Short tutorial for solver SMIQP

23 juin 2020

1 Introduction

The software Solution of Mixed Integer Quadratic Programs (SMIQP) is an open-source C code that implements method Mixed Integer Quadratic Convex Reformulation (MIQCR) [2, 12, 3, 4, 5, 6, 7, 9]. It that solves to $\epsilon-$global optimality MIQCQP (Mixed Integer Quadratically Constrained Quadratic Programs) problems that can be formulated as (P) :

$$(P)\begin{cases} \min\langle Q, xx^T\rangle + c^T x & \\ \text{subject to} & \\ \quad a_r^T x = b_r & r = 0,\dots,m-1 \ (m \text{ linear equalities}) \\ \quad d_r^T x \le e_r & r = 0,\dots,p-1 \ (p \text{ linear inequalities}) \\ \quad \langle Aq_r, xx^T\rangle + c_r^T x = b_r & r = 0,\dots,mq-1 \ (mq \text{ quadratic equalities}) \\ \quad \langle Dq_r, xx^T\rangle + c_r^T x \le b_r & r = 0,\dots,pq-1 \ (pq \text{ quadratic inequalities}) \\ \quad \ell_i \le x_i \le u_i & i = 0,\dots,n-1 \ (n \text{ variables}) \\ \quad x_i \in \mathbb{N} & i = 0,\dots,nb\_int-1 \ (nb\_int \text{ integer variables}) \\ \quad x_i \in \mathbb{R} & i = nb\_int,\dots n-1 \ (n - nb\_int \text{ real variables}) \end{cases}$$

with :

— $(Q, c) \in \mathcal{S}_n \times \mathbb{R}^n$,
— $\forall\ r = 0,\dots m-1$, $(a_r) \in \mathbb{R}^n$, $b \in \mathbb{R}^m$,
— $\forall\ r = 0,\dots p-1$, $(d_r) \in \mathbb{R}^n$, $e \in \mathbb{R}^p$,
— $\forall\ r = 0,\dots mq-1$, $(Aq_r, c_r) \in \mathcal{S}_n \times \mathbb{R}^n$, $bq \in \mathbb{R}^{mq}$,
— $\forall\ r = 0,\dots pq-1$, $(Dq_r, c_r) \in \mathcal{S}_n \times \mathbb{R}^n$, $eq \in \mathbb{R}^{pq}$
— $(\ell, u) \in \mathbb{R}^n \times \mathbb{R}^n$.

MIQCR is a two-phase algorithm. In the first phase, it computes a strong convex formulation of (P) by solving a semi-definite programming problem. Then, the second phase consists in solving (P) with a branch-and-bound algorithm based on this strong quadratic convex relaxation.

It uses the solver csdp of Brian Borchers [8] together with the ConicBundle library of Christoph Helmberg [10] for solving the semi-definite program. It uses the C interface of the solver Cplex [11] for solving the quadratic convex problems at each node of the search tree. For computing feasible local solutions, we use the local solver Ipopt [13], or Scip [1]. SMIQP also works only with sub-solvers Scip or Ipopt.

2 Download and Installation instructions for SMIQP

The software SMIQP can be downloaded at www.cedric.cnam.fr/~lamberta/smiqp.

The build process for SMIQP should be fairly automatic as it uses GNU autotools. SMIQP has to be compiled and installed using the commands :

```
./configure
make
```

The configure script attempts to find all of the machine specific settings (compiler, libraries,...) necessary to compile and run the code. Although configure should find most of the standard ones, you may have to manually specify a few of the settings. The options for the configure script can be found by issuing the command :

```
./configure --help
```

For a more in depth description of these options, the reader is invited to refer to the COIN-OR BuildTools trac page.

If you have Cplex or Scip installed on your machine, you may want to use it as the Mixed Integer Quadratic Programming subsolver. To do so you have to specify the location of the header files and librarie by passing it as an argument to the configure script. More precisely, specify the location of the Cplex/Scip header files by using the argument -with-cplex-cflags and -with-scip-cflags and the location

of the Cplex/Scip library with -with-cplex-lflags and -with-scip-lflags.

For example, on a Linux machine if Cplex is installed in /usr/ilog, and Scip in /usr/local you would invoke configure with the arguments as follows :

```
./configure --with-cplex-cflags="/usr/ilog/CPLEX_Studio129/cplex/include"
--with-cplex-lflags="/usr/ilog/CPLEX_Studio129/cplex/lib/x86-64_linux/static_pic"
--with-scip-lflags="/usr/local/include/scip" --with-scip-cflags="/usr/local/lib"
```

3 Input file format

The solver SMIQP reads the following format :

n nb_int m p mq pq
u
$u_1\ u_2\ \ldots\ u_n$
l
$\ell_1\ \ell_2\ \ldots\ \ell_n$
Q
nnzQ
$i\ j\ q_{ij}$
C
nnzc
$i\ c_i$
A
nnzA $r = 0, \ldots, m-1$
$r\ i\ a_{ri}$
b
nnzb
$r\ b_r$
D
nnzD $r = 0, \ldots, p-1$
$r\ i\ d_{ri}$
e
nnze
$r\ e_r$
Aq
nnzAq_r + nnzc_r $r = 0, \ldots, mq-1$

$r\ \ 0\ \ i+1\ \ c_{ri}$
$r\ \ i+1\ \ j+1\ \ q_{rij}$
`bq`
`nnz`bq
$r\ \ bq_r$
`Dq`
`nnz`Dq_r `+ nnz`$c_r\ \ r = 0, \ldots, pq-1$
$r\ \ 0\ \ i+1\ \ c_{ri}$
$r\ \ i+1\ \ j+1\ \ q_{rij}$
`eq`
`nnz`eq
$r\ \ e_r$

where `nnz`$M_r\ \ r = 1, \ldots m$ is the sum of the number of non zero elements of matrices/vectors M_r.

A toy example :

$$(P)\begin{cases} \min x^T \begin{pmatrix} -7 & 3 & -15 & -4 \\ 3 & -14 & -7 & -13 \\ -15 & -7 & 8 & 7 \\ -4 & -13 & 7 & 12 \end{pmatrix} x + \begin{pmatrix} 15 \\ 10 \\ -7 \\ -4 \end{pmatrix} x & \\ \text{subject to} & \\ \quad 5x_1 + x_2 + 8x_3 + 4x_4 = 95 & \\ \quad 5x_1x_2 + 3x_3^2 + 2x_4^2 + 8x_3 + 4x_4 \leq 105 & \\ \quad 0 \leq x_i \leq 10 & i = 1, \ldots, 4 \\ \quad x_1, x_2 \in \mathbb{N} & \\ \quad x_3, x_4 \in \mathbb{R} & \end{cases}$$

The associated file format is the following :

```
4 2 1 0 0 1
u
10 10 10 10
l
0 0 0 0
Q
```

```
16
0 0 -7
0 1 3
0 2 -15
0 3-4
1 0 3
1 1 -14
1 2 -7
1 3 -13
2 0 -15
2 1 -7
2 2 8
2 3 7
3 0 -4
3 1 -13
3 2 7
3 3 12
c
4
0 15
1 10
2 -7
3 -4
A
4
0 0 5
0 1 1
0 2 8
0 3 4
b
1
0 95
Dq
10
0 0 2 0.5
0 0 3 4.0
0 0 4 2.0
0 1 2 2.5
```

```
0 2 0 0.5
0 2 1 2.5
0 3 0 4.0
0 3 3 3.0
0 4 0 2.0
0 4 4 2.0
eq
1
0 105
```

Remarks :

- Each index starts from 0.
- Each non-zero term of the matrix Q has to be written in the file. If Q is not symmetric, `SMIQP` rewrites Q as matrix Q' where $Q' = \frac{Q+Q^T}{2}$.
- If there is no real variable, then `nb_int` is equal to `n`.
- If the problem does not contain linear/quadratic equality or inequality constraints do not put the corresponding lines in the instance file
- If you write a linear constraints using the linear part of a quadratic constraint, the associated RLT constraints will not be used into the convexification process.

4 Parameters

The user can set several parameters, for this create a `param.smiqp` file that contains the new parameter values. An example of this file is the following :

```
MAX_SOL_BB 1e10
MIN_SOL_BB -1e10
EPS_BETA 1e-6
EPS_INT 1e-2
EPS_LM 1e-6
EPS_ABS_GAP 0.99
EPS_BRANCH 1e-4
EPS_BB 1e-5
FACTOR 1
GAMMA 0
MAX_TIME_SOL_INIT 5
```

```
MAX_TIME_SOL_LOCAL 5
EPS_LOCAL_SOL 1e-4
NB_NODE_COMPUTE_LOCAL 10
TIME_LIMIT_BB 3600
REL_GAP 1e-5
ABS_GAP 0.999
PRINT_LEVEL_NODE 50
TIME_LIMIT_CPLEX 3600
VARSEL 0
PRINT_LEVEL_SCIP 0
PRINT_LEVEL_IPOPT 0
MAX_SOL_SDP 1e12
EPS_VIOL_CB 1e-3
EPS_TERM_CB 1e-4
ACT_BOUND 1
EVAL_LIMIT 1000
UPDATE_LIMIT 500
NB_MAX_ITER 150
TIME_LIMIT_CB 3600
EPS_STEP 1e-2
EPS_SDP 1e-4
```

Where :

- Parameters of `SMIQP` :
  - `MAX_SOL_BB` (`double` parameter : ≥ 0) Starting upper bound on the problem. (default `1e10`).
  - `MIN_SOL_BB` (`double` parameter : ≤ 0) Starting lower bound on the problem. (default `-1e10`).
  - `EPS_BETA` (`double` parameter : ≥ 0) Accuracy of a Zero, positive or negative value. (default `1e-6`).
  - `EPS_INT` (`double` parameter : ≥ 0) Accuracy for considering that a value is integer. (default `1e-2`).
  - `EPS_LM` (`double` parameter : ≥ 0) Accuracy for considering non negative the smallest eigenvalue of S^*. (default `1e-6`).
  - `EPS_ABS_GAP` (`double` parameter : $\in [0, 1]$) Absolute gap of the spatial branch-and-bound. (default `0.99`)
  - `EPS_BRANCH` (`double` parameter : ≥ 0) Acurracy for pruning a branch in the spatial branch-and-bound. (default `1e-2`).
  - `EPS_BB` (`double` parameter : ≥ 0) Relative gap for ending the spatial

branch-and-bound. (default `1e-5`).

- `FACTOR` (`double` parameter : $\in [0,1]$) Proportion of the considered constraints into the SDP solver. (default `1`).
- `GAMMA` (`double` parameter : $\in [0,1]$) Parameter that determines the value for branching. (default `0`).
- `MAX_TIME_SOL_INIT` (`int` parameter : ≥ 0) Maximum time (in seconds) for computing an upper bound at the root node. (default `5`).
- `MAX_TIME_SOL_LOCAL` (`int` parameter : ≥ 0) Maximum time (in seconds) for computing an upper bound at a node. (default `5`).
- `EPS_LOCAL_SOL` (`double` parameter : ≥ 0) Accuracy for computing an upper bound. (default `1e-4`).
- `NB_NODE_COMPUTE_LOCAL` (`int` parameter : ≥ 0) Frequency (in nodes) for running local solver during the spatial branch-and-bound. (default `10`).
- `TIME_LIMIT_BB` (`int` parameter : ≥ 0) Time limit for the spatial branch-and-bound. (default `3600`).
- `REL_GAP` (`double` parameter : ≥ 0) Relative gap for the quadratic convex relaxation solved at each node of the spatial branch-and-bound. 1e-5 (default `1e-5`).
- `ABS_GAP` (`double` parameter : ≥ 0) Absolute gap the quadratic convex reformulation in the pure integer case. (default `0.999`).
- `PRINT_LEVEL_NODE` (`int` parameter : ≥ 0) Frequency (in nodes) of the print of the state of the spatial branch-and-bound(default `10`).

— Parameters of sub-solver `Cplex`

- `TIME_LIMIT_CPLEX` (`int` parameter : ≥ 0) Time limit for solving the quadratic convex reformulation in the pure integer case. (default `3600`).
- `VARSEL` (`int` parameter : $\in \{0,1,2,3,4\}$) Variable selection strategy in `Cplex` in the pure integer case. (default `0`).

— Parameters of sub-solver `Scip`

- `PRINT_LEVEL_SCIP` (`int` parameter : ≥ 0) Print level of `Scip` sub-solver (default `0`).

— Parameters of sub-solver `Ipopt`

- `PRINT_LEVEL_IPOPT` (`int` parameter : ≥ 0) Print level of `Ipopt` sub-solver (default `0`).

— Parameters of the SDP solver :

- `MAX_SOL_SDP` (`double` parameter : ≥ 0) Starting upper SDP bound on the problem. (default `1e12`).
- `EPS_VIOL_CB` (`double` parameter : ≥ 0) Accuracy of the violation of a constraints in the `ConicBundle`. (default `1e-3`).

— `EPS_TERM_CB` (`double` parameter : ≥ 0) Parameter `sg_norm` of the `ConicBundle`. (default `1e-2`).
— `EVAL_LIMIT` (`int` parameter : ≥ 0) Parameter `eval_limit` of the `ConicBundle`. (default `1000`).
— `UPDATE_LIMIT` (`int` parameter : ≥ 0) Parameter `update_limit` of the `ConicBundle`. (default `500`).
— `NB_MAX_ITER` (`int` parameter : ≥ 0) Maximum number of iteration of the SDP solver. (default `150`).
— `TIME_LIMIT_CB` (`int` parameter : ≥ 0) Time limit (in seconds) for the SDP solver. (default `3600`).
— `EPS_STEP` (`double` parameter : ≥ 0) Accuracy of a step of the SDP solver.(default `1e-2`).
— `EPS_SDP` (`double` parameter : ≥ 0) Accuracy of the SDP solver. (default `1e-4`).

Références

[1] T. Achterberg. Scip : solving constraint integer programs. *Mathematical Programming Computation*, (1) :1–41, 2009.

[2] A. Billionnet, S. Elloumi, and A. Lambert. Linear reformulations of integer quadratic programs. In *MCO 2008, september 8-10*, pages 43–51, 2008.

[3] A. Billionnet, S. Elloumi, and A. Lambert. Extending the QCR method to the case of general mixed integer program. *Mathematical Programming*, 131(1) :381–401, 2012.

[4] A. Billionnet, S. Elloumi, and A. Lambert. An efficient compact quadratic convex reformulation for general integer quadratic programs. *Computational Optimization and Applications*, 54(1) :141–162, 2013.

[5] A. Billionnet, S. Elloumi, and A. Lambert. A branch and bound algorithm for general mixed-integer quadratic programs based on quadratic convex relaxation. *Journal of Combinatorial Optimization*, 2(28) :376–399, 2014.

[6] A. Billionnet, S. Elloumi, and A. Lambert. Exact quadratic convex reformulations of mixed-integer quadratically constrained problems. *Mathematical Programming*, 158(1) :235–266, 2016.

[7] A. Billionnet, S. Elloumi, A. Lambert, and A. Wiegele. Using a Conic Bundle method to accelerate both phases of a Quadratic Convex Reformulation. *INFORMS Journal on Computing*, 29(2) :318–331, 2017.

[8] B. Borchers. CSDP, A C Library for Semidefinite Programming. *Optimization Methods and Software*, 11(1) :613–623, 1999.

[9] S. Elloumi and A. Lambert. Global solution of non-convex quadratically constrained quadratic programs. *Optimization Methods and Software*, 34(1) :98–114, 2019.

[10] C. Helmberg. *Conic Bundle v0.3.10*, 2011.

[11] IBM-ILOG. IBM ILOG CPLEX 12.7 Reference Manual. "`http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.7.0/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html`", 2017.

[12] A. Lambert. *Résolution de programmes quadratiques en nombres entiers*. Thèse de doctorat en informatique, Conservatoire National des Arts et Métiers, Paris, 2009.

[13] A. Wächter and L.T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1) :25–57, Mar 2006.