# ROPI

A Robust Optimization Programming Interface
Version 0.1.0

Written by Marc Goerigk
Institute for Numerical and Applied Mathematics
University of Göttingen, Germany
m.goerigk@math.uni-goettingen.de

Documentation generated by Doxygen 1.6.3

Tue Jun 19 10:57:17 2012

# Contents

# Chapter 1

# Class Index

## 1.1 Class List

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

# Chapter 2

# File Index

## 2.1 File List

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

# Chapter 3

# Class Documentation

## 3.1 ROPI::Con Struct Reference

Constraint data structure.

`#include <lp.h>`

### Public Member Functions

- bool operator< (const Con &other) const

  *Relational operator.*

- Con ()

  *Default constructor.*

### Public Attributes

- double rhs

  *Right-hand-side of constraint. Default is 0.*

- CON_SENSE sen

  *Constraint sense (<=, =, >=). Default is equality.*

- int id

  *Unique identificator number. ids must start from 0 and must be numbered consecutively. Default is -1.*

- std::string name

  *Constraint name. Default is "DEFAULTNAME".*

- std::list< std::pair< int, double > > coeffs

  *Variable coefficients of constraint. The pairs consist of variable id and coefficient.*

## 3.2 ROPI::Error Class Reference

Error class thrown by ROPI.

`#include <error.h>`

### Public Member Functions

- Error (std::string _mesg)

  *Constructor with error message.*

- std::string get_mesg ()

  *Returns the error message.*

## 3.3 ROPI::FiniteScenario Struct Reference

A single scenario.

`#include <robustness.h>`

### Public Attributes

- std::map< std::pair< Con, int >, double > coeffs

  *Map containing all coefficients that differ from the nominal case. Con is the affected constraint, int the variable id.*

- std::map< Con, double > rhs

  *Map containing all right-hand-sides that differ from the nominal case.*

## 3.4 ROPI::FiniteUncertainty Struct Reference

A finite uncertainty set.

`#include <robustness.h>`

### Public Attributes

- std::vector< FiniteScenario > scenarios

  *Finite uncertainty consists of a vector of scenarios.*

## 3.5 ROPI::LP Class Reference

Class for (mixed integer) linear programs.

`#include <lp.h>`

### Public Member Functions

- LP ()

  *Default constructor.*

- ~LP ()

  *Default destuctor.*

- void add_var (Var _var)

  *Adds a variable to the problem. Note: a copy of the variable will be created. Ownership of the argument is preserved.*

- void add_con (Con _con)

  *Adds a constraint to the problem. Note: a copy of the constraint will be created. Ownership of the argument is preserved.*

- void set_name (std::string _name)

  *Set the problem name. Default is "DEFAULTNAME".*

- void set_sense (OBJ_SENSE _sen)

  *Set the problem direction (MIN,MAX). Default is MIN.*

- std::string get_name ()

  *Returns the problem name.*

- OBJ_SENSE get_sense ()

  *Returns the problem direction.*

- std::list< Var > * get_vars ()

  *Returns a pointer to the list of variables.*

- std::list< Con > * get_cons ()

  *Returns a pointer to the list of constraints.*

- int get_num_vars ()

  *Returns the number of variables.*

- int get_num_cons ()

  *Returns the number of constraints.*

- LPTYPE get_lptype ()

  *Returns the type of the problem (LP,MIP).*

- void read_mpsfile (std::string file)

*Read a problem from an mps file (fixed or free).*

- void write_plain (std::ostream &out)

  *Write a human-friendly version of the problem to ostream (e.g., cout).*

## 3.6 ROPI::Robustness Class Reference

Robustness class.

`#include <robustness.h>`

### Public Member Functions

- Robustness (LP *_lp, Robustness_Method method)

  *Default constructor. LP ownership is preserved.*

- void set_uncertainty (FiniteUncertainty _unc)

  *Set the uncertainty.*

- void set_options (RobustnessOptions _options)

  *Set robustness options.*

- LP generate_robust ()

  *Generate the robust counterpart.*

- void exclude_variable (int id)

  *RecFeas specific: Exclude variable from objective function.*

## 3.7 ROPI::RobustnessOptions Struct Reference

Options for creating the robust counterpart.

`#include <robustness.h>`

### Public Member Functions

- RobustnessOptions ()

  *Default constructor.*

### Public Attributes

- Norm norm

  *Recovery norm. Default is NORM_UNDEFINED. Relevant for RecOpt and RecFeas.*

- RecoveryObjective recobj

  *Recovery objective. Default is REC_UNDEFINED. Relevant for RecOpt and RecFeas.*

- SOLVERTYPE solvertype

  *The solver used for preprocessing. Default is SOL_UNDEFINED. Relevant for Light Robustness and RecOpt.*

- Recopt_Model recopt_model

  *The RecOpt model applied. Default is RECOPT_UNDEFINED.*

- bool nominal_feasibility

  *Flag that determines if recovery robust solutions need to be feasible for the nominal scenario. Default is true. Relevant for RecOpt and RecFeas.*

- double budget

  *Determines the budget for light robustness (budget) $*$ (nominal objective value) is the worst objective for light robust solution. Default is 1.5.*

## 3.8 ROPI::Solver Class Reference

Solver class.

`#include <solver.h>`

### Public Member Functions

- Solver ()

  *Default constructor.*

- ~Solver ()

  *Default destructor.*

- void init (LP *_lp, SOLVERTYPE _type)

  *Initialize the solver-specific program. LP ownership is preserved.*

- void solve ()

  *Solve the problem.*

- void write_solution (std::ostream &out, double eps)

  *Write solution in human-friendly format to stream. Solution values with $|x| <= eps$ are not shown.*

- void set_options (SolverOptions _options)

  *Set solver options.*

- SOLVERSTATUS get_status ()

  *Returns the current solver status.*

- double get_objective ()

  *Returns the objective value of the current solution.*

- std::vector< double > get_solution ()

  *Returns the current solution.*

### Static Public Member Functions

- static std::list< SOLVERTYPE > get_available ()

  *Returns a list of currently available solvers. Note that solver availability is determined during compilation .*

## 3.9 ROPI::SolverOptions Struct Reference

Solver options data structure.

`#include <solver.h>`

### Public Member Functions

- SolverOptions ()

  *Default constructor.*

### Public Attributes

- bool verbose

  *Flag to turn solver output on/off. Default is true.*

- int timelimit

  *Solver timelimit. 0 means no limit. Default is 0.*

- int threads

  *Number of threads the solver may use. 0 means automatic. Default is 0.*

- int mipfocus

  *Solver focus. 0 means automatic. 1 means improve current solution. 2 means prove optimality. 3 means improve lower bound. Default is 0. Warning: Currently only available for Gurobi.*

- double heuristics

  *Heuristics parameter. Determines the percentage of time for heuristics -1 uses the solver presetting. Default is -1. Warning: Currently only available for Gurobi.*

## 3.10 ROPI::Var Struct Reference

Variable data structure.

`#include <lp.h>`

### Public Member Functions

- bool operator< (const Var &other) const

  *Relational operator.*

- bool operator== (const Var &other) const

  *Equality check.*

- Var ()

  *Default constructor.*

### Public Attributes

- double obj

  *Objective coefficient. Default is 0.*

- int id

  *Unique identificator number. ids must start from 0 and must be numbered consecutively. Default is -1.*

- std::string name

  *Variable name. Default is "DEFAULTNAME".*

- double lb

  *Variable lower bound. Default is 0.*

- double ub

  *Variable upper bound. Default is ROPIPINF.*

- VAR_TYPE type

  *Variable type (continuous, integer, binary). Default is continuous.*

- double solution

  *The value of the variable in the current solution. Default is 0.*

- bool start_given

  *Flag if a start solution provided. Default is 0.*

- double start

  *Variable value in start solution. Default is 0.*

# Chapter 4

# File Documentation

## 4.1 include/error.h File Reference

Header for error class.

```
#include <string>
```

### Classes

- class ROPI::Error

  *Error class thrown by ROPI.*

### 4.1.1 Detailed Description

## 4.2 include/lp.h File Reference

Header for LP class.

```
#include <string>
#include <list>
#include <ostream>
#include <map>
#include <limits>
```

### Classes

- struct ROPI::Var
  *Variable data structure.*
- struct ROPI::Con
  *Constraint data structure.*
- class ROPI::LP
  *Class for (mixed integer) linear programs.*

### Defines

- #define ROPIPINF std::numeric_limits<double>::max( )
  *Definition of ROPI positive infinity.*
- #define ROPIMINF std::numeric_limits<double>::min( )
  *Definition of ROPI negative infinity.*

### Enumerations

- enum **VAR_TYPE** { ROPI::CONTINUOUS = 1, ROPI::INTEGER = 2, ROPI::BINARY = 3 }
  *Variable type enumerator.*
- enum **LPTYPE** { ROPI::LPTYPE_LP = 1, ROPI::LPTYPE_MIP = 2 }
  *Type of program enumerator.*
- enum **CON_SENSE** { ROPI::LEQ = 1, ROPI::EQ = 2, ROPI::GEQ = 3 }
  *Constraint sense enumerator.*
- enum **OBJ_SENSE** { ROPI::MAX = 1, ROPI::MIN = 2 }
  *Objective direction.*

#### 4.2.1 Detailed Description

## 4.3 include/robustness.h File Reference

Header for robustness class.

```
#include "lp.h"
#include <string>
#include <list>
#include <ostream>
#include <map>
#include <limits>
#include <vector>
```

### Classes

- struct ROPI::FiniteScenario

  *A single scenario.*

- struct ROPI::FiniteUncertainty

  *A finite uncertainty set.*

- struct ROPI::RobustnessOptions

  *Options for creating the robust counterpart.*

- class ROPI::Robustness

  *Robustness class.*

### Enumerations

- enum **Robustness_Method** {

  ROPI::ROB_STRICT = 1, ROPI::ROB_LIGHT = 2, ROPI::ROB_RECOPT = 3, ROPI::ROB_-RECFEAS = 4,

  ROPI::ROB_UNDEFINED = 5 }

  *Robustness model enumerator.*

- enum **Uncertainty_Type** { ROPI::UNC_FINITE = 1, ROPI::UNC_UNDEFINED = 2 }

  *Uncertainty type enumerator.*

- enum **Norm** { ROPI::NORM_L1 = 1, ROPI::NORM_LINFTY = 2, ROPI::NORM_UNDEFINED = 3 }

  *Norm enumerator.*

- enum **RecoveryObjective** { ROPI::REC_MEDIAN = 1, ROPI::REC_CENTER = 2, ROPI::REC_-UNDEFINED = 3 }

  *Recovery objective enumerator.*

- enum **Recopt_Model** { ROPI::RECOPT_SIMPLE = 1, ROPI::RECOPT_EXTENDED = 2, ROPI::RECOPT_UNDEFINED = 3 }

  *Recovery to optimality model enumerator.*

### 4.3.1 Detailed Description

## 4.4 include/solver.h File Reference

Header for solver class.

```
#include <list>
#include <ostream>
#include <vector>
```

### Classes

- struct ROPI::SolverOptions

  *Solver options data structure.*

- class ROPI::Solver

  *Solver class.*

### Enumerations

- enum **SOLVERTYPE** { ROPI::SOL_GUROBI = 1, ROPI::SOL_CPLEX = 2, ROPI::SOL_XPRESS = 3, ROPI::SOL_UNDEFINED = 4 }

  *Type of solver enumerator.*

- enum **SOLVERSTATUS** {

  ROPI::SOL_EMPTY = 1, ROPI::SOL_LOADED = 2, ROPI::SOL_OPTIMAL = 3, ROPI::SOL_-INFEASIBLE = 4,

  ROPI::SOL_UNBOUNDED = 5, ROPI::SOL_TIMELIMIT = 6, ROPI::SOL_UNKNOWN = 7 }

  *Solver status enumerator.*

### 4.4.1 Detailed Description

# Index