ROPI

A Robust Optimization Programming Interface

Version 0.1.0

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

June 19, 2012

Introduction

ROPI is a freely available C++ library that facilitates the applicability of robust optimization models. It can be downloaded from

http://num.math.uni-goettingen.de/~m.goerigk/ropi

Structure

ROPI is a C++ library providing two main features:

- A user-friendly MIP class that allows automatic transformation to solver-specific MIP classes. Using this feature, MIPs can be generically written, and solved with whatever solver is currently available. ROPI currently supports transformation to Cplex-, Gurobi-, and Xpress-MIPs. Support for further solvers can be easily added.

- An automatic transformation from a given nominal MIP to a robust MIP. Currently implemented is Strict Robustness, Light Robustness, RecFeas and RecOpt.

In Figure 1, the general structure of ROPI is presented.

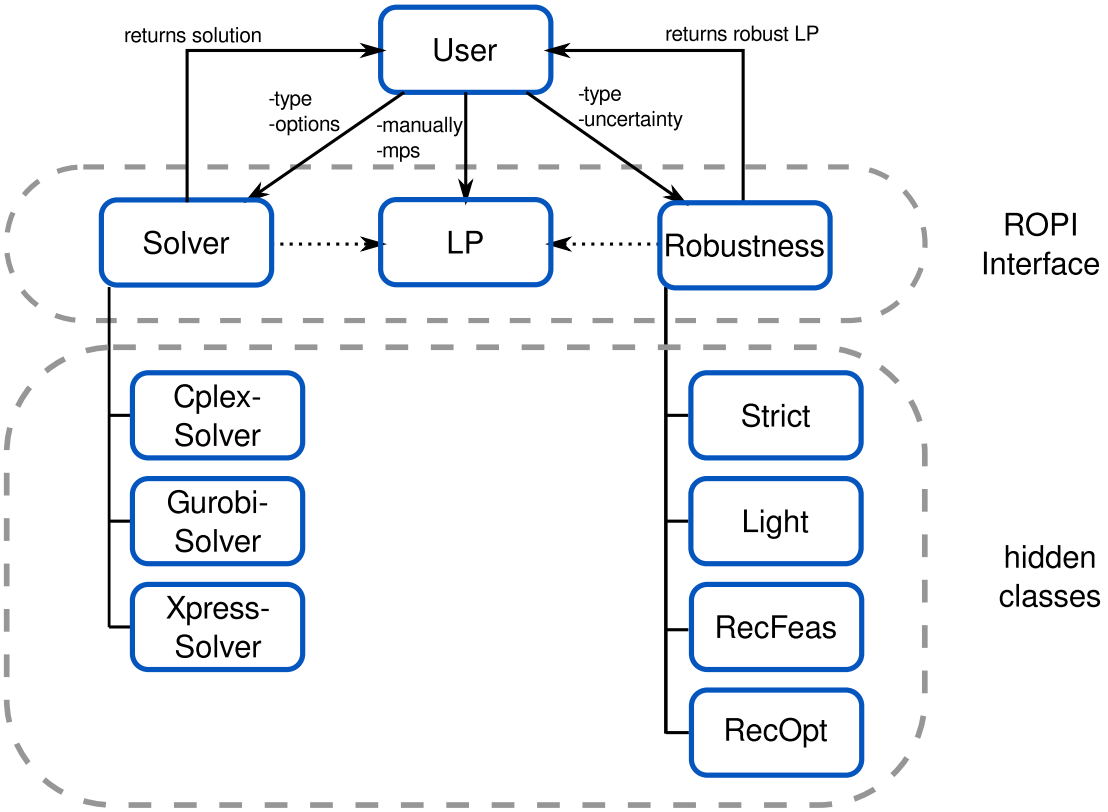

Figure 1: Structure of ROPI

Visible to the library user are three classes: "Solver", "LP", and "Robustness". Each hides its implementation and specified type within hidden classes, using the *opaque pointer* technique (also known as *pointer to implementation idiom*). This creates the possibility for the user to specify which type of solver or robustness he would like to use during runtime.

The currently implemented type of uncertainty is a finite set of scenarios $\mathcal{U}^f = \{\xi^1, \ldots, \xi^n\}$, with further types coming in the next library version. For RecOpt and RecFeas, types of distance measures are the l_1 distance $\sum_{i=1}^{n} |x_i - y_i|$, and the l_∞ distance $\max_{i=1}^{n} |x_i - y_i|$, while the recovery objective can either be the median or the center.

Robustness Transformations

We assume an uncertain optimization problem of the form

$$\begin{aligned} P(A,b) \qquad \min\ & c^t x \\ \text{s.t.}\quad & A_1 x \leq b_1 \\ & A_2 x = b_2 \\ & A_3 x \geq b_3 \\ & l \leq x \leq u \\ & x \in \mathcal{X} \end{aligned}$$

with

$$A = \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

to be given, with a nominal scenario (A^0, b^0). Let the uncertainty set be $\mathcal{U}^f = \{(A^1, b^1), \ldots, (A^N, b^N)\} \subseteq \mathbb{R}^{m \times n} \times \mathbb{R}^m$. We now show the different robust problem counterparts ROPI can produce. The original MIP size is n variables, and m constraints (plus variables bounds). We denote by $Feas(x, A, b)$ the constraints $A_1 x \leq b_1$, $A_2 x = b_2$, and $A_3 x \geq b_3$.

Strict Robustness The concept of strict robustness has been introduced in [BTN98]. The uncertain optimization problem $(P(A,b), (A,b) \in \mathcal{U}^f)$ is transformed to

$$\begin{aligned} \min\ & c^t x \\ \text{s.t.}\quad & Feas(x, A^j, b^j)\ \forall j = 0, \ldots, N \\ & l \leq x \leq u \end{aligned}$$

$$x \in \mathcal{X}$$

The resulting MIP still consists of n variables, but with $N(m+1)$ constraints.

Light Robustness The concept of light robustness has been introduced in [FM09]. In a preprocessing step, the

Solve $P(\tilde{A}, \tilde{b})$, and let $c^t x^*$ be the resulting optimal objective. Transform the uncertain optimization problem $(P(\xi), \xi \in \mathcal{U}^f)$ to

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m} \gamma_i \\
\text{s.t.} \quad & c^t x \leq (1+\rho) c^t x^* \\
& Feas(x, A^0, b^0) \\
& A_1^i x \leq b_1^i + \gamma^1 \ \forall i = 1, \ldots, N \\
& A_2^i x \leq b_2^i + \gamma^2 \ \forall i = 1, \ldots, N \\
& A_2^i x \geq b_2^i - \gamma^2 \ \forall i = 1, \ldots, N \\
& A_3^i x \geq b_3^i - \gamma^3 \ \forall i = 1, \ldots, N \\
& l \leq x \leq u \\
& x \in \mathcal{X}
\end{aligned}$$

where $\gamma = (\gamma_1^t, \gamma_2^t, \gamma_3^t)^t$. The presolving step thus consists of solving a MIP with n variables and m constraints, and the resulting robust counterpart has $n+m$ and $(N+1)m + k$ constraints, where k is the number of equality constraints in the uncertain optimiation problem.

RecOpt, Simple Version The concept of recovery to optimality has been introduced in [GS11]. Solve $P(A^i, b^i)$ for all $i = 0, \ldots, N$. Let x^i be an optimal solution to problem $P(\xi^i)$. Depending on the recovery distance and the recovery objective, return the problem

	l_1	l_∞
median	$\min \sum_{j=1}^{N} \sum_{i=1}^{n} y_{ij}$ s.t. $-y_{ij} \leq x_i - x_i^j \leq y_{ij}$ $\forall i = 1, \ldots, n, \ j = 0, \ldots, N$ $Feas(x, A^0, b^0)$ $l \leq x \leq u$ $x \in \mathcal{X}, y \in \mathbb{R}_{\geq 0}^{n \times N}$	$\min \sum_{j=1}^{N} y_j$ s.t. $-y_j \leq x_i - x_i^j \leq y_j$ $\forall i = 1, \ldots, n, \ j = 0, \ldots, N$ $Feas(x, A^0, b^0)$ $l \leq x \leq u$ $x \in \mathcal{X}, y \in \mathbb{R}_{\geq 0}^{N}$

	l_1	l_∞
center	$\min\ z$ s.t. $z \geq \sum_{i=1}^{n} y_{ij}$ $\forall j = 1, \dots, N$ $-y_{ij} \leq x_i - x_i^j \leq y_{ij}$ $\forall i = 1, \dots, n,\ j = 0, \dots, N$ $Feas(x, A^0, b^0)$ $l \leq x \leq u$ $x \in \mathcal{X}, y \in \mathbb{R}^{n \times N}_{\geq 0}, z \in \mathbb{R}$	$\min\ z$ s.t. $z \geq y_j$ $\forall j = 1, \dots, N$ $-y_j \leq x_i - x_i^j \leq y_j$ $\forall i = 1, \dots, n,\ j = 0, \dots, N$ $Feas(x, A^0, b^0)$ $l \leq x \leq u$ $x \in \mathcal{X}, y \in \mathbb{R}^N_{\geq 0}, z \in \mathbb{R}$

The presolving step thus consists of $(N+1)$ problems with n variables and m constraints each. Concerning the resulting MIP size, we have

	#variables	#constraints
median, l_1	$n(N+1)$	$nN + m$
median, l_∞	$n + N$	$nN + m$
center, l_1	$n(N+1) + 1$	$N(n+1) + m$
center, l_∞	$n + N + 1$	$N(n+1) + m$

The constraints $Feas(x, A^0, b^0)$ ensuring feasibility in the nominal scenario can be left out if not desired, as does $x \in \mathcal{X}$ and the variable bounds $l \leq x \leq u$, especially for combinable problems.

RecOpt, Extended Version In the extended version, we do not precompute the optimal solutions to each of the scenarios, as they might be ambiguous. Instead, we only use the respective optimal objective value to find a recovery robust solution that can be recovered to *any* optimal solution for every scenario.

Solve $P(A^i, b^i)$ for all $i = 0, \dots, N$, where $i = 0$ denotes the nominal problem again, and let $f^*(A^i, b^i)$ be the optimal objective value of problem $P(A^i, b^i)$. Depending on the recovery distance and objective, return the problem

	l_1	l_∞
median	$\min \sum_{j=1}^{N}\sum_{i=1}^{n} y_{ij}$ s.t. $-y_{ij} \le x_i - x_i^j \le y_{ij}$ $\forall i = 1,\dots,n,\ j = 0,\dots,N$ $Feas(x, A^0, b^0)$ $Feas(x^j, A^j, b^j)\ \forall j = 0,\dots,N$ $c^t x^j = f^*(A^j, b^j)\ \forall j = 0,\dots,N$ $x \in \mathcal{X}, y \in \mathbb{R}^{n\times N}$ $x^j \in \mathcal{X}\ \forall j = 0,\dots,N$	$\min \sum_{j=1}^{N} y_j$ s.t. $-y_j \le x_i - x_i^j \le y_j$ $\forall i = 1,\dots,n,\ j = 1,\dots,N$ $Feas(x, A^0, b^0)$ $Feas(x^j, A^j, b^j)\ \forall j = 0,\dots,N$ $c^t x^j = f^*(A^j, b^j)\ \forall j = 0,\dots,N$ $x \in \mathcal{X}, y \in \mathbb{R}^{N}$ $x^j \in \mathcal{X}\ \forall j = 1,\dots,N$

	l_1	l_∞
center	$\min z$ s.t. $z \ge \sum_{i=1}^{n} y_{ij}$ $\forall j = 1,\dots,N$ $-y_{ij} \le x_i - x_i^j \le y_{ij}$ $\forall i = 1,\dots,n,\ j = 1,\dots,N$ $Feas(x, A^0, b^0)$ $Feas(x^j, A^j, b^j)\ \forall j = 0,\dots,N$ $c^t x^j = f^*(A^j, b^j)\ \forall j = 0,\dots,N$ $x \in \mathcal{X}, y \in \mathbb{R}^{n\times N}, z \in \mathbb{R}$ $x^j \in \mathcal{X}\ \forall j = 1,\dots,N$	$\min z$ s.t. $z \ge y_j$ $\forall j = 1,\dots,N$ $-y_j \le x_i - x_i^j \le y_j$ $\forall i = 1,\dots,n,\ j = 1,\dots,N$ $Feas(x, A^0, b^0)$ $Feas(x^j, A^j, b^j)\ \forall j = 0,\dots,N$ $c^t x^j = f^*(A^j, b^j)\ \forall j = 0,\dots,N$ $x \in \mathcal{X}, y \in \mathbb{R}^{N}, z \in \mathbb{R}$ $x^j \in \mathcal{X}\ \forall j = 1,\dots,N$

Presolve: N problems with n variables and m constraints. New MIP size: depending on whether median or center and on l, we have

	#variables	#constraints
median, l_1	$n(2N+1)$	$N(n+m+1)+m$
median, l_∞	$n+N+nN$	$N(n+m+1)+m$
center, l_1	$n(2N+1)+1$	$N(n+m+2)+m$
center, l_∞	$n+N+nN+1$	$N(n+m+2)+m$

RecFeas The concept of recovery to feasibility has been introduced in [CGKS11]. Return the problem

	l_1	l_∞
median	$\min \sum_{j=1}^{N}\sum_{i=1}^{n} y_{ij}$ s.t. $-y_{ij} \le x_i - x_i^j \le y_{ij}$ $\forall i = 1,\dots,n,\ j = 1,\dots,N$ $Feas(x^j, A^j, b^j)\ \forall j = 0,\dots,N$ $\forall j = 1,\dots,N$ $x \in \mathcal{X}, y \in \mathbb{R}^{n\times N}$ $x^j \in \mathcal{X}\ \forall j = 1,\dots,N$	$\min \sum_{j=1}^{N} y_j$ s.t. $-y_j \le x_i - x_i^j \le y_j$ $\forall i = 1,\dots,n,\ j = 1,\dots,N$ $Feas(x, A^0, b^0)$ $Feas(x^j, A^j, b^j)\ \forall j = 0,\dots,N$ $x \in \mathcal{X}, y \in \mathbb{R}^{N}$ $x^j \in \mathcal{X}\ \forall j = 1,\dots,N$
	l_1	l_∞
center	$\min z$ s.t. $z \ge \sum_{i=1}^{n} y_{ij}$ $\forall j = 1,\dots,N$ $-y_{ij} \le x_i - x_i^j \le y_{ij}$ $\forall i = 1,\dots,n,\ j = 1,\dots,N$ $Feas(x, A^0, b^0)$ $Feas(x^j, A^j, b^j)\ \forall j = 0,\dots,N$ $x \in \mathcal{X}, y \in \mathbb{R}^{n\times N}, z \in \mathbb{R}$ $x^j \in \mathcal{X}\ \forall j = 1,\dots,N$	$\min z$ s.t. $z \ge y_j$ $\forall j = 1,\dots,N$ $-y_j \le x_i - x_i^j \le y_j$ $\forall i = 1,\dots,n,\ j = 1,\dots,N$ $Feas(x, A^0, b^0)$ $Feas(x^j, A^j, b^j)\ \forall j = 0,\dots,N$ $x \in \mathcal{X}, y \in \mathbb{R}^{N}, z \in \mathbb{R}$ $x^j \in \mathcal{X}\ \forall j = 1,\dots,N$

No presolve necessary. New MIP size: depending on whether median or center and on l, we have

	#variables	#constraints
median, l_1	$n(2N+1)$	$N(n+m)+m$
median, l_∞	$n+N+nN$	$N(n+m)+m$
center, l_1	$n(2N+1)+1$	$N(n+m+1)+m$
center, l_∞	$n+N+nN+1$	$N(n+m+1)+m$

As it is the case for RecOpt, the user can choose whether to include nominal feasibility and variable constraints, or not.

Example Applications

In this section we present some basic ROPI functionalities on an example problem. Consider the following linear program given in fixed MPS format (taken from http://lpsolve.sourceforge.net/5.0/mps-format.htm):

```
NAME          TESTPROB
ROWS
 N  COST
 L  LIM1
 G  LIM2
 E  MYEQN
COLUMNS
    X         COST                 1   LIM1                 1
    X         LIM2                 1
    Y         COST                 4   LIM1                 1
    Y         MYEQN               -1
    Z         COST                 9   LIM2                 1
    Z         MYEQN                1
RHS
    RHS1      LIM1                 5   LIM2                10
    RHS1      MYEQN                7
BOUNDS
 UP BND1      X                    4
 LO BND1      Y                   -1
 UP BND1      Y                    1
ENDATA
```

In ROPI, it is possible to read in both the fixed and free MPS format. We can read it in using simply

```
LP lp;
lp.read_mpsfile("file.mps");
```

to get the following LP:

$$\begin{aligned} & \min\ x + 4y + 9z \\ \text{s.t.}\quad & x + y \leq 5 \\ & x + z \geq 10 \\ & -y + z = 7 \\ & 0 \leq x \leq 4 \\ & -1 \leq y \leq 1 \end{aligned}$$

$$0 \leq z$$
$$x, y, z \in \mathbb{R}$$

Now let us assume that some of the constraints are uncertain. We build an uncertainty set $\mathcal{U}$ consisting of two scenarios that randomly disturb the right-hand side of one constraint each. In ROPI, this can be achieved using

```
FiniteUncertainty unc;
list<Con>* lpcons = lp.get_cons();
for (int i=0; i<2; ++i)
{
  list<Con>::iterator it = lpcons->begin();
  advance(it,rand()%(lpcons->size()));
  FiniteScenario scen;
  scen.rhs[*it] = (it->rhs) * (1 + (rand()%100)/400.0 - 1.0/8 );
  unc.scenarios.push_back(scen);
}
```

The object `unc` now consists of two scenarios that modify the right-hand side of a constraint by up to 25%. Assume now that we thus generate a scenario set $\mathcal{U} = \{(5, 10, 8), (5, 12, 7)\}$. The optimal objective value for the nominal case is then 54, while it is 62 and 80 in scenario 1 and 2, respectively.

We would like to solve the resulting uncertain optimization problem using the extended RecOpt counterpart with l_1 norm and center objective function. To do so, we create a robustness object that generates the required LP.

```
Robustness rob(&lp,ROB_RECOPT);
rob.set_uncertainty(unc);
RobustnessOptions opt;
opt.recobj = REC_CENTER;
opt.norm = NORM_L1;
opt.recopt_model = RECOPT_EXTENDED;
opt.solvertype = SOL_GUROBI;
rob.set_options(opt);
LP rc = rob.generate_robust();
```

Using only these couple of lines, we get the following robust counterpart for $\mathcal{U}$:

$$\begin{array}{lll} & \min & c \\ \text{s.t.} & \text{Nominal case} \left\{ \begin{array}{l} x^0 - y^0 \leq 5 \\ x^0 + z^0 \geq 10 \\ -y^0 + z^0 = 7 \end{array} \right. & \end{array}$$

$$
\begin{aligned}
\text{Scenario 1} &\left\{\begin{array}{l} x^1 - y^1 \leq 5 \\ x^1 + z^1 \geq 10 \\ -y^1 + z^1 = 8 \end{array}\right. \\
\text{Scenario 2} &\left\{\begin{array}{l} x^2 - y^2 \leq 5 \\ x^2 + z^2 \geq 12 \\ -y^2 + z^2 = 7 \end{array}\right. \\
\text{Optimality} &\left\{\begin{array}{l} x^0 + 4y^0 + 9z^0 = 54 \\ x^1 + 4y^1 + 9z^1 = 62 \\ x^2 + 4y^2 + 9z^2 = 80 \end{array}\right. \\
\text{Nominal feasibility} &\left\{\begin{array}{l} r_x - r_y \leq 5 \\ r_x + r_z \geq 10 \\ -r_y + r_z = 7 \end{array}\right. \\
\text{Distance to nominal solution} &\left\{\begin{array}{l} -a_x^0 \leq r_x - x^0 \leq a_x^0 \\ -a_y^0 \leq r_y - y^0 \leq a_y^0 \\ -a_z^0 \leq r_z - z^0 \leq a_z^0 \end{array}\right. \\
\text{Distance to scenario 1 solution} &\left\{\begin{array}{l} -a_x^1 \leq r_x - x^1 \leq a_x^1 \\ -a_y^1 \leq r_y - y^1 \leq a_y^1 \\ -a_z^1 \leq r_z - z^1 \leq a_z^1 \end{array}\right. \\
\text{Distance to scenario 2 solution} &\left\{\begin{array}{l} -a_x^2 \leq r_x - x^2 \leq a_x^2 \\ -a_y^2 \leq r_y - y^2 \leq a_y^2 \\ -a_z^2 \leq r_z - z^2 \leq a_z^2 \end{array}\right. \\
\text{Objective constraints} &\left\{\begin{array}{l} a_x^0 + a_y^0 + a_z^0 \leq c \\ a_z^1 + a_y^1 + a_z^1 \leq c \\ a_z^2 + a_y^2 + a_z^2 \leq c \end{array}\right. \\
\text{Variable bounds} &\left\{\begin{array}{l} 0 \leq x^0, x^1, x^2, r_x \leq 4 \\ -1 \leq y^0, y^1, y^2, r_y \leq 1 \\ 0 \leq y^0, z^1, z^2, r_z \end{array}\right. \\
& x^i, y^i, z^i \in \mathbb{R}\ \forall i \in \{0,1,2\} \\
& r_x, r_y, r_z \in \mathbb{R} \\
& a_x^i, a_y^i, a_z^i \in \mathbb{R}\ \forall i \in \{0,1,2\}
\end{aligned}
$$

In a preprocessing step, all scenarios are solved to optimality - in this case, using Gurobi (determined by the `solvertype` option). The resulting robust counterpart is handed back to the user, who can then proceed to solve it using a `solver` object:

```
Solver sol;
sol.init(&rc, SOL_CPLEX);
```

```
sol.solve();
if (sol.get_statios() == SOL_OPTIMAL)
  sol.write_solition(cout,-1);
```

These lines generate a solver object, hand it the robust counterpart, and solve it using Cplex. If the problem is solved to optimality, the solution is printed to the standard output. In this example, the recovery robust solution turns out to be $(4, 0, 7)$ with a recovery distance of 2 to the three scenario solutions $(4, -1, 6)$, $(3, -1, 7)$, and $(4, 1, 8)$, while the optimal nominal solution would be $(4, -1, 6)$.

References

[BTN98] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

[CGKS11] E. Carrizosa, M. Goerigk, M.-C. Körner, and A. Schöbel. Recovery to feasibility in robust optimization. Technical report, Preprint-Reihe, Institut fr Numerische und Angewandte Mathematik, Georg-August Universitt Gttingen, 2011. submitted.

[FM09] M. Fischetti and M. Monaci. Light robustness. In R. K. Ahuja, R.H. Möhring, and C.D. Zaroliagis, editors, *Robust and online large-scale optimization*, volume 5868 of *Lecture Note on Computer Science*, pages 61–84. Springer, 2009.

[GS11] Marc Goerigk and Anita Schöbel. A scenario-based approach for robust linear optimization. In *Proceedings of the First international ICST conference on Theory and practice of algorithms in (computer) systems*, TAPAS'11, pages 139–150, Berlin, Heidelberg, 2011. Springer-Verlag.