Global QCQP Solver

Chuwen Zhang

April 6, 2021

Contents

1 Develop Plan **1**

2 Scope of Work **2**

2.1 Modeling interface 2

2.2 SDP interface 2

2.3 Local Refinement 3

2.4 Branch-and-Cut for Global Optimization 3

3 Semidefinite Relaxation **3**

3.1 Method I 3

3.2 Method II 4

3.3 Extending to matrix and tensor case 4

3.4 Tests 4

References **5**

1 Develop Plan

The first part is devoted to SDP relaxations.

- We first test HQCQP for vectors, add support for SeDuMi, should be able to switch SDP solvers
- Extend to non-homogeneous QCQP.
- Develop and test both relaxations, see 3.1, 3.2

- We test on well-known problems (vectors), e.g., QKP (quadratic knapsack), Max-cut, ...
- Start to test matrix variable problems, see 3.3, SNL, QAP, etc using vectorized method.
- Introduce matrix variables, ..., if needed.

Start with Pure Python or Julia interface as a fast prototype. Then migrate to C/C++ interface with Python and Matlab support.

The second part is devoted to "Refinement". to be discussed.

The third part is for Branch-and-cut method, using ideas from integer programming, to create a global QCQP solver using SDP as lower bound in the branching process. to be discussed.

Last part is for modeling and AMLs support.

- in Python one can use cvxpy or other AMLs; in Julia one may extend JuMP.
- add support for COPT?

2 Scope of Work

Develop a QCQP solver that uses SDP-relaxation-and-refinement approach. The QCQP solver should be problem-independent that works for any QCQP instance.

2.1 Modeling interface

The modeling interface is domain specific language, a simple tool for user to define a **QCQP** problem, then the solver translates into canonical form of QCQP. For example, for a HQCQP, canonical form includes parameters Q, $A_i, b_i, \forall i$

To-dos:

- The modeling part does the canonicalization, works like cvxpy, yalmip, etc.
- Directly use COPT.

See Agrawal et al. (2018), Diamond and Boyd (2016), Dunning et al. (2017), Lofberg (2004)

2.2 SDP interface

The SDP interface should be solver **independent**. SDP interface starts with canonical form to create a SDP-relaxation. So the users do not have to derive SDP by themselves. The interface should output a SDP problem in a standard format, e.g., SDPA format, that can be accepted by any SDP solver.

To-dos including:

- starts with canonical form.
- interface with solver: create problems, extract solutions, status, etc.
- consider two types of relaxation: method-I, method-II.

We consider two types of SDP relaxations:

2.3 Local Refinement

Local Refinement from SDP solution to QP seems to be problem dependent, whereas we can start with:

- Use Gurobi to do the refinement
- use existing methods, including residual minimization (SNL), randomization (for BQP), see Luo et al. (2010) and papers for SNL.
- add an **option** for user to choose a refinement method.

2.4 Branch-and-Cut for Global Optimization

3 Semidefinite Relaxation

We consider two types of SDP relaxation for canonical QCQP. We first consider for the case where x is a vector, i.e., $x \in \mathbb{R}^n$.

$$\begin{aligned} &\text{Maximize} \quad x^T Q x \\ \text{s.t.} \quad & \\ & x^T A_i x (\leq, =, \geq) b_i, \forall i = 1, \dots, m \end{aligned}$$

3.1 Method I

$$\begin{aligned} \text{since:} \quad & x^T A_i x = A_i \bullet (x x^T) \\ \text{SDP relaxation:} \quad & \\ & \text{Maximize} \quad Q \bullet Y \\ & Y - x x^T \succeq 0 \text{ or } \begin{bmatrix} 1 & x^T \\ x & Y \end{bmatrix} \succeq 0 \end{aligned}$$

for matrix X:

$$X^T A_i X = A_i \bullet (XX^T)$$

SDP relaxation:

$$\text{Maximize} \quad Q \bullet Y$$

$$Y - XX^T \succeq 0 \text{ or } \begin{bmatrix} I_d & X^T \\ X & Y \end{bmatrix} \succeq 0$$

3.2 Method II

Let $A = A_+ + A_-$ where $A_-, A_+ \succeq 0$, so we do Cholesky $R_+^T R_+ = A_+$, since R_+ may be low-rank, then we can define z_+ according the rank.

$$R_+ x = z_+, x^T A_+ x = ||z_+||^2 = \sum_i (z_+)_i^2$$

$$y_i = (z_+)_i^2, \begin{bmatrix} 1 & (z_+)_i \\ (z_+)_i & y_i \end{bmatrix} \succeq 0, \forall i$$

This is the so-called "many-small-cone" method.

3.3 Extending to matrix and tensor case

We first develop for the vector case: $x \in \mathbb{R}^n$, whereas QCQP is not limited to vector case:

- vectors, $x \in \mathbb{R}^n$, max-cut, quadratic knapsack problem
- matrices, $x \in \mathbb{R}^{n \times d}$, quadratic assignment problem, SNL, kissing number.

For example, SNL uses $X \in \mathbb{R}^{n \times d}$ for d-dimensional coordinates. For higher dimensional case, followings can be done:

- We may first however using vectorized method, i.e., $x = \mathsf{vec}(X)$ to reformulate the matrix-based optimization problem, given the SDP bounds by original and vectorized relaxations are equivalent. (verify this, Ding et al. (2011))
- the above method may create a matrix of very large dimension by Kronecker product. We should test this, if true, then we must include matrix and tensor variables.
- ultimately, the solver should provide an option to use user specified relaxations.

3.4 Tests

We test on applications:

- vectors, $x \in \mathbb{R}^n$, max-cut, quadratic knapsack problem
- matrices, $x \in \mathbb{R}^{n \times d}$, QAP, SNL, kissing number.

References

Agrawal A, Verschueren R, Diamond S, Boyd S (2018) A rewriting system for convex optimization problems. *Journal of Control and Decision* 5(1):42–60, publisher: Taylor & Francis.

Diamond S, Boyd S (2016) CVXPY: A Python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research* 17(1):2909–2913, publisher: JMLR. org.

Ding Y, Ge D, Wolkowicz H (2011) On equivalence of semidefinite relaxations for quadratic matrix programming. *Mathematics of Operations Research* 36(1):88–104, publisher: INFORMS.

Dunning I, Huchette J, Lubin M (2017) JuMP: A modeling language for mathematical optimization. *SIAM review* 59(2):295–320, publisher: SIAM.

Lofberg J (2004) YALMIP: A toolbox for modeling and optimization in MATLAB. *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, 284–289 (IEEE).

Luo ZQ, Ma WK, So AMC, Ye Y, Zhang S (2010) Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine* 27(3):20–34, publisher: IEEE.

Appendix