experiment_open_json (experiment + input->nexperiments,
00927
                                            child, experiment->ninputs))
00928
                  goto exit_on_error;
00929
                if (experiment[experiment->ninputs].template_flags
00930
                    != experiment->template_flags)
00931
00932
                    input error ("bad template inputs");
00933
                    goto exit_on_error;
00934
                  }
00935
00936
            ++input->nexperiments;
00937 #if DEBUG_INPUT
           fprintf (stderr, "input_open_json: nexperiments=%u\n",
00938
00939
                     input->nexperiments);
00940 #endif
00941
00942
           (!input->nexperiments)
00943
00944
            input_error (_("No optimization experiments"));
00945
            goto exit_on_error;
00946
00947
        input->template_flags = experiment->template_flags;
00948
00949
        // Reading the variables data
00950
        array = json_object_get_array_member (object, LABEL_VARIABLES);
00951
        n = json\_array\_get\_length (array);
00952
        input->variable = (Variable *) g_malloc (n * sizeof (Variable));
00953
       for (i = 0; i < n; ++i)
00954
00955 #if DEBUG INPUT
00956 fprintf (stderr, "input_open_json: nvariables=u\n", input->nvariables); 00957 #endif
00958
            child = ison array get element (array, i);
```

```
if (!variable_open_json (input->variable + input->nvariables, child,
00960
                                       input->algorithm, input->nsteps))
00961
               goto exit_on_error;
            ++input->nvariables;
00962
00963
00964
        if (!input->nvariables)
00966
             input_error (_("No optimization variables"));
00967
            goto exit_on_error;
00968
00969
00970
        // Obtaining the error norm
00971
        if (json_object_get_member (object, LABEL_NORM))
00972
00973
            buffer = json_object_get_string_member (object, LABEL_NORM);
            if (!strcmp (buffer, LABEL_EUCLIDIAN))
input->norm = ERROR_NORM_EUCLIDIAN;
00974
00975
            else if (!strcmp (buffer, LABEL_MAXIMUM))
input->norm = ERROR_NORM_MAXIMUM;
00976
00978
            else if (!strcmp (buffer, LABEL_P))
00979
00980
                input->norm = ERROR_NORM_P;
00981
                input->p = jb_json_object_get_float (object, LABEL_P, &error_code);
00982
                if (!error_code)
00983
                  {
00984
                     input_error (_("Bad P parameter"));
00985
                     goto exit_on_error;
00986
00987
              }
            else if (!strcmp (buffer, LABEL_TAXICAB))
00988
              input->norm = ERROR_NORM_TAXICAB;
00989
00990
            else
00991
00992
                input_error (_("Unknown error norm"));
00993
                goto exit_on_error;
00994
00995
          }
00996
       else
00997
          input->norm = ERROR_NORM_EUCLIDIAN;
00998
00999
       // Closing the JSON document
01000 g_object_unref (parser);
01001
01002 #if DEBUG_INPUT
01003 fprintf (stderr, "input_open_json: end\n");
01004 #endif
01005
       return 1;
01006
01007 exit on error:
01008 g_object_unref (parser);
01009 #if DEBUG_INPUT
01010
        fprintf (stderr, "input_open_json: end\n");
01011 #endif
01012
       return 0;
01013 }
```

Here is the call graph for this function:

4.7.3.6 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 138 of file input.c.
00140
        char buffer2[64];
00141
        Experiment *experiment;
00142
        xmlNode *node, *child;
xmlChar *buffer;
00143
       int error_code;
00145
        unsigned int i, nsteps;
00146
00147 #if DEBUG_INPUT
       fprintf (stderr, "input_open_xml: start\n");
00148
00149 #endif
00150
00151
        // Resetting input data
00152
       buffer = NULL;
00153
       input->type = INPUT_TYPE_XML;
00154
00155
        // Getting the root node
00156 #if DEBUG_INPUT
00157
       fprintf (stderr, "input_open_xml: getting the root node\n");
00158 #endif
00159
        node = xmlDocGetRootElement (doc);
        if (xmlStrcmp (node->name, (const xmlChar *) LABEL_OPTIMIZE))
00160
00161
00162
            input_error (_("Bad root XML node"));
00163
            goto exit_on_error;
00164
00165
        \ensuremath{//} Getting result and variables file names
00166
00167
        if (!input->result)
00168
00169
            input->result
00170
              (char *) xmlGetProp (node, (const xmlChar *) LABEL_RESULT_FILE);
00171
            if (!input->result)
00172
              input->result = (char *) xmlStrdup ((const xmlChar *) result_name);
00173
00174 #if DEBUG_INPUT
00175
        fprintf (stderr, "input_open_xml: result file=%s\n", input->result);
00176 #endif
00177
       if (!input->variables)
00178
00179
            input->variables =
              (char *) xmlGetProp (node, (const xmlChar *) LABEL_VARIABLES_FILE);
00180
00181
            if (!input->variables)
              input->variables =
00182
00183
                 (char *) xmlStrdup ((const xmlChar *) variables_name);
00184
00185 #if DEBUG_INPUT
00186
       fprintf (stderr, "input_open_xml: variables file=%s\n", input->variables);
00187 #endif
00188
00189
         // Opening simulator program name
00190
        input->simulator
00191
          (char *) xmlGetProp (node, (const xmlChar *) LABEL_SIMULATOR);
        if (!input->simulator)
00192
00193
         {
00194
            input_error (_("Bad simulator program"));
00195
            goto exit_on_error;
00196
00197
00198
        // Opening evaluator program name
00199
        input->evaluator =
00200
          (char *) xmlGetProp (node, (const xmlChar *) LABEL_EVALUATOR);
00201
00202
        // Opening cleaner program name
00203
        input->cleaner = (char *) xmlGetProp (node, (const xmlChar *) LABEL_CLEANER);
00204
00205
        // Obtaining pseudo-random numbers generator seed
00206
       input->seed
00207
          = jb_xml_node_get_uint_with_default (node, (const xmlChar *) LABEL_SEED,
00208
                                                 &error_code, DEFAULT_RANDOM_SEED);
00209
        if (!error_code)
00210
            input_error (_("Bad pseudo-random numbers generator seed"));
00211
00212
            goto exit_on_error;
00213
00214
00215
        // Opening algorithm
        buffer = xmlGetProp (node, (const xmlChar *) LABEL_ALGORITHM);
if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_MONTE_CARLO))
00216
00217
00218
            input->algorithm = ALGORITHM_MONTE_CARLO;
00220
00221
            // Obtaining simulations number
00222
            input->nsimulations
               = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSIMULATIONS,
00223
00224
                                        &error code);
```

```
00225
           if (!error_code)
00226
00227
                input_error (_("Bad simulations number"));
00228
                goto exit_on_error;
00229
00230
        else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_SWEEP))
00231
00232
          input->algorithm = ALGORITHM_SWEEP;
00233
        else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_ORTHOGONAL))
         input->algorithm = ALGORITHM_ORTHOGONAL;
00234
        else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_GENETIC))
00235
00236
         {
00237
            input->algorithm = ALGORITHM_GENETIC;
00238
00239
            // Obtaining population
00240
            if (xmlHasProp (node, (const xmlChar *) LABEL_NPOPULATION))
00241
00242
                input->nsimulations
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NPOPULATION,
00244
                                           &error_code);
00245
                if (!error_code || input->nsimulations < 3)</pre>
00246
                    input_error (_("Invalid population number"));
00247
00248
                    goto exit_on_error;
00249
                  }
00250
00251
            else
00252
             {
                input_error (_("No population number"));
00253
00254
                goto exit_on_error;
00255
00256
00257
            // Obtaining generations
00258
            if (xmlHasProp (node, (const xmlChar *) LABEL_NGENERATIONS))
00259
                input->niterations
00260
00261
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NGENERATIONS,
                                           &error_code);
00262
00263
                if (!error_code || !input->niterations)
00264
00265
                    input_error (_("Invalid generations number"));
00266
                    goto exit_on_error;
00267
00268
              }
00269
            else
00270
              {
00271
                input_error (_("No generations number"));
00272
                goto exit_on_error;
00273
00274
00275
            // Obtaining mutation probability
00276
            if (xmlHasProp (node, (const xmlChar *) LABEL_MUTATION))
00277
00278
                input->mutation_ratio
                  = jb_xml_node_get_float (node, (const xmlChar *) LABEL_MUTATION,
00279
00280
                                            &error code);
                if (!error_code || input->mutation_ratio < 0.</pre>
00282
                    || input->mutation_ratio >= 1.)
00283
00284
                    input_error (_("Invalid mutation probability"));
00285
                    goto exit_on_error;
00286
00287
00288
00289
00290
                input_error (_("No mutation probability"));
00291
                goto exit_on_error;
              }
00292
00293
00294
            // Obtaining reproduction probability
00295
            if (xmlHasProp (node, (const xmlChar *) LABEL_REPRODUCTION))
00296
00297
                input->reproduction_ratio
                  = jb_xml_node_get_float (node, (const xmlChar *) LABEL_REPRODUCTION,
00298
00299
                                            &error code);
00300
                if (!error_code || input->reproduction_ratio < 0.</pre>
00301
                    || input->reproduction_ratio >= 1.0)
00302
                    input_error (_("Invalid reproduction probability"));
00303
00304
                    goto exit_on_error;
00305
00306
              }
00307
00308
00309
                input_error (_("No reproduction probability"));
00310
                goto exit_on_error;
00311
```

```
00312
00313
            // Obtaining adaptation probability
00314
            if (xmlHasProp (node, (const xmlChar *) LABEL_ADAPTATION))
00315
              {
00316
                input->adaptation ratio
00317
                   = jb_xml_node_get_float (node, (const xmlChar *) LABEL_ADAPTATION,
                                             &error_code);
00318
00319
                if (!error_code || input->adaptation_ratio < 0.</pre>
00320
                    || input->adaptation_ratio >= 1.)
00321
                    input_error (_("Invalid adaptation probability"));
00322
00323
                    goto exit_on_error;
00324
00325
00326
            else
00327
                input_error (_("No adaptation probability"));
00328
00329
                goto exit_on_error;
00330
00331
00332
            // Checking survivals
00333
            i = input->mutation_ratio * input->nsimulations;
            i += input->reproduction_ratio * input->nsimulations;
i += input->adaptation_ratio * input->nsimulations;
00334
00335
00336
            if (i > input->nsimulations - 2)
00337
             {
00338
                input_error
00339
                   (_("No enough survival entities to reproduce the population"));
00340
                goto exit_on_error;
00341
              }
00342
          }
00343
       else
00344
00345
            input_error (_("Unknown algorithm"));
00346
            goto exit_on_error;
00347
00348
        xmlFree (buffer);
00349
        buffer = NULL;
00350
00351
        if (input->algorithm == ALGORITHM_MONTE_CARLO
00352
               input->algorithm == ALGORITHM_SWEEP
            || input->algorithm == ALGORITHM_ORTHOGONAL)
00353
00354
00355
00356
            // Obtaining iterations number
00357
            input->niterations = jb_xml_node_get_uint_with_default
00358
              (node, (const xmlChar *) LABEL_NITERATIONS, &error_code, 1);
00359
            if (!error_code || !input->niterations)
             {
00360
00361
                input_error (_("Bad iterations number"));
00362
                goto exit_on_error;
00363
00364
00365
            // Obtaining best number
00366
            input->nbest
00367
              = jb xml node get uint with default (node,
00368
                                                     (const xmlChar *) LABEL_NBEST,
00369
                                                     &error_code, 1);
00370
            if (!error_code || !input->nbest)
00371
                input_error (_("Invalid best number"));
00372
00373
                goto exit_on_error;
00374
00375
00376
            // Obtaining tolerance
00377
            input->tolerance
00378
              = jb_xml_node_get_float_with_default (node,
00379
                                                      (const xmlChar *) LABEL_TOLERANCE,
00380
                                                      &error code, 0.);
00381
            if (!error_code || input->tolerance < 0.)</pre>
00382
                input_error (_("Invalid tolerance"));
00383
00384
                goto exit_on_error;
00385
00386
            // Getting hill climbing method parameters
00387
00388
            if (xmlHasProp (node, (const xmlChar *) LABEL_NSTEPS))
00389
00390
                input->nsteps =
00391
                   jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSTEPS,
00392
                                         &error_code);
00393
                if (!error_code)
00394
00395
                    input_error (_("Invalid steps number"));
00396
                    goto exit_on_error;
00397
00398
              }
```

```
00399
            else
00400
              input->nsteps = 0;
00401
            if (xmlHasProp (node, (const xmlChar *) LABEL_NFINAL_STEPS))
00402
00403
                input->nfinal steps =
                  jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NFINAL_STEPS,
00404
                                         &error_code);
00406
                if (!error_code)
00407
                    input_error (_("Invalid final steps number"));
00408
00409
                    goto exit_on_error;
00410
00411
00412
00413
              input->nfinal_steps = input->nsteps;
00414
            nsteps = JBM_MAX (input->nsteps, input->nfinal_steps);
00415 #if DEBUG_INPUT
00416
            fprintf (stderr, "input_open_xml: nsteps=%u\n", nsteps);
00418
           if (nsteps)
00419
00420
                buffer = xmlGetProp (node, (const xmlChar *) LABEL_CLIMBING);
                if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_COORDINATES))
  input->climbing = CLIMBING_METHOD_COORDINATES;
00421
00422
00423
                else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_RANDOM))
00424
                  {
00425
                    input->climbing = CLIMBING_METHOD_RANDOM;
                    input->nestimates
00426
00427
                       = jb_xml_node_get_uint (node,
00428
                                                (const xmlChar *) LABEL NESTIMATES.
00429
                                               &error code):
00430
                    if (!error_code || !input->nestimates)
00431
00432
                        input_error (_("Invalid estimates number"));
00433
                        goto exit_on_error;
00434
00435
                  }
00436
                else
00437
                 {
00438
                    input_error (_("Unknown method to estimate the hill climbing"));
00439
                    goto exit_on_error;
00440
                xmlFree (buffer);
00441
00442
                buffer = NULL;
00443
                input->relaxation
00444
                   = jb_xml_node_get_float_with_default (node,
00445
                                                          (const xmlChar *)
00446
                                                          LABEL RELAXATION,
00447
                                                          &error_code,
                                                          DEFAULT_RELAXATION);
00448
                if (!error_code || input->relaxation < 0. || input->relaxation > 2.)
00450
00451
                    input_error (_("Invalid relaxation parameter"));
00452
                    goto exit_on_error;
00453
00454
              }
00455
00456
        // Obtaining the threshold
00457
        input->threshold =
00458
          jb_xml_node_get_float_with_default (node, (const xmlChar *) LABEL_THRESHOLD,
00459
                                               &error code, 0.);
00460
        if (!error code)
00461
         {
00462
            input_error (_("Invalid threshold"));
00463
            goto exit_on_error;
00464
00465
00466
        // Reading the experimental data
00467
        for (child = node->children; child; child = child->next)
00468
00469
            if (xmlStrcmp (child->name, (const xmlChar *) LABEL_EXPERIMENT))
00470
             break;
00471 #if DEBUG_INPUT
           fprintf (stderr, "input_open_xml: nexperiments=%u\n",
00472
00473
                     input->nexperiments);
00474 #endif
00475
           input->experiment = experiment = (Experiment *)
00476
             g_realloc (input->experiment,
00477
                          (1 + input->nexperiments) * sizeof (Experiment));
00478
            if (!input->nexperiments)
00479
00480
                if (!experiment_open_xml (experiment, child, 0))
                  goto exit_on_error;
00481
00482
              }
00483
            else
00484
00485
                if (!experiment open xml (experiment + input->nexperiments.
```

```
child, experiment->ninputs))
00486
00487
                   goto exit_on_error;
00488
                 if (experiment[experiment->ninputs].template_flags
00489
                     != experiment->template_flags)
00490
                     input_error ("bad template inputs");
00491
00492
                     goto exit_on_error;
00493
00494
00495
            ++input->nexperiments;
00496 #if DEBUG_INPUT
       fprintf (stderr, "input_open_xml: nexperiments=%u\n",
00497
00498
                      input->nexperiments);
00499 #endif
00500
00501
        if (!input->nexperiments)
00502
            input_error (_("No optimization experiments"));
goto exit_on_error;
00503
00504
00505
00506
        input->template_flags = experiment->template_flags;
00507
        buffer = NULL;
00508
00509
        // Reading the variables data
if (input->algorithm == ALGORITHM_SWEEP
00510
            || input->algorithm == ALGORITHM_ORTHOGONAL)
00511
00512
          input->nsimulations = 1;
00513
        for (; child; child = child->next)
00514
00515 #if DEBUG_INPUT
00516
            fprintf (stderr, "input_open_xml: nvariables=%u\n", input->nvariables);
00517 #endif
00518
            if (xmlStrcmp (child->name, (const xmlChar *) LABEL_VARIABLE))
00519
                snprintf (buffer2, 64, "%s %u: %s",
    _("Variable"), input->nvariables + 1, _("bad XML node"));
input_error (buffer2);
00520
00521
00522
                goto exit_on_error;
00524
00525
            input->variable = (Variable *)
00526
              g_realloc (input->variable,
00527
                          (1 + input->nvariables) * sizeof (Variable));
            if (!variable_open_xml (input->variable + input->nvariables, child,
00528
00529
                                      input->algorithm, input->nsteps))
00530
               goto exit_on_error;
00531
            if (input->algorithm == ALGORITHM_SWEEP
00532
                 || input->algorithm == ALGORITHM_ORTHOGONAL)
00533
               input->nsimulations *= input->variable[input->nvariables].nsweeps;
            ++input->nvariables;
00534
00535
00536
        if (!input->nvariables)
00537
00538
             input_error (_("No optimization variables"));
00539
            goto exit_on_error;
00540
00541
        if
           (input->nbest > input->nsimulations)
00542
00543
             input_error (_("Best number higher than simulations number"));
00544
            goto exit_on_error;
00545
00546
        buffer = NULL:
00547
00548
        // Obtaining the error norm
00549
        if (xmlHasProp (node, (const xmlChar *) LABEL_NORM))
00550
00551
            buffer = xmlGetProp (node, (const xmlChar *) LABEL_NORM);
00552
            if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_EUCLIDIAN))
input->norm = ERROR_NORM_EUCLIDIAN;
00553
00554
            else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_MAXIMUM))
               input->norm = ERROR_NORM_MAXIMUM;
00556
             else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_P))
00557
00558
                input->norm = ERROR_NORM_P;
00559
                 input->p
00560
                   = jb xml node get float (node, (const xmlChar *) LABEL P,
00561
                                              &error_code);
00562
                 if (!error_code)
00563
                     input_error (_("Bad P parameter"));
00564
00565
                     goto exit_on_error;
00566
00567
00568
             else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_TAXICAB))
00569
               input->norm = ERROR_NORM_TAXICAB;
00570
            else
00571
              {
00572
                 input error ( ("Unknown error norm"));
```

```
goto exit_on_error;
00574
00575
               xmlFree (buffer);
00576
             }
00577
          else
00578
             input->norm = ERROR_NORM_EUCLIDIAN;
00579
00580 // Closing the XML document 00581 xmlFreeDoc (doc);
00582

00583 #if DEBUG_INPUT

00584 fprintf (stderr, "input_open_xml: end\n");
00585 #endif
00586 return 1;
00587
00588 exit_on_error:

00589 xmlFree (buffer);

00590 xmlFreeDoc (doc);

00591 #if DEBUG_INPUT
00592
         fprintf (stderr, "input_open_xml: end\n");
00593 #endif
00594 return 0;
```

Here is the call graph for this function:

4.7.4 Variable Documentation

4.7.4.1 input

```
Input input[1]
```

Global Input struct to set the input data.

Definition at line 57 of file input.c.

4.7.4.2 result_name

```
const char* result_name = "result"
```

Name of the result file.

Definition at line 59 of file input.c.

4.7.4.3 variables name

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

Name of the variables file.

Definition at line 60 of file input.c.

4.8 input.c 101

4.8 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.
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
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 <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/xml.h"
00048 #include "jb/src/json.h'
00049 #include "jb/src/win.h"
00050 #include "tools.h"
00051 #include "experiment.h"
00052 #include "variable.h"
00053 #include "input.h'
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");
00070 #endif
00071 input->nvariables = input->nexperiments = input->nsteps = input->nfinal_steps
00072
           = 0;
00073
        input->simulator = input->evaluator = input->cleaner = input->directory
          = input->name = NULL;
00074
00075
        input->experiment = NULL;
        input->variable = NULL;
00076
00077 #if DEBUG_INPUT
00078 fprintf (stderr, "input_new: end\n");
00075 In
00080 }
00081
00085 void
00086 input_free ()
00087 {
00088
        unsigned int i;
00089 #if DEBUG_INPUT
        fprintf (stderr, "input_free: start\n");
00090
00091 #endif
00092
        g_free (input->name);
00093
        g_free (input->directory);
00094 for (i = 0; i < input->nexperiments; ++i)
```

```
experiment_free (input->experiment + i, input->type);
00096
        for (i = 0; i < input->nvariables; ++i)
00097
         variable_free (input->variable + i, input->type);
        g_free (input->experiment);
00098
        g_free (input->variable);
00099
        if (input->type == INPUT_TYPE_XML)
00100
00102
            xmlFree (input->cleaner);
00103
            xmlFree (input->evaluator);
00104
            xmlFree (input->simulator);
00105
            xmlFree (input->result);
00106
           xmlFree (input->variables);
00107
00108
        else
00109
00110
            g_free (input->cleaner);
00111
            g_free (input->evaluator);
            g_free (input->simulator);
00112
            g_free (input->result);
00113
00114
           g_free (input->variables);
00115
00116
       input->nexperiments = input->nvariables = input->nsteps
00117 = input->nfinal_steps = 0;
00118 #if DEBUG_INPUT
00119
       fprintf (stderr, "input_free: end\n");
00120 #endif
00121 }
00122
00126 static void
00127 input_error (char *message)
00128 {
00129
       error_message = g_strconcat (_("Input"), ": ", message, "\n", NULL);
00130 }
00131
00137 static inline int
00138 input_open_xml (xmlDoc * doc)
00139 {
       char buffer2[64];
00141
       Experiment *experiment;
00142
       xmlNode *node, *child;
00143
       xmlChar *buffer;
00144
       int error_code;
       unsigned int i, nsteps:
00145
00146
00147 #if DEBUG_INPUT
00148
       fprintf (stderr, "input_open_xml: start\n");
00149 #endif
00150
        // Resetting input data
00151
00152
       buffer = NULL;
       input->type = INPUT_TYPE_XML;
00153
00154
00155
       // Getting the root node
00156 #if DEBUG_INPUT
       fprintf (stderr, "input_open_xml: getting the root node\n");
00157
00158 #endif
       node = xmlDocGetRootElement (doc);
        if (xmlStrcmp (node->name, (const xmlChar *) LABEL_OPTIMIZE))
00160
00161
00162
            input_error (_("Bad root XML node"));
00163
           goto exit_on_error;
00164
00165
00166
       // Getting result and variables file names
00167
        if (!input->result)
00168
00169
              (char *) xmlGetProp (node, (const xmlChar *) LABEL_RESULT_FILE);
00170
            if (!input->result)
00171
             input->result = (char *) xmlStrdup ((const xmlChar *) result_name);
00172
00173
00174 #if DEBUG_INPUT
00175
       fprintf (stderr, "input_open_xml: result file=%s\n", input->result);
00176 #endif
00177
       if (!input->variables)
00178
00179
            input->variables =
00180
              (char *) xmlGetProp (node, (const xmlChar *) LABEL_VARIABLES_FILE);
00181
            if (!input->variables)
             input->variables =
00182
00183
                (char *) xmlStrdup ((const xmlChar *) variables_name);
00184
00185 #if DEBUG_INPUT
00186
       fprintf (stderr, "input_open_xml: variables file=%s\n", input->variables);
00187 #endif
00188
00189
       // Opening simulator program name
```

4.8 input.c 103

```
input->simulator =
          (char *) xmlGetProp (node, (const xmlChar *) LABEL_SIMULATOR);
00191
00192
           (!input->simulator)
00193
        {
00194
            input_error (_("Bad simulator program"));
00195
           goto exit_on_error;
00196
00197
00198
        // Opening evaluator program name
        input->evaluator
00199
          (char *) xmlGetProp (node, (const xmlChar *) LABEL EVALUATOR);
00200
00201
00202
        // Opening cleaner program name
        input->cleaner = (char *) xmlGetProp (node, (const xmlChar *) LABEL_CLEANER);
00203
00204
00205
        // Obtaining pseudo-random numbers generator seed
        input->seed
00206
00207
          = jb_xml_node_get_uint_with_default (node, (const xmlChar *) LABEL_SEED,
                                                &error_code, DEFAULT_RANDOM_SEED);
00208
00209
        if (!error_code)
00210
00211
            input_error (_("Bad pseudo-random numbers generator seed"));
00212
           goto exit_on_error;
00213
00214
00215
        // Opening algorithm
00216
        buffer = xmlGetProp (node, (const xmlChar *) LABEL_ALGORITHM);
00217
        if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_MONTE_CARLO))
00218
00219
            input->algorithm = ALGORITHM MONTE CARLO;
00220
00221
            // Obtaining simulations number
00222
            input->nsimulations
00223
              = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSIMULATIONS,
00224
                                      &error_code);
00225
            if (!error_code)
00226
             {
                input_error (_("Bad simulations number"));
00228
                goto exit_on_error;
00229
00230
00231
       else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_SWEEP))
         input->algorithm = ALGORITHM_SWEEP;
00232
        else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_ORTHOGONAL))
00233
00234
         input->algorithm = ALGORITHM_ORTHOGONAL;
00235
        else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_GENETIC))
00236
00237
            input->algorithm = ALGORITHM_GENETIC;
00238
00239
            // Obtaining population
00240
            if (xmlHasProp (node, (const xmlChar *) LABEL_NPOPULATION))
00241
00242
                input->nsimulations
00243
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NPOPULATION,
00244
                                           &error_code);
00245
                if (!error_code || input->nsimulations < 3)</pre>
00246
00247
                    input_error (_("Invalid population number"));
00248
                    goto exit_on_error;
00249
                  }
00250
              }
00251
            else
00252
             {
00253
                input_error (_("No population number"));
00254
                goto exit_on_error;
00255
00256
            // Obtaining generations
00257
00258
            if (xmlHasProp (node, (const xmlChar *) LABEL_NGENERATIONS))
              {
00260
                input->niterations
00261
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NGENERATIONS,
00262
                                           &error_code);
                if (!error_code || !input->niterations)
00263
00264
                 {
                   input_error (_("Invalid generations number"));
00265
00266
                    goto exit_on_error;
00267
00268
             }
00269
            else
00270
             {
00271
                input_error (_("No generations number"));
00272
                goto exit_on_error;
00273
00274
            \//\ Obtaining mutation probability
00275
            if (xmlHasProp (node, (const xmlChar *) LABEL_MUTATION))
00276
```

```
00277
              {
00278
                input->mutation_ratio
00279
                   = jb_xml_node_get_float (node, (const xmlChar *) LABEL_MUTATION,
00280
                                             &error_code);
                if (!error_code || input->mutation_ratio < 0.</pre>
00281
00282
                     || input->mutation ratio >= 1.)
00284
                     input_error (_("Invalid mutation probability"));
00285
                    goto exit_on_error;
00286
00287
00288
            else
00289
00290
                input_error (_("No mutation probability"));
00291
                goto exit_on_error;
00292
00293
00294
            // Obtaining reproduction probability
            if (xmlHasProp (node, (const xmlChar *) LABEL_REPRODUCTION))
00296
              {
00297
                input->reproduction_ratio
00298
                   = jb_xml_node_get_float (node, (const xmlChar *) LABEL_REPRODUCTION,
00299
                                             &error_code);
                if (!error_code || input->reproduction_ratio < 0.</pre>
00300
00301
                     || input->reproduction_ratio >= 1.0)
00302
00303
                     input_error (_("Invalid reproduction probability"));
00304
                    goto exit_on_error;
00305
00306
              }
00307
            else
00308
              {
00309
                input_error (_("No reproduction probability"));
00310
                goto exit_on_error;
00311
00312
00313
            // Obtaining adaptation probability
            if (xmlHasProp (node, (const xmlChar *) LABEL_ADAPTATION))
00314
00315
              {
00316
                input->adaptation_ratio
00317
                   = jb_xml_node_get_float (node, (const xmlChar *) LABEL_ADAPTATION,
00318
                                             &error_code);
                if (!error code || input->adaptation ratio < 0.
00319
00320
                     || input->adaptation_ratio >= 1.)
00321
00322
                     input_error (_("Invalid adaptation probability"));
00323
                    goto exit_on_error;
00324
00325
              }
00326
            else
00327
              {
00328
                input_error (_("No adaptation probability"));
00329
                goto exit_on_error;
00330
00331
00332
            // Checking survivals
            i = input->mutation_ratio * input->nsimulations;
00334
            i += input->reproduction_ratio * input->nsimulations;
00335
            i += input->adaptation_ratio * input->nsimulations;
            if (i > input->nsimulations - 2)
00336
00337
              {
00338
                input error
00339
                   (_("No enough survival entities to reproduce the population"));
00340
                goto exit_on_error;
00341
00342
00343
        else
00344
00345
            input_error (_("Unknown algorithm"));
00346
            goto exit_on_error;
00347
00348
        xmlFree (buffer);
00349
        buffer = NULL;
00350
00351
        if (input->algorithm == ALGORITHM_MONTE_CARLO
            || input->algorithm == ALGORITHM_SWEEP
00352
00353
            || input->algorithm == ALGORITHM_ORTHOGONAL)
00354
00355
00356
            // Obtaining iterations number
            input->niterations = jb_xml_node_get_uint_with_default
  (node, (const xmlChar *) LABEL_NITERATIONS, &error_code, 1);
00357
00358
00359
               (!error_code || !input->niterations)
00360
00361
                input_error (_("Bad iterations number"));
00362
                goto exit_on_error;
00363
```

4.8 input.c 105

```
00364
00365
            // Obtaining best number
            input->nbest
00366
00367
              = jb_xml_node_get_uint_with_default (node,
00368
                                                     (const xmlChar *) LABEL NBEST,
00369
                                                     &error code, 1);
00370
            if (!error_code || !input->nbest)
00371
              {
00372
                input_error (_("Invalid best number"));
00373
                goto exit_on_error;
00374
00375
00376
            // Obtaining tolerance
            input->tolerance
00377
00378
               = jb_xml_node_get_float_with_default (node,
00379
                                                      (const xmlChar *) LABEL_TOLERANCE,
00380
                                                      &error_code, 0.);
00381
            if (!error_code || input->tolerance < 0.)</pre>
00382
                input_error (_("Invalid tolerance"));
00383
00384
                goto exit_on_error;
00385
00386
00387
            // Getting hill climbing method parameters
00388
            if (xmlHasProp (node, (const xmlChar *) LABEL_NSTEPS))
00389
00390
                input->nsteps =
00391
                  jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSTEPS,
00392
                                          &error_code);
00393
                if (!error_code)
00394
00395
                    input_error (_("Invalid steps number"));
00396
                    goto exit_on_error;
00397
00398
00399
            else
00400
              input->nsteps = 0;
               (xmlHasProp (node, (const xmlChar *) LABEL_NFINAL_STEPS))
00402
              {
00403
                input->nfinal_steps =
00404
                   jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NFINAL_STEPS,
00405
                                         &error_code);
00406
                if (!error_code)
00407
                 {
                   input_error (_("Invalid final steps number"));
00408
00409
                    goto exit_on_error;
00410
00411
              }
00412
            else
              input->nfinal_steps = input->nsteps;
00413
00414
            nsteps = JBM_MAX (input->nsteps, input->nfinal_steps);
00415 #if DEBUG_INPUT
00416
            fprintf (stderr, "input_open_xml: nsteps=%u\n", nsteps);
00417 #endif
            if (nsteps)
00418
00419
              {
                buffer = xmlGetProp (node, (const xmlChar *) LABEL_CLIMBING);
                if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_COORDINATES))
  input->climbing = CLIMBING_METHOD_COORDINATES;
00421
00422
00423
                else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_RANDOM))
00424
00425
                    input->climbing = CLIMBING METHOD RANDOM;
00426
                     input->nestimates
00427
                      = jb_xml_node_get_uint (node,
00428
                                                (const xmlChar *) LABEL_NESTIMATES,
00429
                                                &error_code);
00430
                     if (!error_code || !input->nestimates)
00431
00432
                        input_error (_("Invalid estimates number"));
00433
                        goto exit_on_error;
00434
00435
00436
                else
00437
00438
                    input_error (_("Unknown method to estimate the hill climbing"));
00439
                    goto exit_on_error;
00440
00441
                xmlFree (buffer);
00442
                buffer = NULL:
                input->relaxation
00443
00444
                  = jb_xml_node_get_float_with_default (node,
00445
                                                           (const xmlChar *)
                                                          LABEL_RELAXATION,
00446
00447
                                                           &error_code,
00448
                                                          DEFAULT_RELAXATION);
                if (!error_code || input->relaxation < 0. || input->relaxation > 2.)
00449
00450
```

```
input_error (_("Invalid relaxation parameter"));
00452
                   goto exit_on_error;
00453
                 }
00454
             }
00455
00456
        // Obtaining the threshold
       input->threshold =
00457
00458
          jb_xml_node_get_float_with_default (node, (const xmlChar *) LABEL_THRESHOLD,
00459
                                              &error_code, 0.);
00460
       if (!error code)
00461
        {
           input_error (_("Invalid threshold"));
00462
00463
           goto exit_on_error;
00464
00465
00466
        // Reading the experimental data
       for (child = node->children; child; child = child->next)
00467
00468
            if (xmlStrcmp (child->name, (const xmlChar *) LABEL_EXPERIMENT))
00470
              break;
00471 #if DEBUG_INPUT
           fprintf (stderr, "input_open_xml: nexperiments=%u\n",
00472
00473
                    input->nexperiments);
00474 #endif
00475
           input->experiment = experiment = (Experiment *)
             g_realloc (input->experiment,
00476
00477
                         (1 + input->nexperiments) * sizeof (Experiment));
00478
            if (!input->nexperiments)
00479
00480
                if (!experiment_open_xml (experiment, child, 0))
00481
                 goto exit_on_error;
00482
00483
00484
             {
00485
               if (!experiment_open_xml (experiment + input->nexperiments,
00486
                                          child, experiment->ninputs))
00487
                 goto exit on error;
                if (experiment[experiment->ninputs].template_flags
00489
                   != experiment->template_flags)
00490
00491
                   input_error ("bad template inputs");
00492
                   goto exit_on_error;
00493
00494
           ++input->nexperiments;
00495
00496 #if DEBUG_INPUT
00497
       fprintf (stderr, "input_open_xml: nexperiments=%u\n",
00498
                     input->nexperiments);
00499 #endif
00500
          (!input->nexperiments)
00502
00503
            input_error (_("No optimization experiments"));
00504
           goto exit_on_error;
00505
00506
        input->template flags = experiment->template flags;
00507
       buffer = NULL:
00508
00509
        // Reading the variables data
       \begin{array}{lll} \textbf{if} & \texttt{(input->algorithm == ALGORITHM\_SWEEP} \end{array}
00510
            || input->algorithm == ALGORITHM_ORTHOGONAL)
00511
          input->nsimulations = 1;
00512
00513
       for (; child; child = child->next)
00514
00515 #if DEBUG INPUT
00516
           fprintf (stderr, "input_open_xml: nvariables=%u\n", input->nvariables);
00517 #endif
00518
           if (xmlStrcmp (child->name, (const xmlChar *) LABEL VARIABLE))
00519
               00521
00522
               input_error (buffer2);
00523
               goto exit_on_error;
00524
00525
            input->variable = (Variable *)
            g_realloc (input->variable,
00526
00527
                        (1 + input->nvariables) * sizeof (Variable));
00528
            if (!variable_open_xml (input->variable + input->nvariables, child,
00529
                                    input->algorithm, input->nsteps))
00530
              goto exit on error:
            if (input->algorithm == ALGORITHM_SWEEP
00531
                || input->algorithm == ALGORITHM_ORTHOGONAL)
00532
00533
              input->nsimulations *= input->variable[input->nvariables].nsweeps;
00534
            ++input->nvariables;
00535
        if (!input->nvariables)
00536
00537
```

4.8 input.c 107

```
input_error (_("No optimization variables"));
00539
            goto exit_on_error;
00540
00541
        if (input->nbest > input->nsimulations)
00542
00543
            input_error (_("Best number higher than simulations number"));
00544
           goto exit_on_error;
00545
00546
        buffer = NULL;
00547
00548
        // Obtaining the error norm
        if (xmlHasProp (node, (const xmlChar *) LABEL_NORM))
00549
00550
00551
            buffer = xmlGetProp (node, (const xmlChar *) LABEL_NORM);
00552
            if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_EUCLIDIAN))
00553
              input->norm = ERROR_NORM_EUCLIDIAN;
            else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_MAXIMUM))
  input->norm = ERROR_NORM_MAXIMUM;
00554
00555
            else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_P))
00557
              {
00558
                input->norm = ERROR_NORM_P;
00559
                input->p
00560
                  = jb_xml_node_get_float (node, (const xmlChar *) LABEL_P,
00561
                                            &error_code);
00562
                if (!error_code)
00563
00564
                    input_error (_("Bad P parameter"));
00565
                    goto exit_on_error;
00566
00567
              }
00568
            else if (!xmlStrcmp (buffer, (const xmlChar *) LABEL_TAXICAB))
00569
              input->norm = ERROR_NORM_TAXICAB;
00570
00571
             {
00572
                input_error (_("Unknown error norm"));
00573
                goto exit_on_error;
00574
00575
            xmlFree (buffer);
00576
00577
00578
          input->norm = ERROR_NORM_EUCLIDIAN;
00579
00580
       // Closing the XML document
00581
       xmlFreeDoc (doc);
00582
00583 #if DEBUG_INPUT
00584
       fprintf (stderr, "input_open_xml: end\n");
00585 #endif
00586
       return 1:
00587
00588 exit_on_error:
00589 xmlFree (buffer);
00590
        xmlFreeDoc (doc);
00591 #if DEBUG_INPUT
       fprintf (stderr, "input_open_xml: end\n");
00592
00593 #endif
00594
       return 0;
00595 }
00596
00602 static inline int
00603 input_open_json (JsonParser * parser)
00604 {
00605
       Experiment *experiment;
       JsonNode *node, *child;
00606
00607
        JsonObject *object;
00608
       JsonArray *array;
00609
       const char *buffer;
00610
       int error_code;
00611
       unsigned int i, n:
00612
00613 #if DEBUG_INPUT
00614
       fprintf (stderr, "input_open_json: start\n");
00615 #endif
00616
00617
        // Resetting input data
00618
       input->type = INPUT_TYPE_JSON;
00619
00620
       // Getting the root node
00621 #if DEBUG_INPUT
00622
       fprintf (stderr, "input_open_json: getting the root node\n");
00623 #endif
00624
        node = json_parser_get_root (parser);
00625
        object = json_node_get_object (node);
00626
00627
        // Getting result and variables file names
00628
        if (!input->result)
00629
          {
```

```
buffer = json_object_get_string_member (object, LABEL_RESULT_FILE);
           if (!buffer)
00631
00632
             buffer = result_name;
00633
            input->result = g_strdup (buffer);
00634
00635
        else
00636
         input->result = g_strdup (result_name);
00637
        if (!input->variables)
00638
           buffer = json_object_get_string_member (object, LABEL_VARIABLES_FILE);
00639
            if (!buffer)
00640
00641
             buffer = variables name:
00642
            input->variables = g_strdup (buffer);
00643
00644
        else
00645
         input->variables = g_strdup (variables_name);
00646
00647
        // Opening simulator program name
        buffer = json_object_get_string_member (object, LABEL_SIMULATOR);
00648
00649
        if (!buffer)
00650
00651
            input_error (_("Bad simulator program"));
00652
           goto exit_on_error;
00653
00654
        input->simulator = q_strdup (buffer);
00655
        // Opening evaluator program name
00656
00657
        buffer = json_object_get_string_member (object, LABEL_EVALUATOR);
        if (buffer)
00658
00659
          input->evaluator = q_strdup (buffer);
00660
00661
        // Opening cleaner program name
00662
        buffer = json_object_get_string_member (object, LABEL_CLEANER);
00663
        if (buffer)
00664
         input->cleaner = g_strdup (buffer);
00665
00666
        // Obtaining pseudo-random numbers generator seed
00667
00668
          = jb_json_object_get_uint_with_default (object, LABEL_SEED,
00669
                                                   &error_code, DEFAULT_RANDOM_SEED);
00670
        if (!error_code)
00671
        {
00672
           input_error (_("Bad pseudo-random numbers generator seed"));
00673
           goto exit_on_error;
00674
00675
00676
        // Opening algorithm
00677
        buffer = json_object_get_string_member (object, LABEL_ALGORITHM);
        if (!strcmp (buffer, LABEL_MONTE_CARLO))
00678
00679
00680
            input->algorithm = ALGORITHM_MONTE_CARLO;
00681
00682
            // Obtaining simulations number
00683
            input->nsimulations
              = jb_json_object_get_uint (object, LABEL_NSIMULATIONS, &error_code);
00684
00685
            if (!error code)
00686
00687
                input_error (_("Bad simulations number"));
00688
                goto exit_on_error;
00689
00690
        else if (!strcmp (buffer, LABEL_SWEEP))
00691
00692
         input->algorithm = ALGORITHM_SWEEP;
00693
        else if (!strcmp (buffer, LABEL_ORTHOGONAL))
00694
          input->algorithm = ALGORITHM_ORTHOGONAL;
00695
        else if (!strcmp (buffer, LABEL_GENETIC))
00696
00697
            input->algorithm = ALGORITHM_GENETIC;
00698
            // Obtaining population
00700
            if (json_object_get_member (object, LABEL_NPOPULATION))
00701
00702
                input->nsimulations
                  = jb_json_object_get_uint (object, LABEL_NPOPULATION, &error_code);
00703
00704
                if (!error_code || input->nsimulations < 3)</pre>
00705
00706
                    input_error (_("Invalid population number"));
00707
                    goto exit_on_error;
00708
                  }
00709
00710
            else
00711
00712
                input_error (_("No population number"));
00713
                goto exit_on_error;
00714
00715
00716
            // Obtaining generations
```

4.8 input.c 109

```
if (json_object_get_member (object, LABEL_NGENERATIONS))
00718
00719
                 input->niterations
00720
                   = jb_json_object_get_uint_with_default (object, LABEL_NGENERATIONS,
00721
                                                              &error_code, 1);
00722
                 if (!error code || !input->niterations)
00723
00724
                     input_error (_("Invalid generations number"));
00725
                     goto exit_on_error;
00726
00727
00728
            else
00729
              {
00730
                 input_error (_("No generations number"));
00731
                 goto exit_on_error;
00732
00733
00734
             // Obtaining mutation probability
             if (json_object_get_member (object, LABEL_MUTATION))
00736
              {
                 input->mutation_ratio
00737
00738
                   = jb_json_object_get_float (object, LABEL_MUTATION, &error_code);
00739
                 if (!error_code || input->mutation_ratio < 0.</pre>
00740
                     || input->mutation_ratio >= 1.)
00741
                   {
00742
                     input_error (_("Invalid mutation probability"));
00743
                     goto exit_on_error;
00744
00745
              }
00746
            else
00747
              {
00748
                 input_error (_("No mutation probability"));
00749
                 goto exit_on_error;
00750
00751
             \//\ Obtaining reproduction probability
00752
00753
             if (json_object_get_member (object, LABEL_REPRODUCTION))
00754
00755
                 input->reproduction_ratio
00756
                    = jb_json_object_get_float (object, LABEL_REPRODUCTION,
00757
                                                 &error_code);
00758
                 if (!error code || input->reproduction ratio < 0.
00759
                     || input->reproduction_ratio >= 1.0)
00760
00761
                     input_error (_("Invalid reproduction probability"));
00762
                     goto exit_on_error;
00763
00764
              }
00765
            else
00766
              {
00767
                 input_error (_("No reproduction probability"));
00768
                 goto exit_on_error;
00769
00770
00771
             // Obtaining adaptation probability
00772
             if (json_object_get_member (object, LABEL_ADAPTATION))
00773
00774
                 input->adaptation_ratio
00775
                   = jb_json_object_get_float (object, LABEL_ADAPTATION, &error_code);
00776
                 if (!error_code || input->adaptation_ratio < 0.</pre>
00777
                     || input->adaptation_ratio >= 1.)
00778
00779
                     input_error (_("Invalid adaptation probability"));
00780
                     goto exit_on_error;
00781
                   }
00782
00783
             else
00784
00785
                 input_error (_("No adaptation probability"));
00786
                 goto exit_on_error;
00787
00788
00789
             // Checking survivals
            i = input->mutation_ratio * input->nsimulations;
i += input->reproduction_ratio * input->nsimulations;
i += input->adaptation_ratio * input->nsimulations;
00790
00791
00792
00793
             if (i > input->nsimulations - 2)
00794
00795
00796
                   (_("No enough survival entities to reproduce the population"));
00797
                 goto exit_on_error;
00798
               }
00799
00800
        else
00801
            input_error (_("Unknown algorithm"));
00802
00803
             goto exit on error:
```

```
00804
          }
00805
00806
        if (input->algorithm == ALGORITHM_MONTE_CARLO
            || input->algorithm == ALGORITHM_SWEEP
|| input->algorithm == ALGORITHM_ORTHOGONAL)
00807
00808
00809
00811
            // Obtaining iterations number
00812
            input->niterations
               = jb_json_object_get_uint (object, LABEL_NITERATIONS, &error_code);
00813
            if (!error_code || !input->niterations)
00814
00815
00816
                input_error (_("Bad iterations number"));
00817
                goto exit_on_error;
00818
00819
            // Obtaining best number
00820
00821
            input->nbest
00822
              = jb_json_object_get_uint_with_default (object, LABEL_NBEST,
00823
                                                        &error_code, 1);
00824
            if (!error_code || !input->nbest)
00825
                input_error (_("Invalid best number"));
00826
00827
                goto exit_on_error;
00828
00829
00830
            // Obtaining tolerance
00831
            input->tolerance
               = jb_json_object_get_float_with_default (object, LABEL_TOLERANCE,
00832
00833
                                                          &error_code, 0.);
00834
            if (!error code || input->tolerance < 0.)
00835
              {
00836
                input_error (_("Invalid tolerance"));
00837
                goto exit_on_error;
00838
00839
00840
            // Getting hill climbing method parameters
            if (json_object_get_member (object, LABEL_NSTEPS))
00842
              {
00843
                input->nsteps
00844
                   = jb_json_object_get_uint (object, LABEL_NSTEPS, &error_code);
00845
                if (!error_code)
00846
00847
                     input_error (_("Invalid steps number"));
00848
                     goto exit_on_error;
00849
00850
                if (json_object_has_member (object, LABEL_NFINAL_STEPS))
00851
00852
                     input->nfinal steps
                       = jb_json_object_get_uint (object, LABEL_NFINAL_STEPS,
00853
                                                   &error_code);
00855
                     if (!error_code)
00856
00857
                         input_error (_("Invalid final steps number"));
00858
                         goto exit_on_error;
00859
00861
00862
                  input->nfinal_steps = input->nsteps;
00863
                buffer = json_object_get_string_member (object, LABEL_CLIMBING);
                if (!strcmp (buffer, LABEL_COORDINATES))
  input->climbing = CLIMBING_METHOD_COORDINATES;
00864
00865
00866
                else if (!strcmp (buffer, LABEL_RANDOM))
00867
00868
                     input->climbing = CLIMBING_METHOD_RANDOM;
                     input->nestimates
00869
                       = jb_json_object_get_uint (object, LABEL_NESTIMATES,
00870
00871
                                                   &error_code);
00872
                     if (!error_code || !input->nestimates)
00874
                         input_error (_("Invalid estimates number"));
00875
                         goto exit_on_error;
00876
00877
00878
                else
00879
08800
                     input_error (_("Unknown method to estimate the hill climbing"));
00881
                     goto exit_on_error;
00882
00883
                input->relaxation
                   = jb_json_object_get_float_with_default (object, LABEL_RELAXATION,
00884
00885
                                                              &error_code,
00886
                                                              DEFAULT_RELAXATION);
00887
                if (!error_code || input->relaxation < 0. || input->relaxation > 2.)
00888
                    input_error (_("Invalid relaxation parameter"));
00889
00890
                     goto exit on error:
```

4.8 input.c 111

```
00891
00892
00893
             else
00894
              input->nsteps = input->nfinal_steps = 0;
00895
00896
        // Obtaining the threshold
00897
        input->threshold
00898
           = jb_json_object_get_float_with_default (object, LABEL_THRESHOLD,
00899
                                                       &error_code, 0.);
00900
00901
        if (!error_code)
00902
         {
00903
             input_error (_("Invalid threshold"));
00904
            goto exit_on_error;
00905
00906
        \ensuremath{//} Reading the experimental data
00907
        array = json_object_get_array_member (object, LABEL_EXPERIMENTS);
n = json_array_get_length (array);
00908
00909
00910
        input->experiment = experiment = (Experiment *)
00911
          g_malloc (n * sizeof (Experiment));
00912
        for (i = 0; i < n; ++i)
00913
00914 #if DEBUG INPUT
00915
            fprintf (stderr, "input_open_json: nexperiments=%u\n",
                     input->nexperiments);
00916
00917 #endif
00918
            child = json_array_get_element (array, i);
00919
            if (!input->nexperiments)
00920
              {
00921
                if (!experiment_open_json (experiment, child, 0))
00922
                   goto exit_on_error;
00923
00924
            else
00925
00926
                if (!experiment_open_json (experiment + input->nexperiments,
00927
                                              child, experiment->ninputs))
                   goto exit_on_error;
00929
                 if (experiment[experiment->ninputs].template_flags
00930
                     != experiment->template_flags)
00931
00932
                     input error ("bad template inputs");
00933
                     goto exit_on_error;
00934
00935
00936
            ++input->nexperiments;
00937 #if DEBUG_INPUT
            fprintf (stderr, "input_open_json: nexperiments=%u\n",
00938
00939
                      input->nexperiments);
00940 #endif
00941
00942
        if (!input->nexperiments)
00943
00944
            input_error (_("No optimization experiments"));
00945
            goto exit_on_error;
00946
00947
        input->template_flags = experiment->template_flags;
00948
00949
        // Reading the variables data
00950
        array = json_object_get_array_member (object, LABEL_VARIABLES);
00951
        n = json_array_get_length (array);
        input->variable = (Variable *) g_malloc (n * sizeof (Variable));
00952
00953
        for (i = 0; i < n; ++i)
00954
00955 #if DEBUG INPUT
00956
            fprintf (stderr, "input_open_json: nvariables=%u\n", input->nvariables);
00957 #endif
00958
            child = json_array_get_element (array, i);
            if (!variable_open_json (input->variable + input->nvariables, child,
00959
                                        input->algorithm, input->nsteps))
00960
00961
               goto exit_on_error;
00962
             ++input->nvariables;
00963
        if (!input->nvariables)
00964
00965
         {
00966
            input_error (_("No optimization variables"));
00967
            goto exit_on_error;
00968
00969
00970
        // Obtaining the error norm
00971
        if (json_object_get_member (object, LABEL_NORM))
00972
          {
00973
            buffer = json_object_get_string_member (object, LABEL_NORM);
            if (!strcmp (buffer, LABEL_EUCLIDIAN))
input->norm = ERROR_NORM_EUCLIDIAN;
else if (!strcmp (buffer, LABEL_MAXIMUM))
input->norm = ERROR_NORM_MAXIMUM;
00974
00975
00976
00977
```

```
else if (!strcmp (buffer, LABEL_P))
00979
00980
                input->norm = ERROR_NORM_P;
00981
                 input->p = jb_json_object_get_float (object, LABEL_P, &error_code);
00982
                 if (!error_code)
00983
                  {
                    input_error (_("Bad P parameter"));
00985
                     goto exit_on_error;
00986
00987
            else if (!strcmp (buffer, LABEL_TAXICAB))
00988
00989
              input->norm = ERROR_NORM_TAXICAB;
00990
            else
00991
00992
                input_error (_("Unknown error norm"));
00993
                goto exit_on_error;
00994
00995
          }
00996
        else
00997
          input->norm = ERROR_NORM_EUCLIDIAN;
00998
00999
        // Closing the JSON document
01000 g_object_unref (parser);
01001
01002 #if DEBUG_INPUT
01003 fprintf (stderr, "input_open_json: end\n");
01004 #endif
01005
       return 1;
01006
01007 exit_on_error:
01008 g_object_unref (parser);
01009 #if DEBUG_INPUT
01010
       fprintf (stderr, "input_open_json: end\n");
01011 #endif
01012
       return 0;
01013 }
01014
01020 int
01021 input_open (char *filename)
01022 {
        xmlDoc *doc;
01023
01024
       JsonParser *parser;
01025
01026 #if DEBUG_INPUT
01027 fprintf (stderr, "input_open: start\n");
01028 #endif
01029
        // Resetting input data
01030
01031
       input_new ();
01032
        // Opening input file
01034 #if DEBUG_INPUT
01035 fprintf (stderr, "input_open: opening the input file %s\n", filename);
01036 fprintf (stderr, "input_open: trying XML format\n");
01037 #endif
01038
       doc = xmlParseFile (filename);
01039
       if (!doc)
01040
01041 #if DEBUG_INPUT
01042 fprintf (stderr, "input_open: trying JSON format\n"); 01043 #endif
01044
            parser = json_parser_new ();
             if (!json_parser_load_from_file (parser, filename, NULL))
01045
01046
01047
                input_error (_("Unable to parse the input file"));
01048
                goto exit_on_error;
01049
            if (!input_open_json (parser))
01050
              goto exit_on_error;
01051
01053
        else if (!input_open_xml (doc))
01054
          goto exit_on_error;
01055
       // Getting the working directory
01056
        input->directory = g_path_get_dirname (filename);
01057
01058
       input->name = g_path_get_basename (filename);
01059
01060 #if DEBUG_INPUT
01061 fprintf (stderr, "input_open: end\n");
01062 #endif
01063
       return 1;
01064
01065 exit_on_error:
01066 jb_show_error (error_message);
01067 g_free (error_message);
01068 input_free ();
01069 #if DEBUG_INPUT
```

```
01070    fprintf (stderr, "input_open: end\n");
01071 #endif
01072    return 0;
01073 }
```

4.9 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.

4.9.1 Detailed Description

Header file to define the input functions.

Authors

Javier Burguete.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file input.h.

4.9.2 Enumeration Type Documentation

4.9.2.1 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 };
```

4.9.2.2 ErrorNorm

```
enum ErrorNorm
```

Enum to define the error norm.

Enumerator

ERROR_NORM_EUCLIDIAN	Euclidian norm: $\sqrt{\sum_{i}\left(w_{i}x_{i}\right)^{2}}$.
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.

4.9.3 Function Documentation

4.9.3.1 input_free()

```
void input_free ( )
```

Function to free the memory of the input file data.

Definition at line 86 of file input.c.

```
00087 {
00088
       unsigned int i;
00089 #if DEBUG_INPUT
       fprintf (stderr, "input_free: start\n");
00090
00091 #endif
00092
       g free (input->name);
00093
       g_free (input->directory);
00094
       for (i = 0; i < input->nexperiments; ++i)
00095
         experiment_free (input->experiment + i, input->type);
00096
       for (i = 0; i < input->nvariables; ++i)
00097
         variable_free (input->variable + i, input->type);
00098
       g_free (input->experiment);
00099
       g_free (input->variable);
00100
       if (input->type == INPUT_TYPE_XML)
00101
00102
           xmlFree (input->cleaner);
           xmlFree (input->evaluator);
00103
00104
           xmlFree (input->simulator);
00105
           xmlFree (input->result);
00106
           xmlFree (input->variables);
00107
00108
       else
00109
        {
00110
           g_free (input->cleaner);
           g_free (input->evaluator);
00111
00112
            g_free (input->simulator);
00113
            g_free (input->result);
            g_free (input->variables);
00114
00115
       input->nexperiments = input->nvariables = input->nsteps
00116
         = input->nfinal_steps = 0;
00118 #if DEBUG_INPUT
00119
       fprintf (stderr, "input_free: end\n");
00120 #endif
00121 }
```

Here is the call graph for this function:

4.9.3.2 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
       fprintf (stderr, "input_new: start\n");
00069
00070 #endif
00071 input->nvariables = input->nexperiments = input->nsteps = input->nfinal_steps
00072
00073
       input->simulator = input->evaluator = input->cleaner = input->directory
         = input->name = NULL;
00074
00075
       input->experiment = NULL;
00076
       input->variable = NULL;
00077 #if DEBUG_INPUT
00078
       fprintf (stderr, "input_new: end\n");
00079 #endif
00080 }
```

4.9.3.3 input_open()

Function to open the input file.

Returns

1 on success, 0 on error.

Parameters

filename Input data file name.

Definition at line 1021 of file input.c.

```
01022 {
01023
        xmlDoc *doc;
01024
        JsonParser *parser;
01025
01026 #if DEBUG_INPUT
       fprintf (stderr, "input_open: start\n");
01027
01028 #endif
01029
01030
        // Resetting input data
01031
        input_new ();
01032
        // Opening input file
01033
01034 #if DEBUG_INPUT
01035 fprintf (stderr, "input_open: opening the input file %s\n", filename);
01036 fprintf (stderr, "input_open: trying XML format\n");
01037 #endif
       doc = xmlParseFile (filename);
01038
01039
       if (!doc)
01040
01041 #if DEBUG_INPUT
01042
            fprintf (stderr, "input_open: trying JSON format\n");
01043 #endif
01044
            parser = json_parser_new ();
01045
             if (!json_parser_load_from_file (parser, filename, NULL))
01046
              {
                 input_error (_("Unable to parse the input file"));
01047
01048
                 goto exit_on_error;
01049
01050
            if (!input_open_json (parser))
01051
              goto exit_on_error;
01052
01053
       else if (!input_open_xml (doc))
01054
          goto exit on error;
01055
01056
        // Getting the working directory
01057
        input->directory = g_path_get_dirname (filename);
01058
       input->name = g_path_get_basename (filename);
01059
01060 #if DEBUG_INPUT
01061 fprintf (stderr, "input_open: end\n");
01062 #endif
01063 return 1;
01064
01065 exit on error:
01066 jb_show_error (error_message);
01067 g_free (error_message);
01068 input_free ();
01069 #if DEBUG_INPUT
01070
       fprintf (stderr, "input_open: end\n");
01071 #endif
01072
        return 0;
01073 }
```

Here is the call graph for this function:

4.10 input.h 117

4.9.4 Variable Documentation

4.9.4.1 input

```
Input input[1] [extern]
```

Global Input struct to set the input data.

Definition at line 57 of file input.c.

4.9.4.2 result_name

```
const char* result_name [extern]
```

Name of the result file.

Definition at line 59 of file input.c.

4.9.4.3 variables_name

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

Name of the variables file.

Definition at line 60 of file input.c.

4.10 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,
{\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 #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 {
        ERROR_NORM_EUCLIDIAN = 0,
00053
        ERROR_NORM_MAXIMUM = 1,
00055
        ERROR_NORM_P = 2,
       ERROR_NORM_TAXICAB = 3
00057
00059 };
00060
00065 typedef struct
00066 {
00067
        Experiment *experiment;
00068
        Variable *variable;
00069
        char *result;
00070
        char *variables:
00071
        char *simulator;
00072
        char *evaluator;
00074
        char *cleaner;
00075
        char *directory;
00076
        char *name;
00077
        double tolerance;
00078
        double mutation_ratio;
00079
        double reproduction_ratio;
08000
        double adaptation_ratio;
00081
        double relaxation;
00082
       double p;
double threshold;
00083
        unsigned long int seed;
00084
00086
        unsigned int nvariables;
00087
        unsigned int nexperiments;
00088
        unsigned int nsimulations;
00089
        unsigned int algorithm;
00090
        unsigned int nsteps;
00092
        unsigned int nfinal_steps;
00094
        unsigned int climbing;
00095
        unsigned int nestimates;
00097
        unsigned int niterations;
00098
        unsigned int nbest;
00099
        unsigned int norm;
00100
        unsigned int type;
00101
        unsigned int template_flags;
00102 } Input;
00103
00104 extern Input input[1];
00105 extern const char *result_name;
00106 extern const char *variables_name;
00108 // Public functions
00109 void input_new ();
00110 void input_free ();
00111 int input_open (char *filename);
00112
00113 #endif
```

4.11 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 <glib.h>
#include <qlib/qstdio.h>
#include <json-glib/json-glib.h>
#include <mpi.h>
#include <gio/gio.h>
#include <gtk/gtk.h>
#include "jb/src/xml.h"
#include "jb/src/json.h"
#include "jb/src/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 ()
- static void dialog cleaner close (GtkFileChooserDialog *dlg, int response id)
- static void dialog_cleaner ()
- 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.

4.11.1 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.

4.11.2 Macro Definition Documentation

4.11.2.1 DEBUG INTERFACE

```
#define DEBUG_INTERFACE 1
```

Macro to debug interface functions.

Definition at line 69 of file interface.c.

4.11.2.2 INPUT_FILE

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

Macro to define the initial input file.

Definition at line 78 of file interface.c.

4.11.3 Function Documentation

4.11.3.1 dialog_cleaner()

```
static void dialog_cleaner ( ) [static]
```

Function to open a dialog to save the cleaner file.

Definition at line 2340 of file interface.c.

```
02341 {
        GtkFileChooserDialog *dlg;
02342
02343 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_cleaner: start\n");
02345 #endif
02346 dlg = (GtkFileChooserDialog *)
        gtk_file_chooser_dialog_new (_("Open cleaner file"),
02347
02348
                                           window->window.
02349
                                           GTK_FILE_CHOOSER_ACTION_OPEN,
                                           _("_Cancel"), GTK_RESPONSE_CANCEL,
02350
02351 __("_Open"), GTK_RESPONSE_ACCEPT, NULL);
02352 g_signal_connect (dlg, "response", G_CALLBACK (dialog_cleaner_close), NULL);
02353 gtk_window_present (GTK_WINDOW (dlg));
02354 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_cleaner: end\n");
02355
02356 #endif
02357 }
```

Here is the call graph for this function:

4.11.3.2 dialog_cleaner_close()

Function to save the close the cleaner file dialog.

Parameters

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

Definition at line 2309 of file interface.c.

```
02312 {
         GFile *file1, *file2;
char *buffer1, *buffer2;
02313
02314
02315 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_cleaner_close: start\n");
02316
02317 #endif
02318
        if (response_id == GTK_RESPONSE_OK)
02319
02320
              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);
buffer2 = g_file_get_relative_path (file2, file1);
02321
02322
02323
02324
              input->cleaner = g_strdup (buffer2);
02325
              g_free (buffer2);
02326
              g_object_unref (file2);
              g_object_unref (file1);
02327
02328
              g_free (buffer1);
02329
        gtk_window_destroy (GTK_WINDOW (dlg));
02330
02331 #if DEBUG_INTERFACE
02332
        fprintf (stderr, "dialog_cleaner_close: end\n");
02333 #endif
02334 }
```

4.11.3.3 dialog_evaluator()

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

Function to open a dialog to save the evaluator file.

Definition at line 2286 of file interface.c.

```
02287 {
02288
        GtkFileChooserDialog *dlg;
02289 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_evaluator: start\n");
02291 #endif
02292
       dlg = (GtkFileChooserDialog *)
          gtk_file_chooser_dialog_new (_("Open evaluator file"),
02293
02294
                                          window->window.
02295
                                         GTK_FILE_CHOOSER_ACTION_OPEN,
02296
                                         _("_Cancel"), GTK_RESPONSE_CANCEL,
       __("_Open"), GTK_RESPONSE_ACCEPT, NULL);
g_signal_connect (dlg, "response", G_CALLBACK (dialog_evaluator_close), NULL);
02297
02298
        gtk_window_present (GTK_WINDOW (dlg));
02299
02300 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_evaluator: end\n");
02301
02302 #endif
02303 }
```

Here is the call graph for this function:

4.11.3.4 dialog_evaluator_close()

Function to save the close the evaluator file dialog.

Parameters

dlg	GtkFileChooserDialog dialog.
response← id	Response identifier.

Definition at line 2255 of file interface.c.

```
02258
         GFile *file1, *file2;
char *buffer1, *buffer2;
02259
02260
02261 #if DEBUG_INTERFACE
02262
        fprintf (stderr, "dialog_evaluator_close: start\n");
02263 #endif
02264
        if (response_id == GTK_RESPONSE_OK)
02265
02266
             buffer1 = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
             file1 = g_file_new_for_path (bufferl);
file2 = g_file_new_for_path (input->directory);
02267
02268
02269
             buffer2 = g_file_get_relative_path (file2, file1);
             input->evaluator = g_strdup (buffer2);
02270
02271
             g_free (buffer2);
02272
             g_object_unref (file2);
02273
             g_object_unref (file1);
02274
             g_free (buffer1);
02275
02276 gtk_window_destroy (GTK_WINDOW (dlg));
02277 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_evaluator_close: end\n");
02279 #endif
02280 }
```

4.11.3.5 dialog_name_experiment_close()

Function to close the experiment name dialog.

Parameters

dlg	GtkFileChooserDialog struct.
response← _id	Response identifier.
data	Function data.

Definition at line 1487 of file interface.c.

```
01491 {
01492
        char *buffer;
        unsigned int i;
01493
01494 #if DEBUG_INTERFACE
01495
        fprintf (stderr, "window_name_experiment_close: start\n");
01496 #endif
        i = (size_t) data;
01497
        if (response_id == GTK_RESPONSE_OK)
01498
01499
01500
             buffer = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
01501
             g_signal_handler_block (window->combo_experiment, window->id_experiment);
01502
             gtk_combo_box_text_remove (window->combo_experiment, i);
             gtk_combo_box_text_insert_text (window->combo_experiment, i, buffer);
gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), i);
01503
01504
01505
             g_signal_handler_unblock (window->combo_experiment,
01506
                                           window->id_experiment);
```

```
01507     g_free (buffer);
01508     }
01509 #if DEBUG_INTERFACE
01510 fprintf (stderr, "window_name_experiment_close: end\n");
01511 #endif
01512 }
```

4.11.3.6 dialog open close()

Function to close the input data dialog.

Parameters

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

Definition at line 2089 of file interface.c.

```
02092 {
02093
       char *buffer, *directory, *name;
02094
       GFile *file:
02095
02096 #if DEBUG_INTERFACE
02097
       fprintf (stderr, "dialog_open_close: start\n");
02098 #endif
02099
02100
       // Saving a backup of the current input file
02101
       directory = g_strdup (input->directory);
       name = g_strdup (input->name);
02102
02103
02104
       // If OK saving
02105
       if (response_id == GTK_RESPONSE_OK)
02106
02107
02108
            // Traying to open the input file
02109
            file = gtk_file_chooser_get_file (GTK_FILE_CHOOSER (dlg));
           buffer = g_file_get_path (file);
02110
02111 #if DEBUG_INTERFACE
           fprintf (stderr, "dialog_open_close: file name=%s\n", buffer);
02112
02113 #endif
           if (!window_read (buffer))
02114
02115
02116 #if DEBUG_INTERFACE
02117
              fprintf (stderr, "dialog_open_close: error reading input file\n");
02118 #endif
              g_free (buffer);
02119
02120
               // Reading backup file on error
02122
               buffer = g_build_filename (directory, name, NULL);
02123
               input->result = input->variables = NULL;
               if (!input_open (buffer))
02124
02125
02126
02127
                   // Closing on backup file reading error
02128 #if DEBUG_INTERFACE
02129
        fprintf (stderr,
                             "dialog_open_close: error reading backup file\n");
02130
02131 #endif
02132
                 }
02133
02134
           g_free (buffer);
       g_object_unref (file);
}
02135
02136
02137
02138
       // Freeing and closing
02139
      q_free (name);
02140 g_free (directory);
```

```
02141    gtk_window_destroy (GTK_WINDOW (dlg));
02142
02143 #if DEBUG_INTERFACE
02144    fprintf (stderr, "dialog_open_close: end\n");
02145 #endif
02146
02147 }
```

Here is the call graph for this function:

4.11.3.7 dialog_options_close()

Function to close the options dialog.

Parameters

dlg	GtkDialog options dialog.
response <i>⇔</i> id	Response identifier.

Definition at line 656 of file interface.c.

```
00658 {
00659 #if DEBUG_INTERFACE
         fprintf (stderr, "dialog_options_close: start\n");
00660
00661 #endif
         if (response_id == GTK_RESPONSE_OK)
00663
00664
              = (unsigned long int) gtk_spin_button_get_value (options->spin_seed);
nthreads = gtk_spin_button_get_value_as_int (options->spin_threads);
nthreads_climbing
00665
00666
00667
00668
                 = gtk_spin_button_get_value_as_int (options->spin_climbing);
00670 gtk_window_destroy (GTK_WINDOW (dlg));
00671 #if DEBUG_INTERFACE
00672 fprintf (stderr, "dialog_options_close: end\n"); 00673 #endif
00674 }
```

4.11.3.8 dialog save close()

Function to close the save dialog.

Parameters

dlg	GtkFileChooserDialog dialog.
response⇔	Response identifier.
_id	

Definition at line 872 of file interface.c.

```
00876
        GtkFileFilter *filter1;
00877
        char *buffer;
00878 #if DEBUG INTERFACE
00879
        fprintf (stderr, "dialog_save_close: start\n");
00880 #endif
00881
        // If OK response then saving
00882
        if (response_id == GTK_RESPONSE_OK)
00883
00884
            // 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"))
00885
00886
00887
00888
              input->type = INPUT_TYPE_XML;
00889
              input->type = INPUT_TYPE_JSON;
00890
00891
00892
            // Adding properties to the root XML node
            input->simulator
00893
00894
               = g_strdup (gtk_button_get_label (window->button_simulator));
00895
            if (gtk_check_button_get_active (window->check_evaluator))
00896
              input->evaluator
00897
                = g_strdup (gtk_button_get_label (window->button_evaluator));
00898
            else
00899
              input->evaluator = NULL;
00900
            if (gtk_check_button_get_active (window->check_cleaner))
00901
              input->cleaner
00902
                = g_strdup (gtk_button_get_label (window->button_cleaner));
00903
00904
              input->cleaner = NULL:
00905
            if (input->type == INPUT_TYPE_XML)
00906
                input->result
00907
00908
                   = (char *) xmlStrdup ((const xmlChar *)
00909
                                          gtk_entry_get_text (window->entry_result));
00910
                input->variables
00911
                  = (char *) xmlStrdup ((const xmlChar *)
00912
                                          gtk_entry_get_text (window->entry_variables));
00913
            else
00914
00915
              {
00916
                input->result = g_strdup (gtk_entry_get_text (window->entry_result));
00917
                input->variables
00918
                  g_strdup (gtk_entry_get_text (window->entry_variables));
00919
00920
00921
            \ensuremath{//} Setting the algorithm
00922
            switch (window_get_algorithm ())
00923
              {
00924
              case ALGORITHM_MONTE_CARLO:
00925
                input->algorithm = ALGORITHM_MONTE_CARLO;
                input->nsimulations
00926
00927
                   = gtk_spin_button_get_value_as_int (window->spin_simulations);
00928
                input->niterations
00929
                  = gtk spin button get value as int (window->spin iterations);
                input->tolerance = gtk_spin_button_get_value (window->spin_tolerance);
00930
00931
                input->nbest = gtk_spin_button_get_value_as_int (window->spin_bests);
00932
                window_save_climbing ();
              break;
case ALGORITHM SWEEP:
00933
00934
00935
                input->algorithm = ALGORITHM_SWEEP;
00936
                input->niterations
00937
                   = gtk_spin_button_get_value_as_int (window->spin_iterations);
00938
                input->tolerance = gtk_spin_button_get_value (window->spin_tolerance);
00939
                input->nbest = gtk_spin_button_get_value_as_int (window->spin_bests);
00940
                window_save_climbing ();
00941
                break:
00942
              case ALGORITHM_ORTHOGONAL:
00943
                input->algorithm = ALGORITHM_ORTHOGONAL;
00944
                 input->niterations
00945
                   = gtk_spin_button_get_value_as_int (window->spin_iterations);
                input->tolerance = gtk_spin_button_get_value (window->spin_tolerance);
00946
00947
                input->nbest = gtk_spin_button_get_value_as_int (window->spin_bests);
00948
                window_save_climbing ();
00949
                break;
00950
              default:
00951
                input->algorithm = ALGORITHM_GENETIC;
                input->nsimulations
00952
00953
                   = gtk_spin_button_get_value_as_int (window->spin_population);
00954
                input->niterations
00955
                   = gtk_spin_button_get_value_as_int (window->spin_generations);
00956
                input->mutation ratio
00957
                   gtk_spin_button_get_value (window->spin_mutation);
00958
                input->reproduction_ratio
00959
                   = gtk_spin_button_get_value (window->spin_reproduction);
00960
                input->adaptation ratio
```

```
= gtk_spin_button_get_value (window->spin_adaptation);
00962
            input->norm = window_get_norm ();
00963
            input->p = gtk_spin_button_get_value (window->spin_p);
00964
00965
            input->threshold = gtk_spin_button_get_value (window->spin_threshold);
00966
00967
            // Saving the XML file
00968
            buffer = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
00969
            input_save (buffer);
00970
00971
            // Closing and freeing memory
            g_free (buffer);
00972
00973
00974
        gtk_window_destroy (GTK_WINDOW (dlg));
00975 #if DEBUG_INTERFACE
00976 fpri
00977 #endif
       fprintf (stderr, "dialog_save_close: end\n");
00978 }
```

Here is the call graph for this function:

4.11.3.9 dialog_simulator()

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

Function to open a dialog to save the simulator file.

Definition at line 2232 of file interface.c.

```
02233
02234
         GtkFileChooserDialog *dlg;
02235 #if DEBUG_INTERFACE
02236
        fprintf (stderr, "dialog_simulator: start\n");
02237 #endif
       dlg = (GtkFileChooserDialog *)
02238
           gtk_file_chooser_dialog_new (_("Open simulator file"),
02239
02240
                                            window->window,
                                            GTK_FILE_CHOOSER_ACTION_OPEN,
02241
       _("_Cancel"), GTK_RESPONSE_CANCEL,
_("_Open"), GTK_RESPONSE_ACCEPT, NULL);
g_signal_connect (dlg, "response", G_CALLBACK (dialog_simulator_close), NULL);
02242
02243
02244
02245
         gtk_window_present (GTK_WINDOW (dlg));
02246 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_simulator: end\n");
02247
02248 #endif
02249 }
```

Here is the call graph for this function:

4.11.3.10 dialog_simulator_close()

Function to save the close the simulator file dialog.

Parameters

dlg	GtkFileChooserDialog dialog.
response⇔	Response identifier.
_id	

```
Definition at line 2201 of file interface.c.
```

```
02204 {
02205 GFile *file1, *file2;
```

```
char *buffer1, *buffer2;
02207 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_simulator_close: start\n");
02208
02209 #endif
       if (response_id == GTK_RESPONSE_OK)
02210
02211
02212
            buffer1 = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
02213
             file1 = g_file_new_for_path (buffer1);
02214
             file2 = g_file_new_for_path (input->directory);
            buffer2 = g_file_get_relative_path (file2, file1);
02215
            input->simulator = g_strdup (buffer2);
02216
             g_free (buffer2);
02217
            g_object_unref (file2);
g_object_unref (file1);
02218
02219
            g_free (buffer1);
02220
02221
        gtk_window_destroy (GTK_WINDOW (dlg));
02222
02223 #if DEBUG_INTERFACE
02224 fprintf (stderr, "dialog_simulator_close: end\n");
02225 #endif
02226 }
```

4.11.3.11 input_save()

Function to save the input file.

Parameters

filename Input file name.

Definition at line 608 of file interface.c.

```
00610
        xmlDoc *doc;
00611
        JsonGenerator *generator;
00612
00613 #if DEBUG_INTERFACE
       fprintf (stderr, "input_save: start\n");
00614
00615 #endif
00616
00617
        // Getting the input file directory
        input->name = g_path_get_basename (filename);
00618
00619
        input->directory = g_path_get_dirname (filename);
00620
00621
        if (input->type == INPUT_TYPE_XML)
00622
00623
            // Opening the input file
00624
            doc = xmlNewDoc ((const xmlChar *) "1.0");
00625
            input_save_xml (doc);
00626
00627
           // Saving the XML file
           xmlSaveFormatFile (filename, doc, 1);
00628
00629
00630
            // Freeing memory
00631
            xmlFreeDoc (doc);
00632
00633
       else
00634
         {
00635
            // Opening the input file
00636
            generator = json_generator_new ();
00637
            json_generator_set_pretty (generator, TRUE);
00638
           input_save_json (generator);
00639
00640
            // Saving the JSON file
00641
            json_generator_to_file (generator, filename, NULL);
00642
00643
            // Freeing memory
00644
            g_object_unref (generator);
          }
00645
00646
00647 #if DEBUG_INTERFACE
```

```
00648 fprintf (stderr, "input_save: end\n");
00649 #endif
00650 }
```

4.11.3.12 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.

```
00207
00208
        JsonObject *object;
00209 #if DEBUG_INTERFACE
00210
        fprintf (stderr, "input_save_climbing_json: start\n");
00211 #endif
00212
        object = json_node_get_object (node);
        if (input->nsteps)
00213
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
              case CLIMBING_METHOD_COORDINATES:
00220
                json_object_set_string_member (object, LABEL_CLIMBING,
00221
00222
                                                   LABEL_COORDINATES);
00223
                break;
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:

4.11.3.13 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
          \verb|jb_xml_node_set_uint (node, (const xmlChar *) LABEL_NSTEPS,|\\
00180
          input->nsteps);
if (input->relaxation != DEFAULT_RELAXATION)
00181
00182
           jb_xml_node_set_float (node, (const xmlChar *) LABEL_RELAXATION,
00183
                                input->relaxation);
00184
          switch (input->climbing)
00185
           case CLIMBING_METHOD_COORDINATES:
00186
            00187
00188
00189
           default:
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");
00199 #endif
00200 }
```

Here is the call graph for this function:

4.11.3.14 input save json()

Function to save the input file in JSON format.

Parameters

```
generator JsonGenerator struct.
```

Definition at line 427 of file interface.c.

```
00428 {
00429
        unsigned int i, j;
00430
        char *buffer;
00431
         JsonNode *node, *child;
00432
        JsonObject *object;
00433
        JsonArray *array;
00434
        GFile *file, *file2;
00435
00436 #if DEBUG_INTERFACE
00437
        fprintf (stderr, "input_save_json: start\n");
00438 #endif
00439
00440
        // Setting root JSON node
00441
        object = json_object_new ();
00442
        node = json_node_new (JSON_NODE_OBJECT);
00443
         json_node_set_object (node, object);
00444
        json_generator_set_root (generator, node);
00445
00446
         \ensuremath{//} Adding properties to the root JSON node
        if (strcmp (input->result, result_name))
    json_object_set_string_member (object, LABEL_RESULT_FILE, input->result);
if (strcmp (input->variables, variables_name))
00447
00448
00449
00450
           json_object_set_string_member (object, LABEL_VARIABLES_FILE,
00451
                                              input->variables);
00452
        file = g_file_new_for_path (input->directory);
        file2 = g_file_new_for_path (input->simulator);
buffer = g_file_get_relative_path (file, file2);
00453
00454
00455
        g_object_unref (file2);
00456
         json_object_set_string_member (object, LABEL_SIMULATOR, buffer);
00457
        g_free (buffer);
00458
         if (input->evaluator)
00459
00460
             file2 = q_file_new_for_path (input->evaluator);
00461
             buffer = g_file_get_relative_path (file, file2);
00462
             g_object_unref (file2);
```

```
if (strlen (buffer))
00464
                   json_object_set_string_member (object, LABEL_EVALUATOR, buffer);
                g_free (buffer);
00465
00466
00467
          if (input->cleaner)
00468
             {
                file2 = g_file_new_for_path (input->cleaner);
00470
                buffer = g_file_get_relative_path (file, file2);
00471
                g_object_unref (file2);
00472
                if (strlen (buffer))
                   json_object_set_string_member (object, LABEL_CLEANER, buffer);
00473
00474
                g_free (buffer);
00475
00476
              (input->seed != DEFAULT_RANDOM_SEED)
00477
             jb_json_object_set_uint (object, LABEL_SEED, input->seed);
00478
00479
           \ensuremath{//} Setting the algorithm
          buffer = (char *) g_slice_alloc (64);
switch (input->algorithm)
00480
00481
00482
00483
             case ALGORITHM_MONTE_CARLO:
00484
                json_object_set_string_member (object, LABEL_ALGORITHM,
                                                          LABEL_MONTE_CARLO);
00485
                snprintf (buffer, 64, "%u", input->nsimulations);
00486
00487
                json_object_set_string_member (object, LABEL_NSIMULATIONS, buffer);
                snprintf (buffer, 64, "%u", input->niterations);
00488
00489
                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);
00490
00491
00492
                json_object_set_string_member (object, LABEL_NBEST, buffer);
00493
00494
                input_save_climbing_json (node);
00495
                break;
00496
             case ALGORITHM_SWEEP:
                json_object_set_string_member (object, LABEL_ALGORITHM, LABEL_SWEEP); 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);
00497
00498
00499
00501
00502
                snprintf (buffer, 64, "%u", input->nbest);
00503
                json_object_set_string_member (object, LABEL_NBEST, buffer);
00504
                input_save_climbing_json (node);
00505
                break:
00506
             case ALGORITHM_ORTHOGONAL:
00507
                json_object_set_string_member (object, LABEL_ALGORITHM, LABEL_ORTHOGONAL);
00508
                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);
00509
00510
00511
00512
                json_object_set_string_member (object, LABEL_NBEST, buffer);
00514
                input_save_climbing_json (node);
00515
                break;
00516
             default:
                json_object_set_string_member (object, LABEL_ALGORITHM, LABEL_GENETIC);
snprintf (buffer, 64, "%u", input->nsimulations);
00517
00518
                json_object_set_string_member (object, LABEL_NPOPULATION, buffer);
00520
                snprintf (buffer, 64, "%u", input->niterations);
                snprintr (buffer, 64, "%u", input->niterations);
json_object_set_string_member (object, LABEL_NGENERATIONS, buffer);
snprintf (buffer, 64, "%.3lg", input->mutation_ratio);
json_object_set_string_member (object, LABEL_MUTATION, buffer);
snprintf (buffer, 64, "%.3lg", input->reproduction_ratio);
json_object_set_string_member (object, LABEL_REPRODUCTION, buffer);
snprintf (buffer, 64, "%.3lg", input->adaptation_ratio);
json_object_set_string_member (object, LABEL_ADAPTATION, buffer);
00521
00522
00523
00524
00525
00526
00527
00528
                break;
00529
00530
          g slice freel (64, buffer);
00531
          if (input->threshold != 0.)
00532
             jb_json_object_set_float (object, LABEL_THRESHOLD, input->threshold);
00533
00534
          // Setting the experimental data
          array = json_array_new ();
for (i = 0; i < input->nexperiments; ++i)
00535
00536
00537
00538
                child = json_node_new (JSON_NODE_OBJECT);
00539
                object = json_node_get_object (child);
00540
                json_object_set_string_member (object, LABEL_NAME,
00541
                                                           input->experiment[i].name);
                if (input->experiment[i].weight != 1.)
00542
                  jb_json_object_set_float (object, LABEL_WEIGHT,
00543
00544
                                                       input->experiment[i].weight);
00545
                for (j = 0; j < input->experiment->ninputs; ++j)
00546
                   json_object_set_string_member (object, stencil[j],
00547
                                                             input->experiment[i].stencil[j]);
00548
                json_array_add_element (array, child);
00549
```

```
json_object_set_array_member (object, LABEL_EXPERIMENTS, array);
00551
00552
        // Setting the variables data
        array = json_array_new ();
for (i = 0; i < input->nvariables; ++i)
00553
00554
00555
            child = json_node_new (JSON_NODE_OBJECT);
object = json_node_get_object (child);
00556
00557
00558
            json_object_set_string_member (object, LABEL_NAME,
00559
                                            input->variable[i].name);
            00560
00561
00562
            if (input->variable[i].rangeminabs != -G_MAXDOUBLE)
00563
              jb_json_object_set_float (object, LABEL_ABSOLUTE_MINIMUM,
00564
                                         input->variable[i].rangeminabs);
00565
            jb_json_object_set_float (object, LABEL_MAXIMUM,
                                       input->variable[i].rangemax);
00566
            if (input->variable[i].rangemaxabs != G_MAXDOUBLE)
00567
              jb_json_object_set_float (object, LABEL_ABSOLUTE_MAXIMUM,
00568
00569
                                         input->variable[i].rangemaxabs);
00570
            if (input->variable[i].precision != DEFAULT_PRECISION)
00571
              jb_json_object_set_uint (object, LABEL_PRECISION,
00572
                                        input->variable[i].precision);
            if (input->algorithm == ALGORITHM_SWEEP
00573
00574
                || input->algorithm == ALGORITHM_ORTHOGONAL)
00575
              jb_json_object_set_uint (object, LABEL_NSWEEPS,
                                        input->variable[i].nsweeps);
00576
00577
            else if (input->algorithm == ALGORITHM_GENETIC)
            jb_json_object_set_uint (object, LABEL_NBITS, input->variable[i].nbits);
if (input->nsteps)
00578
00579
              (input->nsteps)
00580
              jb_json_object_set_float (object, LABEL_STEP, input->variable[i].step);
00581
            json_array_add_element (array, child);
00582
00583
        json_object_set_array_member (object, LABEL_VARIABLES, array);
00584
00585
        // Saving the error norm
00586
        switch (input->norm)
         {
00588
          case ERROR NORM MAXIMUM:
00589
            json_object_set_string_member (object, LABEL_NORM, LABEL_MAXIMUM);
00590
            break;
00591
          case ERROR NORM P:
00592
            json_object_set_string_member (object, LABEL_NORM, LABEL_P);
00593
            jb_json_object_set_float (object, LABEL_P, input->p);
00594
            break;
00595
          case ERROR_NORM_TAXICAB:
00596
            json_object_set_string_member (object, LABEL_NORM, LABEL_TAXICAB);
00597
00598
00599 #if DEBUG_INTERFACE
       fprintf (stderr, "input_save_json: end\n");
00601 #endif
00602 }
```

Here is the call graph for this function:

4.11.3.15 input_save_xml()

Function to save the input file in XML format.

Parameters

```
doc xmlDoc struct.
```

Definition at line 238 of file interface.c.

```
00239 {
00240 unsigned int i, j;
00241 char *buffer;
00242 xmlNode *node, *child;
00243 GFile *file, *file2;
00244
00245 #if DEBUG_INTERFACE
```

```
fprintf (stderr, "input_save_xml: start\n");
00247 #endif
00248
00249
        // Setting root XML node
00250
        node = xmlNewDocNode (doc, 0, (const xmlChar *) LABEL_OPTIMIZE, 0);
00251
        xmlDocSetRootElement (doc, node);
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
00256
00257
                       (xmlChar *) input->result);
00258
        if (xmlStrcmp
00259
             ((const xmlChar *) input->variables, (const xmlChar *) variables_name))
00260
          xmlSetProp (node, (const xmlChar *) LABEL_VARIABLES_FILE,
00261
                       (xmlChar *) input->variables);
        file = g_file_new_for_path (input->directory);
00262
        file2 = g_file_new_for_path (input->simulator);
buffer = g_file_get_relative_path (file, file2);
00263
00264
00265
        g_object_unref (file2);
00266
        xmlSetProp (node, (const xmlChar *) LABEL_SIMULATOR, (xmlChar *) buffer);
00267
        g_free (buffer);
        if (input->evaluator)
00268
00269
00270
            file2 = q_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
00275
00276
            g_free (buffer);
00277
00278
        if (input->cleaner)
00279
            file2 = g_file_new_for_path (input->cleaner);
buffer = g_file_get_relative_path (file, file2);
00280
00281
            g_object_unref (file2);
00282
            if (xmlStrlen ((xmlChar *) buffer))
00284
              xmlSetProp (node, (const xmlChar *) LABEL_CLEANER, (xmlChar *) buffer);
00285
            g_free (buffer);
00286
        if (input->seed != DEFAULT_RANDOM_SEED)
00287
          jb_xml_node_set_uint (node, (const xmlChar *) LABEL_SEED, input->seed);
00288
00289
00290
        // Setting the algorithm
00291
        buffer = (char *) g_slice_alloc (64);
00292
        switch (input->algorithm)
00293
00294
          case ALGORITHM MONTE CARLO:
            00295
00297
00298
            xmlSetProp (node, (const xmlChar *) LABEL_NSIMULATIONS,
            (xmlChar *) buffer);
snprintf (buffer, 64, "%u", input->niterations);
00299
00300
            xmlSetProp (node, (const xmlChar *) LABEL_NITERATIONS,
00301
                         (xmlChar *) buffer);
00303
            snprintf (buffer, 64, "%.31g", input->tolerance);
            xmlSetProp (node, (const xmlChar *) LABEL_TOLERANCE, (xmlChar *) buffer);
snprintf (buffer, 64, "%u", input->nbest);
00304
00305
            xmlSetProp (node, (const xmlChar *) LABEL_NBEST, (xmlChar *) buffer);
00306
00307
            input_save_climbing_xml (node);
00308
            break;
00309
          case ALGORITHM_SWEEP:
00310
            xmlSetProp (node, (const xmlChar *) LABEL_ALGORITHM,
00311
                         (const xmlChar *) LABEL_SWEEP);
            snprintf (buffer, 64, "%u", input->niterations);
00312
            xmlSetProp (node, (const xmlChar *) LABEL_NITERATIONS,
00313
            (xmlChar *) buffer);
snprintf (buffer, 64, "%.31g", input->tolerance);
00314
00316
            xmlSetProp (node, (const xmlChar *) LABEL_TOLERANCE, (xmlChar *) buffer);
00317
            snprintf (buffer, 64, "%u", input->nbest);
            xmlSetProp (node, (const xmlChar *) LABEL_NBEST, (xmlChar *) buffer);
00318
00319
            input_save_climbing_xml (node);
00320
            break;
          case ALGORITHM_ORTHOGONAL:
00321
00322
            xmlSetProp (node, (const xmlChar *) LABEL_ALGORITHM,
            (const xmlChar *) LABEL_ORTHOGONAL);
snprintf (buffer, 64, "%u", input->niterations);
00323
00324
            xmlSetProp (node, (const xmlChar *) LABEL_NITERATIONS,
00325
            (xmlChar *) buffer);
snprintf (buffer, 64, "%.31g", input->tolerance);
00326
            xmlSetProp (node, (const xmlChar *) LABEL_TOLERANCE, (xmlChar *) buffer);
00328
00329
            snprintf (buffer, 64, "%u", input->nbest);
            xmlSetProp (node, (const xmlChar *) LABEL_NBEST, (xmlChar *) buffer);
00330
00331
            input_save_climbing_xml (node);
00332
            break:
```

```
default:
            xmlSetProp (node, (const xmlChar *) LABEL_ALGORITHM,
00334
             (const xmlChar *) LABEL_GENETIC);
snprintf (buffer, 64, "%u", input->nsimulations);
00335
00336
             xmlSetProp (node, (const xmlChar *) LABEL_NPOPULATION,
00337
             (xmlchar *) buffer);
snprintf (buffer, 64, "%u", input->niterations);
00338
             xmlSetProp (node, (const xmlChar *) LABEL_NGENERATIONS,
00340
             (xmlChar *) buffer);
snprintf (buffer, 64, "%.31g", input->mutation_ratio);
xmlSetProp (node, (const xmlChar *) LABEL_MUTATION, (xmlChar *) buffer);
00341
00342
00343
             snprintf (buffer, 64, "%.31g", input->reproduction_ratio);
00344
00345
             xmlSetProp (node, (const xmlChar *) LABEL_REPRODUCTION,
00346
                          (xmlChar *) buffer);
00347
             snprintf (buffer, 64, "%.31g", input->adaptation_ratio);
00348
             xmlSetProp (node, (const xmlChar *) LABEL_ADAPTATION, (xmlChar *) buffer);
00349
            break:
00350
00351
        g_slice_free1 (64, buffer);
00352
           (input->threshold != 0.)
00353
           jb_xml_node_set_float (node, (const xmlChar *) LABEL_THRESHOLD,
00354
                                    input->threshold);
00355
00356
        // Setting the experimental data
00357
        for (i = 0; i < input->nexperiments; ++i)
00358
00359
             child = xmlNewChild (node, 0, (const xmlChar *) LABEL_EXPERIMENT, 0);
00360
             xmlSetProp (child, (const xmlChar *) LABEL_NAME,
             (xmlChar *) input->experiment[i].name);
if (input->experiment[i].weight != 1.)
  jb_xml_node_set_float (child, (const xmlChar *) LABEL_WEIGHT,
00361
00362
00363
00364
                                        input->experiment[i].weight);
00365
             for (j = 0; j < input->experiment->ninputs; ++j)
00366
              xmlSetProp (child, (const xmlChar *) stencil[j],
00367
                            (xmlChar *) input->experiment[i].stencil[j]);
00368
00369
00370
        // Setting the variables data
        for (i = 0; i < input->nvariables; ++i)
00371
00372
             \label{eq:child} child = xmlNewChild (node, 0, (const xmlChar *) LABEL_VARIABLE, 0); \\ xmlSetProp (child, (const xmlChar *) LABEL_NAME, \\ \\
00373
00374
00375
                          (xmlChar *) input->variable[il.name):
00376
             jb_xml_node_set_float (child, (const xmlChar *) LABEL_MINIMUM,
                                      input->variable[i].rangemin);
00377
00378
             if (input->variable[i].rangeminabs != -G_MAXDOUBLE)
00379
              jb_xml_node_set_float (child, (const xmlChar *) LABEL_ABSOLUTE_MINIMUM,
00380
                                        input->variable[i].rangeminabs);
00381
             jb_xml_node_set_float (child, (const xmlChar *) LABEL_MAXIMUM,
                                      input->variable[i].rangemax);
00382
             if (input->variable[i].rangemaxabs != G_MAXDOUBLE)
00384
               jb_xml_node_set_float (child, (const xmlChar *) LABEL_ABSOLUTE_MAXIMUM,
00385
                                        input->variable[i].rangemaxabs);
             if (input->variable[i].precision != DEFAULT_PRECISION)
00386
               jb_xml_node_set_uint (child, (const xmlChar *) LABEL_PRECISION,
00387
00388
                                       input->variable[i].precision);
             if (input->algorithm == ALGORITHM_SWEEP
00390
                 || input->algorithm == ALGORITHM_ORTHOGONAL)
00391
               jb_xml_node_set_uint (child, (const xmlChar *) LABEL_NSWEEPS,
            input->variable[i].nsweeps);
else if (input->algorithm == ALGORITHM_GENETIC)
00392
00393
               jb_xml_node_set_uint (child, (const xmlChar *) LABEL_NBITS,
00394
00395
                                       input->variable[i].nbits);
00396
             if (input->nsteps)
00397
               jb_xml_node_set_float (child, (const xmlChar *) LABEL_STEP,
00398
                                        input->variable[i].step);
00399
00400
00401
        // Saving the error norm
00402
        switch (input->norm)
00403
00404
           case ERROR NORM MAXIMUM:
00405
            xmlSetProp (node, (const xmlChar *) LABEL_NORM,
00406
                          (const xmlChar *) LABEL_MAXIMUM);
00407
            break;
          case ERROR_NORM_P:
00408
00409
            xmlSetProp (node, (const xmlChar *) LABEL_NORM,
00410
                          (const xmlChar *) LABEL_P);
00411
             jb_xml_node_set_float (node, (const xmlChar *) LABEL_P, input->p);
00412
            break:
          case ERROR_NORM_TAXICAB:
00413
            00414
00415
00416
          }
00417
00418 #if DEBUG_INTERFACE
        fprintf (stderr, "input_save: end\n");
```

```
00420 #endif
00421 }
```

4.11.3.16 options_new()

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

Function to open the options dialog.

Definition at line 680 of file interface.c.

```
00682 #if DEBUG_INTERFACE
       fprintf (stderr, "options_new: start\n");
00683
00684 #endif
00685
       options->label seed = (GtkLabel *)
00686
          gtk_label_new (_("Pseudo-random numbers generator seed"));
00687
        options->spin_seed = (GtkSpinButton *)
00688
          gtk_spin_button_new_with_range (0., (gdouble) G_MAXULONG, 1.);
00689
        {\tt gtk\_widget\_set\_tooltip\_text}
          (GTK_WIDGET (options->spin_seed),
00690
00691
            _("Seed to init the pseudo-random numbers generator"));
        gtk_spin_button_set_value (options->spin_seed, (gdouble) input->seed);
00692
00693
        options->label_threads = (GtkLabel *)
00694
          gtk_label_new (_("Threads number for the stochastic algorithm"));
        options->spin_threads
00695
00696
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 64., 1.);
00697
        gtk_widget_set_tooltip_text
  (GTK_WIDGET (options->spin_threads),
00698
           _("Number of threads to perform the calibration/optimization for "
00699
00700
             "the stochastic algorithm"));
00701
        gtk_spin_button_set_value (options->spin_threads, (gdouble) nthreads);
00702
        options->label_climbing = (GtkLabel *)
00703
         gtk_label_new (_("Threads number for the hill climbing method"));
        options->spin_climbing =
00704
00705
          (GtkSpinButton *) gtk_spin_button_new_with_range (1., 64., 1.);
00706
        gtk_widget_set_tooltip_text
00707
          (GTK_WIDGET (options->spin_climbing),
          _("Number of threads to perform the calibration/optimization for the " "hill climbing method"));
00708
00709
00710
       gtk_spin_button_set_value (options->spin_climbing,
00711
                                   (gdouble) nthreads_climbing);
00712
        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);
00713
00714
00715
        gtk_grid_attach (options->grid, GTK_WIDGET (options->label_threads),
00716
                         0, 1, 1, 1);
00717
        gtk_grid_attach (options->grid, GTK_WIDGET (options->spin_threads),
00718
                         1, 1, 1, 1);
00719
        gtk_grid_attach (options->grid, GTK_WIDGET (options->label_climbing), 0, 2, 1,
00720
                         1);
00721
        gtk_grid_attach (options->grid, GTK_WIDGET (options->spin_climbing), 1, 2, 1,
00722
                         1);
00723 #if !GTK4
00724
        gtk_widget_show_all (GTK_WIDGET (options->grid));
00725 #else
00726
        gtk_widget_show (GTK_WIDGET (options->grid));
00727 #endif
        options->dialog = (GtkDialog *)
00728
00729
          gtk_dialog_new_with_buttons (_("Options"),
00730
                                        window->window
00731
                                        GTK_DIALOG_MODAL,
                                       00732
00733
       00734
00735
00737
00738
        gtk_window_present (GTK_WINDOW (options->dialog));
00739 #if DEBUG_INTERFACE
       fprintf (stderr, "options_new: end\n");
00740
00741 #endif
00742 }
```

4.11.3.17 running_new()

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

Function to open the running dialog.

```
Definition at line 748 of file interface.c.
```

```
00749
00750 #if DEBUG_INTERFACE
00751 fprintf (stderr, "running_new: start\n");
00752 #endif
       running->label = (GtkLabel *) gtk_label_new (_("Calculating ..."));
00753
00754
        running->spinner = (GtkSpinner *) gtk_spinner_new ();
00755
        running->grid = (GtkGrid *) gtk_grid_new ();
00756
        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);
00757
00758
       running->dialog = (GtkDialog *)
00759
         gtk_dialog_new_with_buttons (_("Calculating"),
00760
                                        window->window, GTK_DIALOG_MODAL, NULL, NULL);
00761
       gtk_window_set_child (GTK_WINDOW
00762
                               (gtk_dialog_get_content_area (running->dialog)),
00763
                              GTK_WIDGET (running->grid));
00764
       gtk_spinner_start (running->spinner);
00765 #if !GTK4
       gtk_widget_show_all (GTK_WIDGET (running->dialog));
00767 #else
00768
       gtk_widget_show (GTK_WIDGET (running->dialog));
00769 #endif
00770 #if DEBUG_INTERFACE
00771
       fprintf (stderr, "running_new: end\n");
00772 #endif
00773 }
```

4.11.3.18 window about()

```
static void window about ( ) [static]
```

Function to show an about dialog.

Definition at line 1120 of file interface.c.

```
01121 {
01122
         static const gchar *authors[] = {
           "Javier Burguete Tolosa <jburguete@eead.csic.es>",
"Borja Latorre Garcés <borja.latorre@csic.es>",
01123
01124
01125
           NULL
01126
01127 #if DEBUG_INTERFACE
01128
        fprintf (stderr, "window_about: start\n");
01129 #endif
01130
        gtk_show_about_dialog
           (window->window,
01131
             "program_name", "MPCOTool",
01133
            "comments",
01134
            _("The Multi-Purposes Calibration and Optimization Tool. \normalfont{\sc n} "
             "A software to perform calibrations or optimizations of empirical " "parameters"),
01135
01136
01137
            "authors", authors,
            "translator-credits",
01138
            "Javier Burguete Tolosa <jburguete@eead.csic.es> "
01140
            "(english, french and spanish)\n"
            "Uğur Çayoğlu (german)",
"version", "4.12.0",
"copyright", "Copyright 2012-2023 Javier Burguete Tolosa",
01141
01142
01143
            "logo", window->logo,
01144
01145
            "website", "https://github.com/jburguete/mpcotool",
01146
            "license-type", GTK_LICENSE_BSD, NULL);
01147 #if DEBUG_INTERFACE
01148 fprintf (stderr, "window_about: end\n");
01149 #endif
01150 }
```

4.11.3.19 window_add_experiment()

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

Function to add an experiment in the main window.

Definition at line 1438 of file interface.c.

```
unsigned int i, j;
01440
01441 #if DEBUG_INTERFACE
        fprintf (stderr, "window_add_experiment: start\n");
01442
01443 #endif
01444 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01445
         g_signal_handler_block (window->combo_experiment, window->id_experiment);
01446
         {\tt gtk\_combo\_box\_text\_insert\_text}
01447
            (window->combo_experiment, i, input->experiment[i].name);
         g_signal_handler_unblock (window->combo_experiment, window->id_experiment);
input->experiment = (Experiment *) g_realloc
01448
01449
01450
           (input->experiment, (input->nexperiments + 1) * sizeof (Experiment));
         for (j = input->nexperiments - 1; j > i; --j)
  memcpy (input->experiment + j + 1, input->experiment + j,
01451
01452
         sizeof (Experiment));
input->experiment[j + 1].weight = input->experiment[j].weight;
input->experiment[j + 1].ninputs = input->experiment[j].ninputs;
if (input->type == INPUT_TYPE_XML)
01453
01454
01455
01456
           {
01458
              input->experiment[j + 1].name
01459
                = (char *) xmlStrdup ((xmlChar *) input->experiment[j].name);
              for (j = 0; j < input->experiment->ninputs; ++j)
  input->experiment[i + 1].stencil[j]
01460
01461
01462
                  = (char *) xmlStrdup ((xmlChar *) input->experiment[i].stencil[j]);
01463
01464
         else
01465
01466
              input->experiment[j + 1].name = g_strdup (input->experiment[j].name);
              findut=vexperiment(j + 1].name = g_strump (input=
for (j = 0; j < input=vexperiment=vninputs; ++j)
  input=vexperiment[i + 1].stencil[j]
01467
01468
                   = g_strdup (input->experiment[i].stencil[j]);
01470
01471
         ++input->nexperiments;
01472
        for (j = 0; j < input->experiment->ninputs; ++j)
           g_signal_handler_block (window->button_template[j], window->id_input[j]);
01473
        01474
           g_signal_handler_unblock (window->button_template[j], window->id_input[j]);
01477
        window_update ();
01478 #if DEBUG_INTERFACE
        fprintf (stderr, "window_add_experiment: end\n");
01479
01480 #endif
01481 }
```

Here is the call graph for this function:

4.11.3.20 window_add_variable()

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

Function to add a variable in the main window.

Definition at line 1775 of file interface.c.

```
01776 +
01777
          unsigned int i, j;
01778 #if DEBUG_INTERFACE
01779
          fprintf (stderr, "window_add_variable: start\n");
01780 #endif
01781
          i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01782
          g_signal_handler_block (window->combo_variable, window->id_variable);
          gtk_combo_box_text_insert_text (window->combo_variable, i,
01783
                                                      input->variable[i].name);
01784
          g_signal_handler_unblock (window->combo_variable, window->id_variable);
01785
01786
          input->variable = (Variable *) g_realloc
         (input->variable - (variable *, y_tealloc
  (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));
01787
01788
01789
01790
          if (input->type == INPUT_TYPE_XML)
```

```
input->variable[j + 1].name
01793
           = (char *) xmlStrdup ((xmlChar *) input->variable[j].name);
01794
01795
         input->variable[j + 1].name = g_strdup (input->variable[j].name);
01796
       ++input->nvariables;
01797
       g_signal_handler_block (window->entry_variable, window->id_variable_label);
01798
       gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), i + 1);
01799
       g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
01800
       window_update ();
01801 #if DEBUG_INTERFACE
       fprintf (stderr, "window_add_variable: end\n");
01802
01803 #endif
01804 }
```

Here is the call graph for this function:

4.11.3.21 window_get_algorithm()

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

Function to get the stochastic algorithm number.

Returns

Stochastic algorithm number.

Definition at line 781 of file interface.c.

```
00782 {
00783    unsigned int i;
00784    #if DEBUG_INTERFACE
00785    fprintf (stderr, "window_get_algorithm: start\n");
00786    #endif
00787    i = jbw_array_buttons_get_active (window->button_algorithm, NALGORITHMS);
00788    #if DEBUG_INTERFACE
00789    fprintf (stderr, "window_get_algorithm: %u\n", i);
00790    fprintf (stderr, "window_get_algorithm: end\n");
00791    #endif
00792    return i;
00793 }
```

Here is the call graph for this function:

4.11.3.22 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 801 of file interface.c.

```
00802 {
00803    unsigned int i;
00804    #if DEBUG_INTERFACE
00805    fprintf (stderr, "window_get_climbing: start\n");
00806    #endif
00807    i = jbw_array_buttons_get_active (window->button_climbing, NCLIMBINGS);
00808    #if DEBUG_INTERFACE
00809    fprintf (stderr, "window_get_climbing: %u\n", i);
00810    fprintf (stderr, "window_get_climbing: end\n");
00811    #endif
00812    return i;
00813 }
```

Here is the call graph for this function:

4.11.3.23 window_get_norm()

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

Function to get the norm method number.

Returns

Norm method number.

Definition at line 821 of file interface.c.

```
00822 {
00823    unsigned int i;
00824 #if DEBUG_INTERFACE
00825    fprintf (stderr, "window_get_norm: start\n");
00826 #endif
00827    i = jbw_array_buttons_get_active (window->button_norm, NNORMS);
00828 #if DEBUG_INTERFACE
00829    fprintf (stderr, "window_get_norm: %u\n", i);
fprintf (stderr, "window_get_norm: end\n");
00830    fprintf (stderr, "window_get_norm: end\n");
00831 #endif
00832    return i;
00833 }
```

Here is the call graph for this function:

4.11.3.24 window_help()

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

Function to show a help dialog.

Definition at line 1092 of file interface.c.

```
01093 {
01094 char *buffer, *buffer2;
01095 #if DEBUG_INTERFACE
         fprintf (stderr, "window_help: start\n");
01096
01097 #endif
01098
        buffer2 = g_build_filename (window->application_directory, "..", "manuals",
                                            _("user-manual.pdf"), NULL);
01099
01100 buffer = g_filename_to_uri (buffer2, NULL, NULL);
01101 g_free (buffer2);
01102 #if GTK4
         gtk_show_uri (window->window, buffer, GDK_CURRENT_TIME);
01104 #else
01105
        gtk_show_uri_on_window (window->window, buffer, GDK_CURRENT_TIME, NULL);
gck_snow_uri_on_wi
01106 #endif
01107 #if DEBUG_INTERFACE
01108 fprintf (stderr.'
        fprintf (stderr, "window_help: uri=%s\n", buffer);
01109 #endif
01110
         g_free (buffer);
01111 #if DEBUG_INTERFACE
01112 fprintf (stderr, "window_help: end\n");
01113 #endif
01114 }
```

4.11.3.25 window_inputs_experiment()

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

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

Definition at line 1571 of file interface.c.

```
01572 {
01573
       unsigned int j;
01574 #if DEBUG_INTERFACE
01575
       fprintf (stderr, "window_inputs_experiment: start\n");
01576 #endif
01577
       j = input->experiment->ninputs - 1;
01578
       if (j && !gtk_check_button_get_active (window->check_template[j]))
01579
          --input->experiment->ninputs;
       if (input->experiment->ninputs < MAX_NINPUTS
01580
01581
            && gtk_check_button_get_active (window->check_template[j]))
01582
          ++input->experiment->ninputs;
01583
       window_update ();
01584 #if DEBUG_INTERFACE
01585 fprintf (stderr, "window_inputs_experiment: end\n");
01586 #endif
```

Here is the call graph for this function:

4.11.3.26 window_label_variable()

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

Function to set the variable label in the main window.

Definition at line 1810 of file interface.c.

```
01811 {
01812
         unsigned int i:
01813
         const char *buffer;
01814 #if DEBUG_INTERFACE
01815
         fprintf (stderr, "window_label_variable: start\n");
01816 #endif
        i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
buffer = gtk_entry_get_text (window->entry_variable);
g_signal_handler_block (window->combo_variable, window->id_variable);
01817
01818
         gtk_combo_box_text_remove (window->combo_variable, i);
01821
         gtk_combo_box_text_insert_text (window->combo_variable, i, buffer);
01822
         gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), i);
         g_signal_handler_unblock (window->combo_variable, window->id_variable);
01823
01824 #if DEBUG INTERFACE
        fprintf (stderr, "window_label_variable: end\n");
01825
01826 #endif
01827 }
```

Here is the call graph for this function:

4.11.3.27 window_name_experiment()

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

Function to set the experiment name in the main window.

Definition at line 1518 of file interface.c.

```
01519 {
01520   GtkFileChooserDialog *dlg;
01521   GMainLoop *loop;
01522   const char *buffer;
01523   unsigned int i;
01524   #if DEBUG_INTERFACE
01525   fprintf (stderr, "window_name_experiment: start\n");
01526   #endif
01527   i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
```

```
buffer = gtk_button_get_label (window->button_experiment);
01529
       dlg = (GtkFileChooserDialog *)
01530
         gtk_file_chooser_dialog_new (_("Open experiment file"),
01531
                                   window->window,
01532
                                   GTK_FILE_CHOOSER_ACTION_OPEN,
                                    _("_Cancel"),
01533
                                   GTK_RESPONSE_CANCEL,
01534
01535
                                   _("_Open"), GTK_RESPONSE_OK, NULL);
01536
       gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (dlg), buffer);
       01537
01538
      gtk_window_present (GTK_WINDOW (dlg));
01539
01540
       loop = g_main_loop_new (NULL, 0);
01541
      g_signal_connect_swapped (dlg, "destroy", G_CALLBACK (g_main_loop_quit),
01542
                               loop);
01543
      g_main_loop_run (loop);
01544
       g_main_loop_unref (loop);
01545 #if DEBUG_INTERFACE
      fprintf (stderr, "window_name_experiment: end\n");
01546
01547 #endif
01548 }
```

4.11.3.28 window new()

Function to open the main window.

Parameters

```
application GtkApplication struct.
```

Definition at line 2363 of file interface.c.

```
unsigned int i;
02365
02366
        char *buffer, *buffer2, buffer3[64];
        const char *label_algorithm[NALGORITHMS] = {
   "_Monte-Carlo", _("_Sweep"), _("_Genetic"), _("_Orthogonal")
02367
02368
02369
02370
        const char *tip_algorithm[NALGORITHMS] = {
         _("Monte-Carlo brute force algorithm"),
02371
02372
          _("Sweep brute force algorithm"),
02373
          _("Genetic algorithm"),
02374
          _("Orthogonal sampling brute force algorithm"),
02375
        const char *label_climbing[NCLIMBINGS] = {
02377
          _("_Coordinates climbing"), _("_Random climbing")
02378
        const char *tip_climbing[NCLIMBINGS] = {
    ("Coordinates climbing estimate method"),
02379
02380
           _("Random climbing estimate method")
02381
02382
02383
        const char *label_norm[NNORMS] = { "L2", "L", "Lp", "L1" };
02384
        const char *tip_norm[NNORMS] =
         _("Euclidean error norm (L2)"),
02385
          __("Maximum error norm (L)"),
_("P error norm (Lp)"),
02386
02387
          _("Taxicab error norm (L1)")
02388
02389
02390 #if !GTK4
02391
        const char *close = "delete-event";
02392 #else
        const char *close = "close-request";
02393
02394 #endif
02395
02396 #if DEBUG_INTERFACE
02397
        fprintf (stderr, "window_new: start\n");
02398 #endif
02399
02400
        // Creating the window
02401
        window->window = window_parent = main_window
          = (GtkWindow *) gtk_application_window_new (application);
```

```
02403
        // Finish when closing the window
02404
02405
        g_signal_connect_swapped (window->window, close,
02406
                                    G_CALLBACK (g_application_quit),
02407
                                    G_APPLICATION (application));
02408
02409
        // Setting the window title
02410
        gtk_window_set_title (window->window, "MPCOTool");
02411
02412
        // Creating the open button
02413
        window->button\_open = (GtkButton *)
02414 #if !GTK4
02415
          gtk_button_new_from_icon_name ("document-open", GTK_ICON_SIZE_BUTTON);
02416 #else
02417
          gtk_button_new_from_icon_name ("document-open");
02418 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_open),
02419
        __("Open a case"));
g_signal_connect (window->button_open, "clicked", window_open, NULL);
02420
02421
02422
02423
        // Creating the save button
02424
        window->button_save = (GtkButton *)
02425 #if !GTK4
          gtk_button_new_from_icon_name ("document-save", GTK_ICON_SIZE_BUTTON);
02426
02427 #else
02428
          gtk_button_new_from_icon_name ("document-save");
02429 #endif
02430
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_save),
        _("Save the case"));
g_signal_connect (window->button_save, "clicked", (GCallback) window_save,
02431
02432
02433
                           NULL);
02434
02435
        \ensuremath{//} Creating the run button
02436
        window->button_run = (GtkButton *)
02437 #if !GTK4
          gtk_button_new_from_icon_name ("system-run", GTK_ICON_SIZE_BUTTON);
02438
02439 #else
02440
         gtk_button_new_from_icon_name ("system-run");
02441 #endif
02442
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_run),
        _("Run the optimization"));
g_signal_connect (window->button_run, "clicked", window_run, NULL);
02443
02444
02445
02446
        // Creating the options button
        window->button_options = (GtkButton *)
02447
02448 #if !GTK4
02449
          gtk_button_new_from_icon_name ("preferences-system", GTK_ICON_SIZE_BUTTON);
02450 #else
02451
          gtk_button_new_from_icon_name ("preferences-system");
02452 #endif
02453
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_options),
02454
                                       _("Edit the case"));
02455
        g_signal_connect (window->button_options, "clicked", options_new, NULL);
02456
        // Creating the help button
02457
        window->button\_help = (GtkButton *)
02458
02459 #if !GTK4
02460
          gtk_button_new_from_icon_name ("help-browser", GTK_ICON_SIZE_BUTTON);
02461 #else
02462
          gtk_button_new_from_icon_name ("help-browser");
02463 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_help), _("Help"));
g_signal_connect (window->button_help, "clicked", window_help, NULL);
02464
02465
02466
02467
        // Creating the about button
02468
        window->button_about = (GtkButton *)
02469 #if !GTK4
          gtk_button_new_from_icon_name ("help-about", GTK_ICON_SIZE_BUTTON);
02470
02471 #else
02472
          gtk_button_new_from_icon_name ("help-about");
02473 #endif
02474
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_about), _("About"));
02475
        g_signal_connect (window->button_about, "clicked", window_about, NULL);
02476
02477
        // Creating the exit button
        window->button_exit = (GtkButton *)
02478
02479 #if !GTK4
02480
          gtk_button_new_from_icon_name ("application-exit", GTK_ICON_SIZE_BUTTON);
02481 #else
02482
          gtk button new from icon name ("application-exit");
02483 #endif
02484
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_exit), _("Exit"));
02485
        g_signal_connect_swapped (window->button_exit, "clicked"
02486
                                    G_CALLBACK (g_application_quit),
02487
                                    G_APPLICATION (application));
02488
02489
        // Creating the buttons bar
```

```
window->box_buttons = (GtkBox *) gtk_box_new (GTK_ORIENTATION_HORIZONTAL, 0);
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_open));
02491
02492
         gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_save));
02493
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_run));
02494
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_options));
gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_help));
02495
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_about));
02496
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_exit));
02497
02498
02499
        \ensuremath{//} Creating the simulator program label and entry
        window->label_simulator = (GtkLabel *) gtk_label_new (_("Simulator program"));
window->button_simulator = (GtkButton *)
02500
02501
02502
           gtk_button_new_with_mnemonic (_("Simulator program"));
02503
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_simulator),
02504
                                         _("Simulator program executable file"));
        gtk_widget_set_hexpand (GTK_WIDGET (window->button_simulator), TRUE);
g_signal_connect (window->button_simulator, "clicked",
02505
02506
                            G_CALLBACK (dialog_simulator), NULL);
02507
02508
        // Creating the evaluator program label and entry
window->check_evaluator = (GtkCheckButton *)
02509
02510
02511
          gtk_check_button_new_with_mnemonic (_("_Evaluator program"));
         g_signal_connect (window->check_evaluator, "toggled", window_update, NULL);
02512
        window->button evaluator = (GtkButton *)
02513
02514
          gtk_button_new_with_mnemonic (_("Evaluator program"));
02515
        gtk_widget_set_tooltip_text
02516
           (GTK_WIDGET (window->button_evaluator),
02517
            _("Optional evaluator program executable file"));
        g_signal_connect (window->button_evaluator, "clicked"
02518
                            G_CALLBACK (dialog_evaluator), NULL);
02519
02520
02521
         // Creating the cleaner program label and entry
02522
        window->check_cleaner = (GtkCheckButton *)
        gtk_check_button_new_with_mnemonic (_("_Cleaner program"));
g_signal_connect (window->check_cleaner, "toggled", window_update, NULL);
window->button_cleaner = (GtkButton *)
02523
02524
02525
          gtk_button_new_with_mnemonic (_("Cleaner program"));
02526
02527
        gtk_widget_set_tooltip_text
02528
           (GTK_WIDGET (window->button_cleaner),
02529
            _("Optional cleaner program executable file"));
02530
        g_signal_connect (window->button_cleaner, "clicked",
02531
                            G_CALLBACK (dialog_cleaner), NULL);
02532
02533
        // Creating the results files labels and entries
02534
        window->label_result = (GtkLabel *) gtk_label_new (_("Result file"));
02535
         window->entry_result = (GtkEntry *) gtk_entry_new ();
02536
        gtk_widget_set_tooltip_text
        (GTK_WIDGET (window->entry_result), _("Best results file"));
window->label_variables = (GtkLabel *) gtk_label_new (_("Variables file"));
window->entry_variables = (GtkEntry *) gtk_entry_new ();
02537
02538
02539
        gtk_widget_set_tooltip_text
02541
           (GTK_WIDGET (window->entry_variables), _("All simulated results file"));
02542
02543
         // Creating the files grid and attaching widgets
02544
        window->grid_files = (GtkGrid *) gtk_grid_new ();
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_simulator),
02545
02546
                            0, 0, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_simulator),
02547
02548
                            1, 0, 1, 1);
02549
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->check_evaluator),
02550
                            0, 1, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_evaluator),
02551
02552
                            1, 1, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->check_cleaner),
02553
02554
                            0, 2, 1, 1);
02555
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_cleaner),
02556
                            1, 2, 1, 1);
        qtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_result),
02557
02558
                           0, 3, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->entry_result),
02560
                            1, 3, 1, 1);
02561
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_variables),
02562
                           0, 4, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->entry_variables),
02563
02564
                           1, 4, 1, 1);
02565
02566
         // Creating the algorithm properties
02567
        window->label_simulations = (GtkLabel *) gtk_label_new
02568
           (_("Simulations number"));
02569
        window->spin simulations
02570
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
        gtk_widget_set_tooltip_text
02571
02572
           (GTK_WIDGET (window->spin_simulations),
02573
            _("Number of simulations to perform for each iteration"));
02574
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_simulations), TRUE);
02575
        window->label_iterations = (GtkLabel *)
          gtk_label_new (_("Iterations number"));
02576
```

```
window->spin_iterations
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e6, 1.);
02578
02579
        gtk_widget_set_tooltip_text
02580
          (GTK_WIDGET (window->spin_iterations), _("Number of iterations"));
02581
        g_signal_connect
          (window->spin_iterations, "value-changed", window_update, NULL);
02582
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_iterations), TRUE);
02583
02584
        window->label_tolerance = (GtkLabel *) gtk_label_new (_("Tolerance"));
02585
        window->spin_tolerance =
02586
          (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02587
        gtk_widget_set_tooltip_text
02588
          (GTK WIDGET (window->spin tolerance).
02589
           _("Tolerance to set the variable interval on the next iteration"));
        window->label_bests = (GtkLabel *) gtk_label_new (_("Bests number"));
02590
02591
        window->spin_bests
02592
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e6, 1.);
02593
        gtk_widget_set_tooltip_text
          (GTK_WIDGET (window->spin_bests),
02594
           _("Number of best simulations used to set the variable interval "
02595
02596
             "on the next iteration"));
02597
        window->label_population
02598
          = (GtkLabel *) gtk_label_new (_("Population number"));
02599
        window->spin population
02600
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02601
        gtk_widget_set_tooltip_text
          (GTK_WIDGET (window->spin_population),
02602
02603
           _("Number of population for the genetic algorithm"));
02604
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_population), TRUE);
02605
        window->label_generations
02606
          = (GtkLabel *) gtk_label_new (_("Generations number"));
02607
        window->spin generations
02608
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e6, 1.);
02609
        gtk_widget_set_tooltip_text
          (GTK_WIDGET (window->spin_generations),
02610
        _("Number of generations for the genetic algorithm"));
window->label_mutation = (GtkLabel *) gtk_label_new (_("Mutation ratio"));
02611
02612
        window->spin_mutation
02613
02614
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02615
        gtk_widget_set_tooltip_text
02616
          (GTK_WIDGET (window->spin_mutation),
02617
           _("Ratio of mutation for the genetic algorithm"));
        window->label_reproduction
02618
          = (GtkLabel *) gtk_label_new (_("Reproduction ratio"));
02619
02620
        window->spin_reproduction
02621
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
        gtk_widget_set_tooltip_text
02622
02623
          (GTK_WIDGET (window->spin_reproduction),
02624
           _("Ratio of reproduction for the genetic algorithm"));
        window->label_adaptation = (GtkLabel *) gtk_label_new (_("Adaptation ratio"));
02625
02626
        window->spin adaptation
02627
           = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02628
        gtk_widget_set_tooltip_text
02629
          (GTK_WIDGET (window->spin_adaptation),
02630
           _("Ratio of adaptation for the genetic algorithm"));
        window->label_threshold = (GtkLabel *) gtk_label_new (_("Threshold"));
window->spin_threshold = (GtkSpinButton *)
02631
02632
          gtk_spin_button_new_with_range (-G_MAXDOUBLE, G_MAXDOUBLE,
02633
                                           precision[DEFAULT_PRECISION]);
02634
02635
        gtk_widget_set_tooltip_text
02636
          (GTK_WIDGET (window->spin_threshold),
           ("Threshold in the objective function to finish the simulations"));
02637
02638
        window->scrolled_threshold = (GtkScrolledWindow *)
02639 #if !GTK4
02640
          gtk scrolled window new (NULL, NULL);
02641 #else
02642
          gtk_scrolled_window_new ();
02643 #endi:
        gtk_scrolled_window_set_child (window->scrolled_threshold,
02644
02645
                                        GTK_WIDGET (window->spin_threshold));
02646 //
          gtk_widget_set_hexpand (GTK_WIDGET (window->scrolled_threshold), TRUE);
02647 //
          gtk_widget_set_halign (GTK_WIDGET (window->scrolled_threshold),
02648 //
                                        GTK ALIGN FILL);
02649
02650
        // Creating the hill climbing method properties
        window>check_climbing = (GtkCheckButton *)
   gtk_check_button_new_with_mnemonic (_("_Hill climbing method"));
02651
02652
        g_signal_connect (window->check_climbing, "toggled", window_update, NULL);
02653
02654
        window->grid_climbing = (GtkGrid *) gtk_grid_new ();
02655 #if !GTK4
02656
        window->button climbing[0] = (GtkRadioButton *)
          gtk_radio_button_new_with_mnemonic (NULL, label_climbing[0]);
02657
02658 #else
02659
       window->button_climbing[0] = (GtkCheckButton *)
02660
          gtk_check_button_new_with_mnemonic (label_climbing[0]);
02661 #endif
02662
        gtk_grid_attach (window->grid_climbing,
02663
                          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>
02665
02666
02667 #if !GTK4
            window->button_climbing[i] = (GtkRadioButton *)
02668
02669
              gtk_radio_button_new_with_mnemonic
              (gtk_radio_button_get_group (window->button_climbing[0]),
02670
02671
               label_climbing[i]);
02672 #else
02673
            window->button_climbing[i] = (GtkCheckButton *)
02674
              gtk_check_button_new_with_mnemonic (label_climbing[i]);
02675
            gtk_check_button_set_group (window->button_climbing[i],
                                         window->button_climbing[0]);
02676
02677 #endif
02678
            gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_climbing[i]),
02679
                                          tip_climbing[i]);
            gtk_grid_attach (window->grid_climbing,
02680
                              GTK_WIDGET (window->button_climbing[i]), 0, i, 1, 1);
02681
            g_signal_connect (window->button_climbing[i], "toggled", window_update,
02682
02683
                              NULL);
02684
02685
        window->label_steps = (GtkLabel *) gtk_label_new (_("Steps number"));
        window->spin_steps = (GtkSpinButton *)
02686
          gtk_spin_button_new_with_range (1., 1.e12, 1.);
02687
        qtk_widget_set_hexpand (GTK_WIDGET (window->spin_steps), TRUE);
02688
02689
        window->label_final_steps
          = (GtkLabel *) gtk_label_new (_("Final steps number"));
02690
02691
        window->spin_final_steps = (GtkSpinButton *)
        gtk_spin_button_new_with_range (1., 1.e12, 1.);
gtk_widget_set_hexpand (GTK_WIDGET (window->spin_final_steps), TRUE);
02692
02693
02694
        window->label estimates
        = (GtkLabel *) gtk_label_new (_("Climbing estimates number"));
window->spin_estimates = (GtkSpinButton *)
02695
02696
02697
          gtk_spin_button_new_with_range (1., 1.e3, 1.);
02698
        window->label_relaxation
          = (GtkLabel *) gtk_label_new (_("Relaxation parameter"));
02699
        window->spin_relaxation = (GtkSpinButton *)
  gtk_spin_button_new_with_range (0., 2., 0.001);
02700
02701
02702
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->label_steps),
02703
                          0, NCLIMBINGS, 1, 1);
02704
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_steps),
                          1, NCLIMBINGS, 1, 1);
02705
02706
        gtk_grid_attach (window->grid_climbing,
02707
                         GTK_WIDGET (window->label_final_steps),
02708
                          0, NCLIMBINGS + 1, 1, 1);
02709
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_final_steps),
02710
                          1, NCLIMBINGS + 1, 1, 1);
        02711
02712
02713
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_estimates),
02714
                          1, NCLIMBINGS + 2, 1, 1);
02715
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->label_relaxation),
02716
                          0, NCLIMBINGS + 3, 1, 1);
02717
        gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_relaxation),
02718
                          1, NCLIMBINGS + 3, 1, 1);
02719
02720
        // Creating the array of algorithms
02721
        window->grid_algorithm = (GtkGrid *) gtk_grid_new ();
02722 #if !GTK4
02723
        window->button_algorithm[0] = (GtkRadioButton *)
02724
          gtk_radio_button_new_with_mnemonic (NULL, label_algorithm[0]);
02725 #else
02726
        window->button_algorithm[0] = (GtkCheckButton *)
02727
          gtk_check_button_new_with_mnemonic (label_algorithm[0]);
02728 #endif
02729
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_algorithm[0]),
02730
                                      tip_algorithm[0]);
        gtk_grid_attach (window->grid_algorithm,
02731
02732
                         GTK_WIDGET (window->button_algorithm[0]), 0, 0, 1, 1);
02733
        g_signal_connect (window->button_algorithm[0], "toggled",
02734
                           window_set_algorithm, NULL);
02735
        for (i = 0; ++i < NALGORITHMS;)</pre>
02736
02737 #if !GTK4
02738
            window->button algorithm[i] = (GtkRadioButton *)
02739
              gtk_radio_button_new_with_mnemonic
02740
              (gtk_radio_button_get_group (window->button_algorithm[0]),
02741
               label_algorithm[i]);
02742 #else
02743
            window->button algorithm[i] = (GtkCheckButton *)
02744
              gtk_check_button_new_with_mnemonic (label_algorithm[i]);
02745
            gtk_check_button_set_group (window->button_algorithm[i]
02746
                                         window->button_algorithm[0]),
02747 #endif
02748
              gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_algorithm[i]),
02749
                                            tip_algorithm[i]);
02750
            gtk grid attach (window->grid algorithm,
```

```
GTK_WIDGET (window->button_algorithm[i]), 0, i, 1, 1);
02752
            g_signal_connect (window->button_algorithm[i], "toggled",
02753
                               window_set_algorithm, NULL);
02754
        gtk_grid_attach (window->grid_algorithm,
02755
02756
                         GTK_WIDGET (window->label_simulations),
                          0, NALGORITHMS, 1, 1);
02757
02758
        gtk_grid_attach (window->grid_algorithm,
02759
                         GTK_WIDGET (window->spin_simulations), 1, NALGORITHMS, 1, 1);
        gtk_grid_attach (window->grid_algorithm,
02760
                          GTK_WIDGET (window->label_iterations),
02761
                          0, NALGORITHMS + 1, 1, 1);
02762
02763
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_iterations),
02764
                          1, NALGORITHMS + 1, 1, 1);
02765
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_tolerance),
02766
                          0, NALGORITHMS + 2, 1, 1);
        \verb|gtk_grid_attach| (\verb|window->grid_algorithm|, GTK_WIDGET| (\verb|window->spin_tolerance)|, \\
02767
                          1, NALGORITHMS + 2, 1, 1);
02768
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_bests),
02769
                          0, NALGORITHMS + 3, 1, 1);
02770
02771
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_bests),
02772
                          1, NALGORITHMS + 3, 1, 1);
02773
        gtk_grid_attach (window->grid_algorithm,
02774
                          GTK_WIDGET (window->label_population),
02775
                          0, NALGORITHMS + 4, 1, 1);
02776
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_population),
02777
                          1, NALGORITHMS + 4, 1, 1);
02778
        gtk_grid_attach (window->grid_algorithm,
02779
                          GTK_WIDGET (window->label_generations),
02780
                          0, NALGORITHMS + 5, 1, 1);
        gtk_grid_attach (window->grid_algorithm,
02781
02782
                         GTK_WIDGET (window->spin_generations),
02783
                          1, NALGORITHMS + 5, 1, 1);
02784
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_mutation),
02785
                          0, NALGORITHMS + 6, 1, 1);
        {\tt gtk\_grid\_attach~(window->grid\_algorithm,~GTK\_WIDGET~(window->spin\_mutation),}
02786
                          1, NALGORITHMS + 6, 1, 1);
02787
        gtk_grid_attach (window->grid_algorithm,
02788
02789
                          GTK_WIDGET (window->label_reproduction),
02790
                          0, NALGORITHMS + 7, 1, 1);
02791
        gtk_grid_attach (window->grid_algorithm,
02792
                         GTK_WIDGET (window->spin_reproduction),
02793
                          1. NALGORITHMS + 7, 1, 1);
02794
        gtk_grid_attach (window->grid_algorithm,
                          GTK_WIDGET (window->label_adaptation),
02795
                          0, NALGORITHMS + 8, 1, 1);
02796
02797
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_adaptation),
02798
                          1, NALGORITHMS + 8, 1, 1);
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->check_climbing),
02799
                         0, NALGORITHMS + 9, 2, 1);
02800
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->grid_climbing),
02801
                          0, NALGORITHMS + 10, 2, 1);
02802
02803
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_threshold),
02804
                         0, NALGORITHMS + 11, 1, 1);
        gtk_grid_attach (window->grid_algorithm,
02805
                         GTK_WIDGET (window->scrolled_threshold),
1, NALGORITHMS + 11, 1, 1);
02806
02807
02808
        window->frame_algorithm = (GtkFrame *) gtk_frame_new (_("Algorithm"));
02809
        gtk_frame_set_child (window->frame_algorithm,
02810
                              GTK_WIDGET (window->grid_algorithm));
02811
02812
        // Creating the variable widgets
02813
        window->combo_variable = (GtkComboBoxText *) gtk_combo_box_text_new ();
02814
        gtk_widget_set_tooltip_text
          (GTK_WIDGET (window->combo_variable), _("Variables selector"));
02815
02816
        window->id_variable = g_signal_connect
02817
          (window->combo_variable, "changed", window_set_variable, NULL);
02818 #if !GTK4
02819
       window->button_add_variable = (GtkButton *)
          gtk_button_new_from_icon_name ("list-add", GTK_ICON_SIZE_BUTTON);
02821 #else
02822
        window->button_add_variable = (GtkButton *)
02823
          gtk_button_new_from_icon_name ("list-add");
02824 #endif
        g_signal_connect (window->button_add_variable, "clicked", window_add_variable,
02825
02826
                           NULL):
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_add_variable),
02827
02828
                                      _("Add variable"));
02829 #if !GTK4
        window->button remove_variable = (GtkButton *)
02830
          gtk_button_new_from_icon_name ("list-remove", GTK_ICON_SIZE_BUTTON);
02831
02832 #else
02833
        window->button_remove_variable = (GtkButton *)
02834
          gtk_button_new_from_icon_name ("list-remove");
02835 #endif
02836
        g_signal_connect (window->button_remove_variable, "clicked",
02837
                           window remove variable, NULL);
```

```
gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_remove_variable),
02839
                                         _("Remove variable"));
        window->label_variable = (GtkLabel *) gtk_label_new (_("Name"));
02840
        window->entry_variable = (GtkEntry *) gtk_entry_new ();
02841
02842
        gtk_widget_set_tooltip_text
02843
           (GTK_WIDGET (window->entry_variable), _("Variable name"));
        gtk_widget_set_hexpand (GTK_WIDGET (window->entry_variable), TRUE);
02844
        window->id_variable_label = g_signal_connect
  (window->entry_variable, "changed", window_label_variable, NULL);
02845
02846
02847
        window->label_min = (GtkLabel *) gtk_label_new (_("Minimum"));
        window->spin_min = (GtkSpinButton *) gtk_spin_button_new_with_range
02848
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02849
02850
        gtk widget set tooltip text
           (GTK_WIDGET (window->spin_min), _("Minimum initial value of the variable"));
02851
02852
        window->scrolled_min = (GtkScrolledWindow *)
02853 #if !GTK4
02854
           gtk_scrolled_window_new (NULL, NULL);
02855 #else
02856
          gtk_scrolled_window_new ();
02857 #endif
02858
        gtk_scrolled_window_set_child (window->scrolled_min,
02859
                                           GTK_WIDGET (window->spin_min));
        g_signal_connect (window->spin_min, "value-changed",
02860
02861
                            window_rangemin_variable, NULL);
02862
        window->label_max = (GtkLabel *) gtk_label_new (_("Maximum"));
        window->spin_max = (GtkSpinButton *) gtk_spin_button_new_with_range
02863
02864
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02865
         gtk_widget_set_tooltip_text
        (GTK_WIDGET (window->spin_max), _("Maximum window->scrolled_max = (GtkScrolledWindow *)
02866
                                               _("Maximum initial value of the variable"));
02867
02868 #if !GTK4
02869
          gtk_scrolled_window_new (NULL, NULL);
02870 #else
02871
          gtk_scrolled_window_new ();
02872 #endif
        gtk_scrolled_window_set_child (window->scrolled_max,
02873
02874
                                           GTK WIDGET (window->spin max));
        g_signal_connect (window->spin_max, "value-changed",
02875
02876
                             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
02877
02878
02879
02880
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02881
        gtk_widget_set_tooltip_text
02882
02883
           (GTK_WIDGET (window->spin_minabs),
02884
            _("Minimum allowed value of the variable"));
02885
        window->scrolled_minabs = (GtkScrolledWindow *)
02886 #if !GTK4
02887
           gtk scrolled window new (NULL, NULL);
02888 #else
02889
           gtk_scrolled_window_new ();
02890 #endif
02891
        gtk_scrolled_window_set_child (window->scrolled_minabs,
02892
                                           GTK WIDGET (window->spin_minabs));
        g_signal_connect (window->spin_minabs, "value-changed",
02893
                            window_rangeminabs_variable, NULL);
02894
02895
        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
02896
02897
02898
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02899
02900
        gtk_widget_set_tooltip_text
02901
           (GTK_WIDGET (window->spin_maxabs),
02902
            _("Maximum allowed value of the variable"));
02903
        window->scrolled_maxabs = (GtkScrolledWindow *)
02904 #if !GTK4
02905
          gtk scrolled window new (NULL, NULL);
02906 #else
02907
          gtk_scrolled_window_new ();
02908 #endif
02909
        gtk_scrolled_window_set_child (window->scrolled_maxabs,
02910
                                           GTK_WIDGET (window->spin_maxabs));
        02911
02912
02913
         window->label_precision = (GtkLabel *) gtk_label_new (_("Precision digits"));
         window->spin_precision = (GtkSpinButton *)
02914
02915
           gtk_spin_button_new_with_range (0., (gdouble) DEFAULT_PRECISION, 1.);
02916
        {\tt gtk\_widget\_set\_tooltip\_text}
           (GTR_WIDGET (window->spin_precision),
_("Number of precision floating point digits\n"
02917
02918
              "0 is for integer numbers"));
02919
02920
        g_signal_connect (window->spin_precision, "value-changed",
02921
                            window_precision_variable, NULL);
02922
        window->label_sweeps = (GtkLabel *) gtk_label_new (_("Sweeps number"));
02923
        window->spin sweeps =
02924
           (GtkSpinButton *) gtk spin button new with range (1., 1.e12, 1.);
```

```
gtk_widget_set_tooltip_text (GTK_WIDGET (window->spin_sweeps),
        __("Number of steps sweeping the variable"));
g_signal_connect (window->spin_sweeps, "value-changed",
02926
02927
                           window_update_variable, NULL);
02928
        window->label_bits = (GtkLabel *) gtk_label_new (_("Bits number"));
02929
02930
        window->spin bits
02931
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 64., 1.);
02932
        {\tt gtk\_widget\_set\_tooltip\_text}
02933
          (GTK_WIDGET (window->spin_bits),
02934
            _("Number of bits to encode the variable"));
02935
        g_signal_connect
          (window->spin_bits, "value-changed", window_update_variable, NULL);
02936
        window->label_step = (GtkLabel *) gtk_label_new (_("Step size"));
window->spin_step = (GtkSpinButton *) gtk_spin_button_new_with_range
02937
02938
02939
          (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02940
        gtk_widget_set_tooltip_text
02941
          (GTK_WIDGET (window->spin_step),
           _("Initial step size for the hill climbing method"));
02942
        window->scrolled_step = (GtkScrolledWindow *)
02943
02944 #if !GTK4
02945
          gtk scrolled window new (NULL, NULL);
02946 #else
02947
          gtk_scrolled_window_new ();
02948 #endif
02949
        gtk_scrolled_window_set_child (window->scrolled_step,
02950
                                        GTK_WIDGET (window->spin_step));
02951
        (window->spin_step, "value-changed", window_step_variable, NULL);
window->grid_variable = (GtkGrid *) gtk_grid_new ();
02952
02953
        gtk_grid_attach (window->grid_variable,
02954
02955
                          GTK WIDGET (window->combo variable), 0, 0, 2, 1);
02956
        gtk_grid_attach (window->grid_variable,
02957
                          GTK_WIDGET (window->button_add_variable), 2, 0, 1, 1);
02958
        gtk_grid_attach (window->grid_variable,
02959
                          GTK_WIDGET (window->button_remove_variable), 3, 0, 1, 1);
02960
        gtk_grid_attach (window->grid_variable,
                          GTK_WIDGET (window->label_variable), 0, 1, 1, 1);
02961
02962
        gtk_grid_attach (window->grid_variable,
02963
                          GTK_WIDGET (window->entry_variable), 1, 1, 3, 1);
02964
        gtk_grid_attach (window->grid_variable,
02965
                          GTK_WIDGET (window->label_min), 0, 2, 1, 1);
02966
        gtk_grid_attach (window->grid_variable,
                          GTK WIDGET (window->scrolled_min), 1, 2, 3, 1);
02967
02968
        gtk_grid_attach (window->grid_variable,
02969
                          GTK_WIDGET (window->label_max), 0, 3, 1, 1);
02970
        gtk_grid_attach (window->grid_variable,
02971
                          GTK_WIDGET (window->scrolled_max), 1, 3, 3, 1);
02972
        gtk_grid_attach (window->grid_variable,
02973
                          GTK WIDGET (window->check minabs), 0, 4, 1, 1);
02974
        gtk grid attach (window->grid variable,
                          GTK_WIDGET (window->scrolled_minabs), 1, 4, 3, 1);
02976
        gtk_grid_attach (window->grid_variable,
02977
                          GTK_WIDGET (window->check_maxabs), 0, 5, 1, 1);
02978
        gtk_grid_attach (window->grid_variable,
02979
                          GTK_WIDGET (window->scrolled_maxabs), 1, 5, 3, 1);
02980
        gtk grid attach (window->grid variable,
                          GTK_WIDGET (window->label_precision), 0, 6, 1, 1);
02981
        gtk_grid_attach (window->grid_variable,
02982
02983
                          GTK_WIDGET (window->spin_precision), 1, 6, 3, 1);
02984
        gtk_grid_attach (window->grid_variable,
                          GTK WIDGET (window->label_sweeps), 0, 7, 1, 1);
02985
02986
        gtk_grid_attach (window->grid_variable,
02987
                          GTK_WIDGET (window->spin_sweeps), 1, 7, 3, 1);
        gtk_grid_attach (window->grid_variable,
02988
02989
                          GTK_WIDGET (window->label_bits), 0, 8, 1, 1);
02990
        gtk_grid_attach (window->grid_variable,
02991
                          GTK_WIDGET (window->spin_bits), 1, 8, 3, 1);
02992
        gtk grid attach (window->grid variable,
02993
                          GTK_WIDGET (window->label_step), 0, 9, 1, 1);
        gtk_grid_attach (window->grid_variable,
02995
                          GTK_WIDGET (window->scrolled_step), 1, 9, 3, 1);
02996
        window->frame_variable = (GtkFrame *) gtk_frame_new (_("Variable"));
        gtk_frame_set_child (window->frame_variable,
02997
02998
                              GTK_WIDGET (window->grid_variable));
02999
03000
        // Creating the experiment widgets
        window->combo_experiment = (GtkComboBoxText *) gtk_combo_box_text_new ();
03001
03002
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->combo_experiment),
03003
                                       _("Experiment selector"));
        window->id_experiment = g_signal_connect
03004
          (window->combo_experiment, "changed", window_set_experiment, NULL);
03005
03006 #if
          !GTK4
03007
        window->button_add_experiment = (GtkButton *)
03008
          gtk_button_new_from_icon_name ("list-add", GTK_ICON_SIZE_BUTTON);
03009 #else
       window->button_add_experiment = (GtkButton *)
03010
03011
          gtk button new from icon name ("list-add");
```

```
03012 #endif
03013
        g_signal_connect
          (window->button_add_experiment, "clicked", window_add_experiment, NULL);
03014
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_add_experiment),
03015
03016
                                      _("Add experiment"));
03017 #if !GTK4
03018
       window->button_remove_experiment = (GtkButton *)
03019
         gtk_button_new_from_icon_name ("list-remove", GTK_ICON_SIZE_BUTTON);
03020 #else
03021
        window->button_remove_experiment = (GtkButton *)
         gtk_button_new_from_icon_name ("list-remove");
03022
03023 #endif
03024
        g_signal_connect (window->button_remove_experiment, "clicked",
                           window_remove_experiment, NULL);
03025
03026
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_remove_experiment),
03027
                                      _("Remove experiment"));
03028
        window->label_experiment
03029
          = (GtkLabel *) gtk label new ( ("Experimental data file"));
03030
        window->button_experiment = (GtkButton *)
03031
          gtk_button_new_with_mnemonic (_("Experimental data file"));
03032
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_experiment),
03033
                                      _("Experimental data file"));
        g_signal_connect (window->button_experiment, "clicked",
03034
        window_name_experiment, NULL);
gtk_widget_set_hexpand (GTK_WIDGET (window->button_experiment), TRUE);
03035
03036
03037
        window->label_weight = (GtkLabel *) gtk_label_new (_("Weight"));
03038
        window->spin_weight
03039
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
03040
        gtk_widget_set_tooltip_text
          (GTK_WIDGET (window->spin_weight),
03041
           __("Weight factor to build the objective function"));
03042
03043
        g signal connect
03044
          (window->spin_weight, "value-changed", window_weight_experiment, NULL);
        window->grid_experiment = (GtkGrid *) gtk_grid_new ();
03045
03046
        gtk_grid_attach (window->grid_experiment,
03047
                         GTK_WIDGET (window->combo_experiment), 0, 0, 2, 1);
03048
        gtk grid attach (window->grid experiment,
                         GTK_WIDGET (window->button_add_experiment), 2, 0, 1, 1);
03049
03050
        gtk_grid_attach (window->grid_experiment,
03051
                          GTK_WIDGET (window->button_remove_experiment), 3, 0, 1, 1);
03052
        gtk_grid_attach (window->grid_experiment,
                          GTK_WIDGET (window->label_experiment), 0, 1, 1, 1);
03053
03054
        gtk_grid_attach (window->grid_experiment,
03055
                         GTK_WIDGET (window->button_experiment), 1, 1, 3, 1);
        gtk_grid_attach (window->grid_experiment,
03056
03057
                          GTK_WIDGET (window->label_weight), 0, 2, 1, 1);
03058
        gtk_grid_attach (window->grid_experiment,
03059
                          GTK_WIDGET (window->spin_weight), 1, 2, 3, 1);
03060
        for (i = 0; i < MAX_NINPUTS; ++i)</pre>
03061
            snprintf (buffer3, 64, "%s %u", _("Input template"), i + 1);
window->check_template[i] = (GtkCheckButton *)
03062
03063
03064
              gtk_check_button_new_with_label (buffer3);
03065
            window->id template[i]
03066
              = g_signal_connect (window->check_template[i], "toggled",
03067
                                   window inputs experiment, NULL);
            gtk_grid_attach (window->grid_experiment,
03068
                             GTK_WIDGET (window->check_template[i]), 0, 3 + i, 1, 1);
03069
03070
            window->button_template[i] = (GtkButton *)
03071
              gtk_button_new_with_mnemonic (_("Input template"));
03072
            gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_template[i]),
03073
03074
                                          _("Experimental input template file"));
03075
            window->id input[i] =
03076
              g_signal_connect_swapped (window->button_template[i], "clicked",
03077
                                         (GCallback) window_template_experiment,
03078
                                         (void \star) (size_t) i);
03079
            gtk_grid_attach (window->grid_experiment,
                             GTK_WIDGET (window->button_template[i]), 1, 3 + i, 3, 1);
03080
03081
03082
        window->frame_experiment = (GtkFrame *) gtk_frame_new (_("Experiment"));
03083
        gtk_frame_set_child (window->frame_experiment,
03084
                             GTK_WIDGET (window->grid_experiment));
03085
03086
        // Creating the error norm widgets
        window->frame_norm = (GtkFrame *) gtk_frame_new (_("Error norm"));
03087
        window->grid_norm = (GtkGrid *) gtk_grid_new ();
03088
03089
        gtk_frame_set_child (window->frame_norm, GTK_WIDGET (window->grid_norm));
03090 #if !GTK4
03091
        window->button norm[0] = (GtkRadioButton *)
03092
          gtk_radio_button_new_with_mnemonic (NULL, label_norm[0]);
03093 #else
03094
       window->button_norm[0] = (GtkCheckButton *)
03095
          gtk_check_button_new_with_mnemonic (label_norm[0]);
03096 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_norm[0]),
03097
03098
                                      tip norm[01):
```

```
gtk_grid_attach (window->grid_norm,
                          GTK_WIDGET (window->button_norm[0]), 0, 0, 1, 1);
03100
03101
        g_signal_connect (window->button_norm[0], "toggled", window_update, NULL);
        for (i = 0; ++i < NNORMS;)</pre>
03102
03103
03104 #if !GTK4
03105
             window->button_norm[i] = (GtkRadioButton *)
03106
               gtk_radio_button_new_with_mnemonic
03107
               (gtk_radio_button_get_group (window->button_norm[0]), label_norm[i]);
03108 #else
            window->button norm[i] = (GtkCheckButton *)
03109
               gtk_check_button_new_with_mnemonic (label_norm[i]);
03110
03111
             gtk_check_button_set_group (window->button_norm[i],
03112
                                           window->button_norm[0]);
03113 #endif
0.3114
            gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_norm[i]),
03115
                                            tip_norm[i]);
            gtk_grid_attach (window->grid_norm,
03116
03117
                               GTK_WIDGET (window->button_norm[i]), 0, i, 1, 1);
03118
            g_signal_connect (window->button_norm[i], "toggled", window_update, NULL);
03119
03120
        window->label_p = (GtkLabel *) gtk_label_new (_("P parameter"));
        gtk_grid_attach (window->grid_norm, GTK_WIDGET (window->label_p), 1, 1, 1, 1);
03121
        window->spin_p = (GtkSpinButton *)
0.3122
03123
          gtk_spin_button_new_with_range (-G_MAXDOUBLE, G_MAXDOUBLE, 0.01);
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->spin_p),
03124
03125
                                        _("P parameter for the P error norm"));
03126
        window->scrolled_p = (GtkScrolledWindow *)
03127 #if !GTK4
          gtk_scrolled_window_new (NULL, NULL);
03128
03129 #else
03130
          gtk_scrolled_window_new ();
03131 #endif
03132
        gtk_scrolled_window_set_child (window->scrolled_p,
03133
                                          GTK_WIDGET (window->spin_p));
        gtk_widget_set_hexpand (GTK_WIDGET (window->scrolled_p), TRUE);
03134
        gtk_widget_set_halign (GTK_WIDGET (window->scrolled_p), GTK_ALIGN_FILL);
03135
        gtk_grid_attach (window->grid_norm, GTK_WIDGET (window->scrolled_p),
03136
03137
                           1, 2, 1, 2);
03138
03139
        \ensuremath{//} Creating the grid and attaching the widgets to the grid
        window->grid = (GtkGrid *) gtk_grid_new ();
03140
        gtk_grid_attach (window->grid, GTK_WIDGET (window->box_buttons), 0, 0, 3, 1); gtk_grid_attach (window->grid, GTK_WIDGET (window->grid_files), 0, 1, 1, 1);
0.3141
03142
        gtk_grid_attach (window->grid,
03143
03144
                           GTK_WIDGET (window->frame_algorithm), 0, 2, 1, 1);
03145
        gtk_grid_attach (window->grid,
0.3146
                           GTK_WIDGET (window->frame_variable), 1, 2, 1, 1);
03147
        gtk grid attach (window->grid,
03148
                          GTK_WIDGET (window->frame_experiment), 2, 2, 1, 1);
        gtk_grid_attach (window->grid, GTK_WIDGET (window->frame_norm), 1, 1, 2, 1);
03149
03150
        gtk_window_set_child (window->window, GTK_WIDGET (window->grid));
0.31.51
03152
        // Setting the window logo
03153
        window->logo = gdk_pixbuf_new_from_xpm_data (logo);
03154 #if !GTK4
        gtk_window_set_icon (window->window, window->logo);
03156 #endif
03157
03158
        // Showing the window
03159 #if !GTK4
03160
        gtk_widget_show_all (GTK_WIDGET (window->window));
03161 #else
03162
        gtk_widget_show (GTK_WIDGET (window->window));
03163 #endif
03164
03165
        // In GTK+ 3.16 and 3.18 the default scrolled size is wrong
03166 #if GTK_MINOR_VERSION >= 16
       gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_min), -1, 40);
gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_max), -1, 40);
03167
03168
03169
        gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_minabs), -1, 40);
03170
        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_p), -1, 40);
03171
03172
        gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_threshold), -1, 40);
03173
03174 #endif
03175
03176
        // Reading initial example
        input_new ();
buffer2 = g_get_current_dir ();
03177
03178
        buffer = g_build_filename (buffer2, "..", "tests", "test1", INPUT_FILE, NULL);
03179
03180
        g_free (buffer2);
        window_read (buffer);
03181
03182
        g_free (buffer);
03183
03184 #if DEBUG INTERFACE
        fprintf (stderr, "window_new: start\n");
03185
```

```
03186 #endif
03187 }
```

4.11.3.29 window_open()

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

Function to open the input data.

Definition at line 2153 of file interface.c.

```
02154 {
02155
         GtkFileChooserDialog *dlg;
         GtkFileFilter *filter;
02156
02157
02158 #if DEBUG_INTERFACE
        fprintf (stderr, "window_open: start\n");
02159
02160 #endif
02161
02162
          // Opening dialog
02163
         dlg = (GtkFileChooserDialog *)
02164
           gtk_file_chooser_dialog_new (_("Open input file"),
02165
                                               window->window
                                               GTK_FILE_CHOOSER_ACTION_OPEN,
02166
                                               _("_Cancel"), GTK_RESPONSE_CANCEL,
02167
02168
                                               _("_OK"), GTK_RESPONSE_OK, NULL);
02169
02170
         // Adding XML filter
         filter = (GtkFileFilter *) gtk_file_filter_new ();
gtk_file_filter_set_name (filter, "XML");
02171
02172
         gtk_file_filter_add_pattern (filter, "*.xml");
gtk_file_filter_add_pattern (filter, "*.XML");
02173
02174
02175
         gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter);
02176
02177
         // Adding JSON filter
        filter = (GtkFileFilter *) gtk_file_filter_new ();
gtk_file_filter_set_name (filter, "JSON");
02178
02179
02180
         gtk_file_filter_add_pattern (filter, "*.json");
         gtk_file_filter_add_pattern (filter, "*.JSON");
gtk_file_filter_add_pattern (filter, "*.js");
gtk_file_filter_add_pattern (filter, "*.JS");
02181
02182
02183
02184
         gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter);
02185
02186
        // Connecting the close function
02187
        g_signal_connect (dlg, "response", G_CALLBACK (dialog_open_close), NULL);
02188
02189
         // Showing modal dialog
        gtk_window_present (GTK_WINDOW (dlg));
02190
02191
02192 #if DEBUG_INTERFACE
02193 fprintf (stderr, "window_open: end\n");
02194 #endif
02195 }
```

Here is the call graph for this function:

4.11.3.30 window precision variable()

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

Function to update the variable precision in the main window.

Definition at line 1833 of file interface.c.

```
01834 {
01835    unsigned int i;
01836    #if DEBUG_INTERFACE
01837    fprintf (stderr, "window_precision_variable: start\n");
01838    #endif
01839    i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01840    input->variable[i].precision
01841    = (unsigned int) gtk_spin_button_get_value_as_int (window->spin_precision);
```

```
gtk_spin_button_set_digits (window->spin_min, input->variable[i].precision);
01843
       gtk_spin_button_set_digits (window->spin_max, input->variable[i].precision);
01844
       gtk_spin_button_set_digits (window->spin_minabs,
                                    input->variable[i].precision);
01845
01846
       gtk_spin_button_set_digits (window->spin_maxabs,
                                    input->variable[i].precision);
01847
01848 #if DEBUG_INTERFACE
01849
       fprintf (stderr, "window_precision_variable: end\n");
01850 #endif
01851 }
```

4.11.3.31 window_rangemax_variable()

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

Function to update the variable rangemax in the main window.

Definition at line 1874 of file interface.c.

```
01875 {
01876    unsigned int i;
01877    #if DEBUG_INTERFACE
01878    fprintf (stderr, "window_rangemax_variable: start\n");
01879    #endif
01880    i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01881    input->variable[i].rangemax = gtk_spin_button_get_value (window->spin_max);
01882    #if DEBUG_INTERFACE
01883    fprintf (stderr, "window_rangemax_variable: end\n");
01884    #endif
01885 }
```

4.11.3.32 window_rangemaxabs_variable()

static void window_rangemaxabs_variable () [static]

Function to update the variable rangemaxabs in the main window.

Definition at line 1909 of file interface.c.

```
01910 {
01911
        unsigned int i;
01912 #if DEBUG_INTERFACE
       fprintf (stderr, "window_rangemaxabs_variable: start\n");
01913
01914 #endif
01915
       i = gtk combo box get active (GTK COMBO BOX (window->combo variable));
       input->variable[i].rangemaxabs
01917
          = gtk_spin_button_get_value (window->spin_maxabs);
01918 #if DEBUG_INTERFACE
       fprintf (stderr, "window_rangemaxabs_variable: end\n");
01919
01920 #endif
01921 }
```

4.11.3.33 window_rangemin_variable()

static void window_rangemin_variable () [static]

Function to update the variable rangemin in the main window.

Definition at line 1857 of file interface.c.

```
01858 {
01859    unsigned int i;
01860    #if DEBUG_INTERFACE
01861    fprintf (stderr, "window_rangemin_variable: start\n");
01862    #endif
01863    i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01864    input->variable[i].rangemin = gtk_spin_button_get_value (window->spin_min);
01865    #if DEBUG_INTERFACE
01866    fprintf (stderr, "window_rangemin_variable: end\n");
01867    #endif
01868 }
```

4.11.3.34 window_rangeminabs_variable()

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

Function to update the variable rangeminabs in the main window.

Definition at line 1891 of file interface.c.

```
01892 {
01893    unsigned int i;
01894 #if DEBUG_INTERFACE
01895    fprintf (stderr, "window_rangeminabs_variable: start\n");
01896 #endif
01897    i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01898    input->variable[i].rangeminabs
01899    = gtk_spin_button_get_value (window->spin_minabs);
01900 #if DEBUG_INTERFACE
01901    fprintf (stderr, "window_rangeminabs_variable: end\n");
01902 #endif
01903 }
```

4.11.3.35 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 1983 of file interface.c.

```
01984 {
01985
        unsigned int i;
01986 #if DEBUG_INTERFACE
       fprintf (stderr, "window_read: start\n");
fprintf (stderr, "window_read: file name=%s\n", filename);
01987
01988
01989 #endif
01990
01991
        // Reading new input file
        input_free ();
input->result = input->variables = NULL;
01992
01993
        if (!input_open (filename))
01994
01995
01996 #if DEBUG_INTERFACE
01997
            fprintf (stderr, "window_read: end\n");
01998 #endif
01999
            return 0;
02000
02001
        // Setting GTK+ widgets data
02002
02003
        gtk_entry_set_text (window->entry_result, input->result);
02004
        gtk_entry_set_text (window->entry_variables, input->variables);
02005
        gtk_button_set_label (window->button_simulator, input->simulator);
02006
        gtk_check_button_set_active (window->check_evaluator,
02007
                                       (size_t) input->evaluator);
02008
        if (input->evaluator)
02009
          gtk_button_set_label (window->button_evaluator, input->evaluator);
02010
        gtk_check_button_set_active (window->check_cleaner, (size_t) input->cleaner);
02011
        if (input->cleaner)
02012
          gtk_button_set_label (window->button_cleaner, input->cleaner);
```

```
gtk_check_button_set_active (window->button_algorithm[input->algorithm],
02014
02015
        switch (input->algorithm)
02016
02017
          case ALGORITHM MONTE CARLO:
02018
            gtk_spin_button_set_value (window->spin_simulations,
                                          (gdouble) input->nsimulations);
02020
02021
          case ALGORITHM_SWEEP:
02022
          case ALGORITHM ORTHOGONAL:
02023
            gtk_spin_button_set_value (window->spin_iterations,
                                          (gdouble) input->niterations);
02024
02025
             gtk_spin_button_set_value (window->spin_bests, (gdouble) input->nbest);
02026
             gtk_spin_button_set_value (window->spin_tolerance, input->tolerance);
02027
             gtk_check_button_set_active (window->check_climbing, input->nsteps);
02028
             if (input->nsteps)
02029
02030
                 gtk check button set active
02031
                   (window->button_climbing[input->climbing], TRUE);
02032
                 gtk_spin_button_set_value (window->spin_steps,
02033
                                               (gdouble) input->nsteps);
02034
                 gtk_spin_button_set_value (window->spin_final_steps,
02035
                                               (gdouble) input->nfinal_steps);
02036
                 gtk_spin_button_set_value (window->spin_relaxation,
02037
                                               (gdouble) input->relaxation);
02038
                 switch (input->climbing)
02039
02040
                   case CLIMBING_METHOD_RANDOM:
02041
                     gtk_spin_button_set_value (window->spin_estimates,
02042
                                                   (gdouble) input->nestimates);
02043
                   }
02044
              }
02045
            break;
02046
          default:
02047
            gtk_spin_button_set_value (window->spin_population,
02048
                                          (gdouble) input->nsimulations);
02049
            gtk_spin_button_set_value (window->spin_generations,
02050
                                          (gdouble) input->niterations);
02051
             gtk_spin_button_set_value (window->spin_mutation, input->mutation_ratio);
02052
             gtk_spin_button_set_value (window->spin_reproduction,
02053
                                          input->reproduction_ratio);
02054
             gtk_spin_button_set_value (window->spin_adaptation,
                                          input->adaptation_ratio);
02055
02056
02057
        gtk_check_button_set_active (window->button_norm[input->norm], TRUE);
02058
        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);
02059
02060
        gtk_combo_box_text_remove_all (window->combo_experiment);
02061
02062
             (i = 0; i < input->nexperiments; ++i)
02063
          gtk_combo_box_text_append_text (window->combo_experiment,
02064
                                             input->experiment[i].name);
02065
        g_signal_handler_unblock (window->combo_experiment, window->id_experiment);
02066
        gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), 0);
02067
        g_signal_handler_block (window->combo_variable, window->id_variable);
        g_signal_handler_block (window->entry_variable, window->id_variable_label);
gtk_combo_box_text_remove_all (window->combo_variable);
02068
02069
02070
            (i = 0; i < input->nvariables; ++i)
02071
          gtk_combo_box_text_append_text (window->combo_variable,
02072
                                             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);
02073
02074
02075
        gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), 0);
02076
        window_set_variable ();
02077
        window_update ();
02078
02079 #if DEBUG_INTERFACE
        fprintf (stderr, "window_read: end\n");
02080
02081 #endif
02082
        return 1;
02083 }
```

Here is the call graph for this function:

4.11.3.36 window_remove_experiment()

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

Function to remove an experiment in the main window.

Definition at line 1405 of file interface.c.

```
01407
         unsigned int i, j;
01408 #if DEBUG_INTERFACE
        fprintf (stderr, "window_remove_experiment: start\n");
01409
01410 #endif
01411 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
         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);
01413
01414
        experiment_free (input->experiment + i, input->type);
--input->nexperiments;
01415
01416
        for (j = i; j < input->nexperiments; ++j)
  memcpy (input->experiment + j, input->experiment + j + 1,
01417
01418
01419
                    sizeof (Experiment));
        j = input->nexperiments - 1;
if (i > j)
01420
01421
          i = j;
01422
        for (j = 0; j < input->experiment->ninputs; ++j)
01423
          g_signal_handler_block (window->button_template[j], window->id_input[j]);
01425
         gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), i);
01426
        for (j = 0; j < input->experiment->ninputs; ++j)
          g_signal_handler_unblock (window->button_template[j], window->id_input[j]);
01427
01428 window_update ();
01429 #if DEBUG_INTERFACE
        fprintf (stderr, "window_remove_experiment: end\n");
01430
01431 #endif
01432 }
```

Here is the call graph for this function:

4.11.3.37 window_remove_variable()

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

Function to remove a variable in the main window.

```
Definition at line 1745 of file interface.c.
```

```
01747
         unsigned int i, j;
01748 #if DEBUG_INTERFACE
01749
        fprintf (stderr, "window_remove_variable: start\n");
01750 #endif
01751
        i = qtk_combo_box_qet_active (GTK_COMBO_BOX (window->combo variable));
        g_signal_handler_block (window->combo_variable, window->id_variable);
01752
01753
        gtk_combo_box_text_remove (window->combo_variable, i);
01754
        g_signal_handler_unblock (window->combo_variable, window->id_variable);
01755
        xmlFree (input->variable[i].name);
        --input->nvariables;
for (j = i; j < input->nvariables; ++j)
01756
01757
        memcpy (input->variable + j, input->variable + j + 1, sizeof (Variable));
j = input->nvariables - 1;
01758
01759
01760
        if (i > j)
01761
          i = j;
01762
        g_signal_handler_block (window->entry_variable, window->id_variable_label);
        gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), i);
g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
01763
01764
01765
         window_update ();
01766 #if DEBUG_INTERFACE
01767
        fprintf (stderr, "window_remove_variable: end\n");
01768 #endif
01769 }
```

Here is the call graph for this function:

4.11.3.38 window_run()

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

Function to run a optimization.

Definition at line 1044 of file interface.c.

```
char *msg, *msg2, buffer[64], buffer2[64];
01046
01047
       unsigned int i;
01048 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: start\n");
01049
01050 #endif
01051 window_save ();
01052
       running_new ();
01053
       jbw_process_pending ();
01054
       optimize_open ();
01055 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: closing running dialog\n");
01056
01057 #endif
01058 gtk_spinner_stop (running->spinner);
01059
       gtk_window_destroy (GTK_WINDOW (running->dialog));
01060 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: displaying results\n");
01061
01062 #endif
01063
       snprintf (buffer, 64, "error = %.15le\n", optimize->error_old[0]);
01064
       msg2 = g_strdup (buffer);
01065
       for (i = 0; i < optimize->nvariables; ++i, msg2 = msg)
01066
           snprintf (buffer, 64, "%s = %sn",
01067
01068
                     input->variable[i].name, format[input->variable[i].precision]);
01069
           snprintf (buffer2, 64, buffer, optimize->value_old[i]);
01070
           msg = g_strconcat (msg2, buffer2, NULL);
01071
           g_free (msg2);
01072
       01073
01074
       msg = g_strconcat (msg2, buffer, NULL);
01075
       g_free (msg2);
01077
       jbw_show_message_gtk (_("Best result"), msg, INFO_TYPE);
01078
       g_free (msg);
01079 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: freeing memory\n");
01080
01081 #endif
       optimize_free ();
01083 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: end\n");
01085 #endif
01086 }
```

Here is the call graph for this function:

4.11.3.39 window_save()

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

Function to save the input file.

Definition at line 984 of file interface.c.

```
00985 {
00986
        GtkFileChooserDialog *dlg;
00987
        GtkFileFilter *filter1, *filter2;
00988
        char *buffer;
00989
00990 #if DEBUG_INTERFACE
        fprintf (stderr, "window_save: start\n");
00991
00992 #endif
00993
00994
          // Opening the saving dialog
00995
        dlg = (GtkFileChooserDialog *)
           gtk_file_chooser_dialog_new (_("Save file"),
00996
00997
                                            window->window.
00998
                                            GTK_FILE_CHOOSER_ACTION_SAVE,
                                            __("_Cancel"), GTK_RESPONSE_CANCEL,
00999
01000
                                            _("_OK"), GTK_RESPONSE_OK, NULL);
01001 #if !GTK4
01002
        gtk_file_chooser_set_do_overwrite_confirmation (GTK_FILE_CHOOSER (dlg), TRUE);
01003 #endif
buffer = g_build_filename (input->directory, input->name, NULL);
01005 gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (dlg), buffer);
        g_free (buffer);
01006
01007
01008
        // Adding XML filter
        filter1 = (GtkFileFilter *) gtk_file_filter_new ();
gtk_file_filter_set_name (filter1, "XML");
01009
01010
01011
        gtk_file_filter_add_pattern (filter1, "*.xml");
        gtk_file_filter_add_pattern (filter1, "*.XML");
```

```
gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter1);
01014
01015
         // Adding JSON filter
        filter2 = (GtkFileFilter *) gtk_file_filter_new ();
gtk_file_filter_set_name (filter2, "JSON");
01016
01017
        gtk_file_filter_add_pattern (filter2, "*.json");
gtk_file_filter_add_pattern (filter2, "*.JSON");
01018
01019
        gtk_file_filter_add_pattern (filter2, "*.js");
gtk_file_filter_add_pattern (filter2, "*.JS");
01020
01021
01022
         gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter2);
01023
         if (input->type == INPUT_TYPE_XML)
01024
01025
          gtk_file_chooser_set_filter (GTK_FILE_CHOOSER (dlg), filter1);
01026
01027
           gtk_file_chooser_set_filter (GTK_FILE_CHOOSER (dlg), filter2);
01028
        // Connecting the close function
01029
        g_signal_connect (dlg, "response", G_CALLBACK (dialog_save_close), NULL);
01030
01031
01032
        // Showing modal dialog
        gtk_window_present (GTK_WINDOW (dlg));
01033
01034
01035 #if DEBUG_INTERFACE
01036 fprintf (stderr, "window_save: end\n");
01037 #endif
01038 }
```

4.11.3.40 window_save_climbing()

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

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

Definition at line 839 of file interface.c.

```
00841 #if DEBUG_INTERFACE
00842
       fprintf (stderr, "window_save_climbing: start\n");
00843 #endif
00844
       if (gtk_check_button_get_active (window->check_climbing))
00845
00846
            input->nsteps = gtk_spin_button_get_value_as_int (window->spin_steps);
00847
00848
              = gtk_spin_button_get_value_as_int (window->spin_final_steps);
00849
            input->relaxation = gtk_spin_button_get_value (window->spin_relaxation);
00850
            switch (window_get_climbing ())
00851
             {
00852
             case CLIMBING_METHOD_COORDINATES:
00853
               input->climbing = CLIMBING_METHOD_COORDINATES;
00854
00855
              default:
               input->climbing = CLIMBING_METHOD_RANDOM;
00856
                input->nestimates
00857
00858
                  = gtk_spin_button_get_value_as_int (window->spin_estimates);
00859
              }
00860
00861
       else
00862 input->nsteps = 0;
00863 #if DEBUG_INTERFACE
00864
       fprintf (stderr, "window_save_climbing: end\n");
00865 #endif
00866 }
```

Here is the call graph for this function:

4.11.3.41 window_set_algorithm()

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

Function to avoid memory errors changing the algorithm.

Definition at line 1342 of file interface.c.

```
01344
01345 #if DEBUG_INTERFACE
       fprintf (stderr, "window_set_algorithm: start\n");
01346
01347 #endif
01348 i = window_get_algorithm ();
01349
       switch (i)
01350
01351
         case ALGORITHM_SWEEP:
         case ALGORITHM ORTHOGONAL:
01352
          i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01353
           if (i < 0)
01354
01355
             i = 0;
01356
           gtk_spin_button_set_value (window->spin_sweeps,
01357
                                       (gdouble) input->variable[i].nsweeps);
         break;
case ALGORITHM_GENETIC:
01358
01359
01360
          i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01361
           if (i < 0)</pre>
01362
             i = 0;
01363
            gtk_spin_button_set_value (window->spin_bits,
01364
                                       (gdouble) input->variable[i].nbits);
01365
01366
       window_update ();
01367 #if DEBUG_INTERFACE
01368
       fprintf (stderr, "window_set_algorithm: end\n");
01369 #endif
01370 }
```

Here is the call graph for this function:

4.11.3.42 window_set_experiment()

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

Function to set the experiment data in the main window.

Definition at line 1376 of file interface.c.

```
01378
        unsigned int i, j;
01379 char *buffer1;
01380 #if DEBUG_INTERFACE
01381
       fprintf (stderr, "window_set_experiment: start\n");
01382 #endif
       i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01384
        gtk_spin_button_set_value (window->spin_weight, input->experiment[i].weight);
01385
        buffer1 = gtk_combo_box_text_get_active_text (window->combo_experiment);
01386
        gtk_button_set_label (window->button_experiment, buffer1);
01387
        g free (buffer1);
01388
       for (j = 0; j < input->experiment->ninputs; ++j)
01390
            g_signal_handler_block (window->button_template[j], window->id_input[j]);
01391
            gtk_button_set_label (window->button_template[j],
01392
                                   input->experiment[i].stencil[j]);
            g_signal_handler_unblock
01393
01394
              (window->button_template[j], window->id_input[j]);
01395
01396 #if DEBUG_INTERFACE
01397 fprintf (stderr, "window_set_experiment: end\n");
01398 #endif
01399 }
```

4.11.3.43 window_set_variable()

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

Function to set the variable data in the main window.

Definition at line 1670 of file interface.c.

```
01671 {
01672
        unsigned int i;
01673 #if DEBUG_INTERFACE
01674
       fprintf (stderr, "window_set_variable: start\n");
01675 #endif
01676
       i = gtk combo box get active (GTK COMBO BOX (window->combo variable));
        g_signal_handler_block (window->entry_variable, window->id_variable_label);
01677
        gtk_entry_set_text (window->entry_variable, input->variable[i].name);
01679
        g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
01680
        gtk_spin_button_set_value (window->spin_min, input->variable[i].rangemin);
        gtk_spin_button_set_value (window->spin_max, input->variable[i].rangemax);
01681
        if (input->variable[i].rangeminabs != -G_MAXDOUBLE)
01682
01683
          {
01684
            gtk_spin_button_set_value (window->spin_minabs,
01685
                                        input->variable[i].rangeminabs);
01686
            gtk_check_button_set_active (window->check_minabs, 1);
01687
01688
        else
01689
         {
01690
            gtk_spin_button_set_value (window->spin_minabs, -G_MAXDOUBLE);
            gtk_check_button_set_active (window->check_minabs, 0);
01691
01692
01693
        if (input->variable[i].rangemaxabs != G_MAXDOUBLE)
01694
01695
            gtk_spin_button_set_value (window->spin_maxabs,
                                        input->variable[i].rangemaxabs);
01696
01697
            gtk_check_button_set_active (window->check_maxabs, 1);
01698
01699
        else
01700
         {
01701
            gtk spin button set value (window->spin maxabs, G MAXDOUBLE):
01702
            gtk_check_button_set_active (window->check_maxabs, 0);
01703
01704
        gtk_spin_button_set_value (window->spin_precision,
01705
                                    input->variable[i].precision);
       gtk_spin_button_set_value (window->spin_steps, (gdouble) input->nsteps);
gtk_spin_button_set_value (window->spin_final_steps,
01706
01707
01708
                                    (gdouble) input->nfinal_steps);
01709
01710 gtk_spin_button_set_value (window->spin_step, input->variable[i].step);
01711 #if DEBUG_INTERFACE
01712 fprintf (stderr, "window_set_variable: precision[%u]=%u\n", i,
01713
                 input->variable[i].precision);
01714 #endif
01715 switch (window_get_algorithm ())
01716
01717
          case ALGORITHM_SWEEP:
01718
         case ALGORITHM_ORTHOGONAL:
01719
           gtk_spin_button_set_value (window->spin_sweeps,
01720
                                        (gdouble) input->variable[i].nsweeps);
01721 #if DEBUG_INTERFACE
01722
           fprintf (stderr, "window_set_variable: nsweeps[%u]=%u\n", i,
01723
                     input->variable[i].nsweeps);
01724 #endif
01725
           break:
          case ALGORITHM_GENETIC:
01726
01727
           gtk_spin_button_set_value (window->spin_bits,
01728
                                        (gdouble) input->variable[i].nbits);
01729 #if DEBUG_INTERFACE
           fprintf (stderr, "window_set_variable: nbits[%u]=%u\n", i,
01730
                     input->variable[i].nbits);
01731
01732 #endif
01733
           break;
01734
01735
       window_update ();
01736 #if DEBUG_INTERFACE
       fprintf (stderr, "window_set_variable: end\n");
01737
01738 #endif
01739 }
```

4.11.3.44 window_step_variable()

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

Function to update the variable step in the main window.

Definition at line 1927 of file interface.c.

```
01929 unsigned int i;
01930 #if DEBUG_INTERFACE
01931 fprintf (stderr, "window_step_variable: start\n");
01932 #endif
01933 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01934 input->variable[i].step = gtk_spin_button_get_value (window->spin_step);
01935 #if DEBUG_INTERFACE
01936 fprintf (stderr, "window_step_variable: end\n");
01937 #endif
01938 }
```

4.11.3.45 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 1633 of file interface.c.

```
01635
01636
        GtkFileChooserDialog *dlg;
01637
        GMainLoop *loop;
01638
        const char *buffer;
01639
        unsigned int i;
01640 #if DEBUG_INTERFACE
        fprintf (stderr, "window_template_experiment: start\n");
01641
01642 #endif
01643
       i = (size t) data;
        buffer = gtk_button_get_label (window->button_template[i]);
01645
       dlg = (GtkFileChooserDialog *)
          gtk_file_chooser_dialog_new (_("Open template file"),
01646
01647
                                        window->window.
01648
                                        GTK FILE CHOOSER ACTION OPEN,
01649
                                         _("_Cancel"),
01650
                                        GTK_RESPONSE_CANCEL,
       __("_open"), GTK_RESPONSE_ACCEPT, NULL);
gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (dlg), buffer);
01652
       g_signal_connect (dlg, "response",
01653
                          G_CALLBACK (window_template_experiment_close), data);
01654
       gtk_window_present (GTK_WINDOW (dlg));
01655
01656
        loop = g_main_loop_new (NULL, 0);
01657
       g_signal_connect_swapped (dlg, "destroy", G_CALLBACK (g_main_loop_quit),
01658
                                   loop);
01659
       g_main_loop_run (loop);
01660
        g_main_loop_unref (loop);
01661 #if DEBUG_INTERFACE
       fprintf (stderr, "window_template_experiment: end\n");
01662
01663 #endif
01664 }
```

Here is the call graph for this function:

4.11.3.46 window_template_experiment_close()

Function to close the experiment template dialog.

Parameters

dlg	GtkFileChooserDialg struct.
response← _id	Response identifier.
data	Function data.

Definition at line 1593 of file interface.c.

```
01598
01599
        GFile *file1, *file2;
01600
        char *buffer1, *buffer2;
01601
        unsigned int i, j;
01602 #if DEBUG_INTERFACE
01603
        fprintf (stderr, "window_template_experiment_close: start\n");
01604 #endif
01605
        if (response_id == GTK_RESPONSE OK)
01606
            i = (size_t) data;
01607
01608
             j = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01609
             buffer1 = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
01610
             file1 = g_file_new_for_path (buffer1);
            file2 = g_file_new_for_path (input->directory);
buffer2 = g_file_get_relative_path (file2, file1);
01611
01612
01613
            if (input->type == INPUT_TYPE_XML)
01614
              input->experiment[j].stencil[i]
01615
                 = (char *) xmlStrdup ((xmlChar *) buffer2);
01616
              input->experiment[j].stencil[i] = g_strdup (buffer2);
01617
01618
            a free (buffer2);
            g_object_unref (file2);
01619
            g_object_unref (file1);
            g_free (buffer1);
01621
01622
01623
       gtk_window_destroy (GTK_WINDOW (dlg));
01624 #if DEBUG_INTERFACE
01625
       fprintf (stderr, "window_template_experiment_close: end\n");
01626 #endif
01627 }
```

4.11.3.47 window_update()

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

Function to update the main window view.

Definition at line 1187 of file interface.c.

```
01188 {
        unsigned int i;
01190 #if DEBUG_INTERFACE
01191
       fprintf (stderr, "window_update: start\n");
01192 #endif
01193
       atk widget set sensitive
01194
          (GTK_WIDGET (window->button_evaluator),
01195
           gtk_check_button_get_active (window->check_evaluator));
01196
       gtk_widget_set_sensitive
01197
          (GTK_WIDGET (window->button_cleaner),
          gtk_check_button_get_active (window->check_cleaner));
01198
01199
       gtk_widget_hide (GTK_WIDGET (window->label_simulations));
       gtk_widget_hide (GTK_WIDGET (window->spin_simulations));
01200
       gtk_widget_hide (GTK_WIDGET (window->label_iterations));
01201
01202
       gtk_widget_hide (GTK_WIDGET (window->spin_iterations));
01203
        gtk_widget_hide (GTK_WIDGET (window->label_tolerance));
01204
        gtk_widget_hide (GTK_WIDGET (window->spin_tolerance));
       gtk_widget_hide (GTK_WIDGET (window->label_bests));
01205
01206
       gtk_widget_hide (GTK_WIDGET (window->spin_bests));
01207
       gtk_widget_hide (GTK_WIDGET (window->label_population));
01208
       gtk_widget_hide (GTK_WIDGET (window->spin_population));
01209
        gtk_widget_hide (GTK_WIDGET (window->label_generations));
01210
       gtk_widget_hide (GTK_WIDGET (window->spin_generations));
01211
       gtk_widget_hide (GTK_WIDGET (window->label_mutation));
01212
       gtk_widget_hide (GTK_WIDGET (window->spin_mutation));
01213
       gtk_widget_hide (GTK_WIDGET (window->label_reproduction));
```

```
gtk_widget_hide (GTK_WIDGET (window->spin_reproduction));
        gtk_widget_hide (GTK_WIDGET (window->label_adaptation));
01215
01216
        gtk_widget_hide (GTK_WIDGET (window->spin_adaptation));
01217
        gtk_widget_hide (GTK_WIDGET (window->label_sweeps));
01218
        gtk widget hide (GTK WIDGET (window->spin sweeps));
01219
        qtk_widget_hide (GTK_WIDGET (window->label_bits));
        gtk_widget_hide (GTK_WIDGET (window->spin_bits));
01220
01221
        gtk_widget_hide (GTK_WIDGET (window->check_climbing));
01222
        gtk_widget_hide (GTK_WIDGET (window->grid_climbing));
01223
        gtk widget hide (GTK WIDGET (window->label step));
        gtk_widget_hide (GTK_WIDGET (window->spin_step));
01224
        gtk_widget_hide (GTK_WIDGET (window->label_p));
01225
01226
        gtk_widget_hide (GTK_WIDGET (window->spin_p));
01227
        i = gtk_spin_button_get_value_as_int (window->spin_iterations);
01228
        switch (window_get_algorithm ())
01229
          case ALGORITHM MONTE CARLO:
01230
            gtk_widget_show (GTK_WIDGET (window->label_simulations));
01231
            gtk_widget_show (GTK_WIDGET (window->spin_simulations));
            gtk_widget_show (GTK_WIDGET (window->label_iterations));
01233
            gtk_widget_show (GTK_WIDGET (window->spin_iterations));
01234
01235
            if (i > 1)
01236
             {
                gtk_widget_show (GTK_WIDGET (window->label_tolerance));
01237
01238
                qtk_widget_show (GTK_WIDGET (window->spin_tolerance));
                gtk_widget_show (GTK_WIDGET (window->label_bests));
01239
01240
                gtk_widget_show (GTK_WIDGET (window->spin_bests));
01241
01242
            window_update_climbing ();
01243
            break:
01244
          case ALGORITHM_SWEEP:
01245
          case ALGORITHM_ORTHOGONAL:
01246
            gtk_widget_show (GTK_WIDGET (window->label_iterations));
01247
            gtk_widget_show (GTK_WIDGET (window->spin_iterations));
            <u>if</u> (i > 1)
01248
01249
                gtk_widget_show (GTK_WIDGET (window->label_tolerance));
01250
                gtk_widget_show (GTK_WIDGET (window->spin_tolerance));
01251
01252
                gtk_widget_show (GTK_WIDGET (window->label_bests));
01253
                gtk_widget_show (GTK_WIDGET (window->spin_bests));
01254
01255
            gtk widget show (GTK WIDGET (window->label sweeps));
            gtk_widget_show (GTK_WIDGET (window->spin_sweeps));
01256
            gtk_widget_show (GTK_WIDGET (window->check_climbing));
01257
            window_update_climbing ();
01258
01259
            break;
01260
          default:
01261
            gtk_widget_show (GTK_WIDGET (window->label_population));
            gtk_widget_show (GTK_WIDGET (window->spin_population));
01262
01263
            gtk_widget_show (GTK_WIDGET (window->label_generations));
01264
            gtk_widget_show (GTK_WIDGET (window->spin_generations));
01265
            gtk_widget_show (GTK_WIDGET (window->label_mutation));
01266
            gtk_widget_show (GTK_WIDGET (window->spin_mutation));
            gtk_widget_show (GTK_WIDGET (window->label_reproduction));
01267
            gtk_widget_show (GTK_WIDGET (window->spin_reproduction));
01268
            gtk_widget_show (GTK_WIDGET (window->label_adaptation));
01269
01270
            gtk_widget_show (GTK_WIDGET (window->spin_adaptation));
            gtk_widget_show (GTK_WIDGET (window->label_bits));
01271
01272
            gtk_widget_show (GTK_WIDGET (window->spin_bits));
01273
01274
        atk widget set sensitive
01275
          (GTK WIDGET (window->button remove experiment), input->nexperiments > 1);
        gtk_widget_set_sensitive
          (GTK_WIDGET (window->button_remove_variable), input->nvariables > 1);
01277
01278
        for (i = 0; i < input->experiment->ninputs; ++i)
01279
01280
            gtk_widget_show (GTK_WIDGET (window->check_template[i]));
            gtk_widget_show (GTK_WIDGET (window->button_template[i]));
01281
            gtk_widget_set_sensitive (GTK_WIDGET (window->check_template[i]), 0);
01282
            gtk_widget_set_sensitive (GTK_WIDGET (window->button_template[i]), 1);
01283
01284
            g_signal_handler_block
01285
              (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], 1);
01286
01287
01288
            g_signal_handler_unblock (window->button_template[i],
                                       window->id_input[i]);
01289
01290
            g_signal_handler_unblock (window->check_template[i],
01291
                                       window->id_template[i]);
01292
        if(i > 0)
01293
01294
01295
            gtk_widget_set_sensitive (GTK_WIDGET (window->check_template[i - 1]), 1);
01296
            gtk_widget_set_sensitive (GTK_WIDGET (window->button_template[i - 1]),
01297
                                       gtk_check_button_get_active
01298
                                       (window->check_template[i - 1]));
       }
if (i < MAX_NINPUTS)
01299
01300
```

```
01301
          {
01302
            gtk_widget_show (GTK_WIDGET (window->check_template[i]));
            gtk_widget_show (GTK_WIDGET (window->button_template[i]));
01303
01304
            gtk_widget_set_sensitive (GTK_WIDGET (window->check_template[i]), 1);
01305
            gtk_widget_set_sensitive
               (GTK_WIDGET (window->button_template[i]),
01306
01307
                gtk_check_button_get_active (window->check_template[i]));
01308
            g_signal_handler_block
01309
              (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);
01310
01311
            g_signal_handler_unblock (window->button_template[i],
01312
                                        window->id_input[i]);
01313
01314
            g_signal_handler_unblock (window->check_template[i],
01315
                                        window->id_template[i]);
01316
        while (++i < MAX NINPUTS)
01317
01318
          {
01319
            gtk_widget_hide (GTK_WIDGET (window->check_template[i]));
01320
            gtk_widget_hide (GTK_WIDGET (window->button_template[i]));
01321
01322
        gtk_widget_set_sensitive
01323
          (GTK_WIDGET (window->spin_minabs),
           gtk check button get active (window->check minabs));
01324
01325
        gtk_widget_set_sensitive
01326
         (GTK_WIDGET (window->spin_maxabs),
01327
           gtk_check_button_get_active (window->check_maxabs));
01328
        if (window_get_norm () == ERROR_NORM_P)
01329
            gtk_widget_show (GTK_WIDGET (window->label_p));
01330
01331
            gtk_widget_show (GTK_WIDGET (window->spin_p));
01332
01333 #if DEBUG_INTERFACE
01334
       fprintf (stderr, "window_update: end\n");
01335 #endif
01336 }
```

4.11.3.48 window update climbing()

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

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

Definition at line 1156 of file interface.c.

```
01157
01158 #if DEBUG INTERFACE
01159
       fprintf (stderr, "window_update_climbing: start\n");
01160 #endif
       gtk_widget_show (GTK_WIDGET (window->check_climbing));
01161
01162
           (gtk_check_button_get_active (window->check_climbing))
01163
            gtk_widget_show (GTK_WIDGET (window->grid_climbing));
01164
01165
            gtk_widget_show (GTK_WIDGET (window->label_step));
01166
            gtk_widget_show (GTK_WIDGET (window->spin_step));
01167
01168
       switch (window_get_climbing ())
01169
         case CLIMBING_METHOD_COORDINATES:
01170
          gtk_widget_hide (GTK_WIDGET (window->label_estimates));
01171
            gtk_widget_hide (GTK_WIDGET (window->spin_estimates));
01172
01173
            break;
01174
         default:
           gtk_widget_show (GTK_WIDGET (window->label_estimates));
01175
           gtk_widget_show (GTK_WIDGET (window->spin_estimates));
01176
01177
01178 #if DEBUG_INTERFACE
       fprintf (stderr, "window_update_climbing: end\n");
01179
01180 #endif
01181 }
```

Here is the call graph for this function:

4.11.3.49 window_update_variable()

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

Function to update the variable data in the main window.

Definition at line 1944 of file interface.c.

```
01945 {
01946
01947 #if DEBUG_INTERFACE
       fprintf (stderr, "window_update_variable: start\n");
01948
01949 #endif
01950
       i = gtk combo box get active (GTK COMBO BOX (window->combo variable));
01951
01952
         i = 0;
01953
       switch (window_get_algorithm ())
01954
01955
         case ALGORITHM SWEEP:
01956
         case ALGORITHM_ORTHOGONAL:
01957
           input->variable[i].nsweeps
01958
              = gtk_spin_button_get_value_as_int (window->spin_sweeps);
01959 #if DEBUG_INTERFACE
01960
         fprintf (stderr, "window_update_variable: nsweeps[%d]=%u\n", i,
01961
                    input->variable[i].nsweeps);
01962 #endif
01963
       break;
01964
         case ALGORITHM_GENETIC:
         input->variable[i].nbits
01965
01966 = gtk_spin_button_get_value_as_int (window->spin_bits);
01967 #if DEBUG_INTERFACE
           fprintf (stderr, "window_update_variable: nbits[%d]=%u\n", i,
01968
01969
                    input->variable[i].nbits);
01970 #endif
01971
01972 #if DEBUG_INTERFACE
01973 fprintf (stderr, "window_update_variable: end\n");
01974 #endif
01975 }
```

Here is the call graph for this function:

4.11.3.50 window weight experiment()

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

Function to update the experiment weight in the main window.

Definition at line 1554 of file interface.c.

```
01555 {
01556    unsigned int i;
01557    #if DEBUG_INTERFACE
01558    fprintf (stderr, "window_weight_experiment: start\n");
01559    #endif
01560    i = gtk_combo_box_get_active (GTK_COMBo_BOX (window->combo_experiment));
01561    input->experiment[i].weight = gtk_spin_button_get_value (window->spin_weight);
01562    #if DEBUG_INTERFACE
01563    fprintf (stderr, "window_weight_experiment: end\n");
01564    #endif
01565 }
```

4.11.4 Variable Documentation

4.11.4.1 logo

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

Logo pixmap.

Definition at line 84 of file interface.c.

4.11.4.2 options

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

Options struct to define the options dialog.

Definition at line 163 of file interface.c.

4.11.4.3 running

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

Running struct to define the running dialog.

Definition at line 165 of file interface.c.

4.11.4.4 window

```
Window window[1]
```

Window struct to define the main interface window.

Definition at line 81 of file interface.c.

4.12 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.
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, 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/xml.h"
00059 #include "jb/src/json.h"
00060 #include "jb/src/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 #00000000FFFF",
00128 "X
            c #FFFF00000000",
00129 "
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
00143
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
00180
            if (input->relaxation != DEFAULT_RELAXATION)
00181
00182
              jb_xml_node_set_float (node, (const xmlChar *) LABEL_RELAXATION,
                                       input->relaxation);
00183
            switch (input->climbing)
00184
00185
              {
```

```
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;
00219 #if DEBUG_INTERFACE
00210 fprintf (stderr, "input_save_climbing_json: start\n");
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);
               break;
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
00245 #if DEBUG_INTERFACE
00246 fprintf (stderr, "input_save_xml: start\n");
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
        g object unref (file2);
00266
        xmlSetProp (node, (const xmlChar *) LABEL_SIMULATOR, (xmlChar *) buffer);
00267
        g_free (buffer);
00268
        if (input->evaluator)
00269
           file2 = g_file_new_for_path (input->evaluator);
00270
00271
            buffer = g_file_get_relative_path (file, file2);
00272
            g_object_unref (file2);
            if (xmlStrlen ((xmlChar *) buffer))
00273
00274
              xmlSetProp (node, (const xmlChar *) LABEL_EVALUATOR,
00275
                           (xmlChar *) buffer);
00276
            g_free (buffer);
00277
00278
       if (input->cleaner)
```

```
00280
            file2 = g_file_new_for_path (input->cleaner);
00281
            buffer = g_file_get_relative_path (file, file2);
            g_object_unref (file2);
00282
00283
            if (xmlStrlen ((xmlChar *) buffer))
              xmlSetProp (node, (const xmlChar *) LABEL_CLEANER, (xmlChar *) buffer);
00284
            g_free (buffer);
00286
00287
        if (input->seed != DEFAULT_RANDOM_SEED)
00288
          jb_xml_node_set_uint (node, (const xmlChar *) LABEL_SEED, input->seed);
00289
00290
        // Setting the algorithm
00291
        buffer = (char *) g_slice_alloc (64);
00292
        switch (input->algorithm)
00293
          case ALGORITHM_MONTE_CARLO:
00294
            00295
00296
00298
            xmlSetProp (node, (const xmlChar *) LABEL_NSIMULATIONS,
            (xmlChar *) buffer);
snprintf (buffer, 64, "%u", input->niterations);
00299
00300
            xmlSetProp (node, (const xmlChar *) LABEL_NITERATIONS,
00301
            (xmlChar *) buffer);
snprintf (buffer, 64, "%.3lg", input->tolerance);
00302
00303
            xmlSetProp (node, (const xmlChar *) LABEL_TOLERANCE, (xmlChar *) buffer);
00304
00305
            snprintf (buffer, 64, "%u", input->nbest);
00306
            xmlSetProp (node, (const xmlChar *) LABEL_NBEST, (xmlChar *) buffer);
00307
            input_save_climbing_xml (node);
00308
            break:
00309
          case ALGORITHM_SWEEP:
            00310
00311
00312
00313
            xmlSetProp (node, (const xmlChar *) LABEL_NITERATIONS,
            (xmlChar *) buffer);
snprintf (buffer, 64, "%.3lg", input->tolerance);
00314
00315
            xmlSetProp (node, (const xmlChar *) LABEL_TOLERANCE, (xmlChar *) buffer);
00316
00317
            snprintf (buffer, 64, "%u", input->nbest);
00318
            xmlSetProp (node, (const xmlChar *) LABEL_NBEST, (xmlChar *) buffer);
00319
            input_save_climbing_xml (node);
00320
            break;
          case ALGORITHM ORTHOGONAL:
00321
            00322
00323
00324
00325
            xmlSetProp (node, (const xmlChar *) LABEL_NITERATIONS,
            (xmlChar *) buffer);
snprintf (buffer, 64, "%.31g", input->tolerance);
00326
00327
            xmlSetProp (node, (const xmlChar *) LABEL_TOLERANCE, (xmlChar *) buffer);
00328
            snprintf (buffer, 64, "%u", input->nbest);
00329
00330
            xmlSetProp (node, (const xmlChar *) LABEL_NBEST, (xmlChar *) buffer);
00331
            input_save_climbing_xml (node);
00332
            break;
00333
          default:
00334
            xmlSetProp (node, (const xmlChar *) LABEL_ALGORITHM,
            (const xmlChar *) LABEL_GENETIC);
snprintf (buffer, 64, "%u", input->nsimulations);
00336
00337
            xmlSetProp (node, (const xmlChar *) LABEL_NPOPULATION,
           (xmlChar *) buffer);
snprintf (buffer, 64, "%u", input->niterations);
xmlSetProp (node, (const xmlChar *) LABEL_NGENERATIONS,
00338
00339
00340
00341
                        (xmlChar *) buffer);
            snprintf (buffer, 64, "%.31g", input->mutation_ratio);
00342
            xmlSetProp (node, (const xmlChar *) LABEL_MUTATION, (xmlChar *) buffer);
00343
00344
            snprintf (buffer, 64, "%.31g", input->reproduction_ratio);
00345
            xmlSetProp (node, (const xmlChar *) LABEL_REPRODUCTION,
            (xmlChar *) buffer);
snprintf (buffer, 64, "%.31g", input->adaptation_ratio);
00346
00347
            xmlSetProp (node, (const xmlChar *) LABEL_ADAPTATION, (xmlChar *) buffer);
00348
00349
00350
00351
        g_slice_free1 (64, buffer);
00352
        if (input->threshold != 0.)
          jb_xml_node_set_float (node, (const xmlChar *) LABEL_THRESHOLD,
00353
                                 input->threshold);
00354
00355
00356
        // Setting the experimental data
00357
        for (i = 0; i < input->nexperiments; ++i)
00358
00359
            child = xmlNewChild (node, 0, (const xmlChar *) LABEL_EXPERIMENT, 0);
            xmlSetProp (child, (const xmlChar *) LABEL_NAME,
00360
                        (xmlChar *) input->experiment[i].name);
00361
00362
            if (input->experiment[i].weight != 1.)
00363
             jb_xml_node_set_float (child, (const xmlChar *) LABEL_WEIGHT,
00364
                                      input->experiment[i].weight);
00365
            for (j = 0; j < input->experiment->ninputs; ++j)
```

```
xmlSetProp (child, (const xmlChar *) stencil[j],
                          (xmlChar *) input->experiment[i].stencil[j]);
00367
00368
00369
00370
        \ensuremath{//} Setting the variables data
        for (i = 0; i < input->nvariables; ++i)
00371
00372
00373
            child = xmlNewChild (node, 0, (const xmlChar *) LABEL_VARIABLE, 0);
00374
            xmlSetProp (child, (const xmlChar *) LABEL_NAME,
00375
                        (xmlChar *) input->variable[i].name);
00376
            jb_xml_node_set_float (child, (const xmlChar *) LABEL_MINIMUM,
00377
                                   input->variable[i].rangemin);
            if (input->variable[i].rangeminabs != -G_MAXDOUBLE)
00378
00379
              jb_xml_node_set_float (child, (const xmlChar *) LABEL_ABSOLUTE_MINIMUM,
00380
                                      input->variable[i].rangeminabs);
00381
            jb_xml_node_set_float (child, (const xmlChar *) LABEL_MAXIMUM,
                                   input->variable[i].rangemax);
00382
            if (input->variable[i].rangemaxabs != G_MAXDOUBLE)
00383
              jb_xml_node_set_float (child, (const xmlChar *) LABEL_ABSOLUTE_MAXIMUM,
00385
                                     input->variable[i].rangemaxabs);
00386
            if (input->variable[i].precision != DEFAULT_PRECISION)
00387
              jb_xml_node_set_uint (child, (const xmlChar *) LABEL_PRECISION,
            input->variable[i].precision);
if (input->algorithm == ALGORITHM_SWEEP
00388
00389
00390
                || input->algorithm == ALGORITHM_ORTHOGONAL)
              jb_xml_node_set_uint (child, (const xmlChar *) LABEL_NSWEEPS,
00391
                                    input->variable[i].nsweeps);
00392
            else if (input->algorithm == ALGORITHM_GENETIC)
00393
              jb_xml_node_set_uint (child, (const xmlChar *) LABEL_NBITS,
00394
00395
                                    input->variable[i].nbits);
00396
            if (input->nsteps)
00397
              jb_xml_node_set_float (child, (const xmlChar *) LABEL_STEP,
00398
                                      input->variable[i].step);
00399
00400
        // Saving the error norm
00401
00402
       switch (input->norm)
00404
         case ERROR_NORM_MAXIMUM:
00405
            xmlSetProp (node, (const xmlChar *) LABEL_NORM,
00406
                        (const xmlChar *) LABEL_MAXIMUM);
00407
           break:
          case ERROR NORM P:
00408
00409
           xmlSetProp (node, (const xmlChar *) LABEL_NORM,
                        (const xmlChar *) LABEL_P);
00410
00411
            jb_xml_node_set_float (node, (const xmlChar *) LABEL_P, input->p);
00412
          case ERROR_NORM_TAXICAB:
00413
           00414
00415
00416
         }
00417
00418 #if DEBUG_INTERFACE
00419
       fprintf (stderr, "input_save: end\n");
00420 #endif
00421 }
00422
00426 static inline void
00427 input_save_json (JsonGenerator * generator)
00428 {
       unsigned int i, j;
00429
00430
       char *buffer;
00431
       JsonNode *node, *child;
       JsonObject *object;
00432
00433
       JsonArray *array;
00434
       GFile *file, *file2;
00435
00436 #if DEBUG_INTERFACE
00437 fprintf (stderr, "input_save_json: start\n");
00438 #endif
00439
00440
        // Setting root JSON node
00441
       object = json_object_new ();
       node = json_node_new (JSON_NODE_OBJECT);
00442
        json_node_set_object (node, object);
00443
00444
        json_generator_set_root (generator, node);
00445
00446
        // Adding properties to the root JSON node
00447
        if (strcmp (input->result, result_name))
         json_object_set_string_member (object, LABEL_RESULT_FILE, input->result);
00448
          (strcmp (input->variables, variables_name))
00449
          json_object_set_string_member (object, LABEL_VARIABLES_FILE,
00450
                                         input->variables);
00451
00452
       file = g_file_new_for_path (input->directory);
       file2 = g_file_new_for_path (input->simulator);
buffer = g_file_get_relative_path (file, file2);
00453
00454
00455
       g_object_unref (file2);
```

```
json_object_set_string_member (object, LABEL_SIMULATOR, buffer);
00457
           g_free (buffer);
00458
              (input->evaluator)
00459
             {
00460
                file2 = g_file_new_for_path (input->evaluator);
00461
                buffer = g_file_get_relative_path (file, file2);
                g_object_unref (file2);
00462
00463
                    (strlen (buffer))
                   json_object_set_string_member (object, LABEL_EVALUATOR, buffer);
00464
                g_free (buffer);
00465
00466
00467
          if (input->cleaner)
00468
             {
                file2 = g_file_new_for_path (input->cleaner);
00469
00470
                buffer = g_file_get_relative_path (file, file2);
00471
                g_object_unref (file2);
00472
                if (strlen (buffer))
00473
                   json_object_set_string_member (object, LABEL_CLEANER, buffer);
                g_free (buffer);
00475
00476
              (input->seed != DEFAULT_RANDOM_SEED)
00477
             jb_json_object_set_uint (object, LABEL_SEED, input->seed);
00478
          // Setting the algorithm
buffer = (char *) g_slice_alloc (64);
00479
00480
          switch (input->algorithm)
00482
00483
             case ALGORITHM_MONTE_CARLO:
                00484
00485
                snprintf (buffer, 64, "%u", input->nsimulations);
00486
00487
                json_object_set_string_member (object, LABEL_NSIMULATIONS, buffer);
00488
                snprintf (buffer, 64, "%u", input->niterations);
00489
                json_object_set_string_member (object, LABEL_NITERATIONS, buffer);
                json_object_set_string_member (object, habel_strickflows, buffer
snprintf (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);
00490
00491
00492
00494
                 input_save_climbing_json (node);
00495
                break;
00496
              case ALGORITHM_SWEEP:
                json_object_set_string_member (object, LABEL_ALGORITHM, LABEL_SWEEP);
snprintf (buffer, 64, "%u", input->niterations);
00497
00498
                json_object_set_string_member (object, LABEL_NITERATIONS, buffer); snprintf (buffer, 64, "%.31g", input->tolerance); json_object_set_string_member (object, LABEL_TOLERANCE, buffer);
00499
00500
00501
00502
                snprintf (buffer, 64, "%u", input->nbest);
00503
                json_object_set_string_member (object, LABEL_NBEST, buffer);
00504
                input_save_climbing_json (node);
00505
                break:
             case ALGORITHM_ORTHOGONAL:
00507
                json_object_set_string_member (object, LABEL_ALGORITHM, LABEL_ORTHOGONAL);
00508
                 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);
00509
00510
00511
00513
                json_object_set_string_member (object, LABEL_NBEST, buffer);
00514
                 input_save_climbing_json (node);
00515
                break;
00516
             default:
                json_object_set_string_member (object, LABEL_ALGORITHM, LABEL_GENETIC);
snprintf (buffer, 64, "%u", input->nsimulations);
00517
00519
                json_object_set_string_member (object, LABEL_NPOPULATION, buffer);
00520
                snprintf (buffer, 64, "%u", input->niterations);
                snprintr (buffer, 64, "%u", input->niterations);
json_object_set_string_member (object, LABEL_NGENERATIONS, buffer);
snprintf (buffer, 64, "%.3lg", input->mutation_ratio);
json_object_set_string_member (object, LABEL_MUTATION, buffer);
snprintf (buffer, 64, "%.3lg", input->reproduction_ratio);
json_object_set_string_member (object, LABEL_REPRODUCTION, buffer);
snprintf (buffer, 64, "%.3lg", input->adaptation_ratio);
json_object_set_string_member (object, LABEL_ADAPTATION, buffer);
00521
00522
00523
00524
00525
00526
00527
00528
                break;
00529
00530
          g slice freel (64, buffer);
              (input->threshold != 0.)
00531
00532
             jb_json_object_set_float (object, LABEL_THRESHOLD, input->threshold);
00533
00534
           // Setting the experimental data
          array = json_array_new ();
for (i = 0; i < input->nexperiments; ++i)
00535
00536
00537
                child = json_node_new (JSON_NODE_OBJECT);
object = json_node_get_object (child);
00538
00539
00540
                json_object_set_string_member (object, LABEL_NAME,
00541
                                                            input->experiment[i].name);
00542
                if (input->experiment[i].weight != 1.)
```

```
jb_json_object_set_float (object, LABEL_WEIGHT,
00544
                                        input->experiment[i].weight);
00545
            for (j = 0; j < input->experiment->ninputs; ++j)
00546
             json_object_set_string_member (object, stencil[j],
00547
                                             input->experiment[i].stencil[j]);
00548
            json_array_add_element (array, child);
00549
00550
        json_object_set_array_member (object, LABEL_EXPERIMENTS, array);
00551
00552
       // Setting the variables data
       array = json_array_new ();
for (i = 0; i < input->nvariables; ++i)
00553
00554
00555
00556
            child = json_node_new (JSON_NODE_OBJECT);
00557
            object = json_node_get_object (child);
00558
            json_object_set_string_member (object, LABEL_NAME,
00559
                                           input->variable[i].name);
            00560
00561
00562
           if (input->variable[i].rangeminabs != -G_MAXDOUBLE)
00563
             jb_json_object_set_float (object, LABEL_ABSOLUTE_MINIMUM,
00564
                                        input->variable[i].rangeminabs);
           00565
00566
00567
            if (input->variable[i].rangemaxabs != G_MAXDOUBLE)
             jb_json_object_set_float (object, LABEL_ABSOLUTE_MAXIMUM,
00568
00569
                                        input->variable[i].rangemaxabs);
00570
           if (input->variable[i].precision != DEFAULT_PRECISION)
00571
             jb_json_object_set_uint (object, LABEL_PRECISION,
           input->variable[i].precision);
if (input->algorithm == ALGORITHM_SWEEP
00572
00573
00574
               || input->algorithm == ALGORITHM_ORTHOGONAL)
00575
              jb_json_object_set_uint (object, LABEL_NSWEEPS,
                                       input->variable[i].nsweeps);
00576
           jb_json_object_set_uint (object, LABEL_NBITS, input->variable[i].nbits);
if (input->nsteps)
            else if (input->algorithm == ALGORITHM_GENETIC)
00577
00578
00579
             jb_json_object_set_float (object, LABEL_STEP, input->variable[i].step);
00581
            json_array_add_element (array, child);
00582
        json_object_set_array_member (object, LABEL_VARIABLES, array);
00583
00584
00585
       // Saving the error norm
00586
       switch (input->norm)
00587
00588
         case ERROR_NORM_MAXIMUM:
00589
           json_object_set_string_member (object, LABEL_NORM, LABEL_MAXIMUM);
00590
           break;
00591
         case ERROR NORM P:
00592
           json_object_set_string_member (object, LABEL_NORM, LABEL_P);
00593
            jb_json_object_set_float (object, LABEL_P, input->p);
00594
00595
         case ERROR_NORM_TAXICAB:
00596
           json_object_set_string_member (object, LABEL_NORM, LABEL_TAXICAB);
00597
00598
00599 #if DEBUG_INTERFACE
       fprintf (stderr, "input_save_json: end\n");
00600
00601 #endif
00602 }
00603
00607 static inline void
00608 input_save (char *filename)
00609 {
       xmlDoc *doc;
00610
00611
       JsonGenerator *generator;
00612
00613 #if DEBUG INTERFACE
00614 fprintf (stderr, "input_save: start\n");
00615 #endif
00616
00617
        // Getting the input file directory
00618
       input->name = g_path_get_basename (filename);
00619
       input->directory = g_path_get_dirname (filename);
00620
       if (input->type == INPUT_TYPE_XML)
00621
00622
         {
           // Opening the input file
00623
           doc = xmlNewDoc ((const xmlChar *) "1.0");
00624
           input_save_xml (doc);
00625
00626
00627
            // Saving the XML file
           xmlSaveFormatFile (filename, doc, 1);
00628
00629
00630
            // Freeing memory
00631
           xmlFreeDoc (doc);
00632
```

```
00633
        else
00634
00635
            // Opening the input file
00636
            generator = json_generator_new ();
00637
             json_generator_set_pretty (generator, TRUE);
00638
            input save ison (generator);
00639
00640
             // Saving the JSON file
00641
            json_generator_to_file (generator, filename, NULL);
00642
00643
            // Freeing memory
            g_object_unref (generator);
00644
00645
00646
00647 #if DEBUG_INTERFACE
       fprintf (stderr, "input_save: end\n");
00648
00649 #endif
00650 }
00651
00655 static void
00656 dialog_options_close (GtkDialog * dlg,
00657
                             int response_id)
00658
00659 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_options_close: start\n");
00660
00661 #endif
        if (response_id == GTK_RESPONSE_OK)
00662
00663
00664
            input->seed
00665
              = (unsigned long int) gtk_spin_button_get_value (options->spin_seed);
00666
            nthreads = gtk_spin_button_get_value_as_int (options->spin_threads);
00667
            nthreads_climbing
00668
              = gtk_spin_button_get_value_as_int (options->spin_climbing);
00669
00670
        gtk_window_destroy (GTK_WINDOW (dlg));
00671 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_options_close: end\n");
00672
00673 #endif
00674 }
00675
00679 static void
00680 options_new ()
00681 (
00682 #if DEBUG_INTERFACE
        fprintf (stderr, "options_new: start\n");
00684 #endif
00685
       options->label_seed = (GtkLabel *)
00686
          gtk_label_new (_("Pseudo-random numbers generator seed"));
        options->spin_seed = (GtkSpinButton *)
00687
00688
          gtk_spin_button_new_with_range (0., (gdouble) G_MAXULONG, 1.);
00689
        gtk_widget_set_tooltip_text
00690
          (GTK_WIDGET (options->spin_seed),
00691
            _("Seed to init the pseudo-random numbers generator"));
00692
        gtk_spin_button_set_value (options->spin_seed, (gdouble) input->seed);
00693
        options->label_threads = (GtkLabel *)
00694
          gtk_label_new (_("Threads number for the stochastic algorithm"));
        options->spin_threads
00695
00696
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 64., 1.);
00697
        gtk_widget_set_tooltip_text
00698
          (GTK_WIDGET (options->spin_threads),
           _("Number of threads to perform the calibration/optimization for " \,
00699
00700
              "the stochastic algorithm"));
        gtk_spin_button_set_value (options->spin_threads, (gdouble) nthreads);
options->label_climbing = (GtkLabel *)
00701
00702
00703
          gtk_label_new (_("Threads number for the hill climbing method"));
00704
        options->spin_climbing =
00705
          (GtkSpinButton \star) gtk_spin_button_new_with_range (1., 64., 1.);
00706
        gtk_widget_set_tooltip_text
  (GTK_WIDGET (options->spin_climbing),
00707
00708
           _("Number of threads to perform the calibration/optimization for the "
00709
             "hill climbing method"));
00710
        gtk_spin_button_set_value (options->spin_climbing,
00711
                                     (gdouble) nthreads_climbing);
        options->grid = (GtkGrid *) gtk_grid_new ();
00712
        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);
00713
00714
00715
        gtk_grid_attach (options->grid, GTK_WIDGET (options->label_threads),
00716
                          0, 1, 1, 1);
00717
        gtk_grid_attach (options->grid, GTK_WIDGET (options->spin_threads),
00718
                          1, 1, 1, 1);
00719
        gtk_grid_attach (options->grid, GTK_WIDGET (options->label_climbing), 0, 2, 1,
00720
                          1);
00721
        gtk_grid_attach (options->grid, GTK_WIDGET (options->spin_climbing), 1, 2, 1,
00722
                          1);
00723 #if !GTK4
00724
       gtk_widget_show_all (GTK_WIDGET (options->grid));
00725 #else
```

```
gtk_widget_show (GTK_WIDGET (options->grid));
00727 #endif
        options->dialog = (GtkDialog *)
00728
00729
          {\tt gtk\_dialog\_new\_with\_buttons~(\_("Options"),}
00730
                                          window->window
00731
                                         GTK_DIALOG_MODAL,
00732
                                         _("_OK"), GTK_RESPONSE_OK,
00733
                                          _("_Cancel"), GTK_RESPONSE_CANCEL, NULL);
00734
       gtk_box_append (GTK_BOX (gtk_dialog_get_content_area (options->dialog)),
       GTK_WIDGET (options->grid));
g_signal_connect (options->dialog, "response",
00735
00736
                           G_CALLBACK (dialog_options_close), NULL);
00737
00738
        gtk_window_present (GTK_WINDOW (options->dialog));
00739 #if DEBUG_INTERFACE
00740
       fprintf (stderr, "options_new: end\n");
00741 #endif
00742 }
00743
00747 static inline void
00748 running_new ()
00749
00750 #if DEBUG_INTERFACE
       fprintf (stderr, "running_new: start\n");
00751
00752 #endif
00753
        running->label = (GtkLabel *) gtk_label_new (_("Calculating ..."));
00754
        running->spinner = (GtkSpinner *) gtk_spinner_new ();
00755
        running->grid = (GtkGrid *) gtk_grid_new ();
00756
        gtk_grid_attach (running->grid, GTK_WIDGET (running->label), 0, 0, 1, 1);
00757
        gtk_grid_attach (running->grid, GTK_WIDGET (running->spinner), 0, 1, 1, 1);
00758
        running->dialog = (GtkDialog *)
          gtk_dialog_new_with_buttons (_("Calculating"),
00759
00760
                                          window->window, GTK_DIALOG_MODAL, NULL, NULL);
00761
       gtk_window_set_child (GTK_WINDOW
00762
                                (gtk_dialog_get_content_area (running->dialog)),
00763
                               GTK_WIDGET (running->grid));
00764
        gtk_spinner_start (running->spinner);
00765 #if !GTK4
00766
       gtk_widget_show_all (GTK_WIDGET (running->dialog));
00767 #else
00768
       gtk_widget_show (GTK_WIDGET (running->dialog));
00769 #endif
00770 #if DEBUG INTERFACE
00771 fprintf (stderr, "running_new: end\n");
00772 #endif
00773 }
00774
00780 static unsigned int
00781 window_get_algorithm ()
00782 {
00783
        unsigned int i:
00784 #if DEBUG_INTERFACE
00785
        fprintf (stderr, "window_get_algorithm: start\n");
00786 #endif
00787
       i = jbw_array_buttons_get_active (window->button_algorithm, NALGORITHMS);
00788 #if DEBUG_INTERFACE
00789 fprintf (stderr, "window_get_algorithm: u^n, i);
00790 fprintf (stderr, "window_get_algorithm: end\n");
00791 #endif
00792
       return i;
00793 }
00794
00800 static unsigned int
00801 window_get_climbing ()
00802 {
00803
        unsigned int i;
00804 #if DEBUG_INTERFACE
       fprintf (stderr, "window_get_climbing: start\n");
00805
00806 #endif
00807 i = jbw_array_buttons_get_active (window->button_climbing, NCLIMBINGS);
00808 #if DEBUG_INTERFACE
      fprintf (stderr, "window_get_climbing: %u\n", i);
fprintf (stderr, "window_get_climbing: end\n");
00809
00810
00811 #endif
00812
       return i;
00813 }
00814
00820 static unsigned int
00821 window_get_norm ()
00822 {
00823
        unsigned int i:
00824 #if DEBUG_INTERFACE
        fprintf (stderr, "window_get_norm: start\n");
00826 #endif
00827
       i = jbw_array_buttons_get_active (window->button_norm, NNORMS);
00828 #if DEBUG_INTERFACE
00829 fprintf (stderr, "window_get_norm: %u\n", i);
00830 fprintf (stderr, "window_get_norm: end\n");
```

```
00831 #endif
00832
       return i;
00833 }
00834
00838 static void
00839 window save climbing ()
00841 #if DEBUG_INTERFACE
       fprintf (stderr, "window_save_climbing: start\n");
00842
00843 #endif
00844
       if (gtk_check_button_get_active (window->check_climbing))
00845
00846
            input->nsteps = gtk spin button get value as int (window->spin steps);
00847
            input->nfinal_steps
00848
               = gtk_spin_button_get_value_as_int (window->spin_final_steps);
00849
            input->relaxation = gtk_spin_button_get_value (window->spin_relaxation);
00850
            switch (window_get_climbing ())
00851
              {
00852
              case CLIMBING_METHOD_COORDINATES:
                input->climbing = CLIMBING_METHOD_COORDINATES;
00853
00854
                break;
00855
              default:
               input->climbing = CLIMBING_METHOD_RANDOM;
00856
                input->nestimates
00857
00858
                  = gtk_spin_button_get_value_as_int (window->spin_estimates);
00859
00860
00861
       else
00862 input->nsteps = 0;
00863 #if DEBUG_INTERFACE
00864 fprintf (stderr, "window_save_climbing: end\n");
00865 #endif
00866 }
00867
00871 static void
00872 dialog_save_close (GtkFileChooserDialog * dlg,
00874
                         int response_id)
00876
       GtkFileFilter *filter1;
00877
       char *buffer;
00878 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_save_close: start\n");
00879
00880 #endif
00881
       // If OK response then saving
           (response_id == GTK_RESPONSE_OK)
00882
        if
00883
00884
            // Setting input file type
00885
            filter1 = gtk_file_chooser_get_filter (GTK_FILE_CHOOSER (dlg));
            buffer = (char *) gtk_file_filter_get_name (filter1);
if (!strcmp (buffer, "XML"))
00886
00887
              input->type = INPUT_TYPE_XML;
00888
00889
00890
              input->type = INPUT_TYPE_JSON;
00891
00892
            // Adding properties to the root XML node
00893
            input->simulator
              = g_strdup (gtk_button_get_label (window->button_simulator));
00894
00895
            if (gtk_check_button_get_active (window->check_evaluator))
00896
             input->evaluator
00897
                = g_strdup (gtk_button_get_label (window->button_evaluator));
00898
            else
00899
              input->evaluator = NULL;
00900
            if (gtk_check_button_get_active (window->check_cleaner))
00901
             input->cleaner
00902
                 = g_strdup (gtk_button_get_label (window->button_cleaner));
00903
            else
00904
              input->cleaner = NULL;
00905
            if (input->type == INPUT_TYPE_XML)
00906
              {
00907
                input->result
00908
                   = (char *) xmlStrdup ((const xmlChar *)
00909
                                         gtk_entry_get_text (window->entry_result));
00910
                input->variables
00911
                  = (char *) xmlStrdup ((const xmlChar *)
00912
                                         gtk_entry_get_text (window->entry_variables));
00913
00914
            else
00915
00916
                input->result = g_strdup (gtk_entry_get_text (window->entry_result));
00917
                input->variables =
00918
                  g_strdup (gtk_entry_get_text (window->entry_variables));
00919
00920
00921
            // Setting the algorithm
00922
            switch (window_get_algorithm ())
00923
              {
00924
              case ALGORITHM MONTE CARLO:
```

```
input->algorithm = ALGORITHM_MONTE_CARLO;
00926
                input->nsimulations
00927
                  = gtk_spin_button_get_value_as_int (window->spin_simulations);
00928
                input->niterations
00929
                  = qtk spin button qet value as int (window->spin iterations);
00930
                input->tolerance = gtk_spin_button_get_value (window->spin_tolerance);
                input->nbest = gtk_spin_button_get_value_as_int (window->spin_bests);
00932
                window_save_climbing ();
00933
                break;
              case ALGORITHM_SWEEP:
00934
                input->algorithm = ALGORITHM_SWEEP;
00935
00936
                input->niterations
00937
                  = gtk_spin_button_get_value_as_int (window->spin_iterations);
00938
                input->tolerance = gtk_spin_button_get_value (window->spin_tolerance);
00939
                input->nbest = gtk_spin_button_get_value_as_int (window->spin_bests);
00940
                window_save_climbing ();
00941
                break:
00942
              case ALGORITHM_ORTHOGONAL:
                input->algorithm = ALGORITHM_ORTHOGONAL;
00944
                input->niterations
00945
                  = gtk_spin_button_get_value_as_int (window->spin_iterations);
00946
                input->tolerance = gtk_spin_button_get_value (window->spin_tolerance);
                input->nbest = gtk_spin_button_get_value_as_int (window->spin_bests);
00947
00948
                window_save_climbing ();
00949
                break;
              default:
00950
00951
                input->algorithm = ALGORITHM_GENETIC;
                input->nsimulations
00952
00953
                   = gtk_spin_button_get_value_as_int (window->spin_population);
00954
                input->niterations
00955
                  = gtk_spin_button_get_value_as_int (window->spin_generations);
00956
                input->mutation_ratio
00957
                   = gtk_spin_button_get_value (window->spin_mutation);
00958
                input->reproduction_ratio
00959
                  = gtk_spin_button_get_value (window->spin_reproduction);
00960
                input->adaptation ratio
00961
                  = gtk_spin_button_get_value (window->spin_adaptation);
00963
            input->norm = window_get_norm ();
00964
            input->p = gtk_spin_button_get_value (window->spin_p);
00965
            input->threshold = gtk_spin_button_get_value (window->spin_threshold);
00966
00967
            // Saving the XML file
00968
            buffer = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
00969
            input_save (buffer);
00970
00971
            // Closing and freeing memory
00972
            g_free (buffer);
00973
00974
        atk window destroy (GTK WINDOW (dla));
00975 #if DEBUG_INTERFACE
00976
       fprintf (stderr, "dialog_save_close: end\n");
00977 #endif
00978 }
00979
00983 static void
00984 window_save ()
00985 {
00986
        GtkFileChooserDialog *dlg;
00987
        GtkFileFilter *filter1, *filter2;
00988
       char *buffer:
00989
00990 #if DEBUG_INTERFACE
00991
       fprintf (stderr, "window_save: start\n");
00992 #endif
00993
00994
        // Opening the saving dialog
00995
        dlg = (GtkFileChooserDialog *)
00996
          gtk_file_chooser_dialog_new (_("Save file"),
00997
                                        window->window,
00998
                                        GTK_FILE_CHOOSER_ACTION_SAVE,
00999
                                        _("_Cancel"), GTK_RESPONSE_CANCEL,
                                        _("_OK"), GTK_RESPONSE_OK, NULL);
01000
01001 #if !GTK4
01002
       gtk file chooser set do overwrite confirmation (GTK FILE CHOOSER (dlg), TRUE);
01003 #endif
       buffer = g_build_filename (input->directory, input->name, NULL);
01004
01005
        gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (dlg), buffer);
        g_free (buffer);
01006
01007
01008
       // Adding XML filter
        filter1 = (GtkFileFilter *) gtk_file_filter_new ();
01010
        gtk_file_filter_set_name (filter1, "XML");
        gtk_file_filter_add_pattern (filter1, "*.xml");
gtk_file_filter_add_pattern (filter1, "*.XML");
01011
01012
        gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter1);
01013
01014
```

```
// Adding JSON filter
        filter2 = (GtkFileFilter *) gtk_file_filter_new ();
01016
01017
        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");
01018
01019
01020
01021
01022
        gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter2);
01023
        if (input->type == INPUT_TYPE_XML)
01024
          gtk_file_chooser_set_filter (GTK_FILE_CHOOSER (dlg), filter1);
01025
01026
01027
          gtk_file_chooser_set_filter (GTK_FILE_CHOOSER (dlg), filter2);
01028
01029
        \ensuremath{//} Connecting the close function
        g_signal_connect (dlg, "response", G_CALLBACK (dialog_save_close), NULL);
01030
01031
01032
        // Showing modal dialog
01033
        gtk_window_present (GTK_WINDOW (dlg));
01034
01035 #if DEBUG_INTERFACE
        fprintf (stderr, "window_save: end\n");
01036
01037 #endif
01038 }
01039
01043 static void
01044 window_run ()
01045 {
01046
        char *msg, *msg2, buffer[64], buffer2[64];
01047
        unsigned int i;
01048 #if DEBUG_INTERFACE
01049
        fprintf (stderr, "window_run: start\n");
01050 #endif
       window_save ();
01051
01052
        running_new ();
01053
        jbw_process_pending ();
01054
        optimize open ();
01055 #if DEBUG_INTERFACE
01056
        fprintf (stderr, "window_run: closing running dialog\n");
01057 #endif
01058
       gtk_spinner_stop (running->spinner);
01059
        gtk_window_destroy (GTK_WINDOW (running->dialog));
01060 #if DEBUG INTERFACE
01061
        fprintf (stderr, "window_run: displaying results\n");
01062 #endif
01063
        snprintf (buffer, 64, "error = %.15le\n", optimize->error_old[0]);
01064
        msg2 = g\_strdup (buffer);
01065
        for (i = 0; i < optimize->nvariables; ++i, msg2 = msg)
01066
            snprintf (buffer, 64, "%s = %sn",
01067
01068
                        input->variable[i].name, format[input->variable[i].precision]);
01069
             snprintf (buffer2, 64, buffer, optimize->value_old[i]);
01070
            msg = g_strconcat (msg2, buffer2, NULL);
01071
            g_free (msg2);
01072
01073
        snprintf (buffer, 64, "%s = %.61g s", _("Calculation time"),
01074
                   optimize->calculation_time);
01075
       msg = g_strconcat (msg2, buffer, NULL);
01076
        g_free (msg2);
01077
         jbw_show_message_gtk (_("Best result"), msg, INFO_TYPE);
01078 g_free (msg);
01079 #if DEBUG_INTERFACE
01080
        fprintf (stderr, "window_run: freeing memory\n");
01081 #endif
01082
        optimize_free ();
01083 #if DEBUG_INTERFACE
       fprintf (stderr, "window_run: end\n");
01084
01085 #endif
01086 }
01087
01091 static void
01092 window_help ()
01093 {
01094
        char *buffer, *buffer2;
01095 #if DEBUG_INTERFACE
        fprintf (stderr, "window_help: start\n");
01096
01097 #endif
01098 buffer2 = g_build_filename (window->application_directory, "..", "manuals",
01099
                                        ("user-manual.pdf"), NULL);
       buffer = g_filename_to_uri (buffer2, NULL, NULL);
01100
        g_free (buffer2);
01101
01102 #if GTK4
01103
        gtk_show_uri (window->window, buffer, GDK_CURRENT_TIME);
01104 #else
01105
       gtk_show_uri_on_window (window->window, buffer, GDK_CURRENT_TIME, NULL);
01106 #endif
01107 #if DEBUG_INTERFACE
```

```
fprintf (stderr, "window_help: uri=%s\n", buffer);
01109 #endif
01110
       g_free (buffer);
01111 #if DEBUG_INTERFACE
       fprintf (stderr, "window_help: end\n");
01112
01113 #endif
01114 }
01115
01119 static void
01120 window_about ()
01121 {
01122
        static const gchar *authors[] = {
           "Javier Burguete Tolosa <jburguete@eead.csic.es>",
01123
          "Borja Latorre Garcés <borja.latorre@csic.es>",
01124
01125
          NULL
01126
01127 #if DEBUG_INTERFACE
       fprintf (stderr, "window_about: start\n");
01128
01129 #endif
01130
       gtk_show_about_dialog
         (window->window,
  "program_name", "MPCOTool",
01131
01132
           "comments",
01133
           _("The Multi-Purposes Calibration and Optimization Tool.\n"
01134
              "A software to perform calibrations or optimizations of empirical "
01135
            "parameters"),
01136
01137
           "authors", authors,
01138
           "translator-credits",
           "Javier Burguete Tolosa <jburguete@eead.csic.es> "
01139
           "(english, french and spanish)\n"
"Uğur Çayoğlu (german)",
"version", "4.12.0",
"copyright", "Copyright 2012-2023 Javier Burguete Tolosa",
01140
01141
01142
01143
01144
           "logo", window->logo,
           "website", "https://github.com/jburguete/mpcotool", "license-type", GTK_LICENSE_BSD, NULL);
01145
01146
01147 #if DEBUG_INTERFACE
01148 fprintf (stderr, "window_about: end\n");
01149 #endif
01150 }
01151
01155 static void
01156 window_update_climbing ()
01157 {
01158 #if DEBUG_INTERFACE
       fprintf (stderr, "window_update_climbing: start\n");
01159
01160 #endif
        gtk_widget_show (GTK_WIDGET (window->check_climbing));
01161
01162
        if (gtk_check_button_get_active (window->check_climbing))
01163
            gtk_widget_show (GTK_WIDGET (window->grid_climbing));
01164
01165
            gtk_widget_show (GTK_WIDGET (window->label_step));
01166
            gtk_widget_show (GTK_WIDGET (window->spin_step));
01167
01168
        switch (window_get_climbing ())
01169
         case CLIMBING_METHOD_COORDINATES:
01171
          gtk_widget_hide (GTK_WIDGET (window->label_estimates));
01172
            gtk_widget_hide (GTK_WIDGET (window->spin_estimates));
            break;
01173
01174
          default:
           gtk_widget_show (GTK_WIDGET (window->label_estimates));
01175
01176
            gtk_widget_show (GTK_WIDGET (window->spin_estimates));
01177
01178 #if DEBUG_INTERFACE
01179 fprintf (stderr, "window_update_climbing: end\n");
01180 #endif
01181 }
01182
01186 static void
01187 window_update ()
01188 {
01189
        unsigned int i;
01190 #if DEBUG INTERFACE
       fprintf (stderr, "window_update: start\n");
01191
01192 #endif
01193
       gtk_widget_set_sensitive
01194
        (GTK_WIDGET (window->button_evaluator),
01195
           gtk_check_button_get_active (window->check_evaluator));
01196
       gtk_widget_set_sensitive
         (GTK WIDGET (window->button cleaner),
01197
01198
           gtk_check_button_get_active (window->check_cleaner));
01199
        gtk_widget_hide (GTK_WIDGET (window->label_simulations));
01200
        gtk_widget_hide (GTK_WIDGET (window->spin_simulations));
01201
        gtk_widget_hide (GTK_WIDGET (window->label_iterations));
        gtk widget hide (GTK WIDGET (window->spin iterations));
01202
01203
       gtk_widget_hide (GTK_WIDGET (window->label_tolerance));
```

```
gtk_widget_hide (GTK_WIDGET (window->spin_tolerance));
        gtk_widget_hide (GTK_WIDGET (window->label_bests));
01205
01206
        gtk_widget_hide (GTK_WIDGET (window->spin_bests));
01207
        gtk_widget_hide (GTK_WIDGET (window->label_population));
01208
       gtk_widget_hide (GTK_WIDGET (window->spin_population));
01209
       gtk_widget_hide (GTK_WIDGET (window->label_generations));
01210
       gtk_widget_hide (GTK_WIDGET (window->spin_generations));
       gtk_widget_hide (GTK_WIDGET (window->label_mutation));
01211
01212
       gtk_widget_hide (GTK_WIDGET (window->spin_mutation));
01213
        gtk_widget_hide (GTK_WIDGET (window->label_reproduction));
01214
       gtk_widget_hide (GTK_WIDGET (window->spin_reproduction));
01215
       gtk widget hide (GTK WIDGET (window->label adaptation));
01216
       gtk_widget_hide (GTK_WIDGET (window->spin_adaptation));
01217
       gtk_widget_hide (GTK_WIDGET (window->label_sweeps));
01218
        gtk_widget_hide (GTK_WIDGET (window->spin_sweeps));
01219
        gtk_widget_hide (GTK_WIDGET (window->label_bits));
        qtk_widget_hide (GTK_WIDGET (window->spin bits));
01220
       gtk_widget_hide (GTK_WIDGET (window->check_climbing));
01221
       gtk_widget_hide (GTK_WIDGET (window->grid_climbing));
01223
       gtk_widget_hide (GTK_WIDGET (window->label_step));
        gtk_widget_hide (GTK_WIDGET (window->spin_step));
01224
01225
        gtk_widget_hide (GTK_WIDGET (window->label_p));
01226
        gtk_widget_hide (GTK_WIDGET (window->spin_p));
01227
        i = gtk_spin_button_get_value_as_int (window->spin_iterations);
01228
       switch (window_get_algorithm ())
01229
          case ALGORITHM_MONTE_CARLO:
01230
01231
            gtk_widget_show (GTK_WIDGET (window->label_simulations));
01232
            gtk_widget_show (GTK_WIDGET (window->spin_simulations));
            gtk_widget_show (GTK_WIDGET (window->label_iterations));
01233
01234
            gtk_widget_show (GTK_WIDGET (window->spin_iterations));
01235
            <u>if</u> (i > 1)
01236
01237
                gtk_widget_show (GTK_WIDGET (window->label_tolerance));
                gtk_widget_show (GTK_WIDGET (window->spin_tolerance));
gtk_widget_show (GTK_WIDGET (window->label_bests));
01238
01239
                gtk_widget_show (GTK_WIDGET (window->spin_bests));
01240
01241
01242
            window_update_climbing ();
01243
          case ALGORITHM_SWEEP:
01244
          case ALGORITHM_ORTHOGONAL:
01245
            gtk_widget_show (GTK_WIDGET (window->label_iterations));
01246
01247
            gtk_widget_show (GTK_WIDGET (window->spin_iterations));
            if (i > 1)
01248
01249
01250
                gtk_widget_show (GTK_WIDGET (window->label_tolerance));
01251
                gtk_widget_show (GTK_WIDGET (window->spin_tolerance));
                gtk_widget_show (GTK_WIDGET (window->label_bests));
01252
01253
                gtk_widget_show (GTK_WIDGET (window->spin_bests));
01255
            gtk_widget_show (GTK_WIDGET (window->label_sweeps));
01256
            gtk_widget_show (GTK_WIDGET (window->spin_sweeps));
01257
            gtk_widget_show (GTK_WIDGET (window->check_climbing));
            window_update_climbing ();
01258
01259
            break;
01260
          default:
01261
            gtk_widget_show (GTK_WIDGET (window->label_population));
01262
            gtk_widget_show (GTK_WIDGET (window->spin_population));
01263
            gtk_widget_show (GTK_WIDGET (window->label_generations));
            gtk_widget_show (GTK_WIDGET (window->spin_generations));
01264
01265
            gtk widget show (GTK WIDGET (window->label mutation));
01266
            gtk_widget_show (GTK_WIDGET (window->spin_mutation));
            gtk_widget_show (GTK_WIDGET (window->label_reproduction));
01267
01268
            gtk_widget_show (GTK_WIDGET (window->spin_reproduction));
01269
            gtk_widget_show (GTK_WIDGET (window->label_adaptation));
01270
            gtk_widget_show (GTK_WIDGET (window->spin_adaptation));
01271
            gtk widget show (GTK WIDGET (window->label bits));
01272
            gtk_widget_show (GTK_WIDGET (window->spin_bits));
01274
        gtk_widget_set_sensitive
01275
          (GTK_WIDGET (window->button_remove_experiment), input->nexperiments > 1);
01276
        gtk_widget_set_sensitive
          (GTK WIDGET (window->button_remove_variable), input->nvariables > 1);
01277
01278
        for (i = 0; i < input->experiment->ninputs; ++i)
01279
            gtk_widget_show (GTK_WIDGET (window->check_template[i]));
01280
01281
            gtk_widget_show (GTK_WIDGET (window->button_template[i]));
            gtk_widget_set_sensitive (GTK_WIDGET (window->check_template[i]), 0);
01282
            gtk_widget_set_sensitive (GTK_WIDGET (window->button_template[i]), 1);
01283
01284
            g_signal_handler_block
01285
              (window->check_template[i], window->id_template[i]);
            g_signal_handler_block (window->button_template[i], window->id_input[i]);
01286
01287
            gtk_check_button_set_active (window->check_template[i], 1);
01288
            g_signal_handler_unblock (window->button_template[i],
01289
                                       window->id input[i]);
01290
            q_signal_handler_unblock (window->check_template[i],
```

```
01291
                                          window->id_template[i]);
01292
        if (i > 0)
01293
01294
         {
             gtk_widget_set_sensitive (GTK_WIDGET (window->check_template[i - 1]), 1);
gtk_widget_set_sensitive (GTK_WIDGET (window->button_template[i - 1]),
01295
01296
                                          gtk_check_button_get_active
01297
01298
                                           (window->check_template[i - 1]));
01299
        if (i < MAX_NINPUTS)</pre>
01300
01301
         {
01302
             gtk_widget_show (GTK_WIDGET (window->check_template[i]));
             gtk_widget_show (GTK_WIDGET (window->button_template[i]));
01303
01304
             gtk_widget_set_sensitive (GTK_WIDGET (window->check_template[i]), 1);
01305
             gtk_widget_set_sensitive
01306
               (GTK_WIDGET (window->button_template[i]),
                gtk_check_button_get_active (window->check_template[i]));
01307
01308
             g_signal_handler_block
01309
               (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);
01310
01311
01312
             g_signal_handler_unblock (window->button_template[i],
01313
                                          window->id_input[i]);
             g_signal_handler_unblock (window->check_template[i],
01314
01315
                                          window->id_template[i]);
01316
01317
        while (++i < MAX_NINPUTS)</pre>
01318
             gtk_widget_hide (GTK_WIDGET (window->check_template[i]));
gtk_widget_hide (GTK_WIDGET (window->button_template[i]));
01319
01320
01321
01322
        gtk widget set sensitive
01323
          (GTK_WIDGET (window->spin_minabs),
01324
            gtk_check_button_get_active (window->check_minabs));
01325
        {\tt gtk\_widget\_set\_sensitive}
01326
           (GTK_WIDGET (window->spin_maxabs),
        gtk_check_button_get_active (window->check_maxabs));
if (window_get_norm () == ERROR_NORM_P)
01327
01328
01329
         {
01330
             gtk_widget_show (GTK_WIDGET (window->label_p));
01331
             gtk_widget_show (GTK_WIDGET (window->spin_p));
01332
01333 #if DEBUG_INTERFACE
01334 fprintf (stderr, "window_update: end\n");
01335 #endif
01336 }
01337
01341 static void
01342 window_set_algorithm ()
01343 {
01344
        int i;
01345 #if DEBUG_INTERFACE
01346
        fprintf (stderr, "window_set_algorithm: start\n");
01347 #endif
01348 i = window_get_algorithm ();
01349
        switch (i)
01350
         {
          case ALGORITHM_SWEEP:
01351
01352
          case ALGORITHM_ORTHOGONAL:
01353
            i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01354
            if (i < 0)
               i = 0;
01355
01356
            gtk_spin_button_set_value (window->spin_sweeps,
                                            (gdouble) input->variable[i].nsweeps);
01357
01358
01359
          case ALGORITHM_GENETIC:
            i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01360
             if (i < 0)
01361
01362
               i = 0;
01363
             gtk_spin_button_set_value (window->spin_bits,
01364
                                            (gdouble) input->variable[i].nbits);
01365
01366 window_update ();
01367 #if DEBUG_INTERFACE
        fprintf (stderr, "window_set_algorithm: end\n");
01368
01369 #endif
01370 }
01371
01375 static void
01376 window_set_experiment ()
01377 {
01378
        unsigned int i, j;
01379
        char *buffer1;
01380 #if DEBUG_INTERFACE
01381
        fprintf (stderr, "window_set_experiment: start\n");
01382 #endif
01383
        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);
01385
01386
         gtk_button_set_label (window->button_experiment, buffer1);
01387
         g_free (buffer1);
01388
         for (j = 0; j < input->experiment->ninputs; ++j)
01389
01390
             g_signal_handler_block (window->button_template[j], window->id_input[j]);
             gtk_button_set_label (window->button_template[j],
01391
01392
                                      input->experiment[i].stencil[j]);
01393
             g_signal_handler_unblock
01394
               (window->button_template[j], window->id_input[j]);
01395
01396 #if DEBUG_INTERFACE
        fprintf (stderr, "window_set_experiment: end\n");
01397
01398 #endif
01399 }
01400
01404 static void
01405 window_remove_experiment ()
01406 {
01407
        unsigned int i, j;
01408 #if DEBUG_INTERFACE
        fprintf (stderr, "window_remove_experiment: start\n");
01409
01410 #endif
01411
        i = qtk_combo_box_qet_active (GTK_COMBO_BOX (window->combo_experiment));
        g_signal_handler_block (window->combo_experiment, window->id_experiment);
01412
        gtk_combo_box_text_remove (window->combo_experiment, i);
01413
01414
         g_signal_handler_unblock (window->combo_experiment, window->id_experiment);
01415
         experiment_free (input->experiment + i, input->type);
01416
         --input->nexperiments;
01417
        for (j = i; j < input->nexperiments; ++j)
01418
          memcpy (input->experiment + j, input->experiment + j + 1,
01419
                   sizeof (Experiment));
01420
         j = input->nexperiments - 1;
01421
        if (i > j)
01422
          i = j;
        for (j = 0; j < input->experiment->ninputs; ++j)
01423
          g_signal_handler_block (window->button_template[j], window->id_input[j]);
01425
        gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), i);
01426
        for (j = 0; j < input->experiment->ninputs; ++j)
01427
          g_signal_handler_unblock (window->button_template[j], window->id_input[j]);
01428 window_update ();
01429 #if DEBUG_INTERFACE
        fprintf (stderr, "window_remove_experiment: end\n");
01430
01431 #endif
01432 }
01433
01437 static void
01438 window_add_experiment ()
01439 {
        unsigned int i, j;
01441 #if DEBUG_INTERFACE
01442
        fprintf (stderr, "window_add_experiment: start\n");
01443 #endif
01444 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
        g_signal_handler_block (window->combo_experiment, window->id_experiment);
01445
        gtk_combo_box_text_insert_text
           (window->combo_experiment, i, input->experiment[i].name);
01447
01448
         g_signal_handler_unblock (window->combo_experiment, window->id_experiment);
        input->experiment = (Experiment *) g_realloc
  (input->experiment, (input->nexperiments + 1) * sizeof (Experiment));
for (j = input->nexperiments - 1; j > i; --j)
  memcpy (input->experiment + j + 1, input->experiment + j,
01449
01450
01451
01452
                   sizeof (Experiment));
01453
        input->experiment[j + 1] weight = input->experiment[j] .weight;
input->experiment[j + 1] .ninputs = input->experiment[j] .ninputs;
01454
01455
         if (input->type == INPUT_TYPE_XML)
01456
01457
           {
01458
             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]
01460
01461
01462
                  = (char *) xmlStrdup ((xmlChar *) input->experiment[i].stencil[j]);
01463
01464
        else
01465
01466
             input->experiment[j + 1].name = g_strdup (input->experiment[j].name);
             for (j = 0; j < input->experiment->ninputs; ++j)
  input->experiment[i + 1].stencil[j]
01467
01468
01469
                 = g_strdup (input->experiment[i].stencil[j]);
01470
01471
         ++input->nexperiments;
         for (j = 0; j < input->experiment->ninputs; ++j)
01472
01473
          g_signal_handler_block (window->button_template[j], window->id_input[j]);
01474
         gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), i + 1);
         for (j = 0; j < input->experiment->ninputs; ++j)
  q_signal_handler_unblock (window->button_template[j], window->id_input[j]);
01475
01476
```

```
window_update ();
01478 #if DEBUG_INTERFACE
       fprintf (stderr, "window_add_experiment: end\n");
01479
01480 #endif
01481 }
01482
01486 static void
01487 dialog_name_experiment_close (GtkFileChooserDialog * dlg,
01489
                                    int response_id,
01490
                                    void *data)
01491 {
01492
      char *buffer;
01493
       unsigned int i;
01494 #if DEBUG_INTERFACE
01495
       fprintf (stderr, "window_name_experiment_close: start\n");
01496 #endif
01497
       i = (size_t) data;
       if (response_id == GTK_RESPONSE_OK)
01498
01499
01500
            buffer = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
01501
            g_signal_handler_block (window->combo_experiment, window->id_experiment);
01502
            gtk_combo_box_text_remove (window->combo_experiment, i);
            gtk_combo_box_text_insert_text (window->combo_experiment, i, buffer); gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), i);
01503
01504
01505
            g_signal_handler_unblock (window->combo_experiment,
01506
                                      window->id_experiment);
01507
            g_free (buffer);
01508
01509 #if DEBUG_INTERFACE
       fprintf (stderr, "window_name_experiment_close: end\n");
01510
01511 #endif
01512 }
01513
01517 static void
01518 window_name_experiment ()
01519 {
01520
       GtkFileChooserDialog *dlg;
01521
       GMainLoop *loop;
       const char *buffer;
01522
01523
       unsigned int i;
01524 #if DEBUG_INTERFACE
       fprintf (stderr, "window_name_experiment: start\n");
01525
01526 #endif
01527
       i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01528
        buffer = gtk_button_get_label (window->button_experiment);
01529
        dlg = (GtkFileChooserDialog *)
01530
         gtk_file_chooser_dialog_new (_("Open experiment file"),
01531
                                       window->window
                                       GTK_FILE_CHOOSER_ACTION_OPEN,
01532
01533
                                        _("_Cancel"),
                                       GTK_RESPONSE_CANCEL,
01534
01535
                                       _("_Open"), GTK_RESPONSE_OK, NULL);
01536
        gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (dlg), buffer);
       01537
01538
01539
       gtk_window_present (GTK_WINDOW (dlg));
01540
       loop = g_main_loop_new (NULL, 0);
01541
       g_signal_connect_swapped (dlg, "destroy", G_CALLBACK (g_main_loop_quit),
01542
                                  loop);
01543
       g_main_loop_run (loop);
01544    g_main_loop_unref (loop);
01545 #if DEBUG_INTERFACE
01546
       fprintf (stderr, "window_name_experiment: end\n");
01547 #endif
01548 }
01549
01553 static void
01554 window_weight_experiment ()
01555 {
        unsigned int i;
01557 #if DEBUG_INTERFACE
       fprintf (stderr, "window_weight_experiment: start\n");
01558
01559 #endif
01560 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01561
        input->experiment[i].weight = gtk_spin_button_get_value (window->spin_weight);
01562 #if DEBUG_INTERFACE
       fprintf (stderr, "window_weight_experiment: end\n");
01563
01564 #endif
01565 }
01566
01570 static void
01571 window_inputs_experiment ()
01572 {
01573
       unsigned int j;
01574 #if DEBUG_INTERFACE
01575 fprintf (stderr, "window_inputs_experiment: startn");
01576 #endif
```

```
j = input->experiment->ninputs - 1;
01578
        if (j && !gtk_check_button_get_active (window->check_template[j]))
01579
          --input->experiment->ninputs;
        if (input->experiment->ninputs < MAX_NINPUTS
    && gtk_check_button_get_active (window->check_template[j]))
01580
01581
          ++input->experiment->ninputs;
01582
       window_update ();
01584 #if DEBUG_INTERFACE
01585
       fprintf (stderr, "window_inputs_experiment: end\n");
01586 #endif
01587 }
01588
01592 static void
01593 window_template_experiment_close (GtkFileChooserDialog * dlg,
01595
                                          int response_id,
01597
                                          void *data)
01598 {
01599
       GFile *file1, *file2;
       char *buffer1, *buffer2;
01600
01601
        unsigned int i, j;
01602 #if DEBUG_INTERFACE
       fprintf (stderr, "window_template_experiment_close: start\n");
01603
01604 #endif
       if (response_id == GTK_RESPONSE_OK)
01605
01606
          {
            i = (size_t) data;
01607
01608
            j = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_experiment));
01609
            buffer1 = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
01610
            file1 = g_file_new_for_path (buffer1);
            file2 = g_file_new_for_path (input->directory);
buffer2 = g_file_get_relative_path (file2, file1);
01611
01612
01613
            if (input->type == INPUT_TYPE_XML)
01614
              input->experiment[j].stencil[i]
01615
                 = (char *) xmlStrdup ((xmlChar *) buffer2);
            else
01616
              input->experiment[j].stencil[i] = g_strdup (buffer2);
01617
            q free (buffer2);
01618
            g_object_unref (file2);
01619
01620
            g_object_unref (file1);
01621
            g_free (buffer1);
01622
01623
       gtk_window_destroy (GTK_WINDOW (dlg));
01624 #if DEBUG_INTERFACE
01625
       fprintf (stderr, "window_template_experiment_close: end\n");
01626 #endif
01627 }
01628
01632 static void
01633 window_template_experiment (void *data)
01635 {
01636
        GtkFileChooserDialog *dlg;
01637
        GMainLoop *loop;
01638
        const char *buffer;
01639
       unsigned int i;
01640 #if DEBUG_INTERFACE
       fprintf (stderr, "window_template_experiment: start\n");
01641
01642 #endif
01643
       i = (size_t) data;
01644
        buffer = gtk_button_get_label (window->button_template[i]);
01645
        dlg = (GtkFileChooserDialog *)
          {\tt gtk\_file\_chooser\_dialog\_new~(\_("Open~template~file"),}
01646
01647
                                         window->window,
01648
                                         GTK_FILE_CHOOSER_ACTION_OPEN,
                                         _("_Cancel"),
01649
                                         GTK_RESPONSE_CANCEL,
01650
01651
                                         _("_Open"), GTK_RESPONSE_ACCEPT, NULL);
        gtk_file_chooser_set_current_name (GTK_FILE_CHOOSER (dlg), buffer);
01652
        g_signal_connect (dlg, "response",

G_CALLBACK (window_template_experiment_close), data);
01653
01654
        gtk_window_present (GTK_WINDOW (dlg));
01655
01656
        loop = g_main_loop_new (NULL, 0);
01657
        g_signal_connect_swapped (dlg, "destroy", G_CALLBACK (g_main_loop_quit),
01658
                                   loop);
        g_main_loop_run (loop);
01659
01660
        g_main_loop_unref (loop);
01661 #if DEBUG_INTERFACE
       fprintf (stderr, "window_template_experiment: end\n");
01662
01663 #endif
01664 }
01665
01669 static void
01670 window_set_variable ()
01671 {
01672
       unsigned int i;
01673 #if DEBUG_INTERFACE
       fprintf (stderr, "window_set_variable: start\n");
01674
01675 #endif
```

```
i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
        g_signal_handler_block (window->entry_variable, window->id_variable_label);
01677
01678
        gtk_entry_set_text (window->entry_variable, input->variable[i].name);
        g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
01679
01680
        gtk_spin_button_set_value (window->spin_min, input->variable[i].rangemin);
qtk_spin_button_set_value (window->spin_max, input->variable[i].rangemax);
01681
        if (input->variable[i].rangeminabs != -G_MAXDOUBLE)
01682
01683
01684
            gtk_spin_button_set_value (window->spin_minabs,
01685
                                         input->variable[i].rangeminabs);
            gtk_check_button_set_active (window->check_minabs, 1);
01686
01687
01688
        else
01689
         {
01690
            gtk_spin_button_set_value (window->spin_minabs, -G_MAXDOUBLE);
01691
            gtk_check_button_set_active (window->check_minabs, 0);
01692
01693
        if (input->variable[i].rangemaxabs != G MAXDOUBLE)
01694
01695
            gtk_spin_button_set_value (window->spin_maxabs,
                                         input->variable[i].rangemaxabs);
01696
01697
            gtk_check_button_set_active (window->check_maxabs, 1);
01698
          }
01699
        else
01700
         {
01701
            gtk_spin_button_set_value (window->spin_maxabs, G_MAXDOUBLE);
01702
            gtk_check_button_set_active (window->check_maxabs, 0);
01703
01704
        gtk_spin_button_set_value (window->spin_precision,
                                     input->variable[i].precision);
01705
        gtk_spin_button_set_value (window->spin_steps, (gdouble) input->nsteps);
gtk_spin_button_set_value (window->spin_final_steps,
01706
01707
01708
                                    (gdouble) input->nfinal_steps);
        if (input->nsteps)
01709
01710     gtk_spin_button_set_value (window->spin_step, input->variable[i].step);
01711 #if DEBUG_INTERFACE
       01712
01713
01714 #endif
01715
       switch (window_get_algorithm ())
01716
01717
          case ALGORITHM SWEEP:
         case ALGORITHM ORTHOGONAL:
01718
01719
           gtk_spin_button_set_value (window->spin_sweeps,
01720
                                         (gdouble) input->variable[i].nsweeps);
01721 #if DEBUG_INTERFACE
01722 fprintf (stderr, "window_set_variable: nsweeps[%u]=%u\n", i,
01723
                     input->variable[i].nsweeps);
01724 #endif
01725 break;
          case ALGORITHM_GENETIC:
01726
01727
          gtk_spin_button_set_value (window->spin_bits,
01728
                                         (gdouble) input->variable[i].nbits);
01729 #if DEBUG INTERFACE
           fprintf (stderr, "window_set_variable: nbits[%u]=%u\n", i,
01730
01731
                      input->variable[i].nbits);
01732 #endif
01733
           break:
01734
01735 window_update ();
01736 #if DEBUG_INTERFACE
01737 fprintf (stderr, "window_set_variable: end\n");
01738 #endif
01739 }
01740
01744 static void
01745 window_remove_variable ()
01746 {
01747
        unsigned int i, j;
01748 #if DEBUG_INTERFACE
01749
        fprintf (stderr, "window_remove_variable: start\n");
01750 #endif
01751
        i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01752
        q_signal_handler_block (window->combo_variable, window->id_variable);
        gtk_combo_box_text_remove (window->combo_variable, i);
01753
01754
        g_signal_handler_unblock (window->combo_variable, window->id_variable);
01755
        xmlFree (input->variable[i].name);
01756
        --input->nvariables;
        for (j = i; j < input->nvariables; ++j)
  memcpy (input->variable + j, input->variable + j + 1, sizeof (Variable));
01757
01758
01759
        j = input->nvariables - 1;
01760
        if (i > j)
01761
          i
            = j;
01762
        g_signal_handler_block (window->entry_variable, window->id_variable_label);
01763
        \verb|gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), i);|\\
01764
        g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
01765
        window update ():
```

```
01766 #if DEBUG_INTERFACE
       fprintf (stderr, "window_remove_variable: end\n");
01767
01768 #endif
01769 }
01770
01774 static void
01775 window_add_variable ()
01776 {
01777
        unsigned int i, j;
01778 #if DEBUG_INTERFACE
01779
        fprintf (stderr, "window_add_variable: start\n");
01780 #endif
01781
        i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01782
        g_signal_handler_block (window->combo_variable, window->id_variable);
01783
        gtk_combo_box_text_insert_text (window->combo_variable, i,
01784
                                           input->variable[i].name);
01785
        g_signal_handler_unblock (window->combo_variable, window->id_variable);
01786
        input->variable = (Variable *) g realloc
          (input->variable, (input->nvariables + 1) * sizeof (Variable));
01787
        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));
01788
01789
01790
        if (input->type == INPUT_TYPE_XML)
input->variable[j + 1].name
01791
01792
01793
            = (char *) xmlStrdup ((xmlChar *) input->variable[j].name);
01794
01795
          input->variable[j + 1].name = g_strdup (input->variable[j].name);
01796
        ++input->nvariables;
01797
        g_signal_handler_block (window->entry_variable, window->id_variable_label);
        gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), i + 1);
01798
01799
        q_signal_handler_unblock (window->entry_variable, window->id_variable_label);
01800
        window_update ();
01801 #if DEBUG_INTERFACE
01802
       fprintf (stderr, "window_add_variable: end\n");
01803 #endif
01804 }
01805
01809 static void
01810 window_label_variable ()
01811 {
01812
        unsigned int i;
        const char *buffer;
01813
01814 #if DEBUG_INTERFACE
01815
        fprintf (stderr, "window_label_variable: start\n");
01816 #endif
01817
        i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01818
       buffer = gtk_entry_get_text (window->entry_variable);
01819
        g_signal_handler_block (window->combo_variable, window->id_variable);
        gtk_combo_box_text_remove (window->combo_variable, i);
01820
        gtk_combo_box_text_insert_text (window->combo_variable, i, buffer);
01821
01822
        gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), i);
        g_signal_handler_unblock (window->combo_variable, window->id_variable);
01823
01824 #if DEBUG INTERFACE
01825
        fprintf (stderr, "window_label_variable: end\n");
01826 #endif
01827 }
01828
01832 static void
01833 window_precision_variable ()
01834 {
01835
        unsigned int i:
01836 #if DEBUG_INTERFACE
01837
        fprintf (stderr, "window_precision_variable: start\n");
01838 #endif
        i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01839
01840
       input->variable[i].precision
01841
           = (unsigned int) gtk_spin_button_get_value_as_int (window->spin_precision);
        gtk_spin_button_set_digits (window->spin_min, input->variable[i].precision);
gtk_spin_button_set_digits (window->spin_max, input->variable[i].precision);
01842
01843
01844
        gtk_spin_button_set_digits (window->spin_minabs,
01845
                                       input->variable[i].precision);
01846
        gtk_spin_button_set_digits (window->spin_maxabs
01847
                                      input->variable[i].precision);
01848 #if DEBUG_INTERFACE
       fprintf (stderr, "window_precision_variable: end\n");
01849
01850 #endif
01851 }
01852
01856 static void
01857 window_rangemin_variable ()
01858 {
01859
        unsigned int i;
01860 #if DEBUG_INTERFACE
01861
       fprintf (stderr, "window_rangemin_variable: start\n");
01862 #endif
       i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01863
01864
        input->variable[i].rangemin = qtk_spin_button_get_value (window->spin_min);
```

```
01865 #if DEBUG_INTERFACE
      fprintf (stderr, "window_rangemin_variable: end\n");
01867 #endif
01868 }
01869
01873 static void
01874 window_rangemax_variable ()
01875 {
01876
        unsigned int i;
01877 #if DEBUG_INTERFACE
        fprintf (stderr, "window_rangemax_variable: start\n");
01878
01879 #endif
01880 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01881 input->variable[i].rangemax = gtk_spin_button_get_value (window->spin_max);
01882 #if DEBUG_INTERFACE
01883
        fprintf (stderr, "window_rangemax_variable: end\n");
01884 #endif
01885 }
01886
01890 static void
01891 window_rangeminabs_variable ()
01892 {
01893
        unsigned int i;
01894 #if DEBUG_INTERFACE
01895
        fprintf (stderr, "window_rangeminabs_variable: start\n");
01896 #endif
01897 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01898 input->variable[i].rangeminabs
01899
          = gtk_spin_button_get_value (window->spin_minabs);
01900 #if DEBUG_INTERFACE
01901 fprintf (stderr, "window_rangeminabs_variable: end\n");
01902 #endif
01903 }
01904
01908 static void
01909 window_rangemaxabs_variable ()
01910 {
01911
        unsigned int i;
01912 #if DEBUG_INTERFACE
01913
        fprintf (stderr, "window_rangemaxabs_variable: start\n");
01914 #endif
01915     i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01916     input->variable[i].rangemaxabs
01917
           = gtk_spin_button_get_value (window->spin_maxabs);
01918 #if DEBUG_INTERFACE
01919
        fprintf (stderr, "window_rangemaxabs_variable: end\n");
01920 #endif
01921 }
01922
01926 static void
01927 window_step_variable ()
01928 {
01929
        unsigned int i;
01930 #if DEBUG_INTERFACE
        fprintf (stderr, "window_step_variable: start\n");
01931
01932 #endif
01933 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01934 input->variable[i].step = gtk_spin_button_get_value (window->spin_step
        input->variable[i].step = gtk_spin_button_get_value (window->spin_step);
01935 #if DEBUG_INTERFACE
        fprintf (stderr, "window_step_variable: end\n");
01936
01937 #endif
01938 }
01939
01943 static void
01944 window_update_variable ()
01945 {
01946
01947 #if DEBUG_INTERFACE
        fprintf (stderr, "window_update_variable: start\n");
01948
01949 #endif
01950 i = gtk_combo_box_get_active (GTK_COMBO_BOX (window->combo_variable));
01951
        if (i < 0)</pre>
01952
          i = 0;
        switch (window_get_algorithm ())
01953
01954
          case ALGORITHM_SWEEP:
01955
01956
          case ALGORITHM_ORTHOGONAL:
01957
            input->variable[i].nsweeps
01958
               = gtk_spin_button_get_value_as_int (window->spin_sweeps);
01959 #if DEBUG INTERFACE
            fprintf (stderr, "window_update_variable: nsweeps[%d]=%u\n", i,
01960
01961
                       input->variable[i].nsweeps);
01962 #endif
01963
             break;
01964
           case ALGORITHM_GENETIC:
01965
            input->variable[i].nbits
01966
               = gtk_spin_button_get_value_as_int (window->spin_bits);
```

```
01967 #if DEBUG_INTERFACE
01968
       fprintf (stderr, "window_update_variable: nbits[%d]=%u\n", i,
01969
                      input->variable[i].nbits);
01970 #endif
01971
01972 #if DEBUG_INTERFACE
      fprintf (stderr, "window_update_variable: end\n");
01974 #endif
01975 }
01976
01982 static int
01983 window read (char *filename)
01984 {
01985
        unsigned int i;
01986 #if DEBUG_INTERFACE
      fprintf (stderr, "window_read: start\n");
fprintf (stderr, "window_read: file name=%s\n", filename);
01987
01988
01989 #endif
01990
01991
        // Reading new input file
        input_free ();
input->result = input->variables = NULL;
01992
01993
01994
        if (!input_open (filename))
01995
01996 #if DEBUG_INTERFACE
01997
            fprintf (stderr, "window_read: end\n");
01998 #endif
01999
            return 0;
02000
02001
02002
        // Setting GTK+ widgets data
02003
        gtk_entry_set_text (window->entry_result, input->result);
02004
        gtk_entry_set_text (window->entry_variables, input->variables);
02005
        gtk_button_set_label (window->button_simulator, input->simulator);
02006
        gtk_check_button_set_active (window->check_evaluator,
02007
                                        (size_t) input->evaluator);
02008
        if (input->evaluator)
          gtk_button_set_label (window->button_evaluator, input->evaluator);
02009
02010
        gtk_check_button_set_active (window->check_cleaner, (size_t) input->cleaner);
02011
        if (input->cleaner)
02012
          gtk_button_set_label (window->button_cleaner, input->cleaner);
02013
        gtk_check_button_set_active (window->button_algorithm[input->algorithm],
02014
                                       TRUE):
02015
        switch (input->algorithm)
02016
02017
          case ALGORITHM_MONTE_CARLO:
02018
            gtk_spin_button_set_value (window->spin_simulations,
02019
                                          (gdouble) input->nsimulations);
02020
            // fallthrough
          case ALGORITHM_SWEEP:
02021
          case ALGORITHM_ORTHOGONAL:
02022
02023
            gtk_spin_button_set_value (window->spin_iterations,
02024
                                          (gdouble) input->niterations);
            gtk_spin_button_set_value (window->spin_bests, (gdouble) input->nbest);
gtk_spin_button_set_value (window->spin_tolerance, input->tolerance);
02025
02026
             gtk_check_button_set_active (window->check_climbing, input->nsteps);
02027
02028
             if (input->nsteps)
02029
02030
                {\tt gtk\_check\_button\_set\_active}
02031
                   (window->button_climbing[input->climbing], TRUE);
                 gtk_spin_button_set_value (window->spin_steps,
02032
                                              (gdouble) input->nsteps);
02033
02034
                 gtk_spin_button_set_value (window->spin_final_steps,
02035
                                              (gdouble) input->nfinal_steps);
02036
                 gtk_spin_button_set_value (window->spin_relaxation,
02037
                                              (gdouble) input->relaxation);
02038
                 switch (input->climbing)
02039
02040
                   case CLIMBING_METHOD_RANDOM:
02041
                    gtk_spin_button_set_value (window->spin_estimates,
02042
                                                   (gdouble) input->nestimates);
02043
02044
02045
            break:
02046
          default:
02047
            gtk_spin_button_set_value (window->spin_population,
                                          (gdouble) input->nsimulations);
02048
02049
            gtk_spin_button_set_value (window->spin_generations,
02050
                                          (gdouble) input->niterations);
            gtk_spin_button_set_value (window->spin_mutation, input->mutation_ratio);
gtk_spin_button_set_value (window->spin_reproduction,
02051
02052
02053
                                          input->reproduction_ratio);
02054
             gtk_spin_button_set_value (window->spin_adaptation,
02055
                                          input->adaptation_ratio);
02056
02057
        gtk_check_button_set_active (window->button_norm[input->norm], TRUE);
02058
        gtk_spin_button_set_value (window->spin_p, input->p);
```

```
gtk_spin_button_set_value (window->spin_threshold, input->threshold);
02060
        g_signal_handler_block (window->combo_experiment, window->id_experiment);
02061
        gtk_combo_box_text_remove_all (window->combo_experiment);
02062
        for (i = 0; i < input->nexperiments; ++i)
02063
         gtk_combo_box_text_append_text (window->combo_experiment,
02064
                                           input->experiment[i].name);
        g_signal_handler_unblock (window->combo_experiment, window->id_experiment);
02065
02066
        gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_experiment), 0);
02067
        g_signal_handler_block (window->combo_variable, window->id_variable);
02068
        g_signal_handler_block (window->entry_variable, window->id_variable_label);
        gtk_combo_box_text_remove_all (window->combo_variable);
02069
02070
        for (i = 0; i < input->nvariables; ++i)
02071
          gtk_combo_box_text_append_text (window->combo_variable,
02072
                                           input->variable[i].name);
02073
        g_signal_handler_unblock (window->entry_variable, window->id_variable_label);
02074
        g_signal_handler_unblock (window->combo_variable, window->id_variable);
        gtk_combo_box_set_active (GTK_COMBO_BOX (window->combo_variable), 0);
02075
02076
        window_set_variable ();
02077
       window_update ();
02078
02079 #if DEBUG_INTERFACE
02080 fprintf (stderr, "window_read: end\n");
02081 #endif
02082 return 1;
02083 }
02084
02088 static void
02089 dialog_open_close (GtkFileChooserDialog * dlg,
02091
                         int response_id)
02092 {
02093
       char *buffer, *directory, *name;
02094
       GFile *file;
02095
02096 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_open_close: start\n");
02097
02098 #endif
02099
02100
        // Saving a backup of the current input file
02101
        directory = g_strdup (input->directory);
02102
       name = g_strdup (input->name);
02103
02104
        // If OK saving
       if (response_id == GTK_RESPONSE_OK)
02105
02106
         {
02107
            // Traying to open the input file
02108
           file = gtk_file_chooser_get_file (GTK_FILE_CHOOSER (dlg));
buffer = g_file_get_path (file);
02109
02110
02111 #if DEBUG_INTERFACE
           fprintf (stderr, "dialog_open_close: file name=%s\n", buffer);
02112
02113 #endif
02114
           if (!window_read (buffer))
02115
02116 #if DEBUG_INTERFACE
                fprintf (stderr, "dialog_open_close: error reading input file\n");
02117
02118 #endif
               g_free (buffer);
02120
02121
                // Reading backup file on error
                buffer = g_build_filename (directory, name, NULL);
02122
                input->result = input->variables = NULL;
02123
02124
                if (!input_open (buffer))
02125
02126
02127
                   // Closing on backup file reading error
02128 #if DEBUG_INTERFACE
             fprintf (stderr,
02129
02130
                              "dialog_open_close: error reading backup file\n");
02131 #endif
02132
                 }
02133
02134
           g_free (buffer);
02135
           g_object_unref (file);
02136
02137
02138
       // Freeing and closing
02139
       g_free (name);
02140
       g_free (directory);
02141
       gtk_window_destroy (GTK_WINDOW (dlg));
02142
02143 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_open_close: end\n");
02144
02145 #endif
02146
02147 }
02148
02152 static void
```

```
02153 window_open ()
02154 {
02155
        GtkFileChooserDialog *dlg;
02156
        GtkFileFilter *filter;
02157
02158 #if DEBUG_INTERFACE
      fprintf (stderr, "window_open: start\n");
02159
02160 #endif
02161
02162
         // Opening dialog
02163
        dlg = (GtkFileChooserDialog *)
          gtk_file_chooser_dialog_new (_("Open input file"),
02164
02165
                                            window->window,
02166
                                            GTK_FILE_CHOOSER_ACTION_OPEN,
                                            _("_Cancel"), GTK_RESPONSE_CANCEL,
02167
                                            _("_OK"), GTK_RESPONSE_OK, NULL);
02168
02169
02170
         // Adding XML filter
        filter = (GtkFileFilter *) gtk_file_filter_new ();
02171
        gtk_file_filter_set_name (filter, "XML");
02172
        gtk_file_filter_add_pattern (filter, "*.xml");
gtk_file_filter_add_pattern (filter, "*.XML");
02173
02174
        {\tt gtk\_file\_chooser\_add\_filter~(GTK\_FILE\_CHOOSER~(dlg),~filter);}
02175
02176
02177
         // Adding JSON filter
02178
        filter = (GtkFileFilter *) gtk_file_filter_new ();
02179
        gtk_file_filter_set_name (filter, "JSON");
        gtk_file_filter_add_pattern (filter, "*.json");
gtk_file_filter_add_pattern (filter, "*.JSON");
02180
02181
        gtk_file_filter_add_pattern (filter, "*.js");
gtk_file_filter_add_pattern (filter, "*.JS");
02182
02183
02184
        gtk_file_chooser_add_filter (GTK_FILE_CHOOSER (dlg), filter);
02185
        // Connecting the close function
g_signal_connect (dlg, "response", G_CALLBACK (dialog_open_close), NULL);
02186
02187
02188
02189
        // Showing modal dialog
02190
        gtk_window_present (GTK_WINDOW (dlg));
02191
02192 #if DEBUG_INTERFACE
02193 fprintf (stderr, "window_open: end\n");
02194 #endif
02195 }
02196
02200 static void
02201 dialog_simulator_close (GtkFileChooserDialog * dlg,
02203
                                  int response_id)
02204 {
        GFile *file1, *file2;
02205
02206
        char *buffer1, *buffer2;
02207 #if DEBUG_INTERFACE
02208
        fprintf (stderr, "dialog_simulator_close: start\n");
02209 #endif
02210
        if (response id == GTK RESPONSE OK)
02211
02212
             buffer1 = gtk file chooser get current name (GTK FILE CHOOSER (dlg));
02213
             file1 = g_file_new_for_path (buffer1);
02214
             file2 = g_file_new_for_path (input->directory);
02215
             buffer2 = g_file_get_relative_path (file2, file1);
02216
             input->simulator = g_strdup (buffer2);
             q free (buffer2);
02217
             g_object_unref (file2);
g_object_unref (file1);
02218
02219
             g_free (buffer1);
02220
02221
02222
        gtk_window_destroy (GTK_WINDOW (dlg));
02223 #if DEBUG_INTERFACE
02224 fprintf (stderr, "dialog_simulator_close: end\n");
02225 #endif
02226 }
02227
02231 static void
02232 dialog_simulator ()
02233 {
02234
        GtkFileChooserDialog *dlg;
02235 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_simulator: start\n");
02236
02237 #endif
        dlg = (GtkFileChooserDialog *)
02238
          gtk_file_chooser_dialog_new (_("Open simulator file"),
02239
02240
                                            window->window,
02241
                                            GTK_FILE_CHOOSER_ACTION_OPEN,
                                           _("_Cancel"), GTK_RESPONSE_CANCEL,
_("_Open"), GTK_RESPONSE_ACCEPT, NULL);
02242
02243
        g_signal_connect (dlg, "response", G_CALLBACK (dialog_simulator_close), NULL);
02244
02245
        gtk_window_present (GTK_WINDOW (dlg));
02246 #if DEBUG_INTERFACE
```

```
fprintf (stderr, "dialog_simulator: end\n");
02248 #endif
02249 }
02250
02254 static void
02255 dialog_evaluator_close (GtkFileChooserDialog * dlg,
                                int response_id)
02258 {
02259 GFile *file1, *file2;
02260 char *buffer1, *buffer2;
02261 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_evaluator_close: start\n");
02262
02263 #endif
      if (response_id == GTK_RESPONSE_OK)
02264
02265
02266
            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);
buffer2 = g_file_get_relative_path (file2, file1);
02267
02268
02269
02270
             input->evaluator = g_strdup (buffer2);
02271
             g_free (buffer2);
02272
             g_object_unref (file2);
02273
             g_object_unref (file1);
02274
            g_free (buffer1);
02275
02276 gtk_window_destroy (GTK_WINDOW (dlg));
02277 #if DEBUG_INTERFACE
02278 fprintf (stderr, "dialog_evaluator_close: end\n");
02279 #endif
02280 }
02281
02285 static void
02286 dialog_evaluator ()
02287 {
02288
        GtkFileChooserDialog *dlg;
02289 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_evaluator: start\n");
02290
02291 #endif
02292
      dlg = (GtkFileChooserDialog *)
02293
         gtk_file_chooser_dialog_new (_("Open evaluator file"),
02294
                                          window->window.
                                          GTK FILE CHOOSER ACTION OPEN,
02295
                                          _("_Cancel"), GTK_RESPONSE_CANCEL,
_("_Open"), GTK_RESPONSE_ACCEPT, NULL);
02296
02297
       g_signal_connect (dlg, "response", G_CALLBACK (dialog_evaluator_close), NULL);
02298
02299
        gtk_window_present (GTK_WINDOW (dlg));
02300 #if DEBUG_INTERFACE
02301 fprintf (stderr, "dialog_evaluator: end\n");
02302 #endif
02303 }
02304
02308 static void
02309 dialog_cleaner_close (GtkFileChooserDialog * dlg,
02311
                              int response_id)
02312 {
02313
        GFile *file1, *file2;
02314
        char *buffer1, *buffer2;
02315 #if DEBUG_INTERFACE
02316
        fprintf (stderr, "dialog_cleaner_close: start\n");
02317 #endif
        if (response_id == GTK_RESPONSE_OK)
02318
02319
02320
            buffer1 = gtk_file_chooser_get_current_name (GTK_FILE_CHOOSER (dlg));
            file1 = g_file_new_for_path (bufferl);
file2 = g_file_new_for_path (input->directory);
02321
02322
02323
            buffer2 = g_file_get_relative_path (file2, file1);
            input->cleaner = g_strdup (buffer2);
02324
02325
            g free (buffer2);
02326
            g_object_unref (file2);
             g_object_unref (file1);
02327
02328
            g_free (buffer1);
02329
02330
        gtk_window_destroy (GTK_WINDOW (dlg));
02331 #if DEBUG_INTERFACE
        fprintf (stderr, "dialog_cleaner_close: end\n");
02332
02333 #endif
02334 }
02335
02339 static void
02340 dialog_cleaner ()
02341 {
02342
        GtkFileChooserDialog *dlg;
02343 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_cleaner: start\n");
02344
02345 #endif
02346 dlg = (GtkFileChooserDialog *)
          gtk_file_chooser_dialog_new (_("Open cleaner file"),
02347
```

```
window->window.
02349
                                        GTK_FILE_CHOOSER_ACTION_OPEN,
                                        __("_Cancel"), GTK_RESPONSE_CANCEL,
_("_Open"), GTK_RESPONSE_ACCEPT, NULL);
02350
02351
        g_signal_connect (dlg, "response", G_CALLBACK (dialog_cleaner_close), NULL);
02352
        gtk_window_present (GTK_WINDOW (dlg));
02353
02354 #if DEBUG_INTERFACE
       fprintf (stderr, "dialog_cleaner: end\n");
02355
02356 #endif
02357 }
02358
02362 void
02363 window_new (GtkApplication * application)
02364 {
02365
        unsigned int i;
        char *buffer, *buffer2, buffer3[64];
02366
        const char *label_algorithm[NALGORITHMS] = {
02367
           "_Monte-Carlo", _("_Sweep"), _("_Genetic"), _("_Orthogonal")
02368
02369
02370
        const char *tip_algorithm[NALGORITHMS] = {
02371
         _("Monte-Carlo brute force algorithm"),
02372
          _("Sweep brute force algorithm"),
         _("Genetic algorithm"),
02373
          \_("Orthogonal sampling brute force algorithm"),
02374
02375
02376
        const char *label_climbing[NCLIMBINGS] = {
02377
          _("_Coordinates climbing"), _("_Random climbing")
02378
02379
        const char *tip_climbing[NCLIMBINGS] = {
         _("Coordinates climbing estimate method"),
02380
          _("Random climbing estimate method")
02381
02382
02383
        const char *label_norm[NNORMS] = { "L2", "L", "Lp", "L1" };
02384
        const char *tip_norm[NNORMS] = {
         _("Euclidean error norm (L2)"),
02385
          _("Maximum error norm (L)"),
02386
         _("P error norm (Lp)"),
02387
          _("Taxicab error norm (L1)")
02388
02389
02390 #if !GTK4
02391
       const char *close = "delete-event";
02392 #else
       const_char *close = "close-request":
02393
02394 #endif
02395
02396 #if DEBUG_INTERFACE
02397
       fprintf (stderr, "window_new: start\n");
02398 #endif
02399
02400
        // Creating the window
02401
       window->window = window_parent = main_window
02402
          = (GtkWindow *) gtk_application_window_new (application);
02403
02404
        // Finish when closing the window
        g_signal_connect_swapped (window->window, close,
02405
                                   G_CALLBACK (g_application_quit),
02406
02407
                                   G_APPLICATION (application));
02408
02409
        \ensuremath{//} Setting the window title
        gtk_window_set_title (window->window, "MPCOTool");
02410
02411
02412
        // Creating the open button
02413
        window->button_open = (GtkButton *)
02414 #if !GTK4
02415
          gtk_button_new_from_icon_name ("document-open", GTK_ICON_SIZE_BUTTON);
02416 #else
02417
          gtk_button_new_from_icon_name ("document-open");
02418 #endif
       gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_open),
02419
02420
                                       _("Open a case"));
02421
        g_signal_connect (window->button_open, "clicked", window_open, NULL);
02422
02423
        // Creating the save button
02424
        window->button_save = (GtkButton *)
02425 #if !GTK4
02426
          gtk_button_new_from_icon_name ("document-save", GTK_ICON_SIZE_BUTTON);
02427 #else
02428
          gtk_button_new_from_icon_name ("document-save");
02429 #endif
02430
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_save),
                                      _("Save the case"));
02431
        g_signal_connect (window->button_save, "clicked", (GCallback) window_save,
02432
02433
                          NULL);
02434
02435
        // Creating the run button
02436
        window->button_run = (GtkButton *)
02437 #if !GTK4
```

```
02438
           gtk_button_new_from_icon_name ("system-run", GTK_ICON_SIZE_BUTTON);
02439 #else
02440
          gtk_button_new_from_icon_name ("system-run");
02441 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_run),
02442
02443
                                          ("Run the optimization"));
        g_signal_connect (window->button_run, "clicked", window_run, NULL);
02444
02445
02446
         // Creating the options button
02447
        window->button_options = (GtkButton *)
02448 #if !GTK4
02449
          gtk_button_new_from_icon_name ("preferences-system", GTK_ICON_SIZE_BUTTON);
02450 #else
02451
          gtk_button_new_from_icon_name ("preferences-system");
02452 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_options),
02453
02454
                                          _("Edit the case"));
        g_signal_connect (window->button_options, "clicked", options_new, NULL);
02455
02456
02457
         // Creating the help button
02458
        window->button_help = (GtkButton *)
02459 #if !GTK4
02460
           gtk_button_new_from_icon_name ("help-browser", GTK_ICON_SIZE_BUTTON);
02461 #else
02462
          gtk_button_new_from_icon_name ("help-browser");
02463 #endif
02464
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_help), _("Help"));
02465
        g_signal_connect (window->button_help, "clicked", window_help, NULL);
02466
02467
        // Creating the about button
02468
        window->button about = (GtkButton *)
02469 #if !GTK4
02470
           gtk_button_new_from_icon_name ("help-about", GTK_ICON_SIZE_BUTTON);
02471 #else
02472
           gtk_button_new_from_icon_name ("help-about");
02473 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_about), _("About"));
g_signal_connect (window->button_about, "clicked", window_about, NULL);
02474
02475
02476
02477
         // Creating the exit button
02478
        window->button_exit = (GtkButton *)
02479 #if !GTK4
          gtk_button_new_from_icon_name ("application-exit", GTK_ICON_SIZE_BUTTON);
02480
02481 #else
02482
           gtk_button_new_from_icon_name ("application-exit");
02483 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_exit), _("Exit"));
g_signal_connect_swapped (window->button_exit, "clicked",
02484
02485
                                      G_CALLBACK (g_application_quit),
G_APPLICATION (application));
02486
02487
02488
02489
         // Creating the buttons bar
02490
        window->box_buttons = (GtkBox *) gtk_box_new (GTK_ORIENTATION_HORIZONTAL, 0);
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_open));
gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_save));
02491
02492
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_run));
02493
02494
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_options));
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_help));
02495
02496
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_about));
02497
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_exit));
02498
02499
        \ensuremath{//} Creating the simulator program label and entry
        window->label_simulator = (GtkLabel *) gtk_label_new (_("Simulator program"));
window->button_simulator = (GtkButton *)
02500
02501
02502
           gtk_button_new_with_mnemonic (_("Simulator program"));
02503
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_simulator),
02504
                                          _("Simulator program executable file"));
        gtk_widget_set_hexpand (GTK_WIDGET (window->button_simulator), TRUE);
02505
        g_signal_connect (window->button_simulator, "clicked",
02506
                             G_CALLBACK (dialog_simulator), NULL);
02508
02509
        \ensuremath{//} Creating the evaluator program label and entry
02510
        window->check_evaluator = (GtkCheckButton *)
        gtk_check_button_new_with_mnemonic (_("_Evaluator program"));
g_signal_connect (window->check_evaluator, "toggled", window_update, NULL);
02511
02512
        window->button_evaluator = (GtkButton *)
02513
02514
           gtk_button_new_with_mnemonic (_("Evaluator program"));
02515
         gtk_widget_set_tooltip_text
02516
           (GTK_WIDGET (window->button_evaluator),
02517
             _("Optional evaluator program executable file"));
        g_signal_connect (window->button_evaluator, "clicked"
02518
02519
                             G_CALLBACK (dialog_evaluator), NULL);
02520
02521
         // Creating the cleaner program label and entry
02522
        window->check_cleaner = (GtkCheckButton *)
        gtk_check_button_new_with_mnemonic (_("_Cleaner program"));
g_signal_connect (window->check_cleaner, "toggled", window_update, NULL);
02523
02524
```

```
window->button_cleaner = (GtkButton *)
02526
          gtk_button_new_with_mnemonic (_("Cleaner program"));
02527
        gtk_widget_set_tooltip_text
02528
           (GTK_WIDGET (window->button_cleaner),
02529
        _("Optional cleaner program executable file"));
g_signal_connect (window->button_cleaner, "clicked"
02530
                            G_CALLBACK (dialog_cleaner), NULL);
02532
02533
        // Creating the results files labels and entries
        window->label_result = (GtkLabel *) gtk_label_new (_("Result file"));
window->entry_result = (GtkEntry *) gtk_entry_new ();
02534
02535
02536
        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 ();
02537
02538
02539
02540
        gtk_widget_set_tooltip_text
02541
           (GTK_WIDGET (window->entry_variables), _("All simulated results file"));
02542
02543
        // Creating the files grid and attaching widgets
02544
        window->grid_files = (GtkGrid *) gtk_grid_new ();
02545
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_simulator),
02546
                           0, 0, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_simulator),
02547
02548
                           1, 0, 1, 1);
02549
        qtk_grid_attach (window->grid_files, GTK_WIDGET (window->check_evaluator),
02550
                           0, 1, 1, 1);
02551
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_evaluator),
02552
                           1, 1, 1, 1);
02553
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->check_cleaner),
02554
                           0, 2, 1, 1);
02555
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_cleaner),
02556
                           1, 2, 1, 1);
02557
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_result),
02558
                           0, 3, 1, 1);
02559
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->entry_result),
02560
                           1, 3, 1, 1);
02561
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_variables),
02562
                           0, 4, 1, 1);
02563
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->entry_variables),
02564
                           1, 4, 1, 1);
02565
02566
        // Creating the algorithm properties
02567
        window->label simulations = (GtkLabel *) gtk label new
02568
           (_("Simulations number"));
02569
        window->spin_simulations
02570
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02571
        gtk_widget_set_tooltip_text
02572
           (GTK_WIDGET (window->spin_simulations),
02573
            ("Number of simulations to perform for each iteration"));
02574
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_simulations), TRUE);
        window->label_iterations = (GtkLabel *)
          gtk_label_new (_("Iterations number"));
02576
02577
        window->spin_iterations
02578
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e6, 1.);
02579
        {\tt gtk\_widget\_set\_tooltip\_text}
           (GTK_WIDGET (window->spin_iterations), _("Number of iterations"));
02580
02581
        g_signal_connect
           (window->spin_iterations, "value-changed", window_update, NULL);
02582
02583
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_iterations), TRUE);
02584
         window->label_tolerance = (GtkLabel *) gtk_label_new (_("Tolerance"));
02585
        window->spin tolerance =
02586
           (GtkSpinButton \star) gtk_spin_button_new_with_range (0., 1., 0.001);
02587
        gtk_widget_set_tooltip_text
02588
          (GTK_WIDGET (window->spin_tolerance),
02589
            _("Tolerance to set the variable interval on the next iteration"));
02590
        window->label_bests = (GtkLabel *) gtk_label_new (_("Bests number"));
02591
        window->spin bests
02592
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e6, 1.);
        gtk_widget_set_tooltip_text
  (GTK_WIDGET (window->spin_bests),
02593
02594
02595
           \_("\mbox{Number of best simulations used to set the variable interval"}
02596
              "on the next iteration"));
02597
        window->label_population
02598
           = (GtkLabel *) gtk_label_new (_("Population number"));
02599
        window->spin population
02600
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02601
        gtk_widget_set_tooltip_text
02602
           (GTK_WIDGET (window->spin_population),
        _("Number of population for the genetic algorithm"));
gtk_widget_set_hexpand (GTK_WIDGET (window->spin_population), TRUE);
02603
02604
02605
        window->label_generations
           = (GtkLabel *) gtk_label_new (_("Generations number"));
02606
        window->spin_generations
02607
02608
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e6, 1.);
02609
        gtk_widget_set_tooltip_text
02610
           (GTK_WIDGET (window->spin_generations),
02611
            _("Number of generations for the genetic algorithm"));
```

```
window->label_mutation = (GtkLabel *) gtk_label_new (_("Mutation ratio"));
02613
        window->spin_mutation
02614
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02615
        {\tt gtk\_widget\_set\_tooltip\_text}
02616
           (GTK_WIDGET (window->spin_mutation),
            _("Ratio of mutation for the genetic algorithm"));
02617
        window->label_reproduction
02618
           = (GtkLabel *) gtk_label_new (_("Reproduction ratio"));
02619
02620
        window->spin_reproduction
02621
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
        gtk_widget_set_tooltip_text
02622
02623
          (GTK_WIDGET (window->spin_reproduction),
02624
            _("Ratio of reproduction for the genetic algorithm"));
        window->label_adaptation = (GtkLabel *) gtk_label_new (_("Adaptation ratio"));
02625
02626
        window->spin_adaptation
02627
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02628
        {\tt gtk\_widget\_set\_tooltip\_text}
          02629
02630
        window->label_threshold = (GtkLabel *) gtk_label_new (_("Threshold"));
window->spin_threshold = (GtkSpinButton *)
02631
02632
02633
          gtk_spin_button_new_with_range (-G_MAXDOUBLE, G_MAXDOUBLE,
02634
                                             precision[DEFAULT_PRECISION]);
02635
        {\tt gtk\_widget\_set\_tooltip\_text}
          (GTK_WIDGET (window->spin_threshold),
02636
           _("Threshold in the objective function to finish the simulations"));
        window->scrolled_threshold = (GtkScrolledWindow *)
02638
02639 #if !GTK4
02640
          gtk_scrolled_window_new (NULL, NULL);
02641 #else
02642
          gtk_scrolled_window_new ();
02643 #endif
        gtk_scrolled_window_set_child (window->scrolled_threshold,
02644
02645
                                          GTK_WIDGET (window->spin_threshold));
          gtk_widget_set_hexpand (GTK_WIDGET (window->scrolled_threshold), TRUE);
gtk_widget_set_halign (GTK_WIDGET (window->scrolled_threshold),
02646 //
02647 //
                                          GTK ALIGN FILL);
02648 //
02649
02650
        // Creating the hill climbing method properties
        window->check_climbing = (GtkCheckButton *)
   gtk_check_button_new_with_mnemonic (_("_Hill climbing method"));
g_signal_connect (window->check_climbing, "toggled", window_update, NULL);
02651
02652
02653
02654
        window->grid_climbing = (GtkGrid *) gtk_grid_new ();
02655 #if !GTK4
        window->button_climbing[0] = (GtkRadioButton *)
02657
          gtk_radio_button_new_with_mnemonic (NULL, label_climbing[0]);
02658 #else
02659
        window->button_climbing[0] = (GtkCheckButton *)
02660
          gtk_check_button_new_with_mnemonic (label_climbing[0]);
02661 #endif
02662
        gtk_grid_attach (window->grid_climbing,
                          GTK_WIDGET (window->button_climbing[0]), 0, 0, 1, 1);
02663
02664
        g_signal_connect (window->button_climbing[0], "toggled", window_update, NULL);
02665
        for (i = 0; ++i < NCLIMBINGS;)</pre>
02666
02667 #if !GTK4
02668
            window->button_climbing[i] = (GtkRadioButton *)
               gtk_radio_button_new_with_mnemonic
02669
02670
               (gtk_radio_button_get_group (window->button_climbing[0]),
02671
                label_climbing[i]);
02672 #else
            window->button_climbing[i] = (GtkCheckButton *)
02673
02674
               gtk_check_button_new_with_mnemonic (label_climbing[i]);
             gtk_check_button_set_group (window->button_climbing[i
02675
02676
                                           window->button_climbing[0]);
02677 #endif
02678
            gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_climbing[i]),
02679
                                            tip_climbing[i]);
02680
             gtk_grid_attach (window->grid_climbing,
                               GTK_WIDGET (window->button_climbing[i]), 0, i, 1, 1);
02682
             g_signal_connect (window->button_climbing[i], "toggled", window_update,
02683
                                NULL);
02684
        window->label_steps = (GtkLabel *) gtk_label_new (_("Steps number"));
02685
        window->spin_steps = (GtkSpinButton *)
  gtk_spin_button_new_with_range (1., 1.e12, 1.);
02686
02687
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_steps), TRUE);
02688
02689
        window->label_final_steps
02690
          = (GtkLabel *) gtk_label_new (_("Final steps number"));
        window->spin_final_steps = (GtkSpinButton *)
gtk_spin_button_new_with_range (1., 1.e12, 1.);
02691
02692
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_final_steps), TRUE);
02693
        window->label_estimates
02694
02695
          = (GtkLabel *) gtk_label_new (_("Climbing estimates number"));
02696
        window->spin_estimates = (GtkSpinButton *)
          gtk_spin_button_new_with_range (1., 1.e3, 1.);
02697
02698
        window->label_relaxation
```

```
= (GtkLabel *) gtk_label_new (_("Relaxation parameter"));
02700
       window->spin_relaxation = (GtkSpinButton *)
02701
         gtk_spin_button_new_with_range (0., 2., 0.001);
       gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->label_steps),
02702
       0, NCLIMBINGS, 1, 1);
gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_steps),
02703
02704
02705
                        1, NCLIMBINGS, 1, 1);
02706
       gtk_grid_attach (window->grid_climbing,
02707
                        GTK_WIDGET (window->label_final_steps),
02708
                        0, NCLIMBINGS + 1, 1, 1);
       gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_final_steps),
02709
                        1, NCLIMBINGS + 1, 1, 1);
02710
02711
       gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->label_estimates),
02712
                        0, NCLIMBINGS + 2, 1, 1);
02713
       gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_estimates),
                        1, NCLIMBINGS + 2, 1, 1);
02714
       \tt gtk\_grid\_attach \ (window->grid\_climbing, \ GTK\_WIDGET \ (window->label\_relaxation) \ ,
02715
                        0, NCLIMBINGS + 3, 1, 1);
02716
       gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_relaxation),
02717
02718
                        1, NCLIMBINGS + 3, 1, 1);
02719
02720
       // Creating the array of algorithms
02721
       window->grid_algorithm = (GtkGrid *) gtk_grid_new ();
02722 #if !GTK4
02723
       window->button_algorithm[0] = (GtkRadioButton *)
02724
         gtk_radio_button_new_with_mnemonic (NULL, label_algorithm[0]);
02725 #else
02726
       window->button_algorithm[0] = (GtkCheckButton *)
02727
         gtk_check_button_new_with_mnemonic (label_algorithm[0]);
02728 #endif
       gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_algorithm[0]),
02729
02730
                                   tip algorithm[0]);
02731
       gtk_grid_attach (window->grid_algorithm,
02732
                        GTK_WIDGET (window->button_algorithm[0]), 0, 0, 1, 1);
02733
       g_signal_connect (window->button_algorithm[0], "toggled",
02734
                         window_set_algorithm, NULL);
02735
       for (i = 0; ++i < NALGORITHMS;)</pre>
02736
02737 #if !GTK4
02738
           window->button_algorithm[i] = (GtkRadioButton *)
02739
             gtk_radio_button_new_with_mnemonic
02740
             (gtk_radio_button_get_group (window->button_algorithm[0]),
02741
              label algorithm[i]);
02742 #else
02743
           window->button_algorithm[i] = (GtkCheckButton *)
02744
             gtk_check_button_new_with_mnemonic (label_algorithm[i]);
02745
           gtk_check_button_set_group (window->button_algorithm[i],
02746
                                       window->button_algorithm[0]),
02747 #endif
             gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_algorithm[i]),
02748
02749
                                         tip_algorithm[i]);
02750
           gtk_grid_attach (window->grid_algorithm,
02751
                            GTK_WIDGET (window->button_algorithm[i]), 0, i, 1, 1);
02752
           g_signal_connect (window->button_algorithm[i], "toggled",
02753
                             window_set_algorithm, NULL);
02754
02755
       gtk_grid_attach (window->grid_algorithm,
02756
                        GTK_WIDGET (window->label_simulations),
                        0, NALGORITHMS, 1, 1);
02757
02758
       gtk_grid_attach (window->grid_algorithm,
02759
                        GTK_WIDGET (window->spin_simulations), 1, NALGORITHMS, 1, 1);
02760
       gtk_grid_attach (window->grid_algorithm,
02761
                        GTK_WIDGET (window->label_iterations),
02762
                        0, NALGORITHMS + 1, 1, 1);
02763
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_iterations),
02764
                        1, NALGORITHMS + 1, 1, 1);
       02765
02766
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_tolerance),
02767
                        1, NALGORITHMS + 2, 1, 1);
02768
02769
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_bests),
02770
                        0, NALGORITHMS + 3, 1, 1);
       02771
02772
02773
       gtk_grid_attach (window->grid_algorithm,
02774
                        GTK_WIDGET (window->label_population),
02775
                        0, NALGORITHMS + 4, 1, 1);
02776
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_population),
02777
                        1, NALGORITHMS + 4, 1, 1);
02778
       gtk grid attach (window->grid algorithm,
02779
                        GTK_WIDGET (window->label_generations),
02780
                        0, NALGORITHMS + 5, 1, 1);
02781
       gtk_grid_attach (window->grid_algorithm,
02782
                        GTK_WIDGET (window->spin_generations),
02783
                        1, NALGORITHMS + 5, 1, 1);
       02784
02785
```

```
gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_mutation),
                          1, NALGORITHMS + 6, 1, 1);
02787
02788
        gtk_grid_attach (window->grid_algorithm,
02789
                          GTK_WIDGET (window->label_reproduction),
02790
                          0, NALGORITHMS + 7, 1, 1);
02791
        gtk_grid_attach (window->grid_algorithm,
02792
                          GTK_WIDGET (window->spin_reproduction),
02793
                          1, NALGORITHMS + 7, 1, 1);
02794
        gtk_grid_attach (window->grid_algorithm,
                          GTK_WIDGET (window->label_adaptation),
0, NALGORITHMS + 8, 1, 1);
02795
02796
02797
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_adaptation),
                          1, NALGORITHMS + 8, 1, 1);
02798
02799
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->check_climbing),
02800
                          0, NALGORITHMS + 9, 2, 1);
02801
        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),
02802
02803
                          0, NALGORITHMS + 11, 1, 1);
02804
02805
        gtk_grid_attach (window->grid_algorithm,
02806
                          GTK_WIDGET (window->scrolled_threshold),
02807
                          1, NALGORITHMS + 11, 1, 1);
02808
        window->frame_algorithm = (GtkFrame *) gtk_frame_new (_("Algorithm"));
        gtk_frame_set_child (window->frame_algorithm,
02809
02810
                              GTK_WIDGET (window->grid_algorithm));
02811
02812
        // Creating the variable widgets
02813
        window->combo_variable = (GtkComboBoxText *) gtk_combo_box_text_new ();
02814
        gtk_widget_set_tooltip_text
        (GTK_WIDGET (window->combo_variable), _("Variables selector"));
window->id_variable = g_signal_connect
02815
02816
02817
          (window->combo_variable, "changed", window_set_variable, NULL);
02818 #if !GTK4
02819
        window->button_add_variable = (GtkButton *)
02820
          gtk_button_new_from_icon_name ("list-add", GTK_ICON_SIZE_BUTTON);
02821 #else
02822
        window->button add variable = (GtkButton *)
02823
          gtk_button_new_from_icon_name ("list-add");
02824 #endif
02825
        g_signal_connect (window->button_add_variable, "clicked", window_add_variable,
02826
                           NULL);
02827
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_add_variable),
02828
                                        ("Add variable"));
02829 #if !GTK4
02830
        window->button_remove_variable = (GtkButton *)
02831
          gtk_button_new_from_icon_name ("list-remove", GTK_ICON_SIZE_BUTTON);
02832 #else
02833
        window->button\_remove\_variable = (GtkButton *)
          gtk_button_new_from_icon_name ("list-remove");
02834
02835 #endif
02836
        g_signal_connect (window->button_remove_variable, "clicked",
02837
                            window_remove_variable, NULL);
02838
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_remove_variable),
        __("Remove variable"));
window->label_variable = (GtkLabel *) gtk_label_new (_("Name"));
02839
02840
        window->entry_variable = (GtkEntry *) gtk_entry_new ();
02841
02842
        gtk_widget_set_tooltip_text
          (GTK_WIDGET (window->entry_variable), _("Variable name"));
02843
02844
        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);
window->label_min = (GtkLabel *) gtk_label_new (_("Minimum"));
02845
02846
02847
        02848
02849
02850
        gtk_widget_set_tooltip_text
        (GTK_WIDGET (window->spin_min), _("Minimum initial value of the variable")); window->scrolled_min = (GtkScrolledWindow *)
02851
02852
02853 #if !GTK4
02854
          gtk scrolled window new (NULL, NULL);
02855 #else
02856
          gtk_scrolled_window_new ();
02857 #endif
02858
        gtk_scrolled_window_set_child (window->scrolled_min,
                                         GTK_WIDGET (window->spin_min));
02859
        g_signal_connect (window->spin_min, "value-changed",
02860
                           window_rangemin_variable, NULL);
02861
        window->label_max = (GtkLabel *) gtk_label_new (_("Maximum"));
02862
02863
        window->spin_max = (GtkSpinButton *) gtk_spin_button_new_with_range
02864
          (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02865
        gtk_widget_set_tooltip_text
          (GTK WIDGET (window->spin max),
                                             ("Maximum initial value of the variable"));
02866
        window->scrolled_max = (GtkScrolledWindow *)
02867
02868 #if !GTK4
02869
          gtk_scrolled_window_new (NULL, NULL);
02870 #else
02871
         gtk_scrolled_window_new ();
02872 #endif
```

```
gtk_scrolled_window_set_child (window->scrolled_max,
02874
                                          GTK_WIDGET (window->spin_max));
02875
        g_signal_connect (window->spin_max, "value-changed",
                            window_rangemax_variable, NULL);
02876
        window->check minabs = (GtkCheckButton *)
02877
        gtk_check_button_new_with_mnemonic (_("_Absolute minimum"));
g_signal_connect (window->check_minabs, "toggled", window_update, NULL);
02878
        window->spin_minabs = (GtkSpinButton *) gtk_spin_button_new_with_range
02880
02881
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02882
        gtk_widget_set_tooltip_text
           (GTK_WIDGET (window->spin_minabs),
02883
            _("Minimum allowed value of the variable"));
02884
02885
        window->scrolled_minabs = (GtkScrolledWindow *)
02886 #if !GTK4
02887
          gtk_scrolled_window_new (NULL, NULL);
02888 #else
02889
          gtk_scrolled_window_new ();
02890 #endif
02891
        gtk_scrolled_window_set_child (window->scrolled_minabs,
02892
                                          GTK_WIDGET (window->spin_minabs));
02893
        g_signal_connect (window->spin_minabs, "value-changed",
                            window_rangeminabs_variable, NULL);
02894
02895
        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
02896
02897
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02899
02900
        gtk_widget_set_tooltip_text
02901
           (GTK_WIDGET (window->spin_maxabs),
            _("Maximum allowed value of the variable"));
02902
02903
        window->scrolled maxabs = (GtkScrolledWindow *)
02904 #if !GTK4
02905
          gtk_scrolled_window_new (NULL, NULL);
02906 #else
02907
          gtk_scrolled_window_new ();
02908 #endif
        gtk_scrolled_window_set_child (window->scrolled_maxabs,
02909
02910
                                          GTK_WIDGET (window->spin_maxabs));
02911
        g_signal_connect (window->spin_maxabs, "value-changed",
02912
                            window_rangemaxabs_variable, NULL);
        window->label_precision = (GtkLabel *) gtk_label_new (_("Precision digits"));
window->spin_precision = (GtkSpinButton *)
02913
02914
          gtk_spin_button_new_with_range (0., (gdouble) DEFAULT_PRECISION, 1.);
02915
02916
        gtk_widget_set_tooltip_text
           (GTK_WIDGET (window->spin_precision),
02917
02918
           _("Number of precision floating point digits\n"
02919
             "0 is for integer numbers"));
        g_signal_connect (window->spin_precision, "value-changed",
02920
        window_precision_variable, NULL);
window->label_sweeps = (GtkLabel *) gtk_label_new (_("Sweeps number"));
02921
02922
02923
        window->spin_sweeps =
02924
           (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02925
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->spin_sweeps),
        02926
02927
02928
        window->label_bits = (GtkLabel *) gtk_label_new (_("Bits number"));
02929
02930
        window->spin bits
02931
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 64., 1.);
02932
        {\tt gtk\_widget\_set\_tooltip\_text}
           (GTK_WIDGET (window->spin_bits),
02933
02934
            _("Number of bits to encode the variable"));
02935
        q_signal_connect
02936
           (window->spin_bits, "value-changed", window_update_variable, NULL);
02937
        window->label_step = (GtkLabel *) gtk_label_new (_("Step size"));
        window->spin_step = (GtkSpinButton *) gtk_spin_button_new_with_range
   (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02938
02939
        gtk_widget_set_tooltip_text
02940
           (GTK_WIDGET (window->spin_step),
02941
            _("Initial step size for the hill climbing method"));
02942
02943
        window->scrolled_step = (GtkScrolledWindow *)
02944 #if !GTK4
02945
          gtk_scrolled_window_new (NULL, NULL);
02946 #else
02947
          gtk scrolled window new ();
02948 #endif
02949
        gtk_scrolled_window_set_child (window->scrolled_step,
02950
                                          GTK_WIDGET (window->spin_step));
        g_signal_connect
02951
           (window->spin_step, "value-changed", window_step_variable, NULL);
02952
        window->grid_variable = (GtkGrid *) gtk_grid_new ();
02953
        gtk_grid_attach (window->grid_variable,
02954
02955
                           GTK_WIDGET (window->combo_variable), 0, 0, 2, 1);
02956
        gtk_grid_attach (window->grid_variable,
02957
                           GTK_WIDGET (window->button_add_variable), 2, 0, 1, 1);
02958
        gtk_grid_attach (window->grid_variable,
02959
                           GTK WIDGET (window->button remove variable), 3, 0, 1, 1);
```

```
gtk_grid_attach (window->grid_variable,
                         GTK_WIDGET (window->label_variable), 0, 1, 1, 1);
02961
02962
        gtk_grid_attach (window->grid_variable,
02963
                         GTK_WIDGET (window->entry_variable), 1, 1, 3, 1);
02964
       gtk_grid_attach (window->grid_variable,
                         GTK_WIDGET (window->label_min), 0, 2, 1, 1);
02965
       gtk_grid_attach (window->grid_variable,
                         GTK_WIDGET (window->scrolled_min), 1, 2, 3, 1);
02967
02968
       gtk_grid_attach (window->grid_variable,
02969
                         GTK WIDGET (window->label max), 0, 3, 1, 1);
       gtk_grid_attach (window->grid_variable,
02970
02971
                         GTK WIDGET (window->scrolled max), 1, 3, 3, 1);
02972
       gtk_grid_attach (window->grid_variable,
02973
                         GTK_WIDGET (window->check_minabs), 0, 4, 1, 1);
02974
       gtk_grid_attach (window->grid_variable,
02975
                         GTK_WIDGET (window->scrolled_minabs), 1, 4, 3, 1);
02976
       gtk_grid_attach (window->grid_variable,
02977
                         GTK WIDGET (window->check maxabs), 0, 5, 1, 1);
02978
       gtk_grid_attach (window->grid_variable,
02979
                         GTK_WIDGET (window->scrolled_maxabs), 1, 5, 3, 1);
02980
       gtk_grid_attach (window->grid_variable,
02981
                         GTK_WIDGET (window->label_precision), 0, 6, 1, 1);
02982
       gtk_grid_attach (window->grid_variable,
02983
                         GTK_WIDGET (window->spin_precision), 1, 6, 3, 1);
       gtk_grid_attach (window->grid_variable,
02984
                         GTK_WIDGET (window->label_sweeps), 0, 7, 1, 1);
02985
02986
       gtk_grid_attach (window->grid_variable,
02987
                         GTK_WIDGET (window->spin_sweeps), 1, 7, 3, 1);
02988
       gtk_grid_attach (window->grid_variable,
                         GTK_WIDGET (window->label_bits), 0, 8, 1, 1);
02989
02990
       gtk_grid_attach (window->grid_variable,
02991
                         GTK_WIDGET (window->spin_bits), 1, 8, 3, 1);
02992
       gtk_grid_attach (window->grid_variable,
02993
                         GTK_WIDGET (window->label_step), 0, 9, 1, 1);
02994
       gtk_grid_attach (window->grid_variable,
                         GTK_WIDGET (window->scrolled_step), 1, 9, 3, 1);
02995
       window->frame_variable = (GtkFrame *) gtk_frame_new (_("Variable"));
02996
       gtk_frame_set_child (window->frame_variable,
02997
02998
                             GTK_WIDGET (window->grid_variable));
02999
03000
        \ensuremath{//} Creating the experiment widgets
03001
       window->combo_experiment = (GtkComboBoxText *) gtk_combo_box_text_new ();
03002
       gtk_widget_set_tooltip_text (GTK_WIDGET (window->combo_experiment),
03003
                                      _("Experiment selector"));
03004
        window->id_experiment = g_signal_connect
03005
          (window->combo_experiment, "changed", window_set_experiment, NULL);
03006 #if !GTK4
       window->button_add_experiment = (GtkButton *)
03007
         gtk_button_new_from_icon_name ("list-add", GTK_ICON_SIZE_BUTTON);
03008
03009 #else
03010
       window->button_add_experiment = (GtkButton *)
03011
          gtk_button_new_from_icon_name ("list-add");
03012 #endif
03013
        g_signal_connect
          (window->button_add_experiment, "clicked", window_add_experiment, NULL);
03014
       gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_add_experiment),
03015
03016
                                     _("Add experiment"));
03017 #if !GTK4
       window->button_remove_experiment = (GtkButton *)
   gtk_button_new_from_icon_name ("list-remove", GTK_ICON_SIZE_BUTTON);
03018
03019
03020 #else
03021
       window->button remove experiment = (GtkButton *)
03022
         gtk_button_new_from_icon_name ("list-remove");
03023
03024
        g_signal_connect (window->button_remove_experiment, "clicked",
03025
                          window_remove_experiment, NULL);
        \verb|gtk_widget_set_tooltip_text| (\verb|GTK_WIDGET| (window->button_remove_experiment)|,
03026
                                     _("Remove experiment"));
03027
03028
       window->label experiment
          = (GtkLabel *) gtk_label_new (_("Experimental data file"));
03030
        window->button_experiment = (GtkButton *)
03031
         gtk_button_new_with_mnemonic (_("Experimental data file"));
       03032
03033
03034
                          window_name_experiment, NULL);
03035
       gtk_widget_set_hexpand (GTK_WIDGET (window->button_experiment), TRUE);
03036
03037
        window->label_weight = (GtkLabel *) gtk_label_new (_("Weight"));
03038
        window->spin weight
03039
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
       gtk_widget_set_tooltip_text
  (GTK_WIDGET (window->spin_weight),
03040
03041
03042
           _("Weight factor to build the objective function"));
03043
        g_signal_connect
03044
          (window->spin_weight, "value-changed", window_weight_experiment, NULL);
        window->grid_experiment = (GtkGrid *) gtk_grid_new ();
03045
03046
       gtk_grid_attach (window->grid_experiment,
```

```
03047
                           GTK_WIDGET (window->combo_experiment), 0, 0, 2, 1);
03048
        gtk_grid_attach (window->grid_experiment,
03049
                           GTK_WIDGET (window->button_add_experiment), 2, 0, 1, 1);
03050
        {\tt gtk\_grid\_attach~(window->grid\_experiment,}
03051
                           GTK WIDGET (window->button remove experiment), 3, 0, 1, 1);
03052
        gtk grid attach (window->grid experiment.
03053
                           GTK_WIDGET (window->label_experiment), 0, 1, 1, 1);
03054
        gtk_grid_attach (window->grid_experiment,
03055
                           GTK_WIDGET (window->button_experiment), 1, 1, 3, 1);
03056
        gtk_grid_attach (window->grid_experiment,
                           GTK_WIDGET (window->label_weight), 0, 2, 1, 1);
03057
03058
        gtk_grid_attach (window->grid_experiment,
03059
                           GTK_WIDGET (window->spin_weight), 1, 2, 3, 1);
03060
        for (i = 0; i < MAX_NINPUTS; ++i)</pre>
03061
             snprintf (buffer3, 64, "%s %u", _("Input template"), i + 1);
window->check_template[i] = (GtkCheckButton *)
03062
03063
               gtk_check_button_new_with_label (buffer3);
03064
03065
             window->id_template[i]
03066
              = g_signal_connect (window->check_template[i], "toggled",
                                    window_inputs_experiment, NULL);
03067
03068
             gtk_grid_attach (window->grid_experiment,
            \label{eq:gtk_window} $$ GTK_WIDGET (window->check_template[i]), 0, 3 + i, 1, 1); $$ window->button_template[i] = (GtkButton *) $$
03069
03070
03071
               gtk_button_new_with_mnemonic (_("Input template"));
03072
             gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_template[i]),
03073
03074
                                            _("Experimental input template file"));
03075
             window->id_input[i] =
               g_signal_connect_swapped (window->button_template[i], "clicked",
03076
03077
                                           (GCallback) window_template_experiment,
03078
                                            (void *) (size_t) i);
03079
             gtk_grid_attach (window->grid_experiment,
03080
                               GTK_WIDGET (window->button_template[i]), 1, 3 + i, 3, 1);
03081
03082
        window->frame_experiment = (GtkFrame *) gtk_frame_new (_("Experiment"));
        gtk_frame_set_child (window->frame_experiment,
03083
03084
                               GTK_WIDGET (window->grid_experiment));
03085
03086
        // Creating the error norm widgets
        window->frame_norm = (GtkFrame *) gtk_frame_new (_("Error norm"));
window->grid_norm = (GtkGrid *) gtk_grid_new ();
03087
03088
        gtk_frame_set_child (window->frame_norm, GTK_WIDGET (window->grid_norm));
03089
03090 #if !GTK4
03091
        window->button_norm[0] = (GtkRadioButton *)
03092
          gtk_radio_button_new_with_mnemonic (NULL, label_norm[0]);
03093 #else
03094
        window->button_norm[0] = (GtkCheckButton *)
          gtk_check_button_new_with_mnemonic (label_norm[0]);
03095
03096 #endif
03097
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_norm[0]),
03098
                                        tip_norm[0]);
03099
        gtk_grid_attach (window->grid_norm,
        GTK_WIDGET (window->button_norm[0]), 0, 0, 1, 1);
g_signal_connect (window->button_norm[0], "toggled", window_update, NULL);
03100
03101
        for (i = 0; ++i < NNORMS;)
03102
03103
03104 #if !GTK4
03105
             window->button_norm[i] = (GtkRadioButton *)
03106
               gtk_radio_button_new_with_mnemonic
               (gtk_radio_button_get_group (window->button_norm[0]), label_norm[i]);
03107
03108 #else
03109
            window->button_norm[i] = (GtkCheckButton *)
               gtk_check_button_new_with_mnemonic (label_norm[i]);
03110
03111
             gtk_check_button_set_group (window->button_norm[i]
03112
                                           window->button_norm[0]);
03113 #endif
             gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_norm[i]),
03114
03115
                                            tip_norm[i]);
03116
             gtk_grid_attach (window->grid_norm,
                               GTK_WIDGET (window->button_norm[i]), 0, i, 1, 1);
03117
03118
             g_signal_connect (window->button_norm[i], "toggled", window_update, NULL);
0.3119
        window->label_p = (GtkLabel *) gtk_label_new (_("P parameter"));
03120
        gtk_grid_attach (window->grid_norm, GTK_WIDGET (window->label_p), 1, 1, 1, 1);
window->spin_p = (GtkSpinButton *)
03121
03122
          gtk_spin_button_new_with_range (-G_MAXDOUBLE, G_MAXDOUBLE, 0.01);
03123
03124
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->spin_p),
03125
                                         ("P parameter for the P error norm"));
        window->scrolled_p = (GtkScrolledWindow *)
03126
03127 #if !GTK4
03128
          gtk_scrolled_window_new (NULL, NULL);
03129 #else
03130
          gtk_scrolled_window_new ();
03131 #endif
        gtk_scrolled_window_set_child (window->scrolled_p,
03132
                                          GTK WIDGET (window->spin p));
03133
```

```
gtk_widget_set_hexpand (GTK_WIDGET (window->scrolled_p), TRUE);
        gtk_widget_set_halign (GTK_WIDGET (window->scrolled_p), GTK_ALIGN_FILL);
03135
03136
        gtk_grid_attach (window->grid_norm, GTK_WIDGET (window->scrolled_p),
03137
                          1, 2, 1, 2);
03138
        // Creating the grid and attaching the widgets to the grid
03139
03140
        window->grid = (GtkGrid *) gtk_grid_new ();
03141
        gtk_grid_attach (window->grid, GTK_WIDGET (window->box_buttons), 0, 0, 3, 1);
03142
        gtk_grid_attach (window->grid, GTK_WIDGET (window->grid_files), 0, 1, 1, 1);
        gtk_grid_attach (window->grid,
03143
                          GTK_WIDGET (window->frame_algorithm), 0, 2, 1, 1);
03144
03145
        gtk_grid_attach (window->grid,
03146
                          GTK_WIDGET (window->frame_variable), 1, 2, 1, 1);
03147
        gtk_grid_attach (window->grid,
03148
                          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));
03149
03150
03151
03152
        // Setting the window logo
03153
        window->logo = gdk_pixbuf_new_from_xpm_data (logo);
03154 #if !GTK4
03155
        gtk_window_set_icon (window->window, window->logo);
03156 #endif
03157
03158
        // Showing the window
03159 #if !GTK4
03160
        gtk_widget_show_all (GTK_WIDGET (window->window));
03161 #else
03162
        gtk_widget_show (GTK_WIDGET (window->window));
03163 #endif
03164
03165
           In GTK+ 3.16 and 3.18 the default scrolled size is wrong
03166 #if GTK_MINOR_VERSION >= 16
03167 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);
03168
03169
        gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_maxabs), -1,
03170
                                                                                    40);
03171
        gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_step), -1, 40);
03172
        gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_p), -1, 40);
03173
        gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_threshold), -1, 40);
03174 #endif
03175
03176
        // Reading initial example
03177
        input_new ();
03178
        buffer2 = g_get_current_dir ();
        buffer = g_build_filename (buffer2, "..", "tests", "test1", INPUT_FILE, NULL);
03179
03180 g_free (buffer2);
03181
       window_read (buffer);
       g_free (buffer);
03182
03183
03184 #if DEBUG_INTERFACE
03185 fprintf (stderr, "window_new: start\n");
03186 #endif
03187 }
```

4.13 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.

4.13.1 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.

4.13.2 Macro Definition Documentation

4.13.2.1 MAX_LENGTH

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

Max length of texts allowed in GtkSpinButtons.

Definition at line 42 of file interface.h.

4.13.3 Function Documentation

4.13.3.1 window_new()

Function to open the main window.

Parameters

application | GtkApplication struct.

Definition at line 2363 of file interface.c.

```
02364 {
02365
        unsigned int i;
02366
        char *buffer, *buffer2, buffer3[64];
        const char *label_algorithm[NALGORITHMS] = {
   "_Monte-Carlo", _("_Sweep"), _("_Genetic"), _("_Orthogonal")
02367
02368
02369
        const char *tip_algorithm[NALGORITHMS] = {
         _("Monte-Carlo brute force algorithm"),
02371
          _("Sweep brute force algorithm"),
02372
02373
          _("Genetic algorithm"),
          \_("Orthogonal sampling brute force algorithm"),
02374
02375
02376
        const char *label_climbing[NCLIMBINGS] = {
02377
          _("_Coordinates climbing"), _("_Random climbing")
02378
02379
        const char *tip_climbing[NCLIMBINGS] = {
02380
         _("Coordinates climbing estimate method"),
          _("Random climbing estimate method")
02381
02382
02383
        const char *label_norm[NNORMS] = { "L2", "L", "Lp", "L1" };
02384
        const char *tip_norm[NNORMS] =
         _("Euclidean error norm (L2)"),
02385
         _("Maximum error norm (L)"),
_("P error norm (Lp)"),
02386
02387
02388
          _("Taxicab error norm (L1)")
02389
02390 #if !GTK4
02391
        const char *close = "delete-event";
02392 #else
       const char *close = "close-request";
02393
02394 #endif
02395
02396 #if DEBUG_INTERFACE
02397
        fprintf (stderr, "window_new: start\n");
02398 #endif
02399
02400
        // Creating the window
02401
        window->window = window_parent = main_window
02402
          = (GtkWindow *) gtk_application_window_new (application);
02403
02404
        // Finish when closing the window
02405
        g_signal_connect_swapped (window->window, close,
02406
                                   G_CALLBACK (g_application_quit),
                                   G_APPLICATION (application));
02407
02408
02409
        // Setting the window title
02410
        gtk_window_set_title (window->window, "MPCOTool");
02411
02412
        // Creating the open button
        window->button_open = (GtkButton *)
02413
02414 #if !GTK4
02415
          gtk_button_new_from_icon_name ("document-open", GTK_ICON_SIZE_BUTTON);
02416 #else
02417
          gtk_button_new_from_icon_name ("document-open");
02418 #endi:
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_open),
02419
02420
                                       _("Open a case"));
        g_signal_connect (window->button_open, "clicked", window_open, NULL);
02421
02422
02423
        // Creating the save button
02424
        window->button_save = (GtkButton *)
02425 #if !GTK4
02426
          gtk button new from icon name ("document-save", GTK ICON SIZE BUTTON);
02427 #else
02428
          gtk_button_new_from_icon_name ("document-save");
02429 #endi:
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_save),
02430
        _("Save the case"));
g_signal_connect (window->button_save, "clicked", (GCallback) window_save,
02431
02432
02433
                           NULL);
02434
02435
        // Creating the run button
02436
       window->button_run = (GtkButton *)
02437 #if !GTK4
02438
          gtk button new from icon name ("system-run", GTK ICON SIZE BUTTON);
02439 #else
          gtk_button_new_from_icon_name ("system-run");
02441 #endif
```

```
gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_run),
02443
                                           _("Run the optimization"));
02444
         g_signal_connect (window->button_run, "clicked", window_run, NULL);
02445
02446
        // Creating the options button
02447
        window->button_options = (GtkButton *)
02448 #if !GTK4
02449
           qtk_button_new_from_icon_name ("preferences-system", GTK_ICON_SIZE_BUTTON);
02450 #else
02451
           gtk_button_new_from_icon_name ("preferences-system");
02452 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_options),
02453
02454
                                           ("Edit the case"));
02455
        g_signal_connect (window->button_options, "clicked", options_new, NULL);
02456
         \ensuremath{//} Creating the help button
02457
02458
        window->button_help = (GtkButton *)
02459 #if !GTK4
02460
           gtk_button_new_from_icon_name ("help-browser", GTK_ICON_SIZE_BUTTON);
02461 #else
02462
           gtk button new from icon name ("help-browser");
02463 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_help), _("Help"));
g_signal_connect (window->button_help, "clicked", window_help, NULL);
02464
02465
02466
02467
         // Creating the about button
02468
        window->button_about = (GtkButton *)
02469 #if !GTK4
02470
           gtk_button_new_from_icon_name ("help-about", GTK_ICON_SIZE_BUTTON);
02471 #else
02472
           gtk_button_new_from_icon_name ("help-about");
02473 #endif
02474
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_about), _("About"));
02475
        g_signal_connect (window->button_about, "clicked", window_about, NULL);
02476
02477
        // Creating the exit button
02478
        window->button_exit = (GtkButton *)
02479 #if !GTK4
02480
           gtk_button_new_from_icon_name ("application-exit", GTK_ICON_SIZE_BUTTON);
02481 #else
02482
          gtk_button_new_from_icon_name ("application-exit");
02483 #endif
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_exit), _("Exit"));
02484
        g_signal_connect_swapped (window->button_exit, "clicked",
02485
                                       G_CALLBACK (g_application_quit),
02486
                                       G_APPLICATION (application));
02487
02488
02489
        \ensuremath{//} Creating the buttons bar
        window->box_buttons = (GtkBox *) gtk_box_new (GTK_ORIENTATION_HORIZONTAL, 0);
02490
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_open));
gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_save));
02491
02492
02493
        gtk_box_append (window->box_buttons, GTK_WIDGET (window->button_run));
02494
         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));
02495
02496
02497
        gtk box append (window->box buttons, GTK WIDGET (window->button exit));
02498
02499
         // Creating the simulator program label and entry
02500
         window->label_simulator = (GtkLabel *) gtk_label_new (_("Simulator program"));
        window->button_simulator = (GtkButton *)
02501
        gtk_button_new_with_mnemonic (_("Simulator program"));
gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_simulator),
02502
02503
02504
                                           _("Simulator program executable file"));
02505
        gtk_widget_set_hexpand (GTK_WIDGET (window->button_simulator), TRUE);
02506
        g_signal_connect (window->button_simulator, "clicked",
02507
                             G_CALLBACK (dialog_simulator), NULL);
02508
02509
        // Creating the evaluator program label and entry
window->check_evaluator = (GtkCheckButton *)
02510
02511
          gtk_check_button_new_with_mnemonic (_("_Evaluator program"));
         g_signal_connect (window->check_evaluator, "toggled", window_update, NULL);
window->button_evaluator = (GtkButton *)
02512
02513
02514
           gtk_button_new_with_mnemonic (_("Evaluator program"));
02515
        gtk_widget_set_tooltip_text
  (GTK_WIDGET (window->button_evaluator),
02516
02517
            _("Optional evaluator program executable file"));
02518
        g_signal_connect (window->button_evaluator, "clicked"
02519
                             G_CALLBACK (dialog_evaluator), NULL);
02520
02521
        // Creating the cleaner program label and entry
window->check_cleaner = (GtkCheckButton *)
02522
         gtk_check_button_new_with_mnemonic (_("_Cleaner program"));
g_signal_connect (window->check_cleaner, "toggled", window_update, NULL);
02523
02524
02525
         window->button_cleaner = (GtkButton *)
02526
           gtk_button_new_with_mnemonic (_("Cleaner program"));
02527
        gtk_widget_set_tooltip_text
02528
           (GTK_WIDGET (window->button_cleaner),
```

```
_("Optional cleaner program executable file"));
02530
        g_signal_connect (window->button_cleaner, "clicked"
02531
                            G_CALLBACK (dialog_cleaner), NULL);
02532
02533
        // Creating the results files labels and entries
        window->label_result = (GtkLabel *) gtk_label_new (_("Result file"));
window->entry_result = (GtkEntry *) gtk_entry_new ();
02534
02535
02536
        {\tt gtk\_widget\_set\_tooltip\_text}
        (GTK_WIDGET (window->entry_result), _("Best results file"));
window->label_variables = (GtkLabel *) gtk_label_new (_("Variables file"));
window->entry_variables = (GtkEntry *) gtk_entry_new ();
02537
02538
02539
02540
        gtk_widget_set_tooltip_text
02541
           (GTK WIDGET (window->entry variables), ("All simulated results file"));
02542
02543
         // 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),
02544
02545
02546
                           0, 0, 1, 1);
02547
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_simulator),
02548
                           1, 0, 1, 1);
02549
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->check_evaluator),
02550
                           0, 1, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_evaluator),
02551
02552
                           1, 1, 1, 1);
02553
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->check_cleaner),
02554
                           0, 2, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->button_cleaner),
02555
02556
                           1, 2, 1, 1);
02557
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_result),
02558
                           0, 3, 1, 1);
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->entry_result),
02559
02560
                           1, 3, 1, 1);
02561
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->label_variables),
02562
                           0, 4, 1, 1);
02563
        gtk_grid_attach (window->grid_files, GTK_WIDGET (window->entry_variables),
02564
                           1, 4, 1, 1);
02565
02566
         // Creating the algorithm properties
02567
        window->label_simulations = (GtkLabel *) gtk_label_new
02568
           (_("Simulations number"));
02569
        window->spin_simulations
02570
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02571
        {\tt gtk\_widget\_set\_tooltip\_text}
02572
           (GTK_WIDGET (window->spin_simulations),
            _("Number of simulations to perform for each iteration"));
02573
02574
        gtk_widget_set_hexpand (GTK_WIDGET (window->spin_simulations), TRUE);
02575
        window->label_iterations = (GtkLabel *)
02576
          gtk_label_new (_("Iterations number"));
02577
        window->spin iterations
02578
           = (GtkSpinButton *) gtk spin button new with range (1., 1.e6, 1.);
        gtk_widget_set_tooltip_text
02580
           (GTK_WIDGET (window->spin_iterations), _("Number of iterations"));
02581
        g_signal_connect
        (window->spin_iterations, "value-changed", window_update, NULL);
gtk_widget_set_hexpand (GTK_WIDGET (window->spin_iterations), TRUE);
02582
02583
02584
         window->label_tolerance = (GtkLabel *) gtk_label_new (_("Tolerance"));
        window->spin_tolerance =
02585
02586
           (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02587
         gtk_widget_set_tooltip_text
02588
           (GTK_WIDGET (window->spin_tolerance),
            _("Tolerance to set the variable interval on the next iteration"));
02589
02590
        window->label_bests = (GtkLabel *) gtk_label_new (_("Bests number"));
02591
        window->spin_bests
02592
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e6, 1.);
        gtk_widget_set_tooltip_text
02593
02594
           (GTK_WIDGET (window->spin_bests),
            _("Number of best simulations used to set the variable interval " \,
02595
02596
              "on the next iteration"));
02597
        window->label_population
02598
           = (GtkLabel *) gtk_label_new (_("Population number"));
02599
        window->spin_population
02600
           = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
02601
        {\tt gtk\_widget\_set\_tooltip\_text}
02602
           (GTK_WIDGET (window->spin_population),
        __("Number of population for the genetic algorithm"));
gtk_widget_set_hexpand (GTK_WIDGET (window->spin_population), TRUE);
02603
02604
02605
        window->label_generations
02606
           = (GtkLabel *) gtk_label_new (_("Generations number"));
02607
        window->spin_generations
02608
           = (GtkSpinButton *) gtk spin button new with range (1., 1.e6, 1.);
        gtk_widget_set_tooltip_text
02609
02610
           (GTK_WIDGET (window->spin_generations),
            _("Number of generations for the genetic algorithm"));
02611
02612
        window->label_mutation = (GtkLabel *) gtk_label_new (_("Mutation ratio"));
02613
        window->spin_mutation
02614
           = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02615
        gtk_widget_set_tooltip_text
```

```
(GTK_WIDGET (window->spin_mutation),
            _("Ratio of mutation for the genetic algorithm"));
02617
02618
         window->label_reproduction
02619
           = (GtkLabel *) gtk_label_new (_("Reproduction ratio"));
02620
         window->spin reproduction
           = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
02621
02622
         gtk_widget_set_tooltip_text
           (GTK_WIDGET (window->spin_reproduction),
02623
02624
             _("Ratio of reproduction for the genetic algorithm"));
02625
         window->label_adaptation = (GtkLabel *) gtk_label_new (_("Adaptation ratio"));
02626
         window->spin_adaptation
02627
           = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
         gtk_widget_set_tooltip_text
  (GTK_WIDGET (window->spin_adaptation),
02628
02629
02630
            _("Ratio of adaptation for the genetic algorithm"));
         window->label_threshold = (GtkLabel *) gtk_label_new (_("Threshold"));
window->spin_threshold = (GtkSpinButton *)
02631
02632
           gtk_spin_button_new_with_range (-G_MAXDOUBLE, G_MAXDOUBLE,
02633
                                               precision[DEFAULT_PRECISION]);
02634
02635
         gtk_widget_set_tooltip_text
           (GTK_WIDGET (window->spin_threshold),
02636
02637
             _("Threshold in the objective function to finish the simulations"));
         window->scrolled_threshold = (GtkScrolledWindow *)
02638
02639 #if !GTK4
02640
           gtk_scrolled_window_new (NULL, NULL);
02641 #else
02642
           gtk_scrolled_window_new ();
02643 #endif
02644
         gtk_scrolled_window_set_child (window->scrolled_threshold,
02645
                                             GTK_WIDGET (window->spin_threshold));
           gtk_widget_set_hexpand (GTK_WIDGET (window->scrolled_threshold), TRUE);
02646 //
02647 //
           gtk_widget_set_halign (GTK_WIDGET (window->scrolled_threshold),
                                             GTK_ALIGN_FILL);
02648 //
02649
02650
         // Creating the hill climbing method properties
         window->check_climbing = (GtkCheckButton *)
02651
         gtk_check_button_new_with_mnemonic (_("_Hill climbing method"));
g_signal_connect (window->check_climbing, "toggled", window_update, NULL);
02652
02653
02654
         window->grid_climbing = (GtkGrid *) gtk_grid_new ();
02655 #if !GTK4
02656
         window->button_climbing[0] = (GtkRadioButton *)
02657
           gtk_radio_button_new_with_mnemonic (NULL, label_climbing[0]);
02658 #else
02659
        window->button_climbing[0] = (GtkCheckButton *)
02660
           gtk_check_button_new_with_mnemonic (label_climbing[0]);
02661 #endif
02662
         gtk_grid_attach (window->grid_climbing,
02663
                            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>
02664
02665
02666
02667 #if !GTK4
02668
             window->button_climbing[i] = (GtkRadioButton *)
02669
                {\tt gtk\_radio\_button\_new\_with\_mnemonic}
02670
                ({\tt gtk\_radio\_button\_get\_group} \ ({\tt window->button\_climbing[0]}) \, ,
02671
                 label climbing[i]);
02672 #else
02673
             window->button_climbing[i] = (GtkCheckButton *)
02674
                gtk_check_button_new_with_mnemonic (label_climbing[i]);
02675
              gtk_check_button_set_group (window->button_climbing[i]
                                              window->button_climbing[0]);
02676
02677 #endif
02678
             gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_climbing[i]),
02679
                                               tip_climbing[i]);
02680
              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,
02681
02682
                                  NULL);
02683
02684
         window->label_steps = (GtkLabel *) gtk_label_new (_("Steps number"));
         window->spin_steps = (GtkSpinButton *)
  gtk_spin_button_new_with_range (1., 1.e12, 1.);
02686
02687
02688
         gtk_widget_set_hexpand (GTK_WIDGET (window->spin_steps), TRUE);
02689
         window->label_final_steps
         = (GtkLabel *) gtk_label_new (_("Final steps number"));
window->spin_final_steps = (GtkSpinButton *)
02690
02691
         gtk_spin_button_new_with_range (1, 1.e12, 1.);
gtk_widget_set_hexpand (GTK_WIDGET (window->spin_final_steps), TRUE);
02692
02693
02694
         window->label_estimates
         = (GtkLabel *) gtk_label_new (_("Climbing estimates number"));
window->spin_estimates = (GtkSpinButton *)
02695
02696
02697
           gtk_spin_button_new_with_range (1., 1.e3, 1.);
02698
         window->label_relaxation
02699
           = (GtkLabel *) gtk_label_new (_("Relaxation parameter"));
         window->spin_relaxation = (GtkSpinButton *)
  gtk_spin_button_new_with_range (0., 2., 0.001);
gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->label_steps),
02700
02701
02702
```

```
0, NCLIMBINGS, 1, 1);
02703
02704
       gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_steps),
02705
                        1, NCLIMBINGS, 1, 1);
02706
       gtk_grid_attach (window->grid_climbing,
02707
                        GTK WIDGET (window->label final steps),
02708
                        0, NCLIMBINGS + 1, 1, 1);
       gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_final_steps),
02709
02710
                        1, NCLIMBINGS + 1, 1, 1);
02711
       \verb|gtk_grid_attach| (\verb|window->grid_climbing|, \verb|GTK_WIDGET| (\verb|window->label_estimates|)|, \\
02712
                        0, NCLIMBINGS + 2, 1, 1);
       gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_estimates),
02713
                        1, NCLIMBINGS + 2, 1, 1);
02714
02715
       gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->label_relaxation),
02716
                        0, NCLIMBINGS + 3, 1, 1);
02717
       gtk_grid_attach (window->grid_climbing, GTK_WIDGET (window->spin_relaxation),
02718
                        1, NCLIMBINGS + 3, 1, 1);
02719
02720
       // Creating the array of algorithms
       window->grid_algorithm = (GtkGrid *) gtk_grid_new ();
02721
02722 #if !GTK4
02723
       window->button_algorithm[0] = (GtkRadioButton *)
02724
         gtk_radio_button_new_with_mnemonic (NULL, label_algorithm[0]);
02725 #else
       window->button_algorithm[0] = (GtkCheckButton *)
02726
02727
         gtk_check_button_new_with_mnemonic (label_algorithm[0]);
02728 #endif
02729
       gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_algorithm[0]),
02730
                                    tip_algorithm[0]);
       gtk_grid_attach (window->grid_algorithm,
02731
02732
                        GTK_WIDGET (window->button_algorithm[0]), 0, 0, 1, 1);
       g_signal_connect (window->button_algorithm[0], "toggled",
02733
02734
                         window_set_algorithm, NULL);
02735
       for (i = 0; ++i < NALGORITHMS;)</pre>
02736
02737 #if !GTK4
           window->button_algorithm[i] = (GtkRadioButton *)
02738
02739
             gtk radio button new with mnemonic
02740
              (gtk_radio_button_get_group (window->button_algorithm[0]),
02741
              label_algorithm[i]);
02742 #else
02743
           window->button_algorithm[i] = (GtkCheckButton *)
02744
             gtk_check_button_new_with_mnemonic (label_algorithm[i]);
           gtk_check_button_set_group (window->button_algorithm[i],
02745
02746
                                       window->button_algorithm[0]),
02747 #endif
             gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_algorithm[i]),
02748
02749
                                          tip_algorithm[i]);
           gtk_grid_attach (window->grid_algorithm,
02750
                            GTK_WIDGET (window->button_algorithm[i]), 0, i, 1, 1);
02751
           g_signal_connect (window->button_algorithm[i], "toggled",
02752
02753
                             window_set_algorithm, NULL);
02754
02755
       gtk_grid_attach (window->grid_algorithm,
                        GTK_WIDGET (window->label_simulations),
0, NALGORITHMS, 1, 1);
02756
02757
02758
       gtk_grid_attach (window->grid_algorithm,
02759
                        GTK_WIDGET (window->spin_simulations), 1, NALGORITHMS, 1, 1);
       gtk_grid_attach (window->grid_algorithm,
02760
                        GTK_WIDGET (window->label_iterations),
02761
02762
                        0, NALGORITHMS + 1, 1, 1);
       02763
02764
02765
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_tolerance),
02766
                        0, NALGORITHMS + 2, 1, 1);
02767
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_tolerance),
02768
                        1, NALGORITHMS + 2, 1, 1);
       02769
02770
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_bests),
02771
                        1, NALGORITHMS + 3, 1, 1);
02772
02773
       gtk_grid_attach (window->grid_algorithm,
02774
                        GTK_WIDGET (window->label_population),
02775
                        0, NALGORITHMS + 4, 1, 1);
       02776
02777
02778
       gtk_grid_attach (window->grid_algorithm,
02779
                        GTK_WIDGET (window->label_generations),
02780
                        0, NALGORITHMS + 5, 1, 1);
02781
       gtk_grid_attach (window->grid_algorithm,
02782
                        GTK_WIDGET (window->spin_generations),
                        1, NALGORITHMS + 5, 1, 1);
02783
02784
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_mutation),
02785
                        0, NALGORITHMS + 6, 1, 1);
02786
       gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->spin_mutation),
02787
                        1, NALGORITHMS + 6, 1, 1);
02788
       gtk_grid_attach (window->grid_algorithm
02789
                        GTK WIDGET (window->label reproduction).
```

```
0, NALGORITHMS + 7, 1, 1);
02791
        gtk_grid_attach (window->grid_algorithm,
02792
                           GTK_WIDGET (window->spin_reproduction),
                          1, NALGORITHMS + 7, 1, 1);
02793
02794
        gtk_grid_attach (window->grid_algorithm,
02795
                           GTK_WIDGET (window->label_adaptation),
                           0, NALGORITHMS + 8, 1, 1);
02796
02797
        \verb|gtk_grid_attach| (\verb|window->grid_algorithm|, GTK_WIDGET| (\verb|window->spin_adaptation)|, \\
02798
                           1, NALGORITHMS + 8, 1, 1);
02799
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->check_climbing),
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->grid_climbing),
0, NALGORITHMS + 10, 2, 1);
02800
02801
02802
        gtk_grid_attach (window->grid_algorithm, GTK_WIDGET (window->label_threshold),
02803
02804
                           0, NALGORITHMS + 11, 1, 1);
02805
        gtk_grid_attach (window->grid_algorithm,
                          GTK_WIDGET (window->scrolled_threshold),
02806
        intermediate (window->scribted_intermediate),
1, NALGORITHMS + 11, 1, 1);
window->frame_algorithm = (ftkFrame *) gtk_frame_new (_("Algorithm"));
02807
02808
        gtk_frame_set_child (window->frame_algorithm,
02809
02810
                               GTK WIDGET (window->grid algorithm));
02811
02812
        // Creating the variable widgets
02813
        window->combo variable = (GtkComboBoxText *) gtk combo box text new ();
02814
        gtk_widget_set_tooltip_text
           (GTK_WIDGET (window->combo_variable), _("Variables selector"));
02816
        window->id_variable = g_signal_connect
02817
           (window->combo_variable, "changed", window_set_variable, NULL);
02818 #if !GTK4
02819
        window->button add variable = (GtkButton *)
02820
          gtk_button_new_from_icon_name ("list-add", GTK_ICON_SIZE_BUTTON);
02821 #else
02822
        window->button_add_variable = (GtkButton *)
02823
          gtk_button_new_from_icon_name ("list-add");
02824 #endif
        g_signal_connect (window->button_add_variable, "clicked", window_add_variable,
02825
                           NULL);
02826
02827
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_add_variable),
02828
                                       _("Add variable"));
02829 #if !GTK4
02830
        window->button_remove_variable = (GtkButton *)
02831
          gtk_button_new_from_icon_name ("list-remove", GTK_ICON_SIZE_BUTTON);
02832 #else
02833
        window->button_remove_variable = (GtkButton *)
          gtk_button_new_from_icon_name ("list-remove");
02834
02835 #endif
02836
        g_signal_connect (window->button_remove_variable, "clicked",
                           window_remove_variable, NULL);
02837
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_remove_variable),
02838
                                        _("Remove variable"));
02839
02840
        window->label_variable = (GtkLabel *) gtk_label_new (_("Name"));
02841
        window->entry_variable = (GtkEntry *) gtk_entry_new ();
02842
        gtk_widget_set_tooltip_text
02843
           (GTK_WIDGET (window->entry_variable), _("Variable name"));
        gtk_widget_set_hexpand (GTK_WIDGET (window->entry_variable), TRUE);
02844
        window->id_variable_label = g_signal_connect
  (window->entry_variable, "changed", window_label_variable, NULL);
02845
02846
        window->label_min = (GtkLabel *) gtk_label_new (_("Minimum"));
02847
02848
        window->spin_min = (GtkSpinButton *) gtk_spin_button_new_with_range
02849
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02850
        gtk_widget_set_tooltip_text
02851
          (GTK WIDGET (window->spin min), ("Minimum initial value of the variable"));
02852
        window->scrolled_min = (GtkScrolledWindow *)
02853 #if !GTK4
02854
           gtk_scrolled_window_new (NULL, NULL);
02855 #else
02856
          gtk_scrolled_window_new ();
02857 #endif
02858
        gtk_scrolled_window_set_child (window->scrolled_min,
                                         GTK_WIDGET (window->spin_min));
02860
        g_signal_connect (window->spin_min, "value-changed",
02861
                           window_rangemin_variable, NULL);
        window->label_max = (GtkIabel *) gtk_label_new (_("Maximum"));
window->spin_max = (GtkSpinButton *) gtk_spin_button_new_with_range
02862
02863
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02864
02865
        gtk_widget_set_tooltip_text
02866
           (GTK_WIDGET (window->spin_max), _("Maximum initial value of the variable"));
02867
        window->scrolled_max = (GtkScrolledWindow *)
02868 #if !GTK4
02869
          gtk scrolled window new (NULL, NULL);
02870 #else
02871
          gtk_scrolled_window_new ();
02872 #endif
02873
        gtk_scrolled_window_set_child (window->scrolled_max,
02874
                                          GTK_WIDGET (window->spin_max));
        g_signal_connect (window->spin_max, "value-changed",
02875
                            window_rangemax_variable, NULL);
02876
```

```
window->check_minabs = (GtkCheckButton *)
        gtk_check_button_new_with_mnemonic (_("_Absolute minimum"));
g_signal_connect (window->check_minabs, "toggled", window_update, NULL);
02878
02879
        window->spin_minabs = (GtkSpinButton *) gtk_spin_button_new_with_range
02880
          (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02881
        gtk_widget_set_tooltip_text
02882
           (GTK_WIDGET (window->spin_minabs),
            _("Minimum allowed value of the variable"));
02884
02885
        window->scrolled_minabs = (GtkScrolledWindow *)
02886 #if !GTK4
02887
          gtk_scrolled_window_new (NULL, NULL);
02888 #else
02889
          gtk scrolled window new ();
02890 #endif
02891
        gtk_scrolled_window_set_child (window->scrolled_minabs,
        02892
02893
02894
02895
        window->check_maxabs = (GtkCheckButton *)
        gtk_check_button_new_with_mnemonic (_("_Absolute maximum"));
g_signal_connect (window->check_maxabs, "toggled", window_update, NULL);
02896
02897
02898
        window->spin_maxabs = (GtkSpinButton *) gtk_spin_button_new_with_range
02899
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02900
        gtk_widget_set_tooltip_text
02901
           (GTK_WIDGET (window->spin_maxabs),
            _("Maximum allowed value of the variable"));
02902
        window->scrolled_maxabs = (GtkScrolledWindow *)
02903
02904 #if !GTK4
02905
          gtk_scrolled_window_new (NULL, NULL);
02906 #else
02907
          gtk_scrolled_window_new ();
02908 #endif
02909
        gtk_scrolled_window_set_child (window->scrolled_maxabs,
02910
                                          GTK_WIDGET (window->spin_maxabs));
02911
        g_signal_connect (window->spin_maxabs, "value-changed",
        window_rangemaxabs_variable, NULL);
window->label_precision = (GtkLabel *) gtk_label_new (_("Precision digits"));
window->spin_precision = (GtkSpinButton *)
02912
02913
02914
02915
          gtk_spin_button_new_with_range (0., (gdouble) DEFAULT_PRECISION, 1.);
02916
        gtk_widget_set_tooltip_text
02917
           (GTK_WIDGET (window->spin_precision),
           _("Number of precision floating point digits \n"
02918
              "0 is for integer numbers"));
02919
        g_signal_connect (window->spin_precision, "value-changed",
02920
                            window_precision_variable, NULL);
02921
02922
        window->label_sweeps = (GtkLabel *) gtk_label_new (_("Sweeps number"));
02923
        window->spin_sweeps =
        (GtkSpinButton *) gtk_spin_button_new_with_range (1., 1.e12, 1.);
gtk_widget_set_tooltip_text (GTK_WIDGET (window->spin_sweeps),
02924
02925
                                         _("Number of steps sweeping the variable"));
02926
        g_signal_connect (window->spin_sweeps, "value-changed",
02927
02928
                            window_update_variable, NULL);
02929
        window->label_bits = (GtkLabel *) gtk_label_new (_("Bits number"));
02930
        window->spin bits
02931
          = (GtkSpinButton *) gtk_spin_button_new_with_range (1., 64., 1.);
        gtk_widget_set_tooltip_text
  (GTK_WIDGET (window->spin_bits),
02932
02933
02934
            _("Number of bits to encode the variable"));
02935
        g_signal_connect
           (window->spin_bits, "value-changed", window_update_variable, NULL);
02936
        window->label_step = (GtkLabel *) gtk_label_new (_("Step size"));
window->spin_step = (GtkSpinButton *) gtk_spin_button_new_with_range
02937
02938
02939
           (-G_MAXDOUBLE, G_MAXDOUBLE, precision[DEFAULT_PRECISION]);
02940
        gtk_widget_set_tooltip_text
02941
          (GTK_WIDGET (window->spin_step),
        _("Initial step size for the hill climbing method"));
window->scrolled_step = (GtkScrolledWindow *)
02942
02943
02944 #if !GTK4
02945
          gtk scrolled window new (NULL, NULL);
02946 #else
02947
          gtk_scrolled_window_new ();
02948 #endif
02949
        gtk_scrolled_window_set_child (window->scrolled_step,
02950
                                          GTK_WIDGET (window->spin step));
02951
        g signal connect
02952
           (window->spin_step, "value-changed", window_step_variable, NULL);
02953
        window->grid_variable = (GtkGrid *) gtk_grid_new ();
02954
        gtk_grid_attach (window->grid_variable,
02955
                           GTK_WIDGET (window->combo_variable), 0, 0, 2, 1);
02956
        gtk grid attach (window->grid variable,
02957
                           GTK WIDGET (window->button add variable), 2, 0, 1, 1);
02958
        gtk_grid_attach (window->grid_variable,
02959
                           GTK_WIDGET (window->button_remove_variable), 3, 0, 1, 1);
02960
        gtk_grid_attach (window->grid_variable,
02961
                           GTK_WIDGET (window->label_variable), 0, 1, 1, 1);
02962
        gtk_grid_attach (window->grid_variable,
                           GTK WIDGET (window->entry variable), 1, 1, 3, 1);
02963
```

```
02964
       gtk_grid_attach (window->grid_variable,
02965
                         GTK_WIDGET (window->label_min), 0, 2, 1, 1);
02966
        gtk_grid_attach (window->grid_variable,
                         GTK_WIDGET (window->scrolled_min), 1, 2, 3, 1);
02967
02968
       gtk_grid_attach (window->grid_variable,
                         GTK_WIDGET (window->label_max), 0, 3, 1, 1);
02969
02970
       gtk_grid_attach (window->grid_variable,
02971
                         GTK_WIDGET (window->scrolled_max), 1, 3, 3, 1);
02972
       gtk_grid_attach (window->grid_variable,
02973
                         GTK_WIDGET (window->check_minabs), 0, 4, 1, 1);
02974
       gtk_grid_attach (window->grid_variable,
02975
                         GTK WIDGET (window->scrolled minabs), 1, 4, 3, 1);
02976
       gtk_grid_attach (window->grid_variable,
02977
                         GTK_WIDGET (window->check_maxabs), 0, 5, 1, 1);
02978
       gtk_grid_attach (window->grid_variable,
02979
                         GTK_WIDGET (window->scrolled_maxabs), 1, 5, 3, 1);
02980
       gtk_grid_attach (window->grid_variable,
                         GTK_WIDGET (window->label_precision), 0, 6, 1, 1);
02981
02982
       gtk_grid_attach (window->grid_variable,
02983
                         GTK_WIDGET (window->spin_precision), 1, 6, 3, 1);
       gtk_grid_attach (window->grid_variable,
02984
02985
                         GTK_WIDGET (window->label_sweeps), 0, 7, 1, 1);
02986
       gtk_grid_attach (window->grid_variable,
02987
                         GTK_WIDGET (window->spin_sweeps), 1, 7, 3, 1);
       gtk_grid_attach (window->grid_variable,
02988
02989
                         GTK_WIDGET (window->label_bits), 0, 8, 1, 1);
02990
       gtk_grid_attach (window->grid_variable,
02991
                         GTK_WIDGET (window->spin_bits), 1, 8, 3, 1);
02992
       gtk_grid_attach (window->grid_variable,
                         GTK_WIDGET (window->label_step), 0, 9, 1, 1);
02993
02994
       gtk_grid_attach (window->grid_variable,
02995
                         GTK_WIDGET (window->scrolled_step), 1, 9, 3, 1);
02996
       window->frame_variable = (GtkFrame *) gtk_frame_new (_("Variable"));
       gtk_frame_set_child (window->frame_variable,
02997
02998
                             GTK_WIDGET (window->grid_variable));
02999
03000
       // Creating the experiment widgets
       window->combo_experiment = (GtkComboBoxText *) gtk_combo_box_text_new ();
03001
03002
       gtk_widget_set_tooltip_text (GTK_WIDGET (window->combo_experiment),
03003
                                     _("Experiment selector"));
03004
       window->id_experiment = g_signal_connect
03005
          (window->combo_experiment, "changed", window_set_experiment, NULL);
03006 #if !GTK4
03007
       window->button_add_experiment = (GtkButton *)
03008
          gtk_button_new_from_icon_name ("list-add", GTK_ICON_SIZE_BUTTON);
03009 #else
03010
       window->button_add_experiment = (GtkButton *)
03011
         gtk_button_new_from_icon_name ("list-add");
03012 #endif
03013
       g_signal_connect
03014
          (window->button_add_experiment, "clicked", window_add_experiment, NULL);
03015
       gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_add_experiment),
03016
                                     _("Add experiment"));
03017 #if !GTK4
       window->button remove experiment = (GtkButton *)
03018
         gtk_button_new_from_icon_name ("list-remove", GTK_ICON_SIZE_BUTTON);
03019
03020 #else
03021
       window->button_remove_experiment = (GtkButton *)
03022
          gtk_button_new_from_icon_name ("list-remove");
03023 #endif
03024
       g_signal_connect (window->button_remove_experiment, "clicked",
03025
                          window remove experiment, NULL);
03026
       gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_remove_experiment),
03027
                                     _("Remove experiment"));
03028
       window->label_experiment
03029
          = (GtkLabel *) gtk_label_new (_("Experimental data file"));
03030
        window->button_experiment = (GtkButton *)
         gtk_button_new_with_mnemonic (_("Experimental data file"));
03031
       gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_experiment),
03032
03033
                                     _("Experimental data file"));
03034
        g_signal_connect (window->button_experiment, "clicked",
03035
                          window_name_experiment, NULL);
03036
        gtk_widget_set_hexpand (GTK_WIDGET (window->button_experiment), TRUE);
       window->label_weight = (GtkLabel *) gtk_label_new (_("Weight"));
window->spin_weight
03037
03038
03039
          = (GtkSpinButton *) gtk_spin_button_new_with_range (0., 1., 0.001);
03040
       gtk_widget_set_tooltip_text
03041
          (GTK_WIDGET (window->spin_weight),
03042
           _("Weight factor to build the objective function"));
03043
        g signal connect
          (window->spin_weight, "value-changed", window_weight_experiment, NULL);
03044
        window->grid_experiment = (GtkGrid *) gtk_grid_new ();
03045
03046
       gtk_grid_attach (window->grid_experiment,
03047
                         GTK_WIDGET (window->combo_experiment), 0, 0, 2, 1);
03048
        gtk_grid_attach (window->grid_experiment,
03049
                         GTK_WIDGET (window->button_add_experiment), 2, 0, 1, 1);
03050
       gtk grid attach (window->grid experiment,
```

```
GTK_WIDGET (window->button_remove_experiment), 3, 0, 1, 1);
03052
        gtk_grid_attach (window->grid_experiment,
03053
                           GTK_WIDGET (window->label_experiment), 0, 1, 1, 1);
03054
        {\tt gtk\_grid\_attach~(window->grid\_experiment,}
03055
                           GTK WIDGET (window->button experiment), 1, 1, 3, 1);
03056
        gtk grid attach (window->grid experiment,
                           GTK_WIDGET (window->label_weight), 0, 2, 1, 1);
03058
        gtk_grid_attach (window->grid_experiment,
03059
                           GTK_WIDGET (window->spin_weight), 1, 2, 3, 1);
03060
        for (i = 0; i < MAX_NINPUTS; ++i)</pre>
03061
             snprintf (buffer3, 64, "%s %u", _("Input template"), i + 1);
window->check_template[i] = (GtkCheckButton *)
03062
03063
               gtk_check_button_new_with_label (buffer3);
03064
03065
             window->id_template[i]
03066
               = g_signal_connect (window->check_template[i], "toggled",
03067
                                    window_inputs_experiment, NULL);
            gtk_grid_attach (window->grid_experiment,
03068
                              GTK_WIDGET (window->check_template[i]), 0, 3 + i, 1, 1);
03069
             window->button_template[i] = (GtkButton *)
03070
               gtk_button_new_with_mnemonic (_("Input template"));
03071
03072
03073
             gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_template[i]),
03074
                                            _("Experimental input template file"));
03075
             window->id_input[i] =
03076
               g_signal_connect_swapped (window->button_template[i], "clicked",
03077
                                           (GCallback) window_template_experiment,
03078
                                           (void *) (size_t) i);
03079
             gtk_grid_attach (window->grid_experiment,
                               GTK_WIDGET (window->button_template[i]), 1, 3 + i, 3, 1);
03080
03081
03082
        window->frame_experiment = (GtkFrame *) gtk_frame_new (_("Experiment"));
03083
        gtk_frame_set_child (window->frame_experiment,
03084
                               GTK_WIDGET (window->grid_experiment));
03085
03086
        // Creating the error norm widgets
        window->frame_norm = (GtkFrame *) gtk_frame_new (_("Error norm"));
window->grid_norm = (GtkGrid *) gtk_grid_new ();
03087
03088
        gtk_frame_set_child (window->frame_norm, GTK_WIDGET (window->grid_norm));
03089
03090 #if !GTK4
03091
        window->button_norm[0] = (GtkRadioButton *)
          gtk_radio_button_new_with_mnemonic (NULL, label_norm[0]);
03092
03093 #else
03094
        window->button_norm[0] = (GtkCheckButton *)
          gtk_check_button_new_with_mnemonic (label_norm[0]);
03095
03096 #endif
03097
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_norm[0]),
03098
                                       tip_norm[0]);
        gtk_grid_attach (window->grid_norm,
03099
        GTK_WIDGET (window->button_norm[0]), 0, 0, 1, 1);
g_signal_connect (window->button_norm[0], "toggled", window_update, NULL);
03100
03101
03102
        for (i = 0; ++i < NNORMS;)</pre>
03103
03104 #if !GTK4
            window->button_norm[i] = (GtkRadioButton *)
03105
               gtk_radio_button_new_with_mnemonic
03106
03107
               (gtk_radio_button_get_group (window->button_norm[0]), label_norm[i]);
03108 #else
            window->button_norm[i] = (GtkCheckButton *)
03109
03110
               gtk_check_button_new_with_mnemonic (label_norm[i]);
0.3111
             gtk_check_button_set_group (window->button_norm[i],
03112
                                           window->button norm[0]);
03113 #endif
03114
            gtk_widget_set_tooltip_text (GTK_WIDGET (window->button_norm[i]),
03115
                                            tip_norm[i]);
03116
             gtk_grid_attach (window->grid_norm,
03117
                               GTK_WIDGET (window->button_norm[i]), 0, i, 1, 1);
             g_signal_connect (window->button_norm[i], "toggled", window_update, NULL);
03118
03119
03120
        window->label_p = (GtkLabel *) gtk_label_new (_("P parameter"));
03121
        gtk_grid_attach (window->grid_norm, GTK_WIDGET (window->label_p), 1, 1, 1, 1);
03122
        window->spin p = (GtkSpinButton *)
03123
          gtk_spin_button_new_with_range (-G_MAXDOUBLE, G_MAXDOUBLE, 0.01);
        gtk_widget_set_tooltip_text (GTK_WIDGET (window->spin_p),
03124
                                         ("P parameter for the P error norm"));
03125
        window->scrolled_p = (GtkScrolledWindow *)
03126
03127 #if !GTK4
03128
          gtk_scrolled_window_new (NULL, NULL);
03129 #else
          gtk_scrolled_window_new ();
03130
03131 #endif
03132
        gtk_scrolled_window_set_child (window->scrolled_p,
                                          GTK_WIDGET (window->spin_p));
03133
        gtk_widget_set_hexpand (GTK_WIDGET (window->scrolled_p), TRUE);
gtk_widget_set_halign (GTK_WIDGET (window->scrolled_p), GTK_ALIGN_FILL);
03134
03135
03136
        gtk_grid_attach (window->grid_norm, GTK_WIDGET (window->scrolled_p),
                           1, 2, 1, 2);
03137
```

4.14 interface.h

```
03139
        // Creating the grid and attaching the widgets to the grid
03140
        window->grid = (GtkGrid *) gtk_grid_new ();
        gtk_grid_attach (window->grid, GTK_WIDGET (window->box_buttons), 0, 0, 3, 1); gtk_grid_attach (window->grid, GTK_WIDGET (window->grid_files), 0, 1, 1, 1);
0.3141
03142
03143
        gtk_grid_attach (window->grid,
03144
                            GTK_WIDGET (window->frame_algorithm), 0, 2, 1, 1);
03145
        gtk_grid_attach (window->grid,
03146
                            GTK_WIDGET (window->frame_variable), 1, 2, 1, 1);
03147
        gtk_grid_attach (window->grid,
                           GTK_WIDGET (window->frame_experiment), 2, 2, 1, 1);
03148
        gtk_grid_attach (window->grid, GTK_WIDGET (window->frame_norm), 1, 1, 2, 1);
03149
03150
        gtk_window_set_child (window->window, GTK_WIDGET (window->grid));
03151
03152
        // Setting the window logo
03153
        window->logo = gdk_pixbuf_new_from_xpm_data (logo);
03154 #if !GTK4
03155
        gtk_window_set_icon (window->window, window->logo);
03156 #endif
03157
         // Showing the window
03158
03159 #if !GTK4
03160
       gtk_widget_show_all (GTK_WIDGET (window->window));
03161 #else
        gtk_widget_show (GTK_WIDGET (window->window));
03162
03163 #endif
03164
03165
         // In GTK+ 3.16 and 3.18 the default scrolled size is wrong
03166 #if GTK_MINOR_VERSION >= 16
       gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_min), -1, 40);
gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_max), -1, 40);
03167
03168
03169
        gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_minabs), -1, 40);
03170
        gtk_widget_set_size_request (GTK_WIDGET (window->scrolled_maxabs), -1,
                                                                                        40);
03171
        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);
03172
03173
03174 #endif
03175
03176
        // Reading initial example
03177 input_new ();
03178 buffer2 = g_get_current_dir ();
       buffer = g_build_filename (buffer2, "..", "tests", "test1", INPUT_FILE, NULL);
03179
        g_free (buffer2):
03180
03181
        window_read (buffer);
03182
        g_free (buffer);
03183
03184 #if DEBUG_INTERFACE
03185 fprintf (stderr, "window_new: startn");
03186 #endif
03187 }
```

4.13.4 Variable Documentation

4.13.4.1 window

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

Window struct to define the main interface window.

Definition at line 81 of file interface.c.

4.14 interface.h

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

```
00001 /\star 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
        GtkSpinButton *spin_seed;
00056
        GtkLabel *label_threads;
00057
        GtkSpinButton *spin_threads;
00058
        GtkLabel *label_climbing;
        GtkSpinButton *spin_climbing;
00059
00060 } Options;
00061
00066 typedef struct
00067 {
00068
        GtkDialog *dialog;
00069
        GtkLabel *label;
00070
        GtkSpinner *spinner;
00071
        GtkGrid *grid:
00072 } Running;
00073
00078 typedef struct
00079 {
        GtkWindow *window:
08000
00081
        GtkGrid *grid;
        GtkBox *box_buttons;
00082
00083
        GtkButton *button_open;
00084
        GtkButton *button_save;
00085
        GtkButton *button_run;
        GtkButton *button_options;
00086
00087
        GtkButton *button help;
00088
        GtkButton *button_about;
00089
        GtkButton *button_exit;
00090
        GtkGrid *grid_files;
00091
        GtkLabel *label_simulator;
00092
        GtkButton *button simulator;
00093
        GtkCheckButton *check evaluator:
00094
        GtkButton *button_evaluator;
00095
        GtkCheckButton *check_cleaner;
00096
        GtkButton *button_cleaner;
00097
        GtkLabel *label_result;
        GtkEntry *entry_result;
GtkLabel *label_variables;
00098
00099
00100
        GtkEntry *entry_variables;
00101
        GtkFrame *frame_norm;
        GtkGrid *grid_norm;
00102
00103 #if !GTK4
00104
        GtkRadioButton *button_norm[NNORMS];
00106 #else
00107
        GtkCheckButton *button norm[NNORMS];
00109 #endif
        GtkLabel *label_p;
00110
00111
        GtkSpinButton *spin_p;
00112
        GtkScrolledWindow *scrolled_p;
00114
        GtkFrame *frame algorithm;
00115
       GtkGrid *grid_algorithm;
```

4.14 interface.h

```
00116 #if !GTK4
        GtkRadioButton *button_algorithm[NALGORITHMS];
00117
00119 #else
00120
        GtkCheckButton *button_algorithm[NALGORITHMS];
00122 #endif
00123
        GtkLabel *label simulations:
00124
        GtkSpinButton *spin_simulations;
00126
        GtkLabel *label_iterations;
00127
        GtkSpinButton *spin_iterations;
00129
        GtkLabel *label tolerance;
        GtkSpinButton *spin_tolerance;
GtkLabel *label_bests;
00130
00131
00132
        GtkSpinButton *spin_bests;
00133
        GtkLabel *label_population;
00134
        GtkSpinButton *spin_population;
00136
        GtkLabel *label_generations;
        GtkSpinButton *spin_generations;
GtkLabel *label_mutation;
00137
00139
00140
        GtkSpinButton *spin_mutation;
00141
        GtkLabel *label_reproduction;
00142
        GtkSpinButton *spin_reproduction;
00144
        GtkLabel *label_adaptation;
00145
        GtkSpinButton *spin_adaptation;
00147
        GtkCheckButton *check_climbing;
00149
        GtkGrid *grid_climbing;
00151 #
00152
        GtkRadioButton *button_climbing[NCLIMBINGS];
00154 #else
00155
        GtkCheckButton *button_climbing[NCLIMBINGS];
00157 #endif
00158
        GtkLabel *label steps:
        GtkSpinButton *spin_steps;
GtkLabel *label_final_steps;
00159
00160
00161
        GtkSpinButton *spin_final_steps;
00163
        GtkLabel *label_estimates;
00164
        GtkSpinButton *spin_estimates;
        GtkLabel *label_relaxation;
00166
00168
        GtkSpinButton *spin_relaxation;
00170
        GtkLabel *label_threshold;
00171
        GtkSpinButton *spin_threshold;
00172
        GtkScrolledWindow *scrolled_threshold;
00174
        GtkFrame *frame_variable;
GtkGrid *grid variable;
00175
00176
        GtkComboBoxText *combo_variable;
00178
        GtkButton *button_add_variable;
00179
        GtkButton *button_remove_variable;
00180
        GtkLabel *label_variable;
        GtkEntry *entry_variable;
GtkLabel *label_min;
00181
00182
00183
        GtkSpinButton *spin_min;
00184
        GtkScrolledWindow *scrolled_min;
00185
        GtkLabel *label_max;
00186
        GtkSpinButton *spin_max;
00187
        GtkScrolledWindow *scrolled_max;
00188
        GtkCheckButton *check_minabs;
00189
        GtkSpinButton *spin_minabs;
00190
        GtkScrolledWindow *scrolled_minabs;
        GtkCheckButton *check_maxabs;
00191
00192
        GtkSpinButton *spin_maxabs;
00193
        GtkScrolledWindow *scrolled_maxabs;
        GtkLabel *label_precision;
00194
00195
        GtkSpinButton *spin_precision;
00196
        GtkLabel *label_sweeps;
00197
        GtkSpinButton *spin_sweeps;
00198
        GtkLabel *label_bits;
00199
        GtkSpinButton *spin_bits;
00200
        GtkLabel *label_step;
00201
        GtkSpinButton *spin_step;
00202
        GtkScrolledWindow *scrolled_step;
00203
        GtkFrame *frame_experiment;
00204
        GtkGrid *grid_experiment;
00205
        GtkComboBoxText *combo_experiment;
00206
        GtkButton *button_add_experiment;
00207
        GtkButton *button_remove_experiment;
00208
        GtkLabel *label_experiment;
00209
        GtkButton *button_experiment;
        GtkLabel *label_weight;
00211
00212
        GtkSpinButton *spin_weight;
        GtkCheckButton *check_template[MAX_NINPUTS];
00213
00215
        GtkButton *button_template[MAX_NINPUTS];
GdkPixbuf *logo;
00217
00218
        Experiment *experiment;
00219
        Variable *variable;
00220
        char *application_directory;
00221
        gulong id_experiment;
        gulong id_experiment_name;
00223
        gulong id variable;
```

```
gulong id_variable_label;
00225
       gulong id_template[MAX_NINPUTS];
00227
       gulong id_input[MAX_NINPUTS];
00229 unsigned int nexperiments;
00230
       unsigned int nvariables;
00231 } Window;
00233 // Global variables
00234 extern Window window[1];
00235
00236 // Public functions
00237 void window_new (GtkApplication * application);
00238
00239 #endif
```

4.15 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/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)

4.15 main.c File Reference 215

4.15.1 Detailed Description

Main source file.

Authors

Javier Burguete and Borja Latorre.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file main.c.

4.15.2 Macro Definition Documentation

4.15.2.1 JBW

```
#define JBW 2
```

Definition at line 59 of file main.c.

4.15.3 Function Documentation

4.15.3.1 main()

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

Main function

Returns

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

Definition at line 81 of file main.c.

Here is the call graph for this function:

4.16 main.c

00001 /*

Go to the documentation of this file.

```
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/win.h"
00065 #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 }
```

4.17 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/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)

4.17.1 Detailed Description

Main function source file.

Authors

Javier Burguete and Borja Latorre.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file mpcotool.c.

4.17.2 Macro Definition Documentation

4.17.2.1 DEBUG_MPCOTOOL

```
#define DEBUG_MPCOTOOL 1
```

Macro to debug main functions.

Definition at line 73 of file mpcotool.c.

4.17.3 Function Documentation

4.17.3.1 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 }
```

4.18 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/win.h"
00062 #include "genetic/genetic.h"
00063 #include "tools.h"
00064 #include "experiment.h"
```

4.18 mpcotool.c 221

```
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);
       g_signal_connect (application, "activate", G_CALLBACK (window_new), NULL);
00160
00161
00162
        // Opening the main window
       g_application_run (G_APPLICATION (application), 0, NULL);
00163
00164
00165
       // Freeing memory
       input_free ();
00166
       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"
                   "./mpcotoolbin [-nthreads x] [-seed s] data_file [result_file] "
"[variables_file]\n");
00177
00178
00179
           return 2;
00180
00181
       if (argn > 1)
         input->result = (char *) xmlStrdup ((xmlChar *) argc[optind + 1]);
00182
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
        // Freeing memory
00193
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 }
```

4.19 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)

4.19.1 Detailed Description

Main function header file.

Authors

Javier Burguete and Borja Latorre.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file mpcotool.h.

4.19.2 Function Documentation

4.19.2.1 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.

```
00083 {
00084
        const struct option options[] = {
         {"seed", required_argument, NULL, 's'},
{"nthreads", required_argument, NULL, 't'},
00085
00086
          {NULL, 0, NULL, 0}
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_MPCOTOOL
00096
        fprintf (stderr, "mpcotool: starting pseudo-random numbers generator\n");
00097 #endif
00098
        optimize->rng = gsl_rng_alloc (gsl_rng_taus2);
00099
        // Allowing spaces in the XML data file
00100
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
00108 // Starting MPI
00107 #if HAVE_MPI
00108 #if DEBUG_MPCOTOOL
00109 fprintf (stderr, "mpcotool: starting MPI\n");
00110 #endif
```

```
MPI_Init (&argn, &argc);
        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
       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);
            if (o == -1)
00127
00128
              break:
00129
            switch (o)
00130
              {
00131
              case 's':
              optimize->seed = atol (optarg);
break;
00132
00133
              case 't':
00134
               nthreads_climbing = nthreads = atoi (optarg);
00135
00136
                break;
00137
              default:
              printf ("%s\n%s\n", _("ERROR!"), _("Unknown option"));
00138
00139
                return 1;
              }
00140
00141
00142
       argn -= optind;
00143
00144
       // Resetting result and variables file names
00145 #if DEBUG_MPCOTOOL
       fprintf (stderr, "mpcotool: resetting result and variables file names\n");
00146
00147 #endif
       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 ();
        application = gtk_application_new ("es.csic.eead.auladei.sprinkler",
00158
                                             G_APPLICATION_DEFAULT_FLAGS);
00159
00160
        q_siqnal_connect (application, "activate", G_CALLBACK (window_new), NULL);
00161
00162
        // Opening the main window
00163
        g_application_run (G_APPLICATION (application), 0, NULL);
00164
       // Freeing memory
00165
00166
        input free ();
00167
       gtk_window_destroy (window->window);
00168
        g_object_unref (application);
00169
        g_free (window->application_directory);
00170
00171 #else
00172
00173
        // Checking syntax
00174
        if (argn < 1 || argn > 3)
00175
00176
            printf ("The syntax is:\n"
                    "./mpcotoolbin [-nthreads x] [-seed s] data_file [result_file] "
"[variables_file]\n");
00177
00178
00179
            return 2:
00180
00181
        if (argn > 1)
00182
          input->result = (char *) xmlStrdup ((xmlChar *) argc[optind + 1]);
00183
        if (argn == 2)
          input->variables = (char *) xmlStrdup ((xmlChar *) argc[optind + 2]);
00184
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
00193
00194 #if DEBUG_MPCOTOOL
00195
       fprintf (stderr, "mpcotool: freeing memory and closing\n");
00196 #endif
00197
       optimize free ():
```

4.20 mpcotool.h 225

```
00198
00199 #endif
00200
00201
       // Closing MPI
00202 #if HAVE_MPI
00203
       MPI Finalize ():
00204 #endif
00205
00206
       // Freeing memory
00207
       gsl_rng_free (optimize->rng);
00208
00209
       // Closing
00210
       return 0;
00211 }
```

4.20 mpcotool.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.
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 #ifndef MPCOTOOL_
00039 #define MPCOTOOL H 1
00041 extern int mpcotool (int argn, char **argc);
00042
00043 #endif
```

4.21 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 <gsl/gsl_rng.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/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 CP "cp"

Macro to define the shell copy command.

#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_best ()
- static void optimize climbing (unsigned int nsteps)
- 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 ()
- · static void optimize save optimal ()
- 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.

4.21.1 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.

4.21.2 Macro Definition Documentation

4.21.2.1 CP

```
#define CP "cp"
```

Macro to define the shell copy command.

Definition at line 79 of file optimize.c.

4.21.2.2 DEBUG_OPTIMIZE

```
#define DEBUG_OPTIMIZE 0
```

Macro to debug optimize functions.

Definition at line 67 of file optimize.c.

4.21.2.3 RM

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

Macro to define the shell remove command.

Definition at line 80 of file optimize.c.

4.21.3 Function Documentation

4.21.3.1 optimize best()

Function to save the best simulations.

Parameters

simulation	Simulation number.
value	Objective function value.

Definition at line 478 of file optimize.c.

```
00481
         unsigned int i, j;
00482
        double e;
00483 #if DEBUG_OPTIMIZE
00484 fprintf (stderr, "optimize_best: start\n");
00485 fprintf (stderr, "optimize_best: nsaveds=%u nbest=%u\n",
                    optimize->nsaveds, optimize->nbest);
00486
00488
        if (optimize->nsaveds < optimize->nbest
00489
              || value < optimize->error_best[optimize->nsaveds - 1])
00490
00491
             if (optimize->nsaveds < optimize->nbest)
               ++optimize->nsaveds;
00492
             optimize->error_best[optimize->nsaveds - 1] = value;
optimize->simulation_best[optimize->nsaveds - 1] = simulation;
00493
00494
00495
              for (i = optimize->nsaveds; --i;)
00496
                   if (optimize->error_best[i] < optimize->error_best[i - 1])
00497
00498
                     {
00499
                       j = optimize->simulation_best[i];
00500
                       e = optimize->error_best[i];
                       optimize->simulation_best[i] = optimize->simulation_best[i - 1];
00501
                       optimize->error_best[i] = optimize->error_best[i - 1];
optimize->simulation_best[i - 1] = j;
optimize->error_best[i - 1] = e;
00502
00503
00504
00505
                    }
00506
                  else
00507
00508
                }
00509
00510 #if DEBUG_OPTIMIZE
00511 fprintf (stderr, "optimize_best: end\n");
00512 #endif
00513 }
```

4.21.3.2 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 847 of file optimize.c.

```
00850 #if DEBUG_OPTIMIZE
00851
        fprintf (stderr, "optimize_best_climbing: start\n");
00852
        fprintf (stderr,
                  "optimize_best_climbing: simulation=%u value=%.14le best=%.14le\n",
00853
                 simulation, value, optimize->error_best[0]);
00854
00855 #endif
       if (value < optimize->error_best[0])
00857
00858
            optimize->error_best[0] = value;
            optimize->simulation_best[0] = simulation;
00859
00860 #if DEBUG OPTIMIZE
00861
           fprintf (stderr,
                      "optimize_best_climbing: BEST simulation=%u value=%.14le\n",
00863
                     simulation, value);
00864 #endif
00865
00866 #if DEBUG_OPTIMIZE
00867 fprintf (stderr, "optimize_best_climbing: end\n");
00868 #endif
00869 }
```

4.21.3.3 optimize_climbing()

```
static void optimize_climbing (
          unsigned int nsteps ) [inline], [static]
```

Function to optimize with a hill climbing method.

Parameters

```
nsteps Number of steps.
```

Definition at line 1092 of file optimize.c.

```
01093 {
01094 unsigned int i, j, k, b, s, adjust; 01095 #if DEBUG_OPTIMIZE
01096
       fprintf (stderr, "optimize_climbing: start\n");
01097 #endif
01098
       for (i = 0; i < optimize->nvariables; ++i)
         optimize->climbing[i] = 0.;
01099
        b = optimize->simulation_best[0] * optimize->nvariables;
01100
       s = optimize->nsimulations;
01101
       adjust = 1;
for (i = 0; i < nsteps; ++i, s += optimize->nestimates, b = k)
01102
01103
01104
01105 #if DEBUG_OPTIMIZE
01106
            fprintf (stderr, "optimize_climbing: step=%u old_best=%u\n",
01107
                      i, optimize->simulation_best[0]);
```

```
01108 #endif
      optimize_step_climbing (s);
O1110 k = optimize->simulation_best[0] * optimize->nvariables;
O1111 #if DEBUG_OPTIMIZE
01112 fprintf (stderr, "optimize_climbing: step=%u best=%u\n",
                     i, optimize->simulation_best[0]);
01113
01114 #endif
01115
        if (k == b)
01116
01117
                if (adjust)
                 for (j = 0; j < optimize->nvariables; ++j)
01118
                optimize->step[j] *= 0.5;
for (j = 0; j < optimize->nvariables; ++j)
01119
01120
01121
                  optimize->climbing[j] = 0.;
01122
                adjust = 1;
01123
            else
01124
01125
              {
01126
                for (j = 0; j < optimize->nvariables; ++j)
01128 #if DEBUG_OPTIMIZE
01129
                    fprintf (stderr,
                              "optimize_climbing: best%u=%.14le old%u=%.14le\n",
01130
01131
                              j, optimize->value[k + j], j, optimize->value[b + j]);
01132 #endif
                   optimize->climbing[j]
01133
                     = (1. - optimize->relaxation) * optimize->climbing[j] + optimize->relaxation
01134
01135
01136
                       * (optimize->value[k + j] - optimize->value[b + j]);
01137 #if DEBUG_OPTIMIZE
01138
                   fprintf (stderr, "optimize_climbing: climbing%u=%.14le\n",
01139
                              j, optimize->climbing[j]);
01140 #endif
01141
01142
               adjust = 0;
              }
01143
01144
01145 #if DEBUG_OPTIMIZE
01146
       fprintf (stderr, "optimize_climbing: end\n");
01147 #endif
01148 }
```

Here is the call graph for this function:

4.21.3.4 optimize_climbing_best()

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

Function to select the best simulation to start the hill climbing method.

Definition at line 1075 of file optimize.c.

```
01076 {
01077 #if DEBUG_OPTIMIZE
01078 fprintf (stderr, "optimize_climbing_best: start\n");
01079 #endif
01080 optimize->simulation_best[0] = 0;
01081 memcpy (optimize->value, optimize->value_old,
01082 optimize->nvariables * sizeof (double));
01083 #if DEBUG_OPTIMIZE
01084 fprintf (stderr, "optimize_climbing_best: end\n");
01085 #endif
01086 }
```

4.21.3.5 optimize climbing sequential()

Function to estimate the hill climbing sequentially.

Parameters

simulation Simulation number.

Definition at line 875 of file optimize.c.

```
00876 {
00877
        double e;
00878
        unsigned int i, j;
00879 #if DEBUG_OPTIMIZE
00880 fprintf (stderr, "optimize_climbing_sequential: start\n");
00881 fprintf (stderr, "optimize_climbing_sequential: nstart_climbing=%u "
00882
                  "nend_climbing=%u\n",
00883
                  optimize->nstart_climbing, optimize->nend_climbing);
00884 #endif
00885
       for (i = optimize->nstart_climbing; i < optimize->nend_climbing; ++i)
00887
            j = simulation + i;
00888
             e = optimize_norm (j);
00889
            optimize_best_climbing (j, e);
00890
            optimize_save_variables (j, e);
            if (e < optimize->threshold)
00891
00892
00893
                 optimize->stop = 1;
00894
00895
00896 #if DEBUG_OPTIMIZE
            fprintf (stderr, "optimize_climbing_sequential: i=%u e=%lg\n", i, e);
00897
00898 #endif
00900 #if DEBUG_OPTIMIZE
00901 fprintf (stderr, "optimize_climbing_sequential: end\n");
00902 #endif
00903 }
```

Here is the call graph for this function:

4.21.3.6 optimize_climbing_thread()

Function to estimate the hill climbing on a thread.

Returns

NULL

Parameters

data | Function data.

Definition at line 911 of file optimize.c.

```
00912 {
00913
       unsigned int i, thread;
00914 double e;
00915 #if DEBUG_OPTIMIZE
00916
       fprintf (stderr, "optimize_climbing_thread: start\n");
00917 #endif
       thread = data->thread;
00919 #if DEBUG_OPTIMIZE
00920 fprintf (stderr, "optimize_climbing_thread: thread=%u start=%u end=%u\n",
00921
                 thread,
00922
                 optimize->thread_climbing[thread],
00923
                 optimize->thread_climbing[thread + 1]);
00924 #endif
00925    for (i = optimize->thread_climbing[thread];
00926
             i < optimize->thread_climbing[thread + 1]; ++i)
```

```
{
00928
         e = optimize_norm (i);
00929
           g_mutex_lock (mutex);
00930
           optimize_best_climbing (i, e);
00931
           optimize_save_variables (i, e);
          if (e < optimize->threshold)
00932
            optimize->stop = 1;
00934
           g_mutex_unlock (mutex);
00935
           if (optimize->stop)
00936
             break;
00937 #if DEBUG_OPTIMIZE
           fprintf (stderr, "optimize_climbing_thread: i=%u e=%lg\n", i, e);
00938
00939 #endif
00940
00941 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_climbing_thread: end\n");
00942
00943 #endif
00944 g_thread_exit (NULL);
       return NULL;
00946 }
```

4.21.3.7 optimize_estimate_climbing_coordinates()

Function to estimate a component of the hill climbing vector.

Parameters

variable	Variable number.
estimate	Estimate number.

Definition at line 976 of file optimize.c.

```
00980 {
00981 double x;
00982 #if DEBUG_OPTIMIZE
00983
        fprintf (stderr, "optimize_estimate_climbing_coordinates: start\n");
00984 #endif
00985
       x = optimize->climbing[variable];
00986
        if (estimate >= (2 * variable) && estimate < (2 * variable + 2))</pre>
00987
            if (estimate & 1)
00988
00989
              x += optimize->step[variable];
            else
00990
00991
              x -= optimize->step[variable];
00992
00993 #if DEBUG_OPTIMIZE
00994 fprintf (stderr,
00995
                  "optimize_estimate_climbing_coordinates: climbing%u=%lg\n",
       variable, x);
fprintf (stderr, "optimize_estimate_climbing_coordinates: end\n");
00997
00998 #endif
00999
       return x;
01000 }
```

4.21.3.8 optimize estimate climbing random()

Function to estimate a component of the hill climbing vector.

Parameters

variable	Variable number.
estimate	Estimate number.

Definition at line 952 of file optimize.c.

```
00957 {
00958     double x;
00959     #if DEBUG_OPTIMIZE
00960     fprintf (stderr, "optimize_estimate_climbing_random: start\n");
00961     #endif
00962     x = optimize->climbing[variable]
00963     + (1. - 2. * gsl_rng_uniform (optimize->rng)) * optimize->step[variable];
00964     #if DEBUG_OPTIMIZE
00965     fprintf (stderr, "optimize_estimate_climbing_random: climbing*u=%lg\n",
00966     variable, x);
00967     fprintf (stderr, "optimize_estimate_climbing_random: end\n");
00968     #endif
00969     return x;
00970 }
```

4.21.3.9 optimize_free()

```
void optimize_free ( )
```

Function to free the memory used by the Optimize struct.

Definition at line 1487 of file optimize.c.

```
01488
01489
        unsigned int i, j;
01490 #if DEBUG_OPTIMIZE
01491
        fprintf (stderr, "optimize_free: start\n");
01492 #endif
01493 for (j = 0; j < optimize -> ninputs; ++j)
01494
            for (i = 0; i < optimize->nexperiments; ++i)
01495
               g_mapped_file_unref (optimize->file[j][i]);
01496
01497
             g_free (optimize->file[j]);
01498
01499 g_free (optimize->error_old);
01500 g_free (optimize->value_old);
01501 g_free (optimize->value);
01502 g_free (optimize->genetic_variable);
01503 #if DEBUG_OPTIMIZE
01504 fprintf (stderr, "optimize_free: end\n");
01505 #endif
01506 }
```

4.21.3.10 optimize_genetic()

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

Function to optimize with the genetic algorithm.

Definition at line 1189 of file optimize.c.

```
01199
                 "optimize_genetic: nvariables=%u population=%u generations=%u\n",
01200
                 optimize->nvariables, optimize->nsimulations, optimize->niterations);
01201
        fprintf (stderr,
01202
                  "optimize_genetic: mutation=%lg reproduction=%lg adaptation=%lg\n",
01203
                 optimize->mutation_ratio, optimize->reproduction_ratio,
optimize->adaptation_ratio);
01204
01206
        genetic_algorithm_default (optimize->nvariables,
01207
                                    optimize->genetic_variable,
01208
                                    optimize->nsimulations,
                                    optimize->niterations,
01209
01210
                                    optimize->mutation ratio.
01211
                                    optimize->reproduction_ratio,
01212
                                     optimize->adaptation_ratio,
01213
                                     optimize->seed,
01214
                                     optimize->threshold,
01215
                                     &optimize_genetic_objective,
01216
                                    &best_genome, &best_variable, &best_objective);
01217 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_genetic: the best\n");
01218
01219 #endif
01220
       optimize->error_old = (double *) g_malloc (sizeof (double));
       optimize->value_old
01221
         = (double *) g_malloc (optimize->nvariables * sizeof (double));
01222
01223
       optimize->error_old[0] = best_objective;
01224 memcpy (optimize->value_old, best_variable,
01225
                optimize->nvariables * sizeof (double));
01226
       g_free (best_genome);
01227
       g_free (best_variable);
01228
01228    optimize_print ();
01229 #if DEBUG_OPTIMIZE
01230
       fprintf (stderr, "optimize_genetic: end\n");
01231 #endif
01232 }
```

4.21.3.11 optimize_genetic_objective()

Function to calculate the objective function of an entity.

Returns

objective function value.

Parameters

entity entity data.

Definition at line 1156 of file optimize.c.

```
01157 {
01158
        unsigned int j;
01159
        double objective;
01160
        char buffer[64];
01161 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_genetic_objective: start\n");
01162
01163 #endif
       for (j = 0; j < optimize->nvariables; ++j)
01164
01165
01166
            optimize->value[entity->id * optimize->nvariables + j]
01167
              = genetic_get_variable (entity, optimize->genetic_variable + j);
01168
        objective = optimize_norm (entity->id);
01169
01170
        g_mutex_lock (mutex);
01171
        for (j = 0; j < optimize->nvariables; ++j)
01172
01173
            snprintf (buffer, 64, "%s ", format[optimize->precision[j]]);
            fprintf (optimize->file_variables, buffer,
01174
```

Here is the call graph for this function:

4.21.3.12 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 99 of file optimize.c.

```
00102 {
00103
        char buffer[256], value[32];
00104
       GRegex *regex;
00105
00106
       char *buffer2, *buffer3 = NULL, *content;
00107
       gsize length;
00108
       unsigned int i;
00109
00110 #if DEBUG_OPTIMIZE
00111 fprintf (stderr, "optimize_input: start\n");
00112 #endif
00113
       // Checking the file
00114
       if (!stencil)
00115
00116
         goto optimize_input_end;
00117
00118
       // Opening stencil
00119
       content = g_mapped_file_get_contents (stencil);
00120
       length = g_mapped_file_get_length (stencil);
00121 #if DEBUG_OPTIMIZE
00122
       fprintf (stderr, "optimize_input: length=%lu\ncontent:\n%s", length, content);
00123 #endif
00124
       file = g_fopen (input, "w");
00125
       // Parsing stencil
00126
00127
       for (i = 0; i < optimize->nvariables; ++i)
00128
00129 #if DEBUG_OPTIMIZE
00130
           fprintf (stderr, "optimize_input: variable=%u\n", i);
00131 #endif
00132
           snprintf (buffer, 32, "@variable%u@", i + 1);
           regex = g_regex_new (buffer, (GRegexCompileFlags) 0, (GRegexMatchFlags) 0,
00133
00134
                                NULL);
00135
            if (i == 0)
00136
             {
00137
               buffer2 = g_regex_replace_literal (regex, content, length, 0,
00138
                                                   optimize->label[i]
                                                   (GRegexMatchFlags) 0, NULL);
00139
00140 #if DEBUG_OPTIMIZE
00141
               fprintf (stderr, "optimize_input: buffer2\n%s", buffer2);
00142 #endif
00143
00144
            else
00145
              {
```

```
length = strlen (buffer3);
00147
               buffer2 = g_regex_replace_literal (regex, buffer3, length, 0,
                                                   optimize->label[i],
00148
                                                   (GRegexMatchFlags) 0, NULL);
00149
00150
               g_free (buffer3);
00151
00152
            g_regex_unref (regex);
00153
            length = strlen (buffer2);
00154
            snprintf (buffer, 32, "@value%u@", i + 1);
00155
            regex = g_regex_new (buffer, (GRegexCompileFlags) 0, (GRegexMatchFlags) 0,
00156
                                 NULL);
            snprintf (value, 32, format[optimize->precision[i]],
00157
00158
                      optimize->value[simulation * optimize->nvariables + i]);
00159
00161 fprintf (stderr, "optimize_input: value=%s\n", value);
00162 #endif
00160 #if DEBUG_OPTIMIZE
00163
           buffer3 = g_regex_replace_literal (regex, buffer2, length, 0, value,
                                               (GRegexMatchFlags) 0, NULL);
00164
00165
            g_free (buffer2);
00166
           g_regex_unref (regex);
        }
00167
00168
       // Saving input file
00169
00170
       fwrite (buffer3, strlen (buffer3), sizeof (char), file);
00171
       g_free (buffer3);
00172
       fclose (file);
00173
00174 optimize_input_end:
00175 #if DEBUG_OPTIMIZE
00176
       fprintf (stderr, "optimize_input: end\n");
00177 #endif
00178 return;
00179 }
```

4.21.3.13 optimize_iterate()

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

Function to iterate the algorithm.

```
Definition at line 1420 of file optimize.c.
```

```
01421 {
01422
       unsigned int i:
01423 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_iterate: start\n");
01424
01425 #endif
01426
      optimize->error_old = (double *) g_malloc (optimize->nbest * sizeof (double));
       optimize->value_old =
01427
01428
          (double *) g_malloc (optimize->nbest * optimize->nvariables *
01429
                                sizeof (double));
01430
       optimize step ():
01431
       optimize_save_old ();
        optimize_refine ();
01432
       optimize_print ();
for (i = 1; i < optimize->niterations && !optimize->stop; ++i)
01433
01434
01435
01436
            optimize_step ();
01437
            optimize_merge_old ();
            optimize_refine ();
01438
01439
            optimize_print ();
01440
        if (optimize->nfinal_steps && !optimize->stop)
01441
01442
        {
            optimize climbing best ();
01443
            optimize_climbing (optimize->nfinal_steps);
optimize_merge_old ();
01445
01446
            optimize_print ();
01447
01448 #if DEBUG_OPTIMIZE
01449
       fprintf (stderr, "optimize_iterate: end\n");
01450 #endif
01451 }
```

Here is the call graph for this function:

4.21.3.14 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 591 of file optimize.c.

```
00596 {
00597
        unsigned int i, j, k, s[optimize->nbest];
00598 double e[optimize->nbest]; 00599 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_merge: start\n");
00600
00601 #endif
00602
       i = j = k = 0;
00603
        do
00604
            if (i == optimize->nsaveds)
00605
00606
              {
                s[k] = simulation_best[j];
00607
00608
                e[k] = error_best[j];
00609
00610
                ++k;
00611
                if (j == nsaveds)
00612
                  break:
00613
00614
            else if (j == nsaveds)
00615
              {
00616
                s[k] = optimize->simulation_best[i];
00617
                e[k] = optimize->error_best[i];
00618
                ++i;
00619
                ++k;
00620
                if (i == optimize->nsaveds)
00621
00622
00623
            else if (optimize->error_best[i] > error_best[j])
00624
              {
                s[k] = simulation_best[j];
00625
00626
                e[k] = error_best[j];
00627
                ++j;
00628
                ++k;
00629
              }
00630
            else
00631
              {
00632
                s[k] = optimize->simulation_best[i];
00633
                e[k] = optimize->error_best[i];
00634
                ++i;
00635
                ++k;
00636
00637
       while (k < optimize->nbest);
00638
       optimize->nsaveds = k;
00640
        memcpy (optimize->simulation_best, s, k * sizeof (unsigned int));
00641 memcpy (optimize->error_best, e, k * sizeof (double)); 00642 #if DEBUG_OPTIMIZE
00643 fprintf (stderr, "optimize_merge: end\n");
00644 #endif
00645 }
```

4.21.3.15 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 1270 of file optimize.c.

```
01271
01272
         unsigned int i, j, k;
01273
         double v[optimize->nbest * optimize->nvariables], e[optimize->nbest],
01274
           *enew, *eold;
01275 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_merge_old: start\n");
01276
01277 #endif
01278
        enew = optimize->error_best;
         eold = optimize->error_old;
01280
        i = j = k = 0;
01281
         do
01282
          {
             if (*enew < *eold)</pre>
01283
01284
               {
01285
                 memcpy (v + k * optimize->nvariables,
                          optimize->value
01286
01287
                           + optimize->simulation_best[i] * optimize->nvariables,
01288
                           optimize->nvariables * sizeof (double));
                 e[k] = *enew;
01289
01290
                  ++k;
01291
                  ++enew;
01292
                  ++i;
01293
01294
             else
01295
               {
                  memcpy (v + k * optimize->nvariables,
01296
                           optimize=>value_old + j * optimize=>nvariables,
optimize=>nvariables * sizeof (double));
01297
01298
01299
                  e[k] = *eold;
01300
                  ++k;
01301
                 ++eold;
01302
                  ++j;
01303
               }
01304
        while (k < optimize->nbest);
01305
01306
       memcpy (optimize->value_old, v, k * optimize->nvariables * sizeof (double));
01307 memcpy (optimize->error_old, e, k * sizeof (double)); 01308 #if DEBUG_OPTIMIZE 01309 fprintf (stderr, "optimize_merge_old: end\n");
01310 #endif
01311 }
```

4.21.3.16 optimize MonteCarlo()

static void optimize_MonteCarlo () [static]

Function to optimize with the Monte-Carlo algorithm.

Definition at line 752 of file optimize.c.

```
00753 {
00754
         ParallelData data[nthreads];
00755
         GThread *thread[nthreads];
00756
         double range[optimize->nvariables];
00757
         unsigned int i, j;
00758 #if DEBUG_OPTIMIZE
         fprintf (stderr, "optimize_MonteCarlo: start\n");
00759
00760 #endif
        for (j = 0; j < optimize->nvariables; ++j)
  range[j] = optimize->rangemax[j] - optimize->rangemin[j];
for (i = 0; i < optimize->nsimulations; ++i)
00761
00762
00763
           for (j = 0; j < optimize->nvariables; ++j)
00764
00765
             optimize->value[i * optimize->nvariables + j]
00766
                = optimize->rangemin[j] + gsl_rng_uniform (optimize->rng) * range[j];
         optimize->nsaveds = 0;
00767
00768
         if (nthreads <= 1)
00769
          optimize_sequential ();
00770
         else
00771
          {
00772
              for (i = 0; i < nthreads; ++i)</pre>
00773
00774
                 data[i].thread = i;
00775
                  thread[i]
00776
                    = g_thread_new (NULL, (GThreadFunc) optimize_thread, &data[i]);
00777
```

4.21.3.17 optimize_norm_euclidian()

Function to calculate the Euclidian error norm.

Returns

Euclidian error norm.

Parameters

simulation simulation number.

Definition at line 326 of file optimize.c.

```
00327 {
00328 double e, ei;
00329 unsigned int i;
00330 #if DEBUG_OPTIMIZE
00331
         fprintf (stderr, "optimize_norm_euclidian: start\n");
00332 #endif
00333 e = 0.;
        for (i = 0; i < optimize->nexperiments; ++i)
00334
00335
              ei = optimize_parse (simulation, i);
00336
             e += ei * ei;
00337
00338
           }
00339 e = sqrt (e);
00340 #if DEBUG_OPTIMIZE
00341 fprintf (stderr, "optimize_norm_euclidian: error=%lg\n", e); 00342 fprintf (stderr, "optimize_norm_euclidian: end\n");
00343 #endif
00344 return e;
00345 }
```

Here is the call graph for this function:

4.21.3.18 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 353 of file optimize.c.

```
00354 {
double e, ei;
00356 unsigned int i;
00357 #if DEBUG_OPTIMIZE
00358 fprintf (stderr, "optimize_norm_maximum: start\n");
00359 #endif
00360 e = 0.;
         for (i = 0; i < optimize->nexperiments; ++i)
00361
00362
              ei = fabs (optimize_parse (simulation, i));
e = fmax (e, ei);
00363
00364
00365
00366 #if DEBUG_OPTIMIZE
00367 fprintf (stderr, "optimize_norm_maximum: error=%lg\n", e); 00368 fprintf (stderr, "optimize_norm_maximum: end\n");
00369 #endif
00370 return e;
00371 }
```

Here is the call graph for this function:

4.21.3.19 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 379 of file optimize.c.

```
00380 {
          double e, ei;
00381
         unsigned int i;
00382
00383 #if DEBUG_OPTIMIZE
00384
         fprintf (stderr, "optimize_norm_p: start\n");
00385 #endif
00386 e = 0.;
         for (i = 0; i < optimize->nexperiments; ++i)
00387
00388
          {
00389
              ei = fabs (optimize_parse (simulation, i));
00390
              e += pow (ei, optimize->p);
00391
00392 e = pow (e, 1. / optimize->p);

00393 #if DEBUG_OPTIMIZE

00394 fprintf (stderr, "optimize_norm_p: error=%lg\n", e);

00395 fprintf (stderr, "optimize_norm_p: end\n");
00396 #endif
00397
         return e;
00398 }
```

Here is the call graph for this function:

4.21.3.20 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 406 of file optimize.c.

```
00407
00408
         double e:
00409
          unsigned int i;
00410 #if DEBUG_OPTIMIZE
00411
         fprintf (stderr, "optimize_norm_taxicab: start\n");
00412 #endif
         e = 0.;
00413
         for (i = 0; i < optimize->nexperiments; ++i)
  e += fabs (optimize_parse (simulation, i));
00414
00416 #if DEBUG_OPTIMIZE
00417 fprintf (stderr, "optimize_norm_taxicab: error=%lg\n", e);
00418 fprintf (stderr, "optimize_norm_taxicab: end\n");
00419 #endif
00420
         return e;
00421 }
```

Here is the call graph for this function:

4.21.3.21 optimize_open()

```
void optimize_open ( )
```

Function to open and perform a optimization.

Definition at line 1512 of file optimize.c.

```
01513 {
01514
        GTimeZone *tz;
       GDateTime *t0, *t;
01516
       unsigned int i, j, nsteps;
01517
01518 #if DEBUG_OPTIMIZE
01519 char *buffer;
01520 fprintf (stde:
       fprintf (stderr, "optimize_open: start\n");
01521 #endif
01522
01523
        \ensuremath{//} Getting initial time
01524 #if DEBUG_OPTIMIZE
01525
       fprintf (stderr, "optimize_open: getting initial time\n");
01526 #endif
       tz = g_time_zone_new_utc ();
01528 t0 = g_date_time_new_now (tz);
01529
01530
        \ensuremath{//} Obtaining and initing the pseudo-random numbers generator seed
01531 #if DEBUG_OPTIMIZE
01532
       fprintf (stderr, "optimize_open: getting initial seed\n");
01533 #endif
01534 if (optimize->seed == DEFAULT_RANDOM_SEED)
01535
         optimize->seed = input->seed;
01536
       gsl_rng_set (optimize->rng, optimize->seed);
01537
01538
       // Obtaining template flags
01539 #if DEBUG_OPTIMIZE
```

```
fprintf (stderr, "optimize_open: getting template flags\n");
01541 #endif
01542
        optimize->template_flags = input->template_flags;
01543
01544
        // Replacing the working directory
01545 #if DEBUG_OPTIMIZE
01546
       fprintf (stderr, "optimize_open: replacing the working directory\n");
01547 #endif
01548
        g_chdir (input->directory);
01549
01550
        // Getting results file names
01551
        optimize->result = input->result;
01552
        optimize->variables = input->variables;
01553
01554
        // Obtaining the simulator file
01555
        optimize->simulator = input->simulator;
01556
01557
        // Obtaining the evaluator file
01558
        optimize->evaluator = input->evaluator;
01559
01560
        // Obtaining the cleaner file
01561
        optimize->cleaner = input->cleaner;
01562
01563
        // Reading the algorithm
01564
        optimize->algorithm = input->algorithm;
        switch (optimize->algorithm)
01565
01566
01567
          case ALGORITHM_MONTE_CARLO:
01568
            optimize_algorithm = optimize_MonteCarlo;
01569
            break:
01570
          case ALGORITHM_SWEEP:
01571
           optimize_algorithm = optimize_sweep;
01572
            break;
01573
          case ALGORITHM_ORTHOGONAL:
01574
            optimize_algorithm = optimize_orthogonal;
01575
            break;
01576
          default:
01577
           optimize_algorithm = optimize_genetic;
01578
            optimize->mutation_ratio = input->mutation_ratio;
01579
            optimize->reproduction_ratio = input->reproduction_ratio;
01580
            optimize->adaptation_ratio = input->adaptation_ratio;
01581
        optimize->nvariables = input->nvariables;
01582
01583
        optimize->nsimulations = input->nsimulations;
        optimize->niterations = input->niterations;
01584
01585
        optimize->nbest = input->nbest;
01586
        optimize->tolerance = input->tolerance;
        optimize->nsteps = input->nsteps;
optimize->nfinal_steps = input->nfinal_steps;
01587
01588
01589
        nsteps = JBM_MAX (optimize->nsteps, optimize->nfinal_steps);
        optimize->nestimates = 0;
01591
        optimize->threshold = input->threshold;
01592
        optimize->stop = 0;
01593
        if (nsteps)
01594
01595
            optimize->relaxation = input->relaxation;
01596
            switch (input->climbing)
01597
01598
              case CLIMBING_METHOD_COORDINATES:
01599
                optimize->nestimates = 2 * optimize->nvariables;
                optimize_estimate_climbing = optimize_estimate_climbing_coordinates;
01600
01601
                break;
01602
              default:
               optimize->nestimates = input->nestimates;
01603
01604
                optimize_estimate_climbing = optimize_estimate_climbing_random;
              }
01605
01606
          }
01607
01608 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: nbest=%u\n", optimize->nbest);
01610 #endif
01611
       optimize->simulation_best
01612
          = (unsigned int *) alloca (optimize->nbest * sizeof (unsigned int));
        optimize->error_best = (double *) alloca (optimize->nbest * sizeof (double));
01613
01614
01615
        // Reading the experimental data
01616 #if DEBUG_OPTIMIZE
01617 buffer = g_get_current_dir ();
01618
        fprintf (stderr, "optimize_open: current directory=%s\n", buffer);
01619
        g free (buffer);
01620 #endif
01621
        optimize->nexperiments = input->nexperiments;
01622
        optimize->ninputs = input->experiment->ninputs;
        optimize->experiment
01623
01624
          = (char **) alloca (input->nexperiments * sizeof (char *));
        optimize->weight = (double *) alloca (input->nexperiments * sizeof (double));
for (i = 0; i < input->experiment->ninputs; ++i)
01625
01626
```

```
optimize->file[i] = (GMappedFile **)
            g_malloc (input->nexperiments * sizeof (GMappedFile *));
01628
01629
        for (i = 0; i < input->nexperiments; ++i)
01630
01631 #if DEBUG_OPTIMIZE
            fprintf (stderr, "optimize_open: i=%u\n", i);
01632
01633 #endif
           optimize->experiment[i] = input->experiment[i].name;
01634
01635
            optimize->weight[i] = input->experiment[i].weight;
01636 #if DEBUG OPTIMIZE
           01637
01638
01639 #endif
01640
           for (j = 0; j < input->experiment->ninputs; ++j)
01641
01642 #if DEBUG_OPTIMIZE
                fprintf (stderr, "optimize_open: stencil%u\n", j + 1);
01643
01644 #endif
01645
               optimize->file[j][i]
01646
                  = g_mapped_file_new (input->experiment[i].stencil[j], 0, NULL);
01647
01648
         }
01649
        \ensuremath{//} Reading the variables data
01650
01651 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: reading variables\n");
01652
01653 #endif
01654
       optimize->label = (char **) alloca (input->nvariables * sizeof (char *));
01655
        j = input->nvariables * sizeof (double);
        optimize->rangemin = (double *) alloca (j);
01656
01657
        optimize->rangeminabs = (double *) alloca (j);
01658
        optimize->rangemax = (double *) alloca (j);
01659
        optimize->rangemaxabs = (double *) alloca (j);
01660
        optimize->step = (double *) alloca (j);
        j = input->nvariables * sizeof (unsigned int);
optimize->precision = (unsigned int *) alloca (j);
optimize->nsweeps = (unsigned int *) alloca (j);
01661
01662
01663
        optimize->nbits = (unsigned int *) alloca (j);
01664
        for (i = 0; i < input->nvariables; ++i)
01665
01666
01667
            optimize->label[i] = input->variable[i].name;
            optimize->rangemin[i] = input->variable[i].rangemin;
01668
01669
            optimize->rangeminabs[i] = input->variable[i].rangeminabs;
01670
            optimize->rangemax[i] = input->variable[i].rangemax;
            optimize->rangemaxabs[i] = input->variable[i].rangemaxabs;
01671
01672
            optimize->precision[i] = input->variable[i].precision;
01673
            optimize->step[i] = input->variable[i].step;
01674
            optimize->nsweeps[i] = input->variable[i].nsweeps;
01675
            optimize->nbits[i] = input->variable[i].nbits;
01676
01677
        if (input->algorithm == ALGORITHM_SWEEP
01678
            || input->algorithm == ALGORITHM_ORTHOGONAL)
01679
01680
            optimize->nsimulations = 1;
            for (i = 0; i < input->nvariables; ++i)
01681
01682
             {
01683
                optimize->nsimulations *= optimize->nsweeps[i];
01684 #if DEBUG_OPTIMIZE
01685
               fprintf (stderr, "optimize_open: nsweeps=%u nsimulations=%u\n",
01686
                         optimize->nsweeps[i], optimize->nsimulations);
01687 #endif
01688
01689
        if (nsteps)
01690
        optimize->climbing
01691
01692
            = (double *) alloca (optimize->nvariables * sizeof (double));
01693
01694
        // Setting error norm
01695
       switch (input->norm)
         {
01697
          case ERROR_NORM_EUCLIDIAN:
01698
            optimize_norm = optimize_norm_euclidian;
01699
           break;
          case ERROR NORM MAXIMUM:
          optimize_norm = optimize_norm_maximum;
break;
01700
01701
01702
01703
          case ERROR_NORM_P:
01704
          optimize_norm = optimize_norm_p;
01705
            optimize->p = input->p;
01706
            break:
01707
          default:
01708
           optimize_norm = optimize_norm_taxicab;
01709
01710
01711
       // Allocating values
01712 #if DEBUG OPTIMIZE
01713
       fprintf (stderr, "optimize_open: allocating variables\n");
```

```
fprintf (stderr, "optimize_open: nvariables=%u algorithm=%u\n",
                 optimize->nvariables, optimize->algorithm);
01715
01716 #endif
01717
        optimize->genetic_variable = NULL;
        if (optimize->algorithm == ALGORITHM_GENETIC)
01718
01719
01720
            optimize->genetic_variable = (GeneticVariable *)
01721
              g_malloc (optimize->nvariables * sizeof (GeneticVariable));
01722
            for (i = 0; i < optimize->nvariables; ++i)
01723
01724 #if DEBUG_OPTIMIZE
                fprintf (stderr, "optimize_open: i=%u min=%lg max=%lg nbits=%u\n",
01725
                          i, optimize->rangemin[i], optimize->rangemax[i],
01726
01727
                          optimize->nbits[i]);
01728 #endif
               optimize->genetic_variable[i].minimum = optimize->rangemin[i];
optimize->genetic_variable[i].maximum = optimize->rangemax[i];
01729
01730
               optimize->genetic_variable[i].nbits = optimize->nbits[i];
01731
01733
01734 #if DEBUG_OPTIMIZE
01735 fprintf (stderr, "optimize_open: nvariables=%u nsimulations=%u\n",
                 optimize->nvariables, optimize->nsimulations);
01736
01737 #endif
01738 optimize->value = (double *)
01739
        g_malloc ((optimize->nsimulations + optimize->nestimates * nsteps)
01740
                     * optimize->nvariables * sizeof (double));
01741
01742
       // Calculating simulations to perform for each task
01743 #if HAVE_MPI
01744 #if DEBUG_OPTIMIZE
01745 fprintf (stderr, "optimize_open: rank=%u ntasks=%u\n",
01746
                 optimize->mpi_rank, ntasks);
01747 #endif
       optimize->nstart = optimize->mpi_rank * optimize->nsimulations / ntasks;
optimize->nend = (1 + optimize->mpi_rank) * optimize->nsimulations / ntasks;
01748
01749
01750
       if (nsteps)
01751
01752
            optimize->nstart_climbing
01753
              = optimize->mpi_rank * optimize->nestimates / ntasks;
01754
            optimize->nend_climbing
01755
              = (1 + optimize->mpi_rank) * optimize->nestimates / ntasks;
01756
01757 #else
      optimize->nstart = 0;
01758
01759
        optimize->nend = optimize->nsimulations;
01760
       if (nsteps)
01761
            optimize->nstart climbing = 0;
01762
            optimize->nend_climbing = optimize->nestimates;
01763
01764
01765 #endif
01766 #if DEBUG_OPTIMIZE
01767 fprintf (stderr, "optimize_open: nstart=%u nend=%u\n", optimize->nstart,
01768
                 optimize->nend);
01769 #endif
01770
01771
       // Calculating simulations to perform for each thread
01772
       optimize->thread
01773
          = (unsigned int *) alloca ((1 + nthreads) * sizeof (unsigned int));
        for (i = 0; i <= nthreads; ++i)</pre>
01774
01775
            optimize->thread[i] = optimize->nstart
01777
              + i * (optimize->nend - optimize->nstart) / nthreads;
01778 #if DEBUG OPTIMIZE
01779
            fprintf (stderr, "optimize_open: i=%u thread=%u\n", i,
01780
                     optimize->thread[i]);
01781 #endif
01782
01783
        if (nsteps)
01784
         optimize->thread_climbing = (unsigned int *)
01785
            alloca ((1 + nthreads_climbing) * sizeof (unsigned int));
01786
01787
       // Opening result files
       optimize->file_result = g_fopen (optimize->result, "w");
01788
01789
        optimize->file_variables = g_fopen (optimize->variables, "w");
01790
01791
        // Performing the algorithm
01792
        switch (optimize->algorithm)
01793
         {
01794
           // Genetic algorithm
          case ALGORITHM_GENETIC:
01795
           optimize_genetic ();
01796
01797
01798
           // Iterative algorithm
01799
          default:
01800
```

```
optimize_iterate ();
01802
01803
01804
        // Getting calculation time
01805
        t = g_date_time_new_now (tz);
optimize->calculation_time = 0.000001 * g_date_time_difference (t, t0);
01806
01807
        g_date_time_unref (t);
01808
        g_date_time_unref (t0);
01809
        g_time_zone_unref (tz);
        printf ("%s = %.61g s\n", _("Calculation time"), optimize->calculation_time); fprintf (optimize->file_result, "%s = %.61g s\n",
01810
01811
                  _("Calculation time"), optimize->calculation_time);
01812
01813
01814
        // Closing result files
01815
        optimize_save_optimal ();
01816
        fclose (optimize->file_variables);
01817
        fclose (optimize->file_result);
01818
01819 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_open: end\n");
01821 #endif
01822 }
```

Here is the call graph for this function:

4.21.3.22 optimize_orthogonal()

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

Function to optimize with the orthogonal sampling algorithm.

Definition at line 794 of file optimize.c.

```
00795 {
00796
        ParallelData data[nthreads];
00797
        GThread *thread[nthreads];
00798
        double range[optimize->nvariables];
00799
        double e;
00800 unsigned int i, j, k, 1; 00801 #if DEBUG_OPTIMIZE
00802
       fprintf (stderr, "optimize_orthogonal: start\n");
00803 #endif
00804
       for (j = 0; j < optimize->nvariables; ++j)
00805
         range[j] = (optimize->rangemax[j] - optimize->rangemin[j])
00806
           / optimize->nsweeps[j];
00807
        for (i = 0; i < optimize->nsimulations; ++i)
00808
           k = i:
00809
            for (j = 0; j < optimize->nvariables; ++j)
00810
00811
00812
                1 = k % optimize->nsweeps[j];
00813
               k /= optimize->nsweeps[j];
                e = optimize->rangemin[j];
00814
00815
                if (optimize->nsweeps[j] > 1)
00816
                  e += (l + gsl_rng_uniform (optimize->rng)) * range[j];
                optimize->value[i * optimize->nvariables + j] = e;
00817
00818
00819
00820
        optimize->nsaveds = 0;
00821
        if (nthreads <= 1)</pre>
00822
         optimize_sequential ();
00823
        else
00824
         {
00825
            for (i = 0; i < nthreads; ++i)</pre>
00826
00827
                data[i].thread = i;
00828
                thread[i]
                  = g_thread_new (NULL, (GThreadFunc) optimize_thread, &data[i]);
00829
00830
            for (i = 0; i < nthreads; ++i)</pre>
00832
             g_thread_join (thread[i]);
00833
00834 #if HAVE_MPI
00835 // Communicating tasks results
00836
       optimize_synchronise ();
00837 #endif
00838 #if DEBUG_OPTIMIZE
00839
       fprintf (stderr, "optimize_orthogonal: end\n");
00840 #endif
00841 }
```

4.21.3.23 optimize_parse()

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 188 of file optimize.c.

```
00190 +
00191
       char buffer[512], cinput[MAX_NINPUTS][32], output[32], result[32], *buffer2,
         *buffer3, *buffer4;
00192
00193
       FILE *file_result;
00194
       double e;
00195
       unsigned int i;
00196
       unsigned int flags = 1;
00197
00198 #if DEBUG_OPTIMIZE
     fprintf (stderr, "optimize_parse: start\n");
fprintf (stderr, "optimize_parse: simulation
00199
00200
                                           simulation=%u experiment=%u\n",
00201
                simulation, experiment);
00202 #endif
00203
00204
        // Opening input files
       for (i = 0; i < optimize->ninputs; ++i)
00206
00207
            snprintf (&cinput[i][0], 32, "input-%u-%u-%u", i, simulation, experiment);
00208 #if DEBUG_OPTIMIZE
           fprintf (stderr, "optimize_parse: i=%u input=%s\n", i, &cinput[i][0]);
00209
00210 #endif
00211
           // Checking simple copy
00212
            if (optimize->template_flags & flags)
00213
             optimize_input (simulation, &cinput[i][0],
                              optimize->file[i][experiment]);
00214
00215
00216
               buffer2 = input->experiment[experiment].stencil[i];
snprintf (buffer, 512, CP " %s %s", buffer2, &cinput[i][0]);
00217
00218
00219
                if (system (buffer) == -1)
00220
                 error_message = g_strdup (buffer);
00221
              }
00222
            flags «= 1;
00223
00224
       for (; i < MAX_NINPUTS; ++i)</pre>
00225
         strcpy (&cinput[i][0], "");
00226 #if DEBUG_OPTIMIZE
00227
       fprintf (stderr, "optimize_parse: parsing end\n");
00228 #endif
00229
00230
       // Performing the simulation
00231
        snprintf (output, 32, "output-%u-%u", simulation, experiment);
       buffer2 = g_path_get_dirname (optimize->simulator);
00232
00233
       buffer3 = g_path_get_basename (optimize->simulator);
       00234
00235
00236
00237
00238
       g_free (buffer4);
00239
       g_free (buffer3);
00240
        g_free (buffer2);
00241 #if DEBUG_OPTIMIZE
00242
       fprintf (stderr, "optimize_parse: %s\n", buffer);
00243 #endif
      if (system (buffer) == -1)
00244
00245
         error_message = g_strdup (buffer);
```

```
00247
        // Checking the objective value function
00248
        if (optimize->evaluator)
00249
        {
            snprintf (result, 32, "result-%u-%u", simulation, experiment);
00250
            buffer2 = g_path_get_dirname (optimize->evaluator);
buffer3 = g_path_get_basename (optimize->evaluator);
00251
00253
            buffer4 = g_build_filename (buffer2, buffer3, NULL);
00254
           snprintf (buffer, 512, "\"%s\" %s %s %s",
00255
                       buffer4, output, optimize->experiment[experiment], result);
            g_free (buffer4);
00256
00257
            g_free (buffer3);
00258
             g_free (buffer2);
00259 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_parse: %s\n", buffer);
fprintf (stderr, "optimize_parse: result=%s\n", result);
00260
00261
00262 #endif
           if (system (buffer) == -1)
00263
               error_message = g_strdup (buffer);
            file_result = g_fopen (result, "r");
00266
            e = atof (fgets (buffer, 512, file_result));
00267
            fclose (file_result);
00268
00269
       else
00270
00271 #if DEBUG_OPTIMIZE
00272
             fprintf (stderr, "optimize_parse: output=%s\n", output);
00273 #endif
            strcpy (result, "");
00274
            file_result = g_fopen (output, "r");
e = atof (fgets (buffer, 512, file_result));
00275
00276
00277
            fclose (file_result);
00278
00279
00280
        // Removing files
00281
        if (optimize->cleaner)
00282
        {
            buffer2 = g_path_get_dirname (optimize->cleaner);
00284
            buffer3 = g_path_get_basename (optimize->cleaner);
00285
            buffer4 = g_build_filename (buffer2, buffer3, NULL);
00286
            snprintf (buffer, 512, "\"%s\"", buffer4);
            g_free (buffer4);
00287
00288
            g free (buffer3);
00289
            g_free (buffer2);
00290
            if (system (buffer) == -1)
               error_message = g_strdup (buffer);
00291
00292
00293 #if !DEBUG_OPTIMIZE
        for (i = 0; i < optimize->ninputs; ++i)
00294
00295
            if (optimize->file[i][0])
00297
00298
                snprintf (buffer, 512, RM " %s", &cinput[i][0]);
00299
                 if (system (buffer) == -1)
00300
                   error_message = g_strdup (buffer);
00301
              }
00303
        snprintf (buffer, 512, RM " %s %s", output, result);
        if (system (buffer) == -1)
00304
00305
          error_message = g_strdup (buffer);
00306 #endif
00307
00308
        // Processing pending events
00309
       if (show_pending)
00310
          show_pending ();
00311
00312 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_parse: end\n");
00313
00314 #endif
00316
        // Returning the objective function
00317
        return e * optimize->weight[experiment];
00318 }
```

Here is the call graph for this function:

4.21.3.24 optimize_print()

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

Function to print the results.

Definition at line 427 of file optimize.c.

```
00429
         unsigned int i;
00430
         char buffer[512];
00431 #if HAVE MPI
         if (optimize->mpi_rank)
00432
00433
            return;
00434 #endif
        printf ("%s\n", _("Best result"));
fprintf (optimize->file_result, "%s\n", _("Best result"));
printf ("error = %.15le\n", optimize->error_old[0]);
00435
00436
00437
         fprintf (optimize->file_result, "error = %.15le\n", optimize->error_old[0]);
00438
00439
         for (i = 0; i < optimize->nvariables; ++i)
00440
00441
               snprintf (buffer, 512, "%s = %s\n",
              optimize->label[i], format[optimize->precision[i]]);
printf (buffer, optimize->value_old[i]);
00442
00443
00444
              fprintf (optimize->file_result, buffer, optimize->value_old[i]);
         fflush (optimize->file_result);
00447 }
```

4.21.3.25 optimize_refine()

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

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

Definition at line 1318 of file optimize.c.

```
01319 {
01320
        unsigned int i, j;
01321 double d;
01322 #if HAVE_MPI
      MPI_Status mpi_stat;
01323
01324 #endif
01325 #if DEBUG_OPTIMIZE
01326 fprintf (stderr, "optimize_refine: start\n");
01327 #endif
01328 #if HAVE_MPI
      if (!optimize->mpi_rank)
01330
01331 #endif
01332
            for (j = 0; j < optimize->nvariables; ++j)
01333
01334
                optimize->rangemin[j] = optimize->rangemax[j]
                - optimize->value_old[j];
optimize->step[j] = input->variable[j].step;
01335
01336
01337
01338
            for (i = 0; ++i < optimize->nbest;)
01339
01340
                for (j = 0; j < optimize->nvariables; ++j)
01341
01342
                    optimize->rangemin[j]
01343
                      = fmin (optimize->rangemin[j],
                               optimize->value_old[i * optimize->nvariables + j]);
01344
                    optimize->rangemax[j]
01345
01346
                      = fmax (optimize->rangemax[j],
01347
                               optimize->value_old[i * optimize->nvariables + j]);
                  }
01349
01350
            for (j = 0; j < optimize->nvariables; ++j)
01351
                d = optimize->tolerance
01352
01353
                  * (optimize->rangemax[j] - optimize->rangemin[j]);
01354
                switch (optimize->algorithm)
01355
                  case ALGORITHM_MONTE_CARLO:
01356
01357
                    d *= 0.5;
01358
                    break:
01359
                  default:
01360
                   if (optimize->nsweeps[j] > 1)
01361
                      d /= optimize->nsweeps[j] - 1;
01362
                    else
01363
                      d = 0.;
01364
01365
                optimize->rangemin[i] -= d:
01366
                optimize->rangemin[j]
01367
                  = fmax (optimize->rangemin[j], optimize->rangeminabs[j]);
```

```
optimize->rangemax[j] += d;
01369
                 optimize->rangemax[j]
                 = fmin (optimize->rangemax[j], optimize->rangemaxabs[j]);
printf ("%s min=%lg max=%lg\n", optimize->label[j],
01370
01371
                 optimize->rangemin[j], optimize->rangemax[j]);
fprintf (optimize->file_result, "%s min=%lg max=%lg\n",
01372
01373
01374
                           optimize->label[j], optimize->rangemin[j],
01375
                           optimize->rangemax[j]);
01376
01377 #if HAVE_MPI
01378
             for (i = 1; (int) i < ntasks; ++i)</pre>
01379
01380
                 MPI_Send (optimize->rangemin, optimize->nvariables, MPI_DOUBLE, i,
01381
                             1, MPI_COMM_WORLD);
01382
                  MPI_Send (optimize->rangemax, optimize->nvariables, MPI_DOUBLE, i,
01383
                            1, MPI_COMM_WORLD);
01384
               }
01385
           }
01386
        else
         {
01387
01388
             MPI_Recv (optimize->rangemin, optimize->nvariables, MPI_DOUBLE, 0, 1,
01389
                        MPI_COMM_WORLD, &mpi_stat);
             MPI_Recv (optimize->rangemax, optimize->nvariables, MPI_DOUBLE, 0, 1,
01390
01391
                        MPI_COMM_WORLD, &mpi_stat);
01392
01393 #endif
01394 #if DEBUG_OPTIMIZE
01395 fprintf (stderr, "optimize_refine: end\n");
01396 #endif
01397 }
```

4.21.3.26 optimize_save_old()

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

Function to save the best results on iterative methods.

```
Definition at line 1238 of file optimize.c.
```

```
01240
        unsigned int i, j;
01241 #if DEBUG_OPTIMIZE
01242 fprintf (stderr, "optimize_save_old: start\n");
01243 fprintf (stderr, "optimize_save_old: nsaveds=%u\n", optimize->nsaveds);
01244 #endif
01245 memcpy (optimize->error_old, optimize->error_best,
01246
                  optimize->nbest * sizeof (double));
        for (i = 0; i < optimize->nbest; ++i)
01247
01248
        {
01249
             j = optimize->simulation_best[i];
01250 #if DEBUG_OPTIMIZE
             fprintf (stderr, "optimize_save_old: i=%u j=%u\n", i, j);
01251
01252 #endif
01253
            memcpy (optimize->value_old + i * optimize->nvariables,
                      optimize->value + j * optimize->nvariables,
optimize->nvariables * sizeof (double));
01254
01255
01257 #if DEBUG_OPTIMIZE
01258 for (i = 0; i < optimize->nvariables; ++i)
01259
         fprintf (stderr, "optimize_save_old: best variable %u=%lg\n",
        i, optimize->value_old[i]);
fprintf (stderr, "optimize_save_old: end\n");
01260
01261
01262 #endif
01263 }
```

4.21.3.27 optimize_save_optimal()

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

Function to save the optimal input files.

Definition at line 1457 of file optimize.c.

```
01459
        char cinput[32];
01460
        unsigned int i, j;
01461
        unsigned int flags = 1;
01462
01463
        // Getting optimal values
01464
        memcpy (optimize->value, optimize->value_old,
01465
                optimize->nvariables * sizeof (double));
01466
        // Saving optimal input files
01467
       for (i = 0; i < optimize->ninputs; ++i)
  for (j = 0; j < optimize->nexperiments; ++j)
01468
01469
01470
01471
              snprintf (cinput, 32, "optimal-%u-%u", i, j);
01472 #if DEBUG_OPTIMIZE
              fprintf (stderr, "optimize_save_optimal: i=%u j=%u input=%s\n",
01473
                        i, j, cinput);
01474
01475 #endif
01476
              // Checking templates
01477
              if (optimize->template_flags & flags)
01478
                optimize_input (0, cinput, optimize->file[i][j]);
              flags «= 1;
01479
01480
01481 }
```

Here is the call graph for this function:

4.21.3.28 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 453 of file optimize.c.

```
00455 {
00456
        unsigned int i;
00457
         char buffer[64];
00458 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_save_variables: start\n");
00459
00460 #endif
        for (i = 0; i < optimize->nvariables; ++i)
00461
00462
             snprintf (buffer, 64, "%s ", format[optimize->precision[i]]);
fprintf (optimize->file_variables, buffer,
00463
00464
00465
                       optimize->value[simulation * optimize->nvariables + i]);
00466
        fprintf (optimize->file_variables, "%.14le\n", error);
00467
00468 fflush (optimize->file_variables);
00469 #if DEBUG_OPTIMIZE
00470
        fprintf (stderr, "optimize_save_variables: end\n");
00471 #endif
00472 }
```

4.21.3.29 optimize_sequential()

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

Function to optimize sequentially.

Definition at line 519 of file optimize.c.

```
00521
          unsigned int i;
double e;

00522 double e;

00523 #if DEBUG_OPTIMIZE

00524 fprintf (stderr, "optimize_sequential: start\n");

00525 fprintf (stderr, "optimize_sequential: nstart=%u nend=%u\n",
00526
                     optimize->nstart, optimize->nend);
00527 #endif
00528
        for (i = optimize->nstart; i < optimize->nend; ++i)
00529
              e = optimize_norm (i);
optimize_best (i, e);
optimize_save_variables (i, e);
00530
00531
00532
00533
               if (e < optimize->threshold)
00534
00535
                    optimize->stop = 1;
00536
                   break;
00537
00538 #if DEBUG_OPTIMIZE
00539
              fprintf (stderr, "optimize_sequential: i=%u e=%lg\n", i, e);
00540 #endif
00541
00542 #if DEBUG_OPTIMIZE
00543
         fprintf (stderr, "optimize_sequential: end\n");
00544 #endif
00545 }
```

Here is the call graph for this function:

4.21.3.30 optimize_step()

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

Function to do a step of the iterative algorithm.

Definition at line 1403 of file optimize.c.

```
01404 {
01405 #if DEBUG_OPTIMIZE
01406 fprintf (stderr, "optimize_step: start\n");
01407 #endif
01408 optimize_algorithm ();
01409 if (optimize->nsteps)
01410 optimize_climbing (optimize->nsteps);
01411 #if DEBUG_OPTIMIZE
01412 fprintf (stderr, "optimize_step: end\n");
01413 #endif
01414 }
```

Here is the call graph for this function:

4.21.3.31 optimize_step_climbing()

Function to do a step of the hill climbing method.

Parameters

```
simulation Simulation number.
```

Definition at line 1006 of file optimize.c.

```
01007 {
01008 GThread *thread[nthreads_climbing];
01009 ParallelData data[nthreads_climbing];
```

```
unsigned int i, j, k, b;
01011 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_step_climbing: start\n");
01012
01013 #endif
01014 for (i = 0; i < optimize->nestimates; ++i)
01015
           k = (simulation + i) * optimize->nvariables;
01017
           b = optimize->simulation_best[0] * optimize->nvariables;
01018 #if DEBUG_OPTIMIZE
01019
           fprintf (stderr, "optimize_step_climbing: simulation=%u best=%u\n",
                    simulation + i, optimize->simulation_best[0]);
01020
01021 #endif
       for (j = 0; j < optimize->nvariables; ++j, ++k, ++b)
01022
01023
01024 #if DEBUG_OPTIMIZE
01025
              fprintf (stderr,
                         "optimize_step_climbing: estimate=%u best%u=%.14le\n",
01026
01027
                        i, j, optimize->value[b]);
01028 #endif
               optimize->value[k]
01030
                  = optimize->value[b] + optimize_estimate_climbing (j, i);
01031
               optimize->value[k] = fmin (fmax (optimize->value[k],
                                                 optimize->rangeminabs[j]),
01032
01033
                                           optimize->rangemaxabs[j]);
01034 #if DEBUG_OPTIMIZE
           fprintf (stderr,
01036
                         "optimize_step_climbing: estimate=%u variable%u=%.14le\n",
01037
                        i, j, optimize->value[k]);
01038 #endif
01039
             }
01040
01041
       if (nthreads_climbing == 1)
01042
         optimize_climbing_sequential (simulation);
       else
01043
01044
           for (i = 0; i <= nthreads_climbing; ++i)</pre>
01045
01046
             {
               optimize->thread_climbing[i]
01048
                 = simulation + optimize->nstart_climbing
01049
                 + i * (optimize->nend_climbing - optimize->nstart_climbing)
01050
                 / nthreads_climbing;
01051 #if DEBUG OPTIMIZE
01052
               fprintf (stderr,
01053
                         "optimize_step_climbing: i=%u thread_climbing=%u\n",
                        i, optimize->thread_climbing[i]);
01054
01055 #endif
01056
01057
            for (i = 0; i < nthreads_climbing; ++i)</pre>
01058
               data[i].thread = i;
01059
01060
               thread[i] = g_thread_new
01061
                 (NULL, (GThreadFunc) optimize_climbing_thread, &data[i]);
01062
01063
           for (i = 0; i < nthreads_climbing; ++i)</pre>
01064
             g_thread_join (thread[i]);
01065
01066 #if DEBUG_OPTIMIZE
01067
       fprintf (stderr, "optimize_step_climbing: end\n");
01068 #endif
01069 3
```

Here is the call graph for this function:

4.21.3.32 optimize_sweep()

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

Function to optimize with the sweep algorithm.

Definition at line 699 of file optimize.c.

```
00700 {
00701
       ParallelData data[nthreads];
       GThread *thread[nthreads];
00703
       double range[optimize->nvariables];
00704
       double e;
00705
       unsigned int i, j, k, l;
00706 #if DEBUG OPTIMIZE
       fprintf (stderr, "optimize_sweep: start\n");
00707
00708 #endif
00709
       for (j = 0; j < optimize->nvariables; ++j)
```

```
range[j] = (optimize->rangemax[j] - optimize->rangemin[j])
00711
              (optimize->nsweeps[j] - 1);
00712
        for (i = 0; i < optimize->nsimulations; ++i)
00713
            k = i:
00714
            for (j = 0; j < optimize->nvariables; ++j)
00715
00716
00717
                1 = k % optimize->nsweeps[j];
00718
               k /= optimize->nsweeps[j];
00719
                e = optimize->rangemin[j];
               if (optimize->nsweeps[j] > 1)
00720
                 e += 1 * range[j];
00721
00722
                optimize->value[i * optimize->nvariables + j] = e;
00723
00724
00725
        optimize->nsaveds = 0;
00726
        if (nthreads <= 1)
00727
         optimize_sequential ();
00728
        else
00729
         {
            for (i = 0; i < nthreads; ++i)</pre>
00730
00731
00732
                data[i].thread = i;
00733
                thread[i]
00734
                  = q_thread_new (NULL, (GThreadFunc) optimize_thread, &data[i]);
00735
00736
            for (i = 0; i < nthreads; ++i)
             g_thread_join (thread[i]);
00737
00738
00739 #if HAVE_MPI
00740 // Communicating tasks results
00741
       optimize_synchronise ();
00742 #endif
00743 #if DEBUG_OPTIMIZE
00744 fprintf (stderr, "optimize_sweep: end\n"); 00745 #endif
00746 }
```

4.21.3.33 optimize_synchronise()

```
static void optimize_synchronise ( ) [static]
```

Function to synchronise the optimization results of MPI tasks.

Definition at line 652 of file optimize.c.

```
00653 {
        unsigned int i, nsaveds, simulation_best[optimize->nbest], stop;
00655
        double error_best[optimize->nbest];
00656
        MPI_Status mpi_stat;
00657 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_synchronise: start\n");
00658
00659 #endif
       if (optimize->mpi_rank == 0)
00661
          {
00662
            for (i = 1; (int) i < ntasks; ++i)</pre>
00663
                MPI_Recv (&nsaveds, 1, MPI_INT, i, 1, MPI_COMM_WORLD, &mpi_stat);
00664
                MPI_Recv (simulation_best, nsaveds, MPI_INT, i, 1,
00665
                           MPI_COMM_WORLD, &mpi_stat);
00666
00667
                MPI_Recv (error_best, nsaveds, MPI_DOUBLE, i, 1,
00668
                           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);
00669
00670
00671
                if (stop)
00672
                  optimize->stop = 1;
00673
00674
            for (i = 1; (int) i < ntasks; ++i)
              MPI_Send (&optimize->stop, 1, MPI_UNSIGNED, i, 1, MPI_COMM_WORLD);
00675
00676
00677
        else
00678
00679
            MPI_Send (&optimize->nsaveds, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);
00680
            MPI_Send (optimize->simulation_best, optimize->nsaveds, MPI_INT, 0, 1,
00681
                       MPI_COMM_WORLD);
00682
            MPI_Send (optimize->error_best, optimize->nsaveds, MPI_DOUBLE, 0, 1,
00683
                       MPI_COMM_WORLD);
00684
            MPI_Send (&optimize->stop, 1, MPI_UNSIGNED, 0, 1, MPI_COMM_WORLD);
00685
            MPI_Recv (&stop, 1, MPI_UNSIGNED, 0, 1, MPI_COMM_WORLD, &mpi_stat);
```

4.21.3.34 optimize_thread()

Function to optimize on a thread.

Returns

NULL.

Parameters

data | Function data.

Definition at line 553 of file optimize.c.

```
00554 {
00555
        unsigned int i, thread;
00556 double e;
00557 #if DEBUG_OPTIMIZE
00558
        fprintf (stderr, "optimize_thread: start\n");
00559 #endif
00560
        thread = data->thread;
00561 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_thread: thread=%u start=%u end=%u\n", thread,
00562
                   optimize->thread[thread], optimize->thread[thread + 1]);
00563
00564 #endif
00565
        for (i = optimize->thread[thread]; i < optimize->thread[thread + 1]; ++i)
00566
            e = optimize_norm (i);
g_mutex_lock (mutex);
optimize_best (i, e);
optimize_save_variables (i, e);
00567
00568
00569
00570
00571
            if (e < optimize->threshold)
             optimize->stop = 1;
g_mutex_unlock (mutex);
00572
00573
00574
            if (optimize->stop)
00575 break;
00576 #if DEBUG_OPTIMIZE
00577
             fprintf (stderr, "optimize_thread: i=%u e=%lg\n", i, e);
00578 #endif
00579
00580 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_thread: end\n");
00581
00582 #endif
00583
        g_thread_exit (NULL);
00584
        return NULL;
00585 }
```

4.21.4 Variable Documentation

4.21.4.1 nthreads_climbing

```
unsigned int nthreads_climbing
```

Number of threads for the hill climbing method.

Definition at line 84 of file optimize.c.

4.21.4.2 optimize

```
Optimize optimize[1]
```

Optimization data.

Definition at line 83 of file optimize.c.

4.21.4.3 optimize_algorithm

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

Pointer to the function to perform a optimization algorithm step.

Definition at line 87 of file optimize.c.

4.21.4.4 optimize_estimate_climbing

Pointer to the function to estimate the climbing.

Definition at line 89 of file optimize.c.

4.21.4.5 optimize_norm

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

Pointer to the error norm function.

Definition at line 92 of file optimize.c.

4.22 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/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
00075 #ifdef G_OS_WIN32
00076 #define CP "copy"
00077 #define RM "del
00078 #else
00079 #define CP "cp"
00080 #define RM "rm"
00081 #endif
00082
00083 Optimize optimize[1]:
00084 unsigned int nthreads_climbing;
00087 static void (*optimize_algorithm) ();
00089 static double (*optimize_estimate_climbing) (unsigned int variable,
00090
                                                           unsigned int estimate);
00092 static double (*optimize_norm) (unsigned int simulation);
00094
00098 static inline void
00099 optimize_input (unsigned int simulation,
00100
                         char *input,
                         GMappedFile * stencil)
00101
```

```
00102 {
00103
        char buffer[256], value[32];
       GRegex *regex;
FILE *file;
00104
00105
00106
       char *buffer2, *buffer3 = NULL, *content;
00107
       asize lenath:
       unsigned int i;
00109
00110 #if DEBUG_OPTIMIZE
00111 fprintf (stderr, "optimize_input: start\n");
00112 #endif
00113
00114
        // Checking the file
00115
       if (!stencil)
00116
         goto optimize_input_end;
00117
00118
       // Opening stencil
       content = g_mapped_file_get_contents (stencil);
00119
        length = g_mapped_file_get_length (stencil);
00121 #if DEBUG_OPTIMIZE
00122
       fprintf (stderr, "optimize_input: length=%lu\ncontent:\n%s", length, content);
00123 #endif
00124
       file = g_fopen (input, "w");
00125
00126
        // Parsing stencil
00127
       for (i = 0; i < optimize->nvariables; ++i)
00128
00130 fprintf (stderr, "optimize_input: variable=%u\n", i); 00131 #endif
00132
           snprintf (buffer, 32, "@variable%u@", i + 1);
00133
           regex = g_regex_new (buffer, (GRegexCompileFlags) 0, (GRegexMatchFlags) 0,
00134
                                 NULL);
00135
            if (i == 0)
00136
               buffer2 = g_regex_replace_literal (regex, content, length, 0,
00137
00138
                                                    optimize->label[i],
                                                    (GRegexMatchFlags) 0, NULL);
00140 #if DEBUG_OPTIMIZE
00141
              fprintf (stderr, "optimize_input: buffer2\n%s", buffer2);
00142 #endif
00143
00144
            else
00145
             {
                length = strlen (buffer3);
00146
00147
                buffer2 = g_regex_replace_literal (regex, buffer3, length, 0,
00148
                                                    optimize->label[i],
00149
                                                    (GRegexMatchFlags) 0, NULL);
               g_free (buffer3);
00150
00151
00152
            g_regex_unref (regex);
00153
            length = strlen (buffer2);
00154
            snprintf (buffer, 32, "@value%u@", i + 1);
00155
            regex = g_regex_new (buffer, (GRegexCompileFlags) 0, (GRegexMatchFlags) 0,
           NULL);
snprintf (value, 32, format[optimize->precision[i]],
00156
00157
                      optimize->value[simulation * optimize->nvariables + i]);
00158
00159
00160 #if DEBUG_OPTIMIZE
           fprintf (stderr, "optimize_input: value=%s\n", value);
00161
00162 #endif
            buffer3 = g_regex_replace_literal (regex, buffer2, length, 0, value,
00163
00164
                                                (GRegexMatchFlags) 0, NULL);
00165
            g free (buffer2);
00166
           g_regex_unref (regex);
00167
00168
        // Saving input file
00169
00170
       fwrite (buffer3, strlen (buffer3), sizeof (char), file);
        g_free (buffer3);
00172
       fclose (file);
00173
00174 optimize_input_end:
00175 #if DEBUG_OPTIMIZE
00176
       fprintf (stderr, "optimize_input: end\n");
00177 #endif
00178
       return;
00179 }
00180
00187 static double
00188 optimize_parse (unsigned int simulation,
                      unsigned int experiment)
00190 {
00191
       char buffer[512], cinput[MAX_NINPUTS][32], output[32], result[32], *buffer2,
00192
         *buffer3, *buffer4;
00193
       FILE *file_result;
00194
       double e:
```

```
unsigned int i;
00196
        unsigned int flags = 1;
00197
00198 #if DEBUG OPTIMIZE
        fprintf (stderr, "optimize_parse: start\n");
fprintf (stderr, "optimize_parse: simulation=%u experiment=%u\n",
00199
00200
                   simulation, experiment);
00202 #endif
00203
00204
         // Opening input files
00205
         for (i = 0; i < optimize->ninputs; ++i)
00206
         {
00207
             snprintf (&cinput[i][0], 32, "input-%u-%u-%u", i, simulation, experiment);
00208 #if DEBUG_OPTIMIZE
00209
             fprintf (stderr, "optimize_parse: i=%u input=%s\n", i, &cinput[i][0]);
00210 #endif
             // Checking simple copy
if (optimize->template_flags & flags)
00211
00212
               optimize_input (simulation, &cinput[i][0],
00214
                                  optimize->file[i][experiment]);
00215
00216
                 buffer2 = input->experiment[experiment].stencil[i];
snprintf (buffer, 512, CP " %s %s", buffer2, &cinput[i][0]);
if (system (buffer) == -1)
00217
00218
00219
                   error_message = g_strdup (buffer);
00220
00221
00222
             flags «= 1;
00223
        for (; i < MAX_NINPUTS; ++i)</pre>
00224
00225 strcpy (&cinput[i][0], "");
00226 #if DEBUG_OPTIMIZE
00227
        fprintf (stderr, "optimize_parse: parsing end\n");
00228 #endif
00229
00230
        // Performing the simulation
         snprintf (output, 32, "output-%u-%u", simulation, experiment);
00231
         buffer2 = g_path_get_dirname (optimize->simulator);
00233
         buffer3 = g_path_get_basename (optimize->simulator);
00234
         buffer4 = g_build_filename (buffer2, buffer3, NULL);
        snprintf (buffer, 512, "\"%s\" %s %s %s %s %s %s %s %s %s",
    buffer4, cinput[0], cinput[1], cinput[2], cinput[3], cinput[4],
    cinput[5], cinput[6], cinput[7], output);
00235
00236
00237
00238
        g_free (buffer4);
00239
        g_free (buffer3);
         g_free (buffer2);
00240
00241 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_parse: %s\n", buffer);
00242
00243 #endif
        if (system (buffer) == -1)
00244
00245
           error_message = g_strdup (buffer);
00246
00247
         // Checking the objective value function
00248
         if (optimize->evaluator)
00249
00250
             snprintf (result, 32, "result-%u-%u", simulation, experiment);
             buffer2 = g_path_get_dirname (optimize->evaluator);
00252
             buffer3 = g_path_get_basename (optimize->evaluator);
             buffer4 = g_build_filename (buffer2, buffer3, NULL); snprintf (buffer, 512, "\"%s\" %s %s %s",
00253
00254
                         buffer4, output, optimize->experiment[experiment], result);
00255
             g_free (buffer4);
00256
00257
             g_free (buffer3);
00258
              g_free (buffer2);
00259 #if DEBUG_OPTIMIZE
             fprintf (stderr, "optimize_parse: %s\n", buffer);
fprintf (stderr, "optimize_parse: result=%s\n", result);
00260
00261
00262 #endif
00263
           if (system (buffer) == -1)
00264
                error_message = g_strdup (buffer);
             file_result = g_fopen (result, "r");
e = atof (fgets (buffer, 512, file_result));
00265
00266
00267
             fclose (file_result);
00268
00269
        else
00270
00271 #if DEBUG_OPTIMIZE
00272
             fprintf (stderr, "optimize_parse: output=%sn", output);
00273 #endif
00274
             strcpy (result, "");
             file_result = g_fopen (output, "r");
e = atof (fgets (buffer, 512, file_result));
00275
00277
              fclose (file_result);
00278
00279
        // Removing files
00280
00281
        if (optimize->cleaner)
```

```
00283
            buffer2 = g_path_get_dirname (optimize->cleaner);
00284
            buffer3 = g_path_get_basename (optimize->cleaner);
            buffer4 = g_build_filename (buffer2, buffer3, NULL);
snprintf (buffer, 512, "\"%s\"", buffer4);
00285
00286
00287
             g_free (buffer4);
            g_free (buffer3);
00289
             g_free (buffer2);
00290
             if (system (buffer) == -1)
00291
               error_message = g_strdup (buffer);
00292
00293 #if !DEBUG_OPTIMIZE
00294
        for (i = 0; i < optimize->ninputs; ++i)
00295
00296
             if (optimize->file[i][0])
00297
                 snprintf (buffer, 512, RM " %s", &cinput[i][0]);
00298
00299
                 if (system (buffer) == -1)
00300
                   error_message = g_strdup (buffer);
00301
               }
00302
        snprintf (buffer, 512, RM " %s %s", output, result);
00303
00304
        if (system (buffer) == -1)
00305
          error_message = g_strdup (buffer);
00306 #endif
00307
00308
        // Processing pending events
00309
       if (show_pending)
00310
          show_pending ();
00311
00312 #if DEBUG_OPTIMIZE
00313
        fprintf (stderr, "optimize_parse: end\n");
00314 #endif
00315
00316
        \ensuremath{//} Returning the objective function
00317
        return e * optimize->weight[experiment];
00318 }
00325 static double
00326 optimize_norm_euclidian (unsigned int simulation)
00327 {
00328
        double e, ei;
        unsigned int i;
00329
00330 #if DEBUG_OPTIMIZE
00331 fprintf (stderr, "optimize_norm_euclidian: start\n");
00332 #endif
00333 e = 0.;
        for (i = 0; i < optimize->nexperiments; ++i)
00334
         {
00335
00336
            ei = optimize_parse (simulation, i);
00337
            e += ei * ei;
00338
00339
        e = sqrt (e);
00340 #if DEBUG_OPTIMIZE
00341 fprintf (stderr, "optimize_norm_euclidian: error=%lg\n", e);
00342 fprintf (stderr, "optimize_norm_euclidian: end\n");
00343 #endif
00344
       return e;
00345 }
00346
00352 static double
00353 optimize_norm_maximum (unsigned int simulation)
00354 {
00355 double e, ei;
00356
        unsigned int i;
00357 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_norm_maximum: start\n");
00358
00359 #endif
00360 e = 0.;
        for (i = 0; i < optimize->nexperiments; ++i)
00362
00363
            ei = fabs (optimize_parse (simulation, i));
00364
           e = fmax (e, ei);
00365
00366 #if DEBUG_OPTIMIZE
00367 fprintf (stderr, "optimize_norm_maximum: error=%lg\n", e);
00368 fprintf (stderr, "optimize_norm_maximum: end\n");
00369 #endif
00370
       return e;
00371 }
00372
00378 static double
00379 optimize_norm_p (unsigned int simulation)
00380 {
        double e, ei;
unsigned int i;
00381
00382
00383 #if DEBUG_OPTIMIZE
```

```
fprintf (stderr, "optimize_norm_p: start\n");
00385 #endif
00386
        e = 0.;
00387
        for (i = 0; i < optimize->nexperiments; ++i)
00388
00389
            ei = fabs (optimize parse (simulation, i));
            e += pow (ei, optimize->p);
00391
00392 e = pow (e, 1.
                         / optimize->p);
00393 #if DEBUG_OPTIMIZE
00394 fprintf (stderr, "optimize_norm_p: error=%lg\n", e);
00395 fprintf (stderr, "optimize_norm_p: end\n");
00396 #endif
00397
      return e;
00398 }
00399
00405 static double
00406 optimize_norm_taxicab (unsigned int simulation)
00407 {
00408
        double e;
00409
        unsigned int i;
00410 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_norm_taxicab: start\n");
00411
00412 #endif
00413
       e = 0.;
        for (i = 0; i < optimize->nexperiments; ++i)
00415
          e += fabs (optimize_parse (simulation, i));
00416 #if DEBUG_OPTIMIZE
00417 fprintf (stderr, "optimize_norm_taxicab: error=%lg\n", e);
00418 fprintf (stderr, "optimize_norm_taxicab: end\n");
00419 #endif
00420
       return e;
00421 }
00422
00426 static void
00427 optimize_print ()
00428 {
      unsigned int i;
00430
        char buffer[512];
00431 #if HAVE_MPI
00432 if (optimize->mpi_rank)
00433
          return;
00434 #endif
00435
       printf ("%s\n", _("Best result"));
        fprintf (optimize->file_result, "%s\n", _("Best result"));
00436
00437
        printf ("error = %.15le\n", optimize->error_old[0]);
00438
        fprintf (optimize->file_result, "error = %.15le\n", optimize->error_old[0]);
00439
        for (i = 0; i < optimize->nvariables; ++i)
00440
            snprintf (buffer, 512, "%s = %s\n",
00441
             optimize->label[i], format[optimize->precision[i]]);
printf (buffer, optimize->value_old[i]);
00442
00443
00444
             fprintf (optimize->file_result, buffer, optimize->value_old[i]);
00445
        fflush (optimize->file_result);
00446
00447 }
00448
00452 static void
00453 optimize_save_variables (unsigned int simulation,
00454
                                  double error)
00455 {
00456 unsigned int i;
00457
        char buffer[64];
00458 #if DEBUG_OPTIMIZE
00459
        fprintf (stderr, "optimize_save_variables: start\n");
00460 #endif
        for (i = 0; i < optimize->nvariables; ++i)
00461
00462
00463
            snprintf (buffer, 64, "%s ", format[optimize->precision[i]]);
             fprintf (optimize->file_variables, buffer,
00464
00465
                      optimize->value[simulation * optimize->nvariables + i]);
00466
00467
        fprintf (optimize->file_variables, "%.14le\n", error);
00468 fflush (optimize->file_variables);
00469 #if DEBUG_OPTIMIZE
00470 fprintf (stderr, "optimize_save_variables: end\n");
00471 #endif
00472 }
00473
00477 static void
00478 optimize_best (unsigned int simulation,
                      double value)
00480 {
00481 unsigned int i, j;
00482 double e;
00483 #if DEBUG_OPTIMIZE
00484 fprintf (stderr, "optimize_best: start\n");
```

```
fprintf (stderr, "optimize_best: nsaveds=%u nbest=%u\n",
                  optimize->nsaveds, optimize->nbest);
00486
00487 #endif
00488
        if (optimize->nsaveds < optimize->nbest
00489
            || value < optimize->error_best[optimize->nsaveds - 1])
00490
            if (optimize->nsaveds < optimize->nbest)
00492
               ++optimize->nsaveds;
00493
             optimize->error_best[optimize->nsaveds - 1] = value;
            optimize->simulation_best[optimize->nsaveds - 1] = simulation;
00494
00495
            for (i = optimize->nsaveds; --i;)
00496
00497
                 if (optimize->error_best[i] < optimize->error_best[i - 1])
00498
00499
                     j = optimize->simulation_best[i];
00500
                     e = optimize->error_best[i];
                     optimize->simulation_best[i] = optimize->simulation_best[i - 1];
00501
                    optimize->error_best[i] = optimize->error_best[i - 1];
optimize->simulation_best[i - 1] = j;
00502
00503
00504
                    optimize->error_best[i - 1] = e;
00505
00506
                 else
00507
                  break;
00508
              }
00509
00510 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_best: end\n");
00511
00512 #endif
00513 }
00514
00518 static void
00519 optimize_sequential ()
00520 {
00521
        unsigned int i;
00522 double e;
00522 double e;
00523 #if DEBUG_OPTIMIZE
00524 fprintf (stderr, "optimize_sequential: start\n");
00525 fprintf (stderr, "optimize_sequential: nstart=%u nend=%u\n",
00526
                 optimize->nstart, optimize->nend);
00527 #endif
00528
       for (i = optimize->nstart; i < optimize->nend; ++i)
00529
         {
00530
            e = optimize_norm (i);
            optimize_best (i, e);
optimize_save_variables (i, e);
00531
00532
00533
            if (e < optimize->threshold)
00534
             {
00535
                optimize->stop = 1;
00536
                break:
00537
00538 #if DEBUG_OPTIMIZE
00539
            fprintf (stderr, "optimize_sequential: i=%u e=%lg\n", i, e);
00540 #endif
00541
00542 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_sequential: end\n");
00543
00544 #endif
00545 }
00546
00552 static void *
00553 optimize_thread (ParallelData * data)
00554 {
       unsigned int i, thread;
      double e;
00556
00557 #if DEBUG_OPTIMIZE
00558 fprintf (stderr, "optimize_thread: startn");
00559 #endif
       thread = data->thread;
00560
00561 #if DEBUG_OPTIMIZE
00562 fprintf (stderr, "optimize_thread: thread=%u start=%u end=%u\n", thread,
00563
                 optimize->thread[thread], optimize->thread[thread + 1]);
00564 #endif
00565
       for (i = optimize->thread[thread]; i < optimize->thread[thread + 1]; ++i)
00566
00567
            e = optimize norm (i);
00568
            g_mutex_lock (mutex);
00569
            optimize_best (i, e);
00570
            optimize_save_variables (i, e);
00571
            if (e < optimize->threshold)
00572
             optimize->stop = 1;
00573
             g_mutex_unlock (mutex);
            if (optimize->stop)
00575
              break;
00576 #if DEBUG_OPTIMIZE
00577
            fprintf (stderr, "optimize_thread: i=%u e=%lg\n", i, e);
00578 #endif
00579
```

```
00580 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_thread: end\n");
00582 #endif
00583 g_thread_exit (NULL);
00584
        return NULL;
00585 }
00590 static inline void
00591 optimize_merge (unsigned int nsaveds,
00592
                       unsigned int *simulation_best,
00594
                       double *error_best)
00596 {
00597 unsigned int i, j, k, s[optimize->nbest];
00598 double e[optimize->nbest];
00599 #if DEBUG_OPTIMIZE
00600
       fprintf (stderr, "optimize_merge: start\n");
00601 #endif
       i = j = k = 0;
00602
00603
        do
00604
         {
00605
            if (i == optimize->nsaveds)
00606
              {
                s[k] = simulation_best[j];
00607
                 e[k] = error_best[j];
00608
00609
                 ++j;
00610
                 ++k;
00611
                 if (j == nsaveds)
00612
                  break;
00613
            else if (j == nsaveds)
00614
00615
              {
00616
                 s[k] = optimize->simulation_best[i];
00617
                 e[k] = optimize->error_best[i];
00618
                ++i;
00619
                 ++k;
                if (i == optimize->nsaveds)
00620
00621
                  break:
00622
00623
            else if (optimize->error_best[i] > error_best[j])
00624
00625
                s[k] = simulation_best[j];
00626
                 e[k] = error_best[j];
00627
                 ++i:
00628
                 ++k;
00629
00630
            else
00631
             {
00632
                s[k] = optimize->simulation_best[i];
                 e[k] = optimize->error_best[i];
00633
00634
                ++i;
00635
                ++k;
00636
00637
00638
       while (k < optimize->nbest);
        optimize->nsaveds = k;
00639
       memcpy (optimize->simulation_best, s, k * sizeof (unsigned int));
memcpy (optimize->error_best, e, k * sizeof (double));
00640
00642 #if DEBUG_OPTIMIZE
00643 fprintf (stderr, "optimize_merge: end\n");
00644 #endif
00645 }
00646
00650 #if HAVE_MPI
00651 static void
00652 optimize_synchronise ()
00653 {
00654 unsigned int i, nsaveds, simulation_best[optimize->nbest], stop;
00655
        double error best[optimize->nbest];
00656
        MPI_Status mpi_stat;
00657 #if DEBUG_OPTIMIZE
00658
       fprintf (stderr, "optimize_synchronise: start\n");
00659 #endif
00660
        if (optimize->mpi_rank == 0)
00661
00662
            for (i = 1; (int) i < ntasks; ++i)</pre>
00663
00664
                 MPI_Recv (&nsaveds, 1, MPI_INT, i, 1, MPI_COMM_WORLD, &mpi_stat);
00665
                 MPI_Recv (simulation_best, nsaveds, MPI_INT, i, 1,
00666
                           MPI_COMM_WORLD, &mpi_stat);
                MPI_Recv (error_best, nsaveds, MPI_DOUBLE, i, 1,
00667
                 MPI_COMM_WORLD, &mpi_stat);
optimize_merge (nsaveds, simulation_best, error_best);
00668
00669
00670
                 MPI_Recv (&stop, 1, MPI_UNSIGNED, i, 1, MPI_COMM_WORLD, &mpi_stat);
00671
                 if (stop)
00672
                  optimize->stop = 1;
00673
00674
            for (i = 1; (int) i < ntasks; ++i)</pre>
```

```
MPI_Send (&optimize->stop, 1, MPI_UNSIGNED, i, 1, MPI_COMM_WORLD);
00676
00677
        else
00678
           MPI_Send (&optimize->nsaveds, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);
00679
           MPI_Send (optimize->simulation_best, optimize->nsaveds, MPI_INT, 0, 1,
00680
                      MPI_COMM_WORLD);
00682
            MPI_Send (optimize->error_best, optimize->nsaveds, MPI_DOUBLE, 0, 1,
                      MPI_COMM_WORLD);
00683
           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);
00684
00685
00686
           if (stop)
00687
              optimize->stop = 1;
00688
00689 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_synchronise: end\n");
00690
00691 #endif
00692 }
00693 #endif
00698 static void
00699 optimize_sweep ()
00700 {
        ParallelData data[nthreads];
00701
00702
        GThread *thread[nthreads];
00703
        double range[optimize->nvariables];
00704
        double e;
00705
       unsigned int i, j, k, l;
00706 #if DEBUG_OPTIMIZE
00707
       fprintf (stderr, "optimize_sweep: start\n");
00708 #endif
00709
       for (j = 0; j < optimize->nvariables; ++j)
00710
        range[j] = (optimize->rangemax[j] - optimize->rangemin[j])
00711
              (optimize->nsweeps[j] - 1);
        for (i = 0; i < optimize->nsimulations; ++i)
00712
00713
00714
           k = i;
            for (j = 0; j < optimize->nvariables; ++j)
00716
              {
00717
               1 = k % optimize->nsweeps[j];
00718
               k /= optimize->nsweeps[j];
00719
                e = optimize->rangemin[j];
00720
               if (optimize->nsweeps[j] > 1)
00721
                 e += 1 * range[j];
00722
                optimize->value[i * optimize->nvariables + j] = e;
00723
              }
00724
00725
        optimize->nsaveds = 0;
00726
       if (nthreads <= 1)
00727
         optimize_sequential ();
00728
        else
00729
00730
            for (i = 0; i < nthreads; ++i)
00731
00732
               data[i].thread = i;
00733
                thread[i]
00734
                 = g_thread_new (NULL, (GThreadFunc) optimize_thread, &data[i]);
00735
00736
            for (i = 0; i < nthreads; ++i)</pre>
00737
              g_thread_join (thread[i]);
00738
00739 #if HAVE_MPI
       // Communicating tasks results
00741
       optimize_synchronise ();
00742 #endif
00743 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_sweep: end\n");
00744
00745 #endif
00746 }
00747
00751 static void
00752 optimize_MonteCarlo ()
00753 {
00754
       ParallelData data[nthreads]:
00755
       GThread *thread[nthreads];
00756
       double range[optimize->nvariables];
00757
        unsigned int i, j;
00758 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_MonteCarlo: start\n");
00759
00760 #endif
00761
       for (j = 0; j < optimize->nvariables; ++j)
00762
         range[j] = optimize->rangemax[j] - optimize->rangemin[j];
00763
        for (i = 0; i < optimize->nsimulations; ++i)
00764
         for (j = 0; j < optimize->nvariables; ++j)
            optimize->value[i * optimize->nvariables + j]
00765
00766
              = optimize->rangemin[j] + gsl_rng_uniform (optimize->rng) * range[j];
00767
       optimize->nsaveds = 0:
```

```
if (nthreads <= 1)</pre>
00769
         optimize_sequential ();
00770
       else
00771
            for (i = 0; i < nthreads; ++i)
00772
00773
             {
00774
               data[i].thread = i;
00775
               thread[i]
00776
                 = g_thread_new (NULL, (GThreadFunc) optimize_thread, &data[i]);
00777
00778
            for (i = 0; i < nthreads: ++i)</pre>
00779
             g_thread_join (thread[i]);
00780
00781 #if HAVE_MPI
00782 // Communicating tasks results
00783
       optimize_synchronise ();
00784 #endif
00785 #if DEBUG_OPTIMIZE
00786 fprintf (stderr, "optimize_MonteCarlo: end\n");
00787 #endif
00788 }
00789
00793 static void
00794 optimize_orthogonal ()
00795 {
00796
       ParallelData data[nthreads];
00797
       GThread *thread[nthreads];
00798
       double range[optimize->nvariables];
00799
       double e;
00800 unsigned int i, j, k, 1;
00801 #if DEBUG_OPTIMIZE
00802
       fprintf (stderr, "optimize_orthogonal: start\n");
00803 #endif
00804
       for (j = 0; j < optimize->nvariables; ++j)
00805
        range[j] = (optimize->rangemax[j] - optimize->rangemin[j])
           / optimize->nsweeps[j];
00806
00807
       for (i = 0; i < optimize->nsimulations; ++i)
00809
           k = i;
00810
            for (j = 0; j < optimize->nvariables; ++j)
00811
              {
               1 = k % optimize->nsweeps[j];
00812
               k /= optimize->nsweeps[j];
00813
00814
               e = optimize->rangemin[j];
               if (optimize->nsweeps[j] > 1)
00815
00816
                 e += (1 + gsl_rng_uniform (optimize->rng)) * range[j];
00817
               optimize->value[i * optimize->nvariables + j] = e;
00818
             }
00819
         }
00820
       optimize->nsaveds = 0;
00821
       if (nthreads <= 1)</pre>
00822
         optimize_sequential ();
00823
       else
00824
        {
00825
           for (i = 0; i < nthreads; ++i)
00826
             {
               data[i].thread = i;
00828
00829
                 = g_thread_new (NULL, (GThreadFunc) optimize_thread, &data[i]);
00830
             }
            for (i = 0: i < nthreads: ++i)
00831
00832
             g_thread_join (thread[i]);
00833
00834 #if HAVE_MPI
00835 // Communicating tasks results
00836
       optimize_synchronise ();
00837 #endif
00838 #if DEBUG_OPTIMIZE
00839 fprintf (stderr, "optimize_orthogonal: end\n");
00840 #endif
00841 }
00842
00846 static void
00847 optimize_best_climbing (unsigned int simulation,
00848
                             double value)
00849
00850 #if DEBUG_OPTIMIZE
00851 fprintf (stderr, "optimize_best_climbing: start\n");
       fprintf (stderr,
                 "optimize_best_climbing: simulation=%u value=%.14le best=%.14le\n",
00853
                simulation, value, optimize->error_best[0]);
00854
00855 #endif
      if (value < optimize->error_best[0])
00856
00857
00858
           optimize->error_best[0] = value;
```

```
fprintf (stderr,
00862
                      "optimize_best_climbing: BEST simulation=%u value=%.14le\n",
00863
                      simulation, value);
00864 #endif
00865
00866 #if DEBUG_OPTIMIZE
00867 fprintf (stderr, "optimize_best_climbing: end\n");
00868 #endif
00869 }
00870
00874 static inline void
00875 optimize_climbing_sequential (unsigned int simulation)
00876 {
00877
        double e;
00878
        unsigned int i, j;
00879 #if DEBUG_OPTIMIZE
00880 fprintf (stderr, "optimize_climbing_sequential: start\n");
00881 fprintf (stderr, "optimize_climbing_sequential: nstart_climbing=%u "
00882 "nend_climbing=%u\n",
                  optimize->nstart_climbing, optimize->nend_climbing);
00883
00884 #endif
00885
        for (i = optimize->nstart_climbing; i < optimize->nend_climbing; ++i)
00886
00887
            j = simulation + i;
00888
            e = optimize_norm (j);
            optimize_best_climbing (j, e);
00889
            optimize_save_variables (j, e);
00890
00891
            if (e < optimize->threshold)
00892
               {
00893
                optimize->stop = 1:
00894
                break:
00895
00896 #if DEBUG_OPTIMIZE
00897
             fprintf (stderr, "optimize_climbing_sequential: i=%u e=%lg\n", i, e);
00898 #endif
00899
00900 #if DEBUG_OPTIMIZE
00901 fprintf (stderr, "optimize_climbing_sequential: end\n");
00902 #endif
00903 }
00904
00910 static void *
00911 optimize_climbing_thread (ParallelData * data)
00912 {
00913
        unsigned int i, thread;
00914
        double e;
00915 #if DEBUG_OPTIMIZE
00916 fprintf (stderr, "optimize_climbing_thread: start\n");
00917 #endif
00918
       thread = data->thread:
00919 #if DEBUG_OPTIMIZE
00920 fprintf (stderr, "optimize_climbing_thread: thread=%u start=%u end=%u\n",
00921
                  thread,
00922
                  optimize->thread_climbing[thread],
00923
                  optimize->thread_climbing[thread + 1]);
00924 #endif
      for (i = optimize->thread_climbing[thread];
00926
             i < optimize->thread_climbing[thread + 1]; ++i)
00927
00928
            e = optimize_norm (i);
            g_mutex_lock (mutex);
optimize_best_climbing (i, e);
00929
00930
00931
            optimize_save_variables (i, e);
00932
            if (e < optimize->threshold)
00933
              optimize->stop = 1;
             g_mutex_unlock (mutex);
00934
00935
            if (optimize->stop)
  break;
00936
00937 #if DEBUG_OPTIMIZE
00938
             fprintf (stderr, "optimize_climbing_thread: i=%u e=%lg\n", i, e);
00939 #endif
00940
00941 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_climbing_thread: end\n");
00942
00943 #endif
00944 g_thread_exit (NULL);
00945
        return NULL;
00946 }
00947
00951 static double
00952 optimize_estimate_climbing_random (unsigned int variable,
00954
                                           unsigned int estimate
00955
                                            __attribute__((unused)))
00957 {
00958 double x;
00959 #if DEBUG_OPTIMIZE
00960
        fprintf (stderr, "optimize_estimate_climbing_random: start\n");
```

```
00961 #endif
00962 x = optimize \rightarrow climbing[variable]
00963 + (1. - 2. * gsl_rng_uniform (optimize->rng)) * optimize->step[variable]; 00964 #if DEBUG_OPTIMIZE
00965 fprintf (stderr, "optimize_estimate_climbing_random: climbing%u=%lq\n",
       variable, x);
fprintf (stderr, "optimize_estimate_climbing_random: end\n");
00966
00968 #endif
00969 return x;
00970 }
00971
00975 static double
00976 optimize_estimate_climbing_coordinates (unsigned int variable,
00978
                                               unsigned int estimate)
00980 {
       double x;
00981
00982 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_estimate_climbing_coordinates: start\n");
00983
00984 #endif
00985
       x = optimize->climbing[variable];
00986
       if (estimate >= (2 * variable) && estimate < (2 * variable + 2))</pre>
00987
           if (estimate & 1)
00988
00989
             x += optimize->step[variable];
00990
            else
00991
            x -= optimize->step[variable];
00992
00993 #if DEBUG_OPTIMIZE
00994 fprintf (stderr,
00995
                  "optimize_estimate_climbing_coordinates: climbing%u=%lg\n",
00996
       variable, x); fprintf (stderr, "optimize_estimate_climbing_coordinates: end\n");
00997
00998 #endif
00999
       return x;
01000 }
01001
01005 static inline void
01006 optimize_step_climbing (unsigned int simulation)
01007 {
01008
        GThread *thread[nthreads_climbing];
01009
       ParallelData data[nthreads_climbing];
01010 unsigned int i, j, k, b;
01011 #if DEBUG_OPTIMIZE
01012
       fprintf (stderr, "optimize_step_climbing: start\n");
01013 #endif
01014
       for (i = 0; i < optimize->nestimates; ++i)
01015
           k = (simulation + i) * optimize->nvariables;
01016
            b = optimize->simulation_best[0] * optimize->nvariables;
01017
01018 #if DEBUG_OPTIMIZE
           fprintf (stderr, "optimize_step_climbing: simulation=%u best=%u\n",
01020
                     simulation + i, optimize->simulation_best[0]);
01021 #endif
       for (j = 0; j < optimize->nvariables; ++j, ++k, ++b)
01022
01023
01024 #if DEBUG_OPTIMIZE
              fprintf (stderr,
01026
                          "optimize_step_climbing: estimate=%u best%u=%.14le\n",
01027
                         i, j, optimize->value[b]);
01028 #endif
               optimize->value[k]
01029
                  = optimize->value[b] + optimize_estimate_climbing (j, i);
01030
               optimize->value[k] = fmin (fmax (optimize->value[k],
                                                  optimize->rangeminabs[j]),
01032
01033
                                            optimize->rangemaxabs[j]);
01034 #if DEBUG_OPTIMIZE
        fprintf (stderr,
01035
                          "optimize_step_climbing: estimate=%u variable%u=%.14le\n",
01036
01037
                         i, j, optimize->value[k]);
01038 #endif
01039
              }
01040
01041
       if (nthreads_climbing == 1)
         optimize_climbing_sequential (simulation);
01042
01043
        else
01044
         {
01045
            for (i = 0; i <= nthreads_climbing; ++i)</pre>
01046
01047
                optimize->thread_climbing[i]
01048
                  = simulation + optimize->nstart climbing
                  + i * (optimize->nend_climbing - optimize->nstart_climbing)
01049
                  / nthreads_climbing;
01051 #if DEBUG_OPTIMIZE
01052
                fprintf (stderr,
01053
                          "optimize_step_climbing: i=%u thread_climbing=%u\n",
                         i, optimize->thread_climbing[i]);
01054
01055 #endif
```

```
01057
            for (i = 0; i < nthreads_climbing; ++i)</pre>
01058
01059
                data[i].thread = i;
01060
                thread[i] = g_thread_new
                  (NULL, (GThreadFunc) optimize_climbing_thread, &data[i]);
01061
01062
01063
            for (i = 0; i < nthreads_climbing; ++i)</pre>
01064
             g_thread_join (thread[i]);
01065
01066 #if DEBUG OPTIMIZE
01067 fprintf (stderr, "optimize_step_climbing: end\n");
01068 #endif
01069 }
01070
01074 static inline void
01075 optimize_climbing_best ()
01076 {
01077 #if DEBUG_OPTIMIZE
01078
       fprintf (stderr, "optimize_climbing_best: start\n");
01079 #endif
01080    optimize->simulation_best[0] = 0;
01081 memcpy (optimize->value, optimize->value_old,
               optimize->nvariables * sizeof (double));
01082
01083 #if DEBUG_OPTIMIZE
01084 fprintf (stderr, "optimize_climbing_best: end\n");
01085 #endif
01086 }
01087
01091 static inline void
01092 optimize_climbing (unsigned int nsteps)
01093 {
01094
       unsigned int i, j, k, b, s, adjust;
01095 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_climbing: start\n");
01096
01097 #endif
01098
       for (i = 0; i < optimize->nvariables; ++i)
         optimize->climbing[i] = 0.;
01100
       b = optimize->simulation_best[0] * optimize->nvariables;
01101
       s = optimize->nsimulations;
01102
       adjust = 1;
       for (i = 0; i < nsteps; ++i, s += optimize->nestimates, b = k)
01103
01104
01105 #if DEBUG_OPTIMIZE
           fprintf (stderr, "optimize_climbing: step=%u old_best=%u\n",
01106
01107
                     i, optimize->simulation_best[0]);
01108 #endif
01109
           optimize_step_climbing (s);
           k = optimize->simulation_best[0] * optimize->nvariables;
01110
01111 #if DEBUG_OPTIMIZE
01112
           fprintf (stderr, "optimize_climbing: step=%u best=%u\n",
01113
                    i, optimize->simulation_best[0]);
01114 #endif
01115
           if (k == b)
01116
              {
01117
                if (adjust)
                for (j = 0; j < optimize->nvariables; ++j)
01118
01119
                   optimize->step[j] *= 0.5;
01120
                for (j = 0; j < optimize->nvariables; ++j)
01121
                  optimize->climbing[j] = 0.;
01122
                adiust = 1:
01123
01124
            else
01125
             {
01126
               for (j = 0; j < optimize->nvariables; ++j)
01127
01128 #if DEBUG OPTIMIZE
                    fprintf (stderr,
01129
01130
                              "optimize_climbing: best%u=%.14le old%u=%.14le\n",
                             j, optimize->value[k + j], j, optimize->value[b + j]);
01131
01132 #endif
01133
                    optimize->climbing[j]
                     = (1. - optimize->relaxation) * optimize->climbing[j] + optimize->relaxation
01134
01135
                      * (optimize->value[k + j] - optimize->value[b + j]);
01136
01137 #if DEBUG_OPTIMIZE
01138
             fprintf (stderr, "optimize_climbing: climbing%u=%.14le\n",
01139
                             j, optimize->climbing[j]);
01140 #endif
01141
               adjust = 0;
01142
             }
01143
01144
01145 #if DEBUG_OPTIMIZE
01146 fprintf (stderr, "optimize_climbing: end\n");
01147 #endif
01148 }
```

```
01149
01155 static double
01156 optimize_genetic_objective (Entity * entity)
01157 {
01158
       unsigned int j;
01159
        double objective:
        char buffer[64];
01160
01161 #if DEBUG_OPTIMIZE
01162
       fprintf (stderr, "optimize_genetic_objective: start\n");
01163 #endif
01164
        for (j = 0; j < optimize->nvariables; ++j)
01165
            optimize->value[entity->id * optimize->nvariables + j]
01166
01167
              = genetic_get_variable (entity, optimize->genetic_variable + j);
01168
01169
        objective = optimize_norm (entity->id);
01170
        g_mutex_lock (mutex);
01171
        for (j = 0; j < optimize->nvariables; ++j)
01172
01173
            snprintf (buffer, 64, "%s ", format[optimize->precision[j]]);
            fprintf (optimize->file_variables, buffer,
01174
01175
                     genetic_get_variable (entity, optimize->genetic_variable + j));
01176
       fprintf (optimize->file_variables, "%.14le\n", objective);
01177
01178
        g_mutex_unlock (mutex);
01179 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_genetic_objective: end\n");
01180
01181 #endif
01182
       return objective;
01183 }
01184
01188 static void
01189 optimize_genetic ()
01190 {
01191
        double *best_variable = NULL;
01192
       char *best_genome = NULL;
        double best_objective = 0.;
01193
01194 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_genetic: start\n");
fprintf (stderr, "optimize_genetic: ntasks=%u nthreads=%u\n", ntasks,
01195
01196
01197
                 nthreads);
       fprintf (stderr,
01198
                  "optimize_genetic: nvariables=%u population=%u generations=%u\n",
01199
01200
                 optimize->nvariables, optimize->nsimulations, optimize->niterations);
01201
       fprintf (stderr,
01202
                  "optimize_genetic: mutation=%lg reproduction=%lg adaptation=%lg\n",
01203
                 optimize->mutation_ratio, optimize->reproduction_ratio,
01204
                 optimize->adaptation_ratio);
01205 #endif
01206
       genetic algorithm default (optimize->nvariables,
                                    optimize->genetic_variable,
01208
                                    optimize->nsimulations,
                                    optimize->niterations,
01209
01210
                                    optimize->mutation_ratio,
01211
                                    optimize->reproduction ratio,
                                    optimize->adaptation_ratio,
01212
01213
                                    optimize->seed,
01214
                                    optimize->threshold,
01215
                                     &optimize_genetic_objective,
01216
                                    &best_genome, &best_variable, &best_objective);
01217 #if DEBUG OPTIMIZE
       fprintf (stderr, "optimize_genetic: the best\n");
01218
01219 #endif
01220    optimize->error_old = (double *) g_malloc (sizeof (double));
01221
       optimize->value_old
01222
         = (double *) g_malloc (optimize->nvariables * sizeof (double));
01223
       optimize->error_old[0] = best_objective;
01224
       memcpy (optimize->value_old, best_variable,
01225
                optimize->nvariables * sizeof (double));
01226
       g_free (best_genome);
01227
       g_free (best_variable);
01228
        optimize_print ();
01229 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_genetic: end\n");
01230
01231 #endif
01232 }
01233
01237 static inline void
01238 optimize_save_old ()
01239 {
01240
        unsigned int i, j;
01241 #if DEBUG_OPTIMIZE
      fprintf (stderr, "optimize_save_old: start\n");
fprintf (stderr, "optimize_save_old: nsaveds=%u\n", optimize->nsaveds);
01242
01243
01244 #endif
01245
       memcpy (optimize->error_old, optimize->error_best,
01246
                optimize->nbest * sizeof (double));
```

```
for (i = 0; i < optimize->nbest; ++i)
01248
       f = optimize->simulation_best[i];
01249
01250 #if DEBUG_OPTIMIZE
            fprintf (stderr, "optimize_save_old: i=%u j=%u\n", i, j);
01251
01252 #endif
            memcpy (optimize->value_old + i * optimize->nvariables,
01253
01254
                     optimize->value + j * optimize->nvariables,
                     optimize->nvariables * sizeof (double));
01255
01256
01257 #if DEBUG_OPTIMIZE
01258 for (i = 0; i < optimize->nvariables; ++i)
        fprintf (stderr, "optimize_save_old: best variable %u=%lg\n",
    i, optimize->value_old[i]);
01259
01260
01261
        fprintf (stderr, "optimize_save_old: end\n");
01262 #endif
01263 }
01264
01269 static inline void
01270 optimize_merge_old ()
01271 {
01272
        unsigned int i, j, k;
        double v[optimize->nbest * optimize->nvariables], e[optimize->nbest],
01273
01274 *enew, *eold;
01275 #if DEBUG_OPTIMIZE
      fprintf (stderr, "optimize_merge_old: start\n");
01276
01277 #endif
01278 enew = optimize->error_best;
01279
        eold = optimize->error_old;
        i = j = k = 0;
01280
01281
        do
01282
          {
01283
             if (*enew < *eold)</pre>
01284
              {
01285
                memcpy (v + k * optimize->nvariables,
01286
                         optimize->value
                          + optimize->simulation_best[i] * optimize->nvariables,
01287
                         optimize->nvariables * sizeof (double));
01288
01289
                e[k] = *enew;
01290
                ++k;
01291
                ++enew;
01292
                ++i:
01293
              }
01294
            else
01295
              {
01296
                memcpy (v + k \star optimize->nvariables,
                         optimize->value_old + j * optimize->nvariables,
optimize->nvariables * sizeof (double));
01297
01298
01299
                e[k] = *eold;
01300
                ++k;
01301
                ++eold;
01302
                ++j;
01303
              }
01304
       while (k < optimize->nbest);
01305
01306 memcpy (optimize->value_old, v, k * optimize->nvariables * sizeof (double));
01307 memcpy (optimize->error_old, e, k * sizeof (double));
        memcpy (optimize->error_old, e, k * sizeof (double));
01308 #if DEBUG_OPTIMIZE
01309 fprintf (stderr, "optimize_merge_old: end\n");
01310 #endif
01311 }
01312
01317 static inline void
01318 optimize_refine ()
01319 {
01320 unsigned int i, j;
01321
        double d;
01322 #if HAVE_MPI
01323 MPI_Status mpi_stat;
01324 #endif
01325 #if DEBUG_OPTIMIZE
01326 fprintf (stderr, "optimize_refine: start\n");
01327 #endif
01328 #if HAVE_MPI
      if (!optimize->mpi_rank)
01329
01330
01331 #endif
01332
            for (j = 0; j < optimize->nvariables; ++j)
01333
01334
                optimize->rangemin[j] = optimize->rangemax[j]
                - optimize->value_old[j];
optimize->step[j] = input->variable[j].step;
01335
01336
01337
01338
             for (i = 0; ++i < optimize->nbest;)
01339
01340
                 for (j = 0; j < optimize->nvariables; ++j)
01341
```

```
01342
                     optimize->rangemin[j]
01343
                       = fmin (optimize->rangemin[j],
01344
                                 optimize->value_old[i * optimize->nvariables + j]);
                      optimize->rangemax[j]
01345
01346
                        = fmax (optimize->rangemax[j],
                                 optimize->value_old[i * optimize->nvariables + j]);
01347
01348
                   }
01349
01350
             for (j = 0; j < optimize->nvariables; ++j)
01351
                 d = optimize->tolerance
01352
                   * (optimize->rangemax[j] - optimize->rangemin[j]);
01353
01354
                 switch (optimize->algorithm)
01355
01356
                   case ALGORITHM_MONTE_CARLO:
                    d *= 0.5;
01357
01358
                     break:
01359
                   default:
01360
                    if (optimize->nsweeps[j] > 1)
01361
                       d /= optimize->nsweeps[j] - 1;
01362
01363
                       d = 0.;
01364
                   }
01365
                 optimize->rangemin[j] -= d;
01366
                 optimize->rangemin[j]
                    = fmax (optimize->rangemin[j], optimize->rangeminabs[j]);
01367
01368
                 optimize->rangemax[j] += d;
01369
                 optimize->rangemax[j]
01370
                    = fmin (optimize->rangemax[j], optimize->rangemaxabs[j]);
                 = fmin (optimize=>rangemax[j], optimize=>rangemaxas;
printf ("%s min=%lg max=%lg\n", optimize=>rangemax[j]);
    optimize=>rangemin[j], optimize=>rangemax[j]);
fprintf (optimize=>file_result, "%s min=%lg max=%lg\n",
    optimize=>label[j], optimize=>rangemin[j],
01371
01372
01373
01374
01375
                           optimize->rangemax[j]);
01376
01377 #if HAVE_MPI
01378
            for (i = 1; (int) i < ntasks; ++i)</pre>
01379
01380
                 MPI_Send (optimize->rangemin, optimize->nvariables, MPI_DOUBLE, i,
01381
                             1, MPI_COMM_WORLD);
01382
                 MPI_Send (optimize->rangemax, optimize->nvariables, MPI_DOUBLE, i,
                            1, MPI_COMM_WORLD);
01383
01384
01385
          }
01386
        else
01387
01388
             MPI_Recv (optimize->rangemin, optimize->nvariables, MPI_DOUBLE, 0, 1,
01389
                        MPI_COMM_WORLD, &mpi_stat);
             MPI_Recv (optimize->rangemax, optimize->nvariables, MPI_DOUBLE, 0, 1,
01390
01391
                       MPI_COMM_WORLD, &mpi_stat);
01392
01393 #endif
01394 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_refine: end\n");
01395
01396 #endif
01397 }
01398
01402 static void
01403 optimize_step ()
01404 {
01405 #if DEBUG_OPTIMIZE
01406 fprintf (stderr, "optimize_step: start\n");
01407 #endif
01408 optimize_algorithm ();
01409 if (optimize->nsteps)
01410
          optimize_climbing (optimize->nsteps);
01411 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_step: end\n");
01412
01413 #endif
01414 }
01415
01419 static inline void
01420 optimize_iterate ()
01421 {
01422
        unsigned int i;
01423 #if DEBUG_OPTIMIZE
01424
        fprintf (stderr, "optimize_iterate: start\n");
01425 #endif
01426
        optimize->error_old = (double *) g_malloc (optimize->nbest * sizeof (double));
        optimize->value old =
01427
           (double *) g_malloc (optimize->nbest * optimize->nvariables *
01428
01429
                                  sizeof (double));
01430
        optimize_step ();
01431
        optimize_save_old ();
01432
        optimize_refine ();
        optimize_print ();
for (i = 1; i < optimize->niterations && !optimize->stop; ++i)
01433
01434
```

```
01435
          {
01436
            optimize_step ();
01437
            optimize_merge_old ();
01438
            optimize_refine ();
01439
            optimize_print ();
01440
01441
        if (optimize->nfinal_steps && !optimize->stop)
01442
          {
01443
            optimize_climbing_best ();
            optimize_climbing (optimize->nfinal_steps);
optimize_merge_old ();
01444
01445
            optimize_print ();
01446
01447
01448 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_iterate: end\n");
01449
01450 #endif
01451 }
01452
01456 static inline void
01457 optimize_save_optimal ()
01458 {
01459
        char cinput[32];
01460
        unsigned int i, j;
        unsigned int flags = 1;
01461
01462
01463
        // Getting optimal values
01464
        memcpy (optimize->value, optimize->value_old,
01465
                optimize->nvariables * sizeof (double));
01466
        // Saving optimal input files
01467
       for (i = 0; i < optimize->ninputs; ++i)
  for (j = 0; j < optimize->nexperiments; ++j)
01468
01469
01470
01471
              snprintf (cinput, 32, "optimal-%u-%u", i, j);
01472 #if DEBUG OPTIMIZE
              fprintf (stderr, "optimize_save_optimal: i=%u j=%u input=%s\n",
01473
                        i, j, cinput);
01474
01475 #endif
01476
              // Checking templates
01477
              if (optimize->template_flags & flags)
01478
                optimize_input (0, cinput, optimize->file[i][j]);
01479
              flags «= 1;
01480
01481 }
01482
01486 void
01487 optimize_free ()
01488 {
        unsigned int i, j;
01489
01490 #if DEBUG_OPTIMIZE
01491
        fprintf (stderr, "optimize_free: start\n");
01492 #endif
01493
        for (j = 0; j < optimize->ninputs; ++j)
01494
            for (i = 0; i < optimize->nexperiments; ++i)
01495
            g_mapped_file_unref (optimize->file[j][i]);
g_free (optimize->file[j]);
01496
01498
01499
       g_free (optimize->error_old);
01500
        g_free (optimize->value_old);
       g_free (optimize->value);
01501
01502
        g_free (optimize->genetic_variable);
01503 #if DEBUG_OPTIMIZE
01504
       fprintf (stderr, "optimize_free: end\n");
01505 #endif
01506 }
01507
01511 void
01512 optimize_open ()
01513 {
01514
        GTimeZone *tz;
01515
        GDateTime *t0, *t;
01516
       unsigned int i, j, nsteps;
01517
01518 #if DEBUG_OPTIMIZE
01519 char *buffer;
01520 fprintf (stde:
        fprintf (stderr, "optimize_open: start\n");
01521 #endif
01522
        // Getting initial time
01523
01524 #if DEBUG_OPTIMIZE
01525
        fprintf (stderr, "optimize_open: getting initial time\n");
01526 #endif
01527
        tz = g_time_zone_new_utc ();
01528
       t0 = g_date_time_new_now (tz);
01529
01530
        // Obtaining and initing the pseudo-random numbers generator seed
```

```
01531 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: getting initial seed\n");
01532
01533 #endif
       if (optimize->seed == DEFAULT_RANDOM_SEED)
01534
          optimize->seed = input->seed;
01535
       gsl_rng_set (optimize->rng, optimize->seed);
01536
01537
01538
        // Obtaining template flags
01539 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: getting template flags\n");
01540
01541 #endif
01542
       optimize->template_flags = input->template_flags;
01543
01544
        // Replacing the working directory
01545 #if DEBUG_OPTIMIZE
01546
       fprintf (stderr, "optimize_open: replacing the working directory\n");
01547 #endif
01548
        g_chdir (input->directory);
01549
01550
        // Getting results file names
01551
        optimize->result = input->result;
01552
        optimize->variables = input->variables;
01553
01554
       // Obtaining the simulator file
01555
       optimize->simulator = input->simulator;
01556
        \ensuremath{//} Obtaining the evaluator file
01557
01558
        optimize->evaluator = input->evaluator;
01559
01560
        // Obtaining the cleaner file
01561
        optimize->cleaner = input->cleaner;
01562
01563
        // Reading the algorithm
01564
        optimize->algorithm = input->algorithm;
01565
        switch (optimize->algorithm)
01566
          case ALGORITHM MONTE CARLO:
01567
01568
            optimize_algorithm = optimize_MonteCarlo;
01569
            break;
01570
          case ALGORITHM_SWEEP:
01571
            optimize_algorithm = optimize_sweep;
01572
           break;
          case ALGORITHM ORTHOGONAL:
01573
           optimize_algorithm = optimize_orthogonal;
01574
01575
           break;
01576
          default:
01577
            optimize_algorithm = optimize_genetic;
01578
            optimize->mutation_ratio = input->mutation_ratio;
            optimize->reproduction_ratio = input->reproduction_ratio;
optimize->adaptation_ratio = input->adaptation_ratio;
01579
01580
01581
01582
        optimize->nvariables = input->nvariables;
01583
        optimize->nsimulations = input->nsimulations;
        optimize->niterations = input->niterations;
01584
        optimize->nbest = input->nbest;
01585
01586
        optimize->tolerance = input->tolerance;
        optimize->nsteps = input->nsteps;
01587
01588
        optimize->nfinal_steps = input->nfinal_steps;
01589
        nsteps = JBM_MAX (optimize->nsteps, optimize->nfinal_steps);
01590
        optimize->nestimates = 0;
        optimize > hestimate;
optimize > threshold = input -> threshold;
01591
01592
        optimize->stop = 0;
01593
        if (nsteps)
01594
01595
            optimize->relaxation = input->relaxation;
01596
            switch (input->climbing)
01597
              case CLIMBING_METHOD_COORDINATES:
01598
01599
               optimize->nestimates = 2 * optimize->nvariables;
01600
                optimize_estimate_climbing = optimize_estimate_climbing_coordinates;
01601
                break;
01602
              default:
                optimize->nestimates = input->nestimates;
optimize_estimate_climbing = optimize_estimate_climbing_random;
01603
01604
              }
01605
01606
01607
01608 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: nbest=%u\n", optimize->nbest);
01609
01610 #endif
       optimize->simulation_best
01611
01612
          = (unsigned int *) alloca (optimize->nbest * sizeof (unsigned int));
01613
        optimize->error_best = (double *) alloca (optimize->nbest * sizeof (double));
01614
01615
       // Reading the experimental data
01616 #if DEBUG OPTIMIZE
01617
       buffer = g_get_current_dir ();
```

4.22 optimize.c 273

```
fprintf (stderr, "optimize_open: current directory=%s\n", buffer);
01619
        g free (buffer);
01620 #endif
01621
        optimize->nexperiments = input->nexperiments;
01622
        optimize->ninputs = input->experiment->ninputs;
01623
        optimize->experiment
01624
          = (char **) alloca (input->nexperiments * sizeof (char *));
        optimize->weight = (double *) alloca (input->nexperiments * sizeof (double));
01625
01626
       for (i = 0; i < input->experiment->ninputs; ++i)
         optimize->file[i] = (GMappedFile **)
01627
            g_malloc (input->nexperiments * sizeof (GMappedFile *));
01628
01629
       for (i = 0; i < input->nexperiments; ++i)
01630
01631 #if DEBUG_OPTIMIZE
01632
            fprintf (stderr, "optimize_open: i=%u\n", i);
01633 #endif
            optimize->experiment[i] = input->experiment[i].name;
01634
            optimize->weight[i] = input->experiment[i].weight;
01635
01636 #if DEBUG OPTIMIZE
            fprintf (stderr, "optimize_open: experiment=%s weight=%lg\n",
                     optimize->experiment[i], optimize->weight[i]);
01638
01639 #endif
01640
            for (j = 0; j < input->experiment->ninputs; ++j)
01641
01642 #if DEBUG_OPTIMIZE
01643
               fprintf (stderr, "optimize_open: stencil%u\n", j + 1);
01644 #endif
01645
               optimize->file[j][i]
01646
                   = g_mapped_file_new (input->experiment[i].stencil[j], 0, NULL);
01647
              }
01648
         }
01649
01650
       // Reading the variables data
01651 #if DEBUG_OPTIMIZE
01652
       fprintf (stderr, "optimize_open: reading variables\n");
01653 #endif
       optimize->label = (char **) alloca (input->nvariables * sizeof (char *));
01654
        j = input->nvariables * sizeof (double);
01655
01656
        optimize->rangemin = (double *) alloca (j);
01657
        optimize->rangeminabs = (double *) alloca (j);
01658
        optimize->rangemax = (double *) alloca (j);
        optimize->rangemaxabs = (double *) alloca (j);
01659
        optimize->step = (double *) alloca (j);
01660
        j = input->nvariables * sizeof (unsigned int);
optimize->precision = (unsigned int *) alloca (j);
01661
01662
01663
        optimize->nsweeps = (unsigned int *) alloca (j);
01664
        optimize->nbits = (unsigned int *) alloca (j);
01665
        for (i = 0; i < input->nvariables; ++i)
01666
01667
            optimize->label[i] = input->variable[i].name;
01668
            optimize->rangemin[i] = input->variable[i].rangemin;
01669
            optimize->rangeminabs[i] = input->variable[i].rangeminabs;
01670
            optimize->rangemax[i] = input->variable[i].rangemax;
            optimize->rangemaxabs[i] = input->variable[i].rangemaxabs;
optimize->precision[i] = input->variable[i].precision;
01671
01672
            optimize->step[i] = input->variable[i].step;
optimize->nsweeps[i] = input->variable[i].nsweeps;
01673
01674
01675
            optimize->nbits[i] = input->variable[i].nbits;
01676
01677
        if (input->algorithm == ALGORITHM_SWEEP
            || input->algorithm == ALGORITHM_ORTHOGONAL)
01678
01679
01680
            optimize->nsimulations = 1;
            for (i = 0; i < input->nvariables; ++i)
01682
01683
                optimize->nsimulations *= optimize->nsweeps[i];
01684 #if DEBUG_OPTIMIZE
                fprintf (stderr, "optimize_open: nsweeps=%u nsimulations=%u\n",
01685
01686
                          optimize->nsweeps[i], optimize->nsimulations);
01687 #endif
01688
01689
01690
       if (nsteps)
         optimize->climbing
01691
            = (double *) alloca (optimize->nvariables * sizeof (double));
01692
01693
01694
        // Setting error norm
01695
        switch (input->norm)
01696
01697
          case ERROR NORM EUCLIDIAN:
01698
           optimize norm = optimize norm euclidian;
01699
            break;
01700
          case ERROR_NORM_MAXIMUM:
01701
            optimize_norm = optimize_norm_maximum;
01702
            break;
          case ERROR NORM P:
01703
01704
            optimize norm = optimize norm p;
```

```
optimize->p = input->p;
01706
           break;
01707
          default:
01708
           optimize_norm = optimize_norm_taxicab;
01709
01710
01711
        // Allocating values
01712 #if DEBUG_OPTIMIZE
01713 fprintf (stderr, "optimize_open: allocating variables\n"); 01714 fprintf (stderr, "optimize_open: nvariables=%u algorithm=%u\n",
                 optimize->nvariables, optimize->algorithm);
01715
01716 #endif
       optimize->genetic_variable = NULL;
if (optimize->algorithm == ALGORITHM_GENETIC)
01717
01718
01719
01720
            optimize->genetic_variable = (GeneticVariable *)
              g_malloc (optimize->nvariables * sizeof (GeneticVariable));
01721
            for (i = 0; i < optimize->nvariables; ++i)
01722
01724 #if DEBUG_OPTIMIZE
01725
                fprintf (stderr, "optimize_open: i=%u min=%lg max=%lg nbits=%u\n",
                          i, optimize->rangemin[i], optimize->rangemax[i],
01726
                         optimize->nbits[i]);
01727
01728 #endif
01729
               optimize->genetic_variable[i].minimum = optimize->rangemin[i];
01730
               optimize->genetic_variable[i].maximum = optimize->rangemax[i];
01731
                optimize->genetic_variable[i].nbits = optimize->nbits[i];
01732
01733
01734 #if DEBUG_OPTIMIZE
01735 fprintf (stderr, "optimize_open: nvariables=%u nsimulations=%u\n",
                 optimize->nvariables, optimize->nsimulations);
01737 #endif
01738 optimize->value = (double *)
         g_malloc ((optimize->nsimulations + optimize->nestimates * nsteps)
    * optimize->nvariables * sizeof (double));
01739
01740
01741
01742
        // Calculating simulations to perform for each task
01743 #if HAVE_MPI
01744 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: rank=%u ntasks=%u\n",
01745
01746
                 optimize->mpi_rank, ntasks);
01747 #endif
01748
       optimize->nstart = optimize->mpi_rank * optimize->nsimulations / ntasks;
01749
        optimize->nend = (1 + optimize->mpi_rank) * optimize->nsimulations / ntasks;
01750
        if (nsteps)
01751
01752
            optimize->nstart_climbing
01753
              = optimize->mpi_rank * optimize->nestimates / ntasks;
01754
            optimize->nend_climbing
              = (1 + optimize->mpi_rank) * optimize->nestimates / ntasks;
01755
01756
01757 #else
01758
       optimize->nstart = 0;
        optimize->nend = optimize->nsimulations;
01759
01760
        if (nsteps)
01761
01762
            optimize->nstart_climbing = 0;
01763
           optimize->nend_climbing = optimize->nestimates;
01764
01765 #endif
01766 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: nstart=%u nend=%u\n", optimize->nstart,
01768
                 optimize->nend);
01769 #endif
01770
01771
        // Calculating simulations to perform for each thread
01772
       optimize->thread
         = (unsigned int *) alloca ((1 + nthreads) * sizeof (unsigned int));
01773
01774
        for (i = 0; i <= nthreads; ++i)</pre>
01775
01776
            optimize->thread[i] = optimize->nstart
01777
              + i * (optimize->nend - optimize->nstart) / nthreads;
01778 #if DEBUG_OPTIMIZE
           01779
01780
01781 #endif
        }
if (nsteps)
01782
01783
01784
         optimize->thread_climbing = (unsigned int *)
01785
            alloca ((1 + nthreads_climbing) * sizeof (unsigned int));
01787
        // Opening result files
01788
        optimize->file_result = g_fopen (optimize->result, "w");
       optimize->file_variables = g_fopen (optimize->variables, "w");
01789
01790
01791
        // Performing the algorithm
```

```
switch (optimize->algorithm)
01793
             // Genetic algorithm
01794
           case ALGORITHM_GENETIC:
01795
01796
             optimize_genetic ();
01797
             break:
01798
01799
              // Iterative algorithm
01800
          optimize_iterate ();
}
           default:
01801
01802
01803
01804
         // Getting calculation time
01805
         t = g_date_time_new_now (tz);
01806
         optimize->calculation_time = 0.000001 * g_date_time_difference (t, t0);
        g_date_time_unref (t);
g_date_time_unref (t0);
01807
g_time_zone_unref (tz);
01810 g_time_zone_unref (tz);
01810 printf ("%s = %.61g s\n", _("Calculation time"), optimize->calculation_time);
01811 fprintf (optimize->file_result, "%s = %.61g s\n"
01812
                   _("Calculation time"), optimize->calculation_time);
01813
         // Closing result files
01814
01815
        optimize_save_optimal ();
fclose (optimize->file_variables);
01816
01817
        fclose (optimize->file_result);
01818
01819 #if DEBUG_OPTIMIZE
01820 fprintf (stderr, "optimize_open: end\n");
01821 #endif
01822 }
```

4.23 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.

4.23.1 Detailed Description

Header file to define the optimization functions.

Authors

Javier Burguete.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file optimize.h.

4.23.2 Function Documentation

4.23.2.1 optimize free()

```
void optimize_free ( )
```

Function to free the memory used by the Optimize struct.

Definition at line 1487 of file optimize.c.

```
01488 {
01489
 01489 unsigned int i, j;
01490 #if DEBUG_OPTIMIZE
 01491
             fprintf (stderr, "optimize_free: start\n");
 01492 #endif
              for (j = 0; j < optimize->ninputs; ++j)
 01493
 01494
                     for (i = 0; i < optimize->nexperiments; ++i)
  g_mapped_file_unref (optimize->file[j][i]);
 01495
 01497
                      g_free (optimize->file[j]);
 01498
01499 g_free (optimize->error_old);
01500 g_free (optimize->value_old);
01501 g_free (optimize->value);
01502 g_free (optimize->genetic_var
g_tree (optimize->value);

01502 g_free (optimize->genetic_variable);

01503 #if DEBUG_OPTIMIZE

01504 fprintf (stderr, "optimize_free: end\n");

01505 #endif
01506 }
```

4.23.2.2 optimize_open()

```
void optimize_open ( )
```

Function to open and perform a optimization.

```
Definition at line 1512 of file optimize.c.
```

```
01513 {
01514
        GTimeZone *tz;
01515
        GDateTime *t0, *t;
        unsigned int i, j, nsteps;
01516
01517
01518 #if DEBUG_OPTIMIZE
      char *buffer;
01519
01520
        fprintf (stderr, "optimize_open: start\n");
01521 #endif
01522
01523
        // Getting initial time
01524 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_open: getting initial time\n");
01525
01526 #endif
01527
       tz = g_time_zone_new_utc ();
01528
       t0 = g_date_time_new_now (tz);
01529
01530
        // Obtaining and initing the pseudo-random numbers generator seed
01531 #if DEBUG_OPTIMIZE
01532
        fprintf (stderr, "optimize_open: getting initial seed\n");
01533 #endif
01534
       if (optimize->seed == DEFAULT_RANDOM_SEED)
         optimize->seed = input->seed;
01535
       gsl_rng_set (optimize->rng, optimize->seed);
01536
01537
01538
        // Obtaining template flags
01539 #if DEBUG_OPTIMIZE
        fprintf (stderr, "optimize_open: getting template flags\n");
01540
01541 #endif
01542
        optimize->template_flags = input->template_flags;
01543
01544
        // Replacing the working directory
01545 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: replacing the working directory\n");
01546
01547 #endif
        g_chdir (input->directory);
01548
01549
01550
        // Getting results file names
01551
        optimize->result = input->result;
01552
        optimize->variables = input->variables;
01553
01554
        // Obtaining the simulator file
01555
        optimize->simulator = input->simulator;
01556
01557
        // Obtaining the evaluator file
01558
        optimize->evaluator = input->evaluator;
01559
01560
        // Obtaining the cleaner file
01561
        optimize->cleaner = input->cleaner;
01562
01563
        // Reading the algorithm
01564
        optimize->algorithm = input->algorithm;
01565
        switch (optimize->algorithm)
01566
01567
          case ALGORITHM_MONTE_CARLO:
01568
            optimize_algorithm = optimize_MonteCarlo;
01569
            break;
01570
          case ALGORITHM_SWEEP:
01571
            optimize_algorithm = optimize_sweep;
01572
            break;
01573
          case ALGORITHM_ORTHOGONAL:
01574
           optimize_algorithm = optimize_orthogonal;
01575
            break;
01576
01577
            optimize_algorithm = optimize_genetic;
            optimize->mutation_ratio = input->mutation_ratio;
optimize->reproduction_ratio = input->reproduction_ratio;
01578
01579
            optimize->adaptation_ratio = input->adaptation_ratio;
01580
01581
01582
        optimize->nvariables = input->nvariables;
        optimize->nsimulations = input->nsimulations;
optimize->niterations = input->niterations;
01583
01584
        optimize->nbest = input->nbest;
optimize->tolerance = input->tolerance;
01585
01586
01587
        optimize->nsteps = input->nsteps;
        optimize->nfinal_steps = input->nfinal_steps;
01588
```

```
nsteps = JBM_MAX (optimize->nsteps, optimize->nfinal_steps);
       optimize->nestimates = 0;
01590
        optimize->threshold = input->threshold;
01591
01592
       optimize->stop = 0;
01593
       if (nsteps)
01594
01595
            optimize->relaxation = input->relaxation;
01596
            switch (input->climbing)
01597
01598
              case CLIMBING METHOD COORDINATES:
01599
               optimize->nestimates = 2 * optimize->nvariables;
01600
                optimize_estimate_climbing = optimize_estimate_climbing_coordinates;
01601
                break;
01602
01603
               optimize->nestimates = input->nestimates;
01604
                optimize_estimate_climbing = optimize_estimate_climbing_random;
01605
01606
         }
01607
01608 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: nbest=%u\n", optimize->nbest);
01609
01610 #endif
01611
       optimize->simulation_best
       = (unsigned int *) alloca (optimize->nbest * sizeof (unsigned int));
optimize->error_best = (double *) alloca (optimize->nbest * sizeof (double));
01612
01613
01614
01615
       // Reading the experimental data
01616 #if DEBUG_OPTIMIZE
01617
       buffer = g_get_current_dir ();
       fprintf (stderr, "optimize_open: current directory=%s\n", buffer);
01618
01619
       g free (buffer);
01620 #endif
01621
       optimize->nexperiments = input->nexperiments;
01622
        optimize->ninputs = input->experiment->ninputs;
01623
       optimize->experiment
         = (char **) alloca (input->nexperiments * sizeof (char *));
01624
       optimize->weight = (double *) alloca (input->nexperiments * sizeof (double));
01625
       for (i = 0; i < input->experiment->ninputs; ++i)
01626
01627
        optimize->file[i] = (GMappedFile **)
01628
            g_malloc (input->nexperiments * sizeof (GMappedFile *));
01629
       for (i = 0; i < input->nexperiments; ++i)
01630
01631 #if DEBUG OPTIMIZE
            fprintf (stderr, "optimize_open: i=%u\n", i);
01632
01633 #endif
01634
           optimize->experiment[i] = input->experiment[i].name;
01635
            optimize->weight[i] = input->experiment[i].weight;
01636 #if DEBUG OPTIMIZE
           01637
01638
01639 #endif
01640
           for (j = 0; j < input->experiment->ninputs; ++j)
01641
01642 #if DEBUG_OPTIMIZE
               fprintf (stderr, "optimize_open: stencil%u\n", j + 1);
01643
01644 #endif
01645
               optimize->file[i][i]
                  = g_mapped_file_new (input->experiment[i].stencil[j], 0, NULL);
01646
01647
01648
         }
01649
01650
        // Reading the variables data
01651 #if DEBUG_OPTIMIZE
       fprintf (stderr, "optimize_open: reading variables\n");
01652
01653 #endif
01654
       optimize->label = (char **) alloca (input->nvariables * sizeof (char *));
01655
        j = input->nvariables * sizeof (double);
01656
       optimize->rangemin = (double *) alloca (i);
01657
       optimize->rangeminabs = (double *) alloca (j);
       optimize->rangemax = (double *) alloca (j);
01658
01659
        optimize->rangemaxabs = (double *) alloca (j);
01660
       optimize->step = (double *) alloca (j);
        j = input->nvariables * sizeof (unsigned int);
01661
       optimize->precision = (unsigned int *) alloca (j);
optimize->nsweeps = (unsigned int *) alloca (j);
01662
01663
01664
        optimize->nbits = (unsigned int *) alloca (j);
01665
        for (i = 0; i < input->nvariables; ++i)
01666
01667
            optimize->label[i] = input->variable[i].name;
            optimize->rangemin[i] = input->variable[i].rangemin;
01668
            optimize->rangeminabs[i] = input->variable[i].rangeminabs;
01669
01670
            optimize->rangemax[i] = input->variable[i].rangemax;
01671
            optimize->rangemaxabs[i] = input->variable[i].rangemaxabs;
01672
            optimize->precision[i] = input->variable[i].precision;
01673
            optimize->step[i] = input->variable[i].step;
            optimize->nsweeps[i] = input->variable[i].nsweeps;
01674
01675
            optimize->nbits[i] = input->variable[i].nbits;
```

```
01677
        if (input->algorithm == ALGORITHM_SWEEP
01678
            || input->algorithm == ALGORITHM_ORTHOGONAL)
01679
01680
            optimize->nsimulations = 1;
            for (i = 0; i < input->nvariables; ++i)
01681
01682
01683
                 optimize->nsimulations *= optimize->nsweeps[i];
01684 #if DEBUG_OPTIMIZE
                 fprintf (stderr, "optimize_open: nsweeps=%u nsimulations=%u\n",
01685
                          optimize->nsweeps[i], optimize->nsimulations);
01686
01687 #endif
01688
              }
01689
01690
        if (nsteps)
01691
        optimize->climbing
             = (double *) alloca (optimize->nvariables * sizeof (double));
01692
01693
        // Setting error norm
01694
01695
        switch (input->norm)
01696
01697
          case ERROR_NORM_EUCLIDIAN:
          optimize_norm = optimize_norm_euclidian;
01698
01699
            break:
01700
          case ERROR_NORM_MAXIMUM:
          optimize_norm = optimize_norm_maximum;
01701
01702
            break;
01703
          case ERROR_NORM_P:
           optimize_norm = optimize_norm_p;
optimize->p = input->p;
01704
01705
01706
            break:
01707
          default:
01708
            optimize_norm = optimize_norm_taxicab;
01709
01710
        // Allocating values
01711
01712 #if DEBUG_OPTIMIZE
01713 fprintf (stderr, "optimize_open: allocating variables\n");
01714 fprintf (stderr, "optimize_open: nvariables=%u algorithm=%u\n",
01715
                 optimize->nvariables, optimize->algorithm);
01716 #endif
        optimize->genetic variable = NULL;
01717
        if (optimize->algorithm == ALGORITHM_GENETIC)
01718
01719
01720
            optimize->genetic_variable = (GeneticVariable *)
01721
               g_malloc (optimize->nvariables * sizeof (GeneticVariable));
01722
             for (i = 0; i < optimize->nvariables; ++i)
01723
01724 #if DEBUG_OPTIMIZE
01725
               fprintf (stderr, "optimize_open: i=%u min=%lg max=%lg nbits=%u\n",
                          i, optimize->rangemin[i], optimize->rangemax[i],
01726
01727
                           optimize->nbits[i]);
01728 #endif
                optimize->genetic_variable[i].minimum = optimize->rangemin[i];
optimize->genetic_variable[i].maximum = optimize->rangemax[i];
01729
01730
                optimize->genetic_variable[i].nbits = optimize->nbits[i];
01731
01732
01733
01734 #if DEBUG_OPTIMIZE
01735 fprintf (stderr, "optimize_open: nvariables=%u nsimulations=%u\n",
01736
                  optimize->nvariables, optimize->nsimulations);
01737 #endif
01738 optimize->value = (double *)
01739
        g_malloc ((optimize->nsimulations + optimize->nestimates * nsteps)
01740
                      * optimize->nvariables * sizeof (double));
01741
        // Calculating simulations to perform for each task
01742
01743 #if HAVE_MPI
01744 #if DEBUG_OPTIMIZE
01745 fprintf (stderr, "optimize_open: rank=%u ntasks=%u\n",
01746
                 optimize->mpi_rank, ntasks);
01747 #endif
       optimize->nstart = optimize->mpi_rank * optimize->nsimulations / ntasks;
optimize->nend = (1 + optimize->mpi_rank) * optimize->nsimulations / ntasks;
01748
01749
01750
        if (nsteps)
01751
01752
             optimize->nstart_climbing
01753
               = optimize->mpi_rank * optimize->nestimates / ntasks;
01754
             optimize->nend_climbing
               = (1 + optimize->mpi_rank) * optimize->nestimates / ntasks:
01755
01756
01757 #else
01758
       optimize->nstart = 0;
01759
        optimize->nend = optimize->nsimulations;
01760
        if (nsteps)
01761
01762
            optimize->nstart climbing = 0:
```

```
optimize->nend_climbing = optimize->nestimates;
01764
01765 #endif
01766 #if DEBUG OPTIMIZE
01767 fprintf (stderr, "optimize_open: nstart=%u nend=%u\n", optimize->nstart,
01768
                optimize->nend);
01769 #endif
01770
01771
       // Calculating simulations to perform for each thread
01772
       optimize->thread
01773
         = (unsigned int *) alloca ((1 + nthreads) * sizeof (unsigned int));
01774
       for (i = 0; i <= nthreads; ++i)</pre>
01775
01776
           optimize->thread[i] = optimize->nstart
01777
             + i * (optimize->nend - optimize->nstart) / nthreads;
01778 #if DEBUG_OPTIMIZE
           fprintf (stderr, "optimize_open: i=%u thread=%u\n", i,
01779
01780
                    optimize->thread[i]);
01781 #endif
01782
       if (nsteps)
01783
01784
        optimize->thread_climbing = (unsigned int *)
           alloca ((1 + nthreads_climbing) * sizeof (unsigned int));
01785
01786
01787
       // Opening result files
01788
       optimize->file_result = g_fopen (optimize->result, "w");
01789
       optimize->file_variables = g_fopen (optimize->variables, "w");
01790
01791
       // Performing the algorithm
01792
       switch (optimize->algorithm)
01793
        {
01794
           // Genetic algorithm
01795
        case ALGORITHM_GENETIC:
01796
          optimize_genetic ();
01797
           break;
01798
01799
           // Iterative algorithm
01800
         default:
01801
           optimize_iterate ();
01802
01803
       // Getting calculation time
01804
      t = g_date_time_new_now (tz);
optimize->calculation_time = 0.000001 * g_date_time_difference (t, t0);
01805
01806
01807
       g_date_time_unref (t);
01808
       g_date_time_unref (t0);
01809
       g_time_zone_unref (tz);
       01810
01811
01812
01813
01814
       // Closing result files
01815
       optimize_save_optimal ();
01816
      fclose (optimize->file_variables);
01817
       fclose (optimize->file_result);
01818
01819 #if DEBUG_OPTIMIZE
01820
      fprintf (stderr, "optimize_open: end\n");
01821 #endif
01822 }
```

Here is the call graph for this function:

4.23.3 Variable Documentation

4.23.3.1 nthreads_climbing

unsigned int nthreads_climbing [extern]

Number of threads for the hill climbing method.

Definition at line 84 of file optimize.c.

4.24 optimize.h 281

4.23.3.2 optimize

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

Optimization data.

Definition at line 83 of file optimize.c.

4.24 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 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 OPTIMIZE__H
00039 #define OPTIMIZE__H 1
00040
00045 typedef struct
00046 {
00047
       GMappedFile **file[MAX_NINPUTS];
00048
       char **experiment;
char **label;
00049
00050
        qsl_rnq *rnq;
00051
        GeneticVariable *genetic_variable;
        FILE *file_result;
00053
00054
       FILE *file_variables;
00055
       char *result;
00056
       char *variables:
00057
       char *simulator;
00058
       char *evaluator;
00060
        char *cleaner;
00061
        double *value;
00062
       double *rangemin;
00063
       double *rangemax;
00064
       double *rangeminabs;
       double *rangemaxabs;
00065
00066
        double *error_best;
00067
        double *weight;
00068
       double *step;
00069
       double *climbing;
00070
       double *value_old;
00072
        double *error_old;
00074
        unsigned int *precision;
00075
        unsigned int *nsweeps;
00076
        unsigned int *nbits;
00078
        unsigned int *thread;
08000
       unsigned int *thread_climbing;
00083
       unsigned int *simulation_best;
00084
       double tolerance;
```

```
double mutation_ratio;
00086
       double reproduction_ratio;
00087
        double adaptation_ratio;
00088
       double relaxation;
00089
       double calculation time;
       double p;
double threshold;
00090
00091
00092
       unsigned long int seed;
00094
       unsigned int nvariables;
00095
       unsigned int nexperiments;
00096
       unsigned int ninputs;
00097
       unsigned int nsimulations;
00098
       unsigned int nsteps;
00100
       unsigned int nfinal_steps;
00102
       unsigned int nestimates;
00104
       unsigned int algorithm;
00105
       unsigned int nstart;
00106
       unsigned int nend;
00107
       unsigned int nstart_climbing;
00109
       unsigned int nend_climbing;
00111
       unsigned int niterations;
00112
       unsigned int nbest;
00113
       unsigned int nsaveds;
00114
       unsigned int stop;
00115
       unsigned int template_flags;
00116 #if HAVE_MPI
00117
       int mpi_rank;
00118 #endif
00119 } Optimize;
00120
00125 typedef struct
00126 {
00127
       unsigned int thread;
00128 } ParallelData;
00129
00130 // Global variables
00131 extern int ntasks;
00132 extern unsigned int nthreads;
00133 extern unsigned int nthreads_climbing;
00134 extern GMutex mutex[1];
00135 extern Optimize optimize[1];
00136
00137 // Public functions
00138 void optimize_free ();
00139 void optimize_open ();
00140
00141 #endif
```

4.25 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/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.

4.25 tools.c File Reference 283

4.25.1 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.

4.25.2 Variable Documentation

4.25.2.1 error_message

char* error_message

Error message.

Definition at line 59 of file tools.c.

4.25.2.2 main_window

 ${\tt GtkWindow*\ main_window}$

Main GtkWindow.

Definition at line 56 of file tools.c.

4.25.2.3 show_pending

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

Pointer to the function to show pending events.

Definition at line 60 of file tools.c.

4.26 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.
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 <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/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;
```

4.27 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.

4.27 tools.h File Reference 285

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.

4.27.1 Detailed Description

Header file to define some useful functions.

Authors

Javier Burguete.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file tools.h.

4.27.2 Macro Definition Documentation

4.27.2.1 ERROR_TYPE

#define ERROR_TYPE GTK_MESSAGE_ERROR

Macro to define the error message type.

Definition at line 48 of file tools.h.

4.27.2.2 INFO_TYPE

#define INFO_TYPE GTK_MESSAGE_INFO

Macro to define the information message type.

Definition at line 49 of file tools.h.

4.27.3 Variable Documentation

4.27.3.1 error_message

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

Error message.

Definition at line 59 of file tools.c.

4.27.3.2 main_window

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

Main GtkWindow.

Definition at line 56 of file tools.c.

4.27.3.3 show_pending

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

Pointer to the function to show pending events.

Definition at line 60 of file tools.c.

4.28 tools.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 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) ();
00062 #endif
```

4.29 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/xml.h"
#include "jb/src/json.h"
#include "jb/src/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.

4.29.1 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.

4.29.2 Macro Definition Documentation

4.29.2.1 DEBUG_VARIABLE

```
#define DEBUG_VARIABLE 0
```

Macro to debug variable functions.

Definition at line 51 of file variable.c.

4.29.3 Function Documentation

4.29.3.1 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.

4.29.3.2 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.

```
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 }
```

4.29.3.3 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.

```
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);
       if (!label)
00284
00285
00286
            variable_error (variable, _("no name"));
00287
           goto exit_on_error;
00288
00289
       variable->name = g_strdup (label);
       if (json_object_get_member (object, LABEL_MINIMUM))
00290
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
                variable_error (variable, _("minimum range not allowed"));
00309
               goto exit_on_error;
00310
00311
00312
00313
       else
00314
            variable_error (variable, _("no minimum range"));
00315
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
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
             {
                variable_error (variable, _("bad range"));
00342
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
         = jb_json_object_get_uint_with_default (object, LABEL_PRECISION,
00352
                                                   &error_code, DEFAULT_PRECISION);
00353
00354
        if (!error code || variable->precision >= NPRECISIONS)
00355
         {
00356
            variable_error (variable, _("bad precision"));
00357
            goto exit_on_error;
00358
        if (algorithm == ALGORITHM_SWEEP || algorithm == ALGORITHM_ORTHOGONAL)
00359
00360
```

```
if (json_object_get_member (object, LABEL_NSWEEPS))
00362
00363
                variable->nsweeps
                  = jb_json_object_get_uint (object, LABEL_NSWEEPS, &error_code);
00364
                if (!error_code || !variable->nsweeps)
00365
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
            fprintf (stderr, "variable_open_json: nsweeps=%u\n", variable->nsweeps);
00377
00378 #endif
00380
           (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
00394
00395
                variable_error (variable, _("no bits number"));
00396
                goto exit_on_error;
00397
00398
00399
       else if (nsteps)
00400
00401
           variable->step
             = jb_json_object_get_float (object, LABEL_STEP, &error_code);
00402
00403
            if (!error_code || variable->step < 0.)</pre>
00404
00405
               variable_error (variable, _("bad step size"));
00406
                goto exit_on_error;
00407
00408
         }
00409
00410 #if DEBUG_VARIABLE
       fprintf (stderr, "variable_open_json: end\n");
00411
00412 #endif
00413
       return 1;
00414 exit_on_error:
00415 variable_free (variable, INPUT_TYPE_JSON); 00416 #if DEBUG_VARIABLE
       fprintf (stderr, "variable_open_json: end\n");
00418 #endif
00419
       return 0;
00420 }
```

Here is the call graph for this function:

4.29.3.4 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.

```
00112 {
        int error code;
00113
00114
00115 #if DEBUG_VARIABLE
00116
       fprintf (stderr, "variable_open_xml: start\n");
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
                variable_error (variable, _("bad minimum"));
00132
00133
               goto exit_on_error;
00134
            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
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;
00155
        if (xmlHasProp (node, (const xmlChar *) LABEL_MAXIMUM))
00156
00157
            variable->rangemax
              = jb_xml_node_get_float (node, (const xmlChar *) LABEL_MAXIMUM,
00158
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)
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
```

```
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,
00192
                                                &error_code, DEFAULT_PRECISION);
00193
        if (!error_code || variable->precision >= NPRECISIONS)
00194
            variable_error (variable, _("bad precision"));
00195
00196
            goto exit_on_error;
00197
00198
           (algorithm == ALGORITHM_SWEEP || algorithm == ALGORITHM_ORTHOGONAL)
00199
00200
            if (xmlHasProp (node, (const xmlChar *) LABEL_NSWEEPS))
00201
00202
               variable->nsweeps
                 = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSWEEPS,
00204
                                           &error_code);
               if (!error_code || !variable->nsweeps)
00205
00206
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
00217
            fprintf (stderr, "variable_open_xml: nsweeps=%u\n", variable->nsweeps);
00218 #endif
00219
       if (algorithm == ALGORITHM_GENETIC)
00220
00221
         {
            // Obtaining bits representing each variable
00223
            if (xmlHasProp (node, (const xmlChar *) LABEL_NBITS))
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
             {
00236
               variable_error (variable, _("no bits number"));
00237
               goto exit_on_error;
00238
             }
00239
00240
       else if (nsteps)
00241
00242
00243
             = jb_xml_node_get_float (node, (const xmlChar *) LABEL_STEP,
00244
                                        &error_code);
            if (!error_code || variable->step < 0.)</pre>
00245
00246
00247
               variable_error (variable, _("bad step size"));
00248
               goto exit_on_error;
00249
00250
        }
00251
00252 #if DEBUG_VARIABLE
00253 fprintf (stderr, "variable_open_xml: end\n");
00254 #endif
00255
        return 1;
00256 exit_on_error:
00257 variable_free (variable, INPUT_TYPE_XML); 00258 #if DEBUG_VARIABLE
       fprintf (stderr, "variable_open_xml: end\n");
00259
00260 #endif
00261
       return 0;
00262 }
```

Here is the call graph for this function:

4.29.4 Variable Documentation

4.29.4.1 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.

4.29.4.2 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.

4.30 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.
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 <libxml/parser.h>
```

4.30 variable.c 295

```
00042 #include <libintl.h>
00043 #include <glib.h>
00044 #include <json-glib/json-glib.h>
00045 #include "jb/src/xml.h"
00046 #include "jb/src/json.h"
00047 #include "jb/src/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
00061 };
00062
00066 void
00067 variable_free (Variable * variable,
00069
                     unsigned int type)
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 }
00083
00087 void
00088 variable_error (Variable * variable,
00090
                      char *message)
00092 {
00093
        char buffer[64];
       if (!variable->name)
00094
         snprintf (buffer, 64, "%s: %s", _("Variable"), message);
00095
00096
00097
          snprintf (buffer, 64, "%s %s: %s", _("Variable"), variable->name, message);
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;
00115 #if DEBUG_VARIABLE
00116 fprintf (stderr, "variable_open_xml: start\n");
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
        if (xmlHasProp (node, (const xmlChar *) LABEL_MINIMUM))
00125
00126
            variable->rangemin
00128
              = jb_xml_node_get_float (node, (const xmlChar *) LABEL_MINIMUM,
00129
                                         &error_code);
            if (!error_code)
00130
00131
                variable_error (variable, _("bad minimum"));
00132
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
                -G MAXDOUBLE):
00137
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)
```

```
{
                variable_error (variable, _("minimum range not allowed"));
00146
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
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
               G MAXDOUBLE);
00167
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
                variable_error (variable, _("bad range"));
00180
                goto exit_on_error;
00181
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,
00191
                                                 (const xmlChar *) LABEL_PRECISION,
        &error_code, DEFAULT_PRECISION);
if (!error_code || variable->precision >= NPRECISIONS)
00192
00193
00194
00195
            variable_error (variable, _("bad precision"));
00196
            goto exit_on_error;
00197
00198
        if (algorithm == ALGORITHM_SWEEP || algorithm == ALGORITHM_ORTHOGONAL)
00199
            if (xmlHasProp (node, (const xmlChar *) LABEL_NSWEEPS))
00200
00202
00203
                  = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSWEEPS,
00204
                                            &error_code);
                if (!error_code || !variable->nsweeps)
00205
00206
00207
                    variable_error (variable, _("bad sweeps"));
00208
                    goto exit_on_error;
00209
                  }
00210
00211
            else
00212
00213
                variable_error (variable, _("no sweeps number"));
                goto exit_on_error;
00215
00216 #if DEBUG_VARIABLE
            fprintf (stderr, "variable_open_xml: nsweeps=%u\n", variable->nsweeps);
00217
00218 #endif
00219
           (algorithm == ALGORITHM_GENETIC)
00220
00221
         {
00222
            // Obtaining bits representing each variable
00223
            if (xmlHasProp (node, (const xmlChar *) LABEL_NBITS))
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
                    variable_error (variable, _("invalid bits number"));
00230
00231
                    goto exit on error:
```

4.30 variable.c 297

```
}
00233
00234
            else
00235
            {
00236
               variable_error (variable, _("no bits number"));
00237
               goto exit_on_error;
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"));
00248
               goto exit_on_error;
             }
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
00259
       fprintf (stderr, "variable_open_xml: end\n");
00260 #endif
00261
       return 0;
00262 }
00263
00269 int
00270 variable_open_json (Variable \star 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");
00280
00281 #endif
       object = json_node_get_object (node);
00283
        label = json_object_get_string_member (object, LABEL_NAME);
00284
       if (!label)
00285
            variable_error (variable, _("no name"));
00286
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
               variable_error (variable, _("bad absolute minimum"));
00304
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
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"));
```

```
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)
00332
                variable_error (variable, _("bad absolute maximum"));
00333
                goto exit_on_error;
00334
            if (variable->rangemax > variable->rangemaxabs)
00335
00336
                variable_error (variable, _("maximum range not allowed"));
00337
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
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))
00362
00363
               variable->nsweeps
00364
                  = 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;
00376 #if DEBUG_VARIABLE
00377
            fprintf (stderr, "variable_open_json: nsweeps=%u\n", variable->nsweeps);
00378 #endif
00379
00380
        if
           (algorithm == ALGORITHM GENETIC)
00382
            // Obtaining bits representing each variable
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
         }
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
             {
00405
               variable_error (variable, _("bad step size"));
00406
               goto exit_on_error;
00407
00408
         }
00409
00410 #if DEBUG VARIABLE
       fprintf (stderr, "variable_open_json: end\n");
00411
```

```
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 }
```

4.31 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.

4.31.1 Detailed Description

Header file to define the variable data.

Authors

Javier Burguete.

Copyright

Copyright 2012-2023, all rights reserved.

Definition in file variable.h.

4.31.2 Enumeration Type Documentation

4.31.2.1 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 };
```

4.31.3 Function Documentation

4.31.3.1 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.

```
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 }
```

4.31.3.2 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 }
```

4.31.3.3 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.

```
00277
       const char *label;
00278
       int error_code;
00279 #if DEBUG_VARIABLE
       fprintf (stderr, "variable_open_json: start\n");
00280
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
00332
               variable_error (variable, _("bad absolute maximum"));
                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
00352
         = jb_json_object_get_uint_with_default (object, LABEL_PRECISION,
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
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
           (algorithm == ALGORITHM_GENETIC)
       if
00380
00381
         {
            // Obtaining bits representing each variable
00382
00383
            if (json_object_get_member (object, LABEL_NBITS))
00384
00385
                variable->nbits
00386
                  = jb_json_object_get_uint (object, LABEL_NBITS, &error_code);
                if (!error_code || !variable->nbits)
00387
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:

4.31.3.4 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.

```
00112 {
        int error code;
00113
00114
00115 #if DEBUG_VARIABLE
00116
       fprintf (stderr, "variable_open_xml: start\n");
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
                variable_error (variable, _("bad minimum"));
00132
00133
               goto exit_on_error;
00134
            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
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;
00155
        if (xmlHasProp (node, (const xmlChar *) LABEL_MAXIMUM))
00156
00157
            variable->rangemax
              = jb_xml_node_get_float (node, (const xmlChar *) LABEL_MAXIMUM,
00158
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)
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
```

```
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,
00192
                                                &error_code, DEFAULT_PRECISION);
00193
        if (!error_code || variable->precision >= NPRECISIONS)
00194
            variable_error (variable, _("bad precision"));
00195
00196
            goto exit_on_error;
00197
00198
           (algorithm == ALGORITHM_SWEEP || algorithm == ALGORITHM_ORTHOGONAL)
00199
00200
            if (xmlHasProp (node, (const xmlChar *) LABEL_NSWEEPS))
00201
00202
               variable->nsweeps
                 = jb_xml_node_get_uint (node, (const xmlChar *) LABEL_NSWEEPS,
00204
                                           &error_code);
               if (!error_code || !variable->nsweeps)
00205
00206
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
00217
            fprintf (stderr, "variable_open_xml: nsweeps=%u\n", variable->nsweeps);
00218 #endif
00219
       if (algorithm == ALGORITHM_GENETIC)
00220
00221
         {
            // Obtaining bits representing each variable
00223
            if (xmlHasProp (node, (const xmlChar *) LABEL_NBITS))
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
             {
00236
               variable_error (variable, _("no bits number"));
00237
               goto exit_on_error;
00238
             }
00239
00240
       else if (nsteps)
00241
00242
00243
             = jb_xml_node_get_float (node, (const xmlChar *) LABEL_STEP,
00244
                                        &error_code);
            if (!error_code || variable->step < 0.)</pre>
00245
00246
00247
               variable_error (variable, _("bad step size"));
00248
               goto exit_on_error;
00249
00250
        }
00251
00252 #if DEBUG_VARIABLE
00253 fprintf (stderr, "variable_open_xml: end\n");
00254 #endif
00255
        return 1;
00256 exit_on_error:
00257 variable_free (variable, INPUT_TYPE_XML); 00258 #if DEBUG_VARIABLE
       fprintf (stderr, "variable_open_xml: end\n");
00259
00260 #endif
00261
       return 0;
00262 }
```

Here is the call graph for this function:

4.31.4 Variable Documentation

4.31.4.1 format

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

Array of C-strings with variable formats.

Definition at line 53 of file variable.c.

4.31.4.2 precision

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

Array of variable precisions.

Definition at line 58 of file variable.c.

4.32 variable.h

Go to the documentation of this file.

```
00001 /4
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,
00045
        ALGORITHM_SWEEP = 1,
ALGORITHM_GENETIC = 2,
00046
00047
        ALGORITHM_ORTHOGONAL = 3
00048 };
00049
00054 typedef struct
00055 {
        char *name;
00056
00057
        double rangemin;
00058
        double rangemax;
00059
        double rangeminabs;
00060
        double rangemaxabs;
```

4.32 variable.h

```
00061 double step;
00062 unsigned int precision;
00063 unsigned int nsweeps;
00064 unsigned int nsweeps;
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, 43
Input, 8	button_run
Optimize, 17	Window, 44
Algorithm	button_save
variable.h, 300	Window, 44
algorithm	button_simulator
Input, 8	Window, 44
Optimize, 17	button_template
ALGORITHM_GENETIC	Window, 44
variable.h, 300	,
ALGORITHM MONTE CARLO	calculation_time
variable.h, 300	Optimize, 17
ALGORITHM_ORTHOGONAL	check cleaner
variable.h, 300	Window, 44
ALGORITHM SWEEP	check_climbing
variable.h, 300	Window, 45
application_directory	check evaluator
Window, 40	Window, 45
Williadw, 40	check maxabs
box buttons	Window, 45
Window, 40	check minabs
button about	Window, 45
Window, 41	check_template
button_add_experiment	Window, 45
Window, 41	cleaner
button_add_variable	Input, 8
Window, 41	Optimize, 17
	•
button_algorithm Window, 41	climbing Input, 9
	•
button_cleaner Window 41	Optimize, 18
Window, 41	CLIMBING_METHOD_COORDINATES
button_climbing	input.h, 114
Window, 42	CLIMBING_METHOD_RANDOM
button_evaluator	input.h, 114
Window, 42	ClimbingMethod
button_exit	input.h, 114
Window, 42	combo_experiment
button_experiment	Window, 46
Window, 42	combo_variable
button_help	Window, 46
Window, 42	config.h, 65
button_norm	CP
Window, 43	optimize.c, 227
button_open	
Window, 43	DEBUG_EXPERIMENT
button_options	experiment.c, 68
Window, 43	DEBUG_INPUT
button_remove_experiment	input.c, 86
Window, 43	DEBUG_INTERFACE
button_remove_variable	interface.c, 121

DEBUG_MPCOTOOL	Optimize, 18
mpcotool.c, 218	Experiment, 5
DEBUG_OPTIMIZE	name, 5
optimize.c, 227	ninputs, 6
DEBUG_VARIABLE	stencil, 6
variable.c, 288	template_flags, 6
dialog	weight, 6
Options, 29	experiment
Running, 32	Input, 9
dialog_cleaner	Optimize, 18
interface.c, 121	Window, 47
dialog_cleaner_close	experiment.c, 67
interface.c, 121	DEBUG_EXPERIMENT, 68
dialog_evaluator	experiment_error, 68
interface.c, 122	· —
	experiment_free, 68
dialog_evaluator_close	experiment_new, 69
interface.c, 122	experiment_open_json, 69
dialog_name_experiment_close	experiment_open_xml, 71
interface.c, 123	stencil, 73
dialog_open_close	stencilbin, 73
interface.c, 124	experiment.h, 78
dialog_options_close	experiment_error, 79
interface.c, 125	experiment_free, 79
dialog_save_close	experiment_open_json, 80
interface.c, 125	experiment_open_xml, 82
dialog_simulator	stencil, 84
interface.c, 127	stencilbin, 84
dialog_simulator_close	experiment_error
interface.c, 127	experiment.c, 68
directory	experiment.h, 79
Input, 9	experiment_free
F	experiment.c, 68
entry_result	experiment.h, 79
Window, 46	experiment_new
entry_variable	experiment.c, 69
Window, 46	experiment_open_json
entry_variables	experiment.c, 69
Window, 46	experiment.h, 80
error best	•
Optimize, 18	experiment_open_xml
error message	experiment.c, 71
tools.c, 283	experiment.h, 82
tools.h, 286	file
ERROR NORM EUCLIDIAN	Optimize, 19
input.h, 114	file result
ERROR_NORM_MAXIMUM	Optimize, 19
input.h, 114	file variables
•	-
ERROR_NORM_P	Optimize, 19
input.h, 114	format
ERROR_NORM_TAXICAB	variable.c, 293
input.h, 114	variable.h, 305
error_old	frame_algorithm
Optimize, 18	Window, 47
ERROR_TYPE	frame_experiment
tools.h, 285	Window, 47
ErrorNorm	frame_norm
input.h, 114	Window, 47
evaluator	frame_variable
Input, 9	Window, 47

genetic_variable	threshold, 13
Optimize, 19	tolerance, 13
grid	type, 13
Options, 29	variable, 14
Running, 32	variables, 14
Window, 48	input
grid_algorithm	input.c, 100
Window, 48	input.h, 117
grid_climbing	input.c, 85
Window, 48	DEBUG_INPUT, 86
grid_experiment	input, 100
Window, 48	input_error, 87
grid_files	input_free, 87
Window, 48	input_new, 88
grid_norm	input_open, 88
Window, 49	input_open_json, 89
grid_variable	input_open_xml, 94
Window, 49	result_name, 100
id_experiment	variables_name, 100
Window, 49	input.h, 113
id_experiment_name	CLIMBING_METHOD_COORDINATES, 114
Window, 49	CLIMBING_METHOD_RANDOM, 114
id_input	ClimbingMethod, 114
Window, 49	ERROR_NORM_EUCLIDIAN, 114 ERROR_NORM_MAXIMUM, 114
id_template	ERROR_NORM_P, 114
Window, 50	ERROR_NORM_TAXICAB, 114
id variable	ErrorNorm, 114
Window, 50	input, 117
id_variable_label	input_free, 114
Window, 50	input_new, 115
INFO TYPE	input_open, 115
tools.h, 285	result name, 117
Input, 7	variables_name, 117
adaptation_ratio, 8	input error
algorithm, 8	input.c, 87
cleaner, 8	INPUT_FILE
climbing, 9	interface.c, 121
directory, 9	input_free
evaluator, 9	input.c, 87
experiment, 9	input.h, 114
mutation_ratio, 9	input_new
name, 10	input.c, 88
nbest, 10	input.h, 115
nestimates, 10	input_open
nexperiments, 10	input.c, 88
nfinal_steps, 10	input.h, 115
niterations, 11	input_open_json
norm, 11	input.c, 89
nsimulations, 11	input_open_xml
nsteps, 11	input.c, 94
nvariables, 11	input_save
p, 12 relaxation, 12	interface.c, 128
reproduction_ratio, 12	input_save_climbing_json
result, 12	interface.c, 129
seed, 12	input_save_climbing_xml
simulator, 13	interface.c, 129
template_flags, 13	input_save_json
tomplato_nago, 10	

interface.c, 130	window_update_variable, 163
input_save_xml	window_weight_experiment, 164
interface.c, 132	interface.h, 200
interface.c, 118	MAX_LENGTH, 201
DEBUG_INTERFACE, 121	window, 211
dialog_cleaner, 121	window_new, 201
dialog_cleaner_close, 121	IDW
dialog_evaluator, 122	JBW
dialog_evaluator_close, 122	main.c, 215
dialog_name_experiment_close, 123	label
dialog_open_close, 124	Optimize, 19
dialog_options_close, 125	Running, 32
dialog_save_close, 125	label_adaptation
dialog_simulator, 127	Window, 50
dialog_simulator_close, 127	label bests
INPUT_FILE, 121	_
input_save, 128	Window, 50
input_save_climbing_json, 129	label_bits
input_save_climbing_xml, 129	Window, 51
input_save_json, 130	label_climbing
input_save_xml, 132	Options, 29
logo, 164	label_estimates
options, 165	Window, 51
options_new, 135	label_experiment
running, 165	Window, 51
running_new, 135	label_final_steps
window, 165	Window, 51
window_about, 136	label_generations
window_add_experiment, 136	Window, 51
window_add_variable, 137	label_iterations
window_get_algorithm, 138	Window, 52
window_get_climbing, 138	label_max
window_get_norm, 138	Window, 52
window_help, 139	label_min
window_inputs_experiment, 139	Window, 52
window_label_variable, 140	label_mutation
window_name_experiment, 140	Window, 52
window_new, 141	label_p
window open, 151	Window, 52
window_precision_variable, 151	label_population
window_rangemax_variable, 152	Window, 53
window_rangemaxabs_variable, 152	label_precision
window_rangemin_variable, 152	Window, 53
window_rangeminabs_variable, 152	label_relaxation
window read, 153	Window, 53
window remove experiment, 154	label_reproduction
window_remove_variable, 155	Window, 53
window run, 155	label_result
window save, 156	Window, 53
window_save_climbing, 157	label_seed
window_set_algorithm, 157	Options, 29
window_set_experiment, 158	label_simulations
window set variable, 158	Window, 54
window step variable, 159	label_simulator
window_template_experiment, 160	Window, 54
window_template_experiment_close, 160	label_step
window_update, 161	Window, 54
window update climbing, 163	label_steps
0,	Window, 54

label_sweeps	nexperiments
Window, 54	Input, 10
label_threads	Optimize, 21
Options, 29	Window, 56
label_threshold	nfinal_steps
Window, 55	Input, 10
label_tolerance	Optimize, 21
Window, 55	ninputs
label_variable	Experiment, 6
Window, 55	Optimize, 21
label_variables	niterations
Window, 55	Input, 11
label_weight	Optimize, 22
Window, 55	norm
logo	Input, 11
interface.c, 164	nsaveds
Window, 56	Optimize, 22
	nsimulations
main	Input, 11
main.c, 215	Optimize, 22
main.c, 214	nstart
JBW, 215	Optimize, 22
main, 215	nstart_climbing
main_window	Optimize, 22
tools.c, 283	nsteps
tools.h, 286	Input, 11
MAX_LENGTH	Optimize, 23
interface.h, 201	nsweeps
mpcotool	Optimize, 23
mpcotool.c, 218	Variable, 34
mpcotool.h, 223	nthreads_climbing
mpcotool.c, 217	optimize.c, 254
DEBUG_MPCOTOOL, 218	optimize.h, 280
mpcotool, 218	nvariables
mpcotool.h, 222	Input, 11
mpcotool, 223	Optimize, 23
mpi_rank	Window, 56
Optimize, 20	,
mutation_ratio	Optimize, 14
Input, 9	adaptation_ratio, 17
Optimize, 20	algorithm, 17
	calculation_time, 17
name	cleaner, 17
Experiment, 5	climbing, 18
Input, 10	error_best, 18
Variable, 33	error_old, 18
nbest	evaluator, 18
Input, 10	experiment, 18
Optimize, 20	file, 19
nbits	file_result, 19
Optimize, 20	file_variables, 19
Variable, 33	genetic_variable, 19
nend	label, 19
Optimize, 20	mpi_rank, 20
nend_climbing	mutation_ratio, 20
Optimize, 21	nbest, 20
nestimates	nbits, 20
Input, 10	nend, 20
Optimize, 21	nend_climbing, 21

nestimates, 21	optimize_iterate, 236
nexperiments, 21	optimize_merge, 236
nfinal_steps, 21	optimize_merge_old, 237
ninputs, 21	optimize_MonteCarlo, 238
niterations, 22	optimize_norm, 255
nsaveds, 22	optimize_norm_euclidian, 239
nsimulations, 22	optimize_norm_maximum, 239
nstart, 22	optimize_norm_p, 240
nstart_climbing, 22	optimize_norm_taxicab, 240
nsteps, 23	optimize open, 241
nsweeps, 23	optimize_orthogonal, 245
nvariables, 23	optimize_parse, 245
p, 23	optimize_print, 247
precision, 23	optimize_refine, 248
rangemax, 24	optimize_save_old, 249
rangemaxabs, 24	optimize_save_optimal, 249
rangemin, 24	optimize_save_variables, 250
rangeminabs, 24	optimize sequential, 250
relaxation, 24	optimize step, 251
reproduction ratio, 25	optimize_step_climbing, 251
· —	optimize_sweep, 252
result, 25	
rng, 25	optimize_synchronise, 253
seed, 25	optimize_thread, 254
simulation_best, 25	RM, 227
simulator, 26	optimize.h, 275
step, 26	nthreads_climbing, 280
stop, 26	optimize, 280
template_flags, 26	optimize_free, 276
thread, 26	optimize_open, 276
thread_climbing, 27	optimize_algorithm
threshold, 27	optimize.c, 255
tolerance, 27	optimize_best
value, 27	optimize.c, 228
value_old, 27	optimize_best_climbing
variables, 28	optimize.c, 228
weight, 28	optimize_climbing
optimize	optimize.c, 229
optimize.c, 255	optimize_climbing_best
optimize.h, 280	optimize.c, 230
optimize.c, 225	optimize_climbing_sequential
CP, 227	optimize.c, 230
DEBUG_OPTIMIZE, 227	optimize_climbing_thread
nthreads_climbing, 254	optimize.c, 231
optimize, 255	optimize_estimate_climbing
optimize_algorithm, 255	optimize.c, 255
optimize_best, 228	optimize_estimate_climbing_coordinates
optimize_best_climbing, 228	optimize.c, 232
optimize climbing, 229	optimize_estimate_climbing_random
optimize_climbing_best, 230	optimize.c, 232
optimize_climbing_sequential, 230	optimize_free
optimize_climbing_thread, 231	optimize.c, 233
optimize_estimate_climbing, 255	optimize.h, 276
optimize_estimate_climbing_coordinates, 232	optimize_genetic
optimize_estimate_climbing_random, 232	optimize.c, 233
optimize_free, 233	optimize_genetic_objective
optimize_genetic, 233	optimize_genetic_objective
optimize_genetic_objective, 234	optimize_input
optimize_input, 235	optimize_riput
spsps.() = 00	

optimize_iterate	options_new
optimize.c, 236	interface.c, 135
optimize_merge	
optimize.c, 236	р
optimize_merge_old	Input, 12
optimize.c, 237	Optimize, 23
optimize_MonteCarlo	ParallelData, 30
optimize.c, 238	thread, 31
optimize_norm	precision
optimize.c, 255	Optimize, 23
optimize_norm_euclidian	Variable, 34
optimize.c, 239	variable.c, 294
optimize_norm_maximum	variable.h, 306
optimize.c, 239	rangemax
optimize_norm_p	Optimize, 24
optimize.c, 240	Variable, 34
optimize_norm_taxicab	rangemaxabs
optimize.c, 240	Optimize, 24
optimize_open	Variable, 34
optimize.c, 241	rangemin
optimize.h, 276	Optimize, 24
optimize_orthogonal	Variable, 34
optimize.c, 245	rangeminabs
optimize_parse	Optimize, 24
optimize.c, 245	Variable, 35
optimize_print	relaxation
optimize.c, 247	Input, 12
optimize_refine	Optimize, 24
optimize.c, 248	reproduction_ratio
optimize_save_old	Input, 12
optimize.c, 249	Optimize, 25
optimize_save_optimal	result
optimize.c, 249	Input, 12
optimize_save_variables	Optimize, 25
optimize.c, 250	result name
optimize_sequential	input.c, 100
optimize.c, 250	input.h, 117
optimize_step	RM
optimize.c, 251	optimize.c, 227
optimize_step_climbing	rng
optimize.c, 251	Optimize, 25
optimize_sweep	Running, 31
optimize.c, 252	dialog, 32
optimize_synchronise	grid, 32
optimize.c, 253	label, 32
optimize_thread	spinner, 32
optimize.c, 254	running
Options, 28	interface.c, 165
dialog, 29	running_new
grid, 29	interface.c, 135
label_climbing, 29	111011400.0, 100
label_seed, 29	scrolled_max
label_threads, 29	Window, 56
spin_climbing, 30	scrolled maxabs
spin_seed, 30	Window, 56
spin_threads, 30	scrolled_min
options	Window, 57
interface.c, 165	scrolled_minabs

Window, 57	spin_step
scrolled_p	Window, 61
Window, 57	spin_steps
scrolled_step	Window, 61
Window, 57	spin_sweeps
scrolled_threshold	Window, 62
Window, 57	spin_threads
seed	Options, 30
Input, 12	spin threshold
Optimize, 25	Window, 62
show_pending	spin_tolerance
tools.c, 283	Window, 62
tools.h, 286	spin_weight
simulation best	Window, 62
-	spinner
Optimize, 25	•
simulator	Running, 32
Input, 13	stencil
Optimize, 26	Experiment, 6
spin_adaptation	experiment.c, 73
Window, 58	experiment.h, 84
spin_bests	stencilbin
Window, 58	experiment.c, 73
spin_bits	experiment.h, 84
Window, 58	step
spin_climbing	Optimize, 26
Options, 30	Variable, 35
spin_estimates	stop
Window, 58	Optimize, 26
spin_final_steps	,
Window, 58	template_flags
spin_generations	Experiment, 6
Window, 59	Input, 13
spin iterations	Optimize, 26
Window, 59	thread
	Optimize, 26
spin_max	ParallelData, 31
Window, 59	thread_climbing
spin_maxabs	Optimize, 27
Window, 59	threshold
spin_min	
Window, 59	Input, 13
spin_minabs	Optimize, 27
Window, 60	tolerance
spin_mutation	Input, 13
Window, 60	Optimize, 27
spin_p	tools.c, 282
Window, 60	error_message, 283
spin_population	main_window, 283
Window, 60	show_pending, 283
spin_precision	tools.h, 284
Window, 60	error_message, 286
spin_relaxation	ERROR_TYPE, 285
Window, 61	INFO TYPE, 285
spin_reproduction	main_window, 286
Window, 61	show_pending, 286
	,,,,
spin_seed	type
	type
Options, 30	type Input, 13
spin_simulations	Input, 13

value_old	box_buttons, 40
Optimize, 27	button_about, 41
Variable, 33	button_add_experiment, 41
name, <mark>33</mark>	button_add_variable, 41
nbits, 33	button_algorithm, 41
nsweeps, 34	button_cleaner, 41
precision, 34	button_climbing, 42
rangemax, 34	button_evaluator, 42
rangemaxabs, 34	button_exit, 42
rangemin, 34	button_experiment, 42
rangeminabs, 35	button_help, 42
step, 35	button_norm, 43
variable	button_open, 43
Input, 14	button_options, 43
Window, 62	button_remove_experiment, 43
variable.c, 287	button_remove_variable, 43
DEBUG_VARIABLE, 288	button_run, 44
format, 293	button_save, 44
precision, 294	button_simulator, 44
variable error, 288	button template, 44
variable free, 289	check cleaner, 44
variable_open_json, 289	check_climbing, 45
variable_open_xml, 291	check_evaluator, 45
variable.h, 299	check maxabs, 45
Algorithm, 300	check minabs, 45
ALGORITHM GENETIC, 300	check template, 45
ALGORITHM MONTE CARLO, 300	combo_experiment, 46
ALGORITHM ORTHOGONAL, 300	combo_variable, 46
ALGORITHM_SWEEP, 300	entry result, 46
format, 305	entry variable, 46
precision, 306	entry_variables, 46
variable_error, 300	experiment, 47
variable free, 301	frame_algorithm, 47
variable_open_json, 301	frame experiment, 47
variable open xml, 303	frame norm, 47
variable error	frame variable, 47
variable.c, 288	grid, 48
variable.h, 300	grid_algorithm, 48
variable free	grid_climbing, 48
variable.c, 289	grid experiment, 48
variable.h, 301	grid_files, 48
variable_open_json	grid_norm, 49
variable.c, 289	grid_variable, 49
variable.h, 301	id_experiment, 49
variable_open_xml	id_experiment_name, 49
variable.c, 291	id_input, 49
variable.h, 303	id_template, 50
variables	id variable, 50
Input, 14	id_variable_label, 50
Optimize, 28	label_adaptation, 50
variables_name	label_bests, 50
input.c, 100	label_bits, 51
input.h, 117	label_estimates, 51
	label_experiment, 51
weight	label_final_steps, 51
Experiment, 6	label_generations, 51
Optimize, 28	label_iterations, 52
Window, 35	label_max, 52
application_directory, 40	asoi_max, or

label_min, 52	window_about
label_mutation, 52	interface.c, 136
label_p, 52	window_add_experiment
label_population, 53	interface.c, 136
label_precision, 53	window_add_variable
label_relaxation, 53	interface.c, 137
label_reproduction, 53	window_get_algorithm
label_result, 53	interface.c, 138
label_simulations, 54	window_get_climbing
label_simulator, 54	interface.c, 138
label_step, 54	window_get_norm
label_steps, 54	interface.c, 138
label_sweeps, 54	window_help
label_threshold, 55	interface.c, 139
label_tolerance, 55	window_inputs_experiment
label_variable, 55	interface.c, 139
label_variables, 55	window_label_variable
label weight, 55	interface.c, 140
logo, 56	window_name_experiment
nexperiments, 56	interface.c, 140
nvariables, 56	window new
scrolled_max, 56	interface.c, 141
scrolled maxabs, 56	interface.h, 201
scrolled min, 57	window open
scrolled minabs, 57	interface.c, 151
scrolled p, 57	window_precision_variable
scrolled step, 57	interface.c, 151
scrolled threshold, 57	window_rangemax_variable
spin_adaptation, 58	interface.c, 152
spin bests, 58	window_rangemaxabs_variable
spin_bits, 58	interface.c, 152
spin_estimates, 58	window_rangemin_variable
spin_final_steps, 58	interface.c, 152
spin_generations, 59	window_rangeminabs_variable
spin_iterations, 59	interface.c, 152
spin_max, 59	window_read
spin_maxabs, 59	interface.c, 153
spin_min, 59	window_remove_experiment
spin_minabs, 60	interface.c, 154
spin_mutation, 60	window_remove_variable
spin_p, 60	interface.c, 155
spin_population, 60	window_run
spin_precision, 60	interface.c, 155
spin_relaxation, 61	window_save
spin_reproduction, 61	interface.c, 156
spin_simulations, 61	window_save_climbing
spin_step, 61	interface.c, 157
spin_steps, 61	window_set_algorithm
spin_sweeps, 62	interface.c, 157
spin_threshold, 62	window_set_experiment
spin_tolerance, 62	interface.c, 158
spin_weight, 62	window_set_variable
variable, 62	interface.c, 158
window, 63	window_step_variable
window	interface.c, 159
interface.c, 165	window_template_experiment
interface.h, 211	interface.c, 160
Window, 63	window_template_experiment_close

interface.c, 160
window_update
 interface.c, 161
window_update_climbing
 interface.c, 163
window_update_variable
 interface.c, 163
window_weight_experiment
 interface.c, 164