# Reformulation methods inside a commercial MIQCQP solver

KTH Royal Institute of Technology
May 20th, 2022

**Sven Wiese**

www.mosek.com

A software package/library for solving:

- Linear and conic problems.
- Convex quadratic and quadratically constrained problems.
- Also mixed-integer versions of the above.

Current version is **MOSEK** 10.

- Currently supported cone types are second-order, exponential, power, geometric mean and semi-definite.

# MOSEK

A software package/library for solving:

- Linear and conic problems.
- Convex quadratic and quadratically constrained problems.
- Also mixed-integer versions of the above.

Current version is **MOSEK** 10.

- Currently supported cone types are second-order, exponential, power, geometric mean and semi-definite.

# MIQCQP

Consider the problem

$$
\begin{array}{lll}
\min & x^T Q^0 x + c^T x & \\
\text{s.t.} & x^T Q^k x + a_k^T x \leq b_k, & k = 1, \ldots, m \\
& l \leq x \leq u & \\
& x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}. &
\end{array} \qquad \text{(P)}
$$

- Decisive question: are all involved $Q$-matrices positive semi-definite (p.s.d.) or not?
- That means, is the model a **convex** or **non-convex** MIQCQP?

# MIQCQP

Consider the problem

$$
\begin{array}{lll}
\min & x^T Q^0 x + c^T x & \\
\text{s.t.} & x^T Q^k x + a_k^T x \le b_k, & k = 1, \dots, m \\
 & l \le x \le u & \\
 & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}. &
\end{array}
\qquad \text{(P)}
$$

- Decisive question: are all involved $Q$-matrices positive semi-definite (p.s.d.) or not?
- That means, is the model a **convex** or **non-convex** MIQCQP?

# Reformulating quadratic terms

A quadratic term $x_i x_j$ may be reformulated to linear constraints:

- $x_i, x_j \in \{0,1\}$: substitute $x_i x_j$ with $X_{ij}$, and impose

$$x_i + x_j - 1 \leq X_{ij} \leq \min\{x_i, x_j\}.$$

- $x_i \in \{0,1\}, l_j \leq x_j \leq u_j$: $x_i x_j \leftarrow X_{ij}$, and (special case of McCormick inequalities):

$$l_j x_i \leq X_{ij} \leq u_j x_i,$$
$$x_j - u_j(1 - x_i) \leq X_{ij} \leq x_j - l_j(1 - x_i).$$

- $0 \leq x_i \leq u_i$ integer: $x_i \leftarrow \sum_{t=0}^{\lfloor \log(u_i) \rfloor} 2^t z_{ti}$, and proceed as above.

# Reformulating quadratic terms

A quadratic term $x_i x_j$ may be reformulated to linear constraints:

- $x_i, x_j \in \{0,1\}$: substitute $x_i x_j$ with $X_{ij}$, and impose

$$x_i + x_j - 1 \le X_{ij} \le \min\{x_i, x_j\}.$$

- $x_i \in \{0,1\}, l_j \le x_j \le u_j$: $x_i x_j \leftarrow X_{ij}$, and (special case of McCormick inequalities):

$$l_j x_i \le X_{ij} \le u_j x_i,$$
$$x_j - u_j(1 - x_i) \le X_{ij} \le x_j - l_j(1 - x_i).$$

- $0 \le x_i \le u_i$ integer: $x_i \leftarrow \sum_{t=0}^{\lfloor \log(u_i) \rfloor} 2^t z_{ti}$, and proceed as above.

# Reformulating quadratic terms

A quadratic term $x_i x_j$ may be reformulated to linear constraints:

- $x_i, x_j \in \{0, 1\}$: substitute $x_i x_j$ with $X_{ij}$, and impose

$$x_i + x_j - 1 \leq X_{ij} \leq \min\{x_i, x_j\}.$$

- $x_i \in \{0, 1\}, l_j \leq x_j \leq u_j$: $x_i x_j \leftarrow X_{ij}$, and (special case of McCormick inequalities):

$$\begin{gathered} l_j x_i \leq X_{ij} \leq u_j x_i, \\ x_j - u_j(1 - x_i) \leq X_{ij} \leq x_j - l_j(1 - x_i). \end{gathered}$$

- $0 \leq x_i \leq u_i$ integer: $x_i \leftarrow \sum_{t=0}^{\lfloor \log(u_i) \rfloor} 2^t z_{ti}$, and proceed as above.

# Reformulating quadratic terms (cont.)

- A quadratic term may be "just linearized" in this way, i.e.,

$$q_{ij}x_ix_j \leftarrow q_{ij}X_{ij},$$

  or "perturbed":

$$q_{ij}x_ix_j \leftarrow (q_{ij} + q'_{ij})x_ix_j - q'_{ij}X_{ij}.$$

- Perturbation can be used, e.g., to replace a non-p.s.d. $Q$-matrix with a p.s.d. one.
- These techniques may in principle be applied to both convex and non-convex MIQCQPs.
- For simplicity assume all possible products $x_ix_j$ can be linearized, i.e., no continuous variables or infinite/huge bounds (extension possible and implemented in **MOSEK**).

# Reformulating quadratic terms (cont.)

- A quadratic term may be "just linearized" in this way, i.e.,

$$q_{ij}x_ix_j \leftarrow q_{ij}X_{ij},$$

  or "perturbed":

$$q_{ij}x_ix_j \leftarrow (q_{ij} + q'_{ij})x_ix_j - q'_{ij}X_{ij}.$$

- Perturbation can be used, e.g., to replace a non-p.s.d. $Q$-matrix with a p.s.d. one.

- These techniques may in principle be applied to both convex and non-convex MIQCQPs.

- For simplicity assume all possible products $x_ix_j$ can be linearized, i.e., no continuous variables or infinite/huge bounds (extension possible and implemented in **MOSEK**).

# Reformulating quadratic terms (cont.)

- A quadratic term may be "just linearized" in this way, i.e.,

$$q_{ij}x_ix_j \leftarrow q_{ij}X_{ij},$$

or "perturbed":

$$q_{ij}x_ix_j \leftarrow (q_{ij} + q'_{ij})x_ix_j - q'_{ij}X_{ij}.$$

- Perturbation can be used, e.g., to replace a non-p.s.d. $Q$-matrix with a p.s.d. one.

- These techniques may in principle be applied to both convex and non-convex MIQCQPs.

- For simplicity assume all possible products $x_ix_j$ can be linearized, i.e., no continuous variables or infinite/huge bounds (extension possible and implemented in **MOSEK**).

# A family of MIQCQP reformulations

As in [Billionnet et al., 2016], consider reformulations of the form

$$
\begin{array}{lll}
\min & x^T(Q^0+P^0)x + c^T x - \langle P^0, X\rangle & \\
\text{s.t.} & x^T(Q^k+P^k)x + a_k^T x - \langle P^k, X\rangle \le b_k, & k = 1,\dots,m \\
& (x,X) \in S_{L_P} & \\
& l \le x \le u & \\
& x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, &
\end{array}
\tag{$\mathrm{R}_{P^0,\dots,P^m}$}
$$

where $S_{L_P}$ contains all linearization constraints over

$$L_P := \{(i,j) \mid \exists k : p_{ij}^k \neq 0\}.$$

- $(x,X) \in S_{L_P}$ encodes $X = xx^T$, i.e., the products $X_{ij} = x_i x_j$.
- We are interested in reformulations ($\mathrm{R}_{P^0,\dots,P^m}$) that are convex MIQCQPs.

# A family of MIQCQP reformulations

As in [Billionnet et al., 2016], consider reformulations of the form

$$
\begin{array}{lll}
\min & x^T(Q^0 + P^0)x + c^T x - \langle P^0, X\rangle & \\
\text{s.t.} & x^T(Q^k + P^k)x + a_k^T x - \langle P^k, X\rangle \le b_k, & k = 1, \dots, m \\
 & (x, X) \in S_{L_P} & \\
 & l \le x \le u & \\
 & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, &
\end{array}
\qquad (\mathrm{R}_{P^0,\dots,P^m})
$$

where $S_{L_P}$ contains all linearization constraints over

$$L_P := \{(i,j) \mid \exists k : p_{ij}^k \neq 0\}.$$

- $(x, X) \in S_{L_P}$ encodes $X = xx^T$, i.e., the products $X_{ij} = x_i x_j$.
- We are interested in reformulations $(\mathrm{R}_{P^0,\dots,P^m})$ that are convex MIQCQPs.

# Reformulation method 1: complete linearization

Setting $P^k = -Q^k$ for all $k$ amounts to getting rid of all products.

- Depending on the size of $L_P$, the problem dimensions may grow considerably.
- The resulting problem is (almost surely) a MILP, leading to a technology shift.
- Denote this method by $(R_Q)$, see [Glover and Wolsey, 1974] or [Furini and Traversi, 2019].

# Reformulation method 1: complete linearization

Setting $P^k = -Q^k$ for all $k$ amounts to getting rid of all products.

- Depending on the size of $L_P$, the problem dimensions may grow considerably.
- The resulting problem is (almost surely) a MILP, leading to a technology shift.
- Denote this method by $(R_Q)$, see [Glover and Wolsey, 1974] or [Furini and Traversi, 2019].

# Reformulation method 1: complete linearization

Setting $P^k = -Q^k$ for all $k$ amounts to getting rid of all products.

- Depending on the size of $L_P$, the problem dimensions may grow considerably.
- The resulting problem is (almost surely) a MILP, leading to a technology shift.
- Denote this method by $(R_Q)$, see [Glover and Wolsey, 1974] or [Furini and Traversi, 2019].

# Reformulation 2: the eigenvalue-method

Let the eigenvalues of $Q^k$ be $\lambda_1 \leq \ldots \leq \lambda_m$.

- Setting $P^k = -\lambda_1 I$ leads to a p.s.d. matrix.
- Originally proposed for 0-1 programming [Hammer and Rubin, 1970]. Denote this method by $(R_\lambda)$.
- Choosing the smallest eigenvalue means the "least convex" function and a (hopefully) better dual bound.

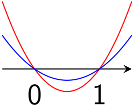



- The amount of linearization tends to be lower than for a complete linearization, the resulting problem remains a MIQCQP.

# Reformulation 2: the eigenvalue-method

Let the eigenvalues of $Q^k$ be $\lambda_1 \leq \ldots \leq \lambda_m$.

- Setting $P^k = -\lambda_1 I$ leads to a p.s.d. matrix.
- Originally proposed for 0-1 programming [Hammer and Rubin, 1970]. Denote this method by $(R_\lambda)$.
- Choosing the smallest eigenvalue means the "least convex" function and a (hopefully) better dual bound.

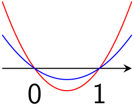



- The amount of linearization tends to be lower than for a complete linearization, the resulting problem remains a MIQCQP.

# Reformulation 2: the eigenvalue-method

Let the eigenvalues of $Q^k$ be $\lambda_1 \leq \ldots \leq \lambda_m$.

- Setting $P^k = -\lambda_1 I$ leads to a p.s.d. matrix.
- Originally proposed for 0-1 programming [Hammer and Rubin, 1970]. Denote this method by $(R_\lambda)$.
- Choosing the smallest eigenvalue means the "least convex" function and a (hopefully) better dual bound.




- The amount of linearization tends to be lower than for a complete linearization, the resulting problem remains a MIQCQP.

# Reformulation 3: the diagonal-method

Generalize the eigenvalue-method: find $P_k = -\textbf{diag}(\mu_1, \ldots, \mu_n)$ s.t. $Q^k + P^k$ is p.s.d., making the $\mu_i$ possibly large.

For example, solve

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \mu_i \\ \text{s.t.} \quad & Q^k - \textbf{diag}(\mu_1, \ldots, \mu_n) \succeq 0. \end{aligned} \tag{$\mu$-SDP}$$

- Similar to the eigenvalue-method as for the amount of linearization, but more "flexible".
- The resulting problem remains a MIQCQP also here.
- Denote this method by $(R_\mu)$, see also [Dong and Lou, 2018].

# Reformulation 3: the diagonal-method

Generalize the eigenvalue-method: find $P_k = -\mathbf{diag}(\mu_1, \ldots, \mu_n)$ s.t. $Q^k + P^k$ is p.s.d., making the $\mu_i$ possibly large.

For example, solve

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \mu_i \\ \text{s.t.} \quad & Q^k - \mathbf{diag}(\mu_1, \ldots, \mu_n) \succeq 0. \end{aligned} \qquad (\mu\text{-SDP})$$

- Similar to the eigenvalue-method as for the amount of linearization, but more "flexible".
- The resulting problem remains a MIQCQP also here.
- Denote this method by $(R_\mu)$, see also [Dong and Lou, 2018].

# Reformulation 3: the diagonal-method

Generalize the eigenvalue-method: find $P_k = -\mathbf{diag}(\mu_1, \ldots, \mu_n)$ s.t. $Q^k + P^k$ is p.s.d., making the $\mu_i$ possibly large.

For example, solve

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \mu_i \\ \text{s.t.} \quad & Q^k - \mathbf{diag}(\mu_1, \ldots, \mu_n) \succeq 0. \end{aligned} \qquad (\mu\text{-SDP})$$

- Similar to the eigenvalue-method as for the amount of linearization, but more "flexible".
- The resulting problem remains a MIQCQP also here.
- Denote this method by $(R_\mu)$, see also [Dong and Lou, 2018].

# Reformulation 3: the diagonal-method

Generalize the eigenvalue-method: find $P_k = -\mathbf{diag}(\mu_1, \ldots, \mu_n)$ s.t. $Q^k + P^k$ is p.s.d., making the $\mu_i$ possibly large.

For example, solve

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \mu_i \\ \text{s.t.} \quad & Q^k - \mathbf{diag}(\mu_1, \ldots, \mu_n) \succeq 0. \end{aligned} \tag{$\mu$-SDP}$$

- Similar to the eigenvalue-method as for the amount of linearization, but more "flexible".
- The resulting problem remains a MIQCQP also here.
- Denote this method by $(R_\mu)$, see also [Dong and Lou, 2018].

# MIQCQP's optimal SDP-relaxation

A strong SDP-relaxation of (P) can be shown to be

$$
\begin{array}{lll}
\min & \langle Q^0, X\rangle + c^T x & \\
\text{s.t.} & \langle Q^k, X\rangle + a_k^T x \le b_k, & k = 1,\dots,m \\
& X_{ij} \le u_j x_i + l_i x_j - u_j l_i & i,j = 1,\dots,p \\
& X_{ij} \le l_j x_i + u_i x_j - l_j u_i & i,j = 1,\dots,p \\
& X_{ij} \ge u_j x_i + u_i x_j - u_j u_i & i,j = 1,\dots,p \\
& X_{ij} \ge l_j x_i + l_i x_j - l_j l_i & i,j = 1,\dots,p \\
& X_{ii} \ge |x_i| & i = 1,\dots,p \\
& l \le x \le u & \\
& \begin{pmatrix} X & x \\ x^T & 1 \end{pmatrix} \succeq 0. &
\end{array}
\qquad \text{(RSDP)}
$$

- $\begin{pmatrix} X & x \\ x^T & 1 \end{pmatrix} \succeq 0$ means $X \succeq xx^T$, thus relaxing $X = xx^T$.
- McCormick constraints give the convex hull of $X_{ij} = x_i x_j$.

# MIQCQP's optimal SDP-relaxation

A strong SDP-relaxation of (P) can be shown to be

$$
\begin{array}{lll}
\min & \langle Q^0, X\rangle + c^T x & \\
\text{s.t.} & \langle Q^k, X\rangle + a_k^T x \le b_k, & k = 1, \dots, m \\
& X_{ij} \le u_j x_i + l_i x_j - u_j l_i & i,j = 1, \dots, p \\
& X_{ij} \le l_j x_i + u_i x_j - l_j u_i & i,j = 1, \dots, p \\
& X_{ij} \ge u_j x_i + u_i x_j - u_j u_i & i,j = 1, \dots, p \\
& X_{ij} \ge l_j x_i + l_i x_j - l_j l_i & i,j = 1, \dots, p \\
& X_{ii} \ge |x_i| & i = 1, \dots, p \\
& l \le x \le u & \\
& \begin{pmatrix} X & x \\ x^T & 1 \end{pmatrix} \succeq 0. &
\end{array}
\qquad \text{(RSDP)}
$$

- $\begin{pmatrix} X & x \\ x^T & 1 \end{pmatrix} \succeq 0$ means $X \succeq xx^T$, thus relaxing $X = xx^T$.
- McCormick constraints give the convex hull of $X_{ij} = x_i x_j$.

# MIQCQP's optimal SDP-relaxation

A strong SDP-relaxation of (P) can be shown to be

$$
\begin{array}{lll}
\min & \langle Q^0, X\rangle + c^T x & \\
\text{s.t.} & \langle Q^k, X\rangle + a_k^T x \le b_k, & k = 1, \dots, m \\
& X_{ij} \le u_j x_i + l_i x_j - u_j l_i & i,j = 1,\dots,p \\
& X_{ij} \le l_j x_i + u_i x_j - l_j u_i & i,j = 1,\dots,p \\
& X_{ij} \ge u_j x_i + u_i x_j - u_j u_i & i,j = 1,\dots,p \\
& X_{ij} \ge l_j x_i + l_i x_j - l_j l_i & i,j = 1,\dots,p \\
& X_{ii} \ge |x_i| & i = 1,\dots,p \\
& l \le x \le u & \\
& \begin{pmatrix} X & x \\ x^T & 1 \end{pmatrix} \succeq 0. &
\end{array}
\qquad \text{(RSDP)}
$$

- $\begin{pmatrix} X & x \\ x^T & 1 \end{pmatrix} \succeq 0$ means $X \succeq xx^T$, thus relaxing $X = xx^T$.
- McCormick constraints give the convex hull of $X_{ij} = x_i x_j$.

# Reformulation method 4: SDP-relaxation

(RSDP) also gives rise to some reformulation ($\mathrm{R}_{P^0,\dots,P^m}$) [Billionnet et al., 2016].

- Namely, denote the optimal dual matrix variable of (RSDP) by $\begin{pmatrix} S & s \\ s^T & \sigma \end{pmatrix}$, and take $P^0 = S - Q^0$ and $P^k = -Q^k$ for all $k \geq 1$.
- Among all possible reformulations ($\mathrm{R}_{P^0,\dots,P^m}$), this one has the best dual bound.
- The resulting problem is (almost surely) a MIQP, and the amount of linearization tends to be higher than for the eiganvalue- or diagonal-method.
- Call this method $(R_S)$.

# Reformulation method 4: SDP-relaxation

(RSDP) also gives rise to some reformulation $(\mathrm{R}_{P^0,\ldots,P^m})$ [Billionnet et al., 2016].

- Namely, denote the optimal dual matrix variable of (RSDP) by $\begin{pmatrix} S & s \\ s^T & \sigma \end{pmatrix}$, and take $P^0 = S - Q^0$ and $P^k = -Q^k$ for all $k \geq 1$.
- Among all possible reformulations $(\mathrm{R}_{P^0,\ldots,P^m})$, this one has the best dual bound.
- The resulting problem is (almost surely) a MIQP, and the amount of linearization tends to be higher than for the eiganvalue- or diagonal-method.
- Call this method $(R_S)$.

# Reformulation method 4: SDP-relaxation

(RSDP) also gives rise to some reformulation $(\mathrm{R}_{P^0,\ldots,P^m})$ [Billionnet et al., 2016].

- Namely, denote the optimal dual matrix variable of (RSDP) by $\begin{pmatrix} S & s \\ s^T & \sigma \end{pmatrix}$, and take $P^0 = S - Q^0$ and $P^k = -Q^k$ for all $k \geq 1$.
- Among all possible reformulations $(\mathrm{R}_{P^0,\ldots,P^m})$, this one has the best dual bound.
- The resulting problem is (almost surely) a MIQP, and the amount of linearization tends to be higher than for the eiganvalue- or diagonal-method.
- Call this method $(R_S)$.

# Reformulation method 4: SDP-relaxation

(RSDP) also gives rise to some reformulation ($\mathrm{R}_{P^0,\ldots,P^m}$) [Billionnet et al., 2016].

- Namely, denote the optimal dual matrix variable of (RSDP) by $\begin{pmatrix} S & s \\ s^T & \sigma \end{pmatrix}$, and take $P^0 = S - Q^0$ and $P^k = -Q^k$ for all $k \geq 1$.

- Among all possible reformulations ($\mathrm{R}_{P^0,\ldots,P^m}$), this one has the best dual bound.

- The resulting problem is (almost surely) a MIQP, and the amount of linearization tends to be higher than for the eiganvalue- or diagonal-method.

- Call this method ($R_S$).

# Practicability of a reformulation

- Performing a reformulation should be fast.
- Especially in a commercial solver setting, excessive reformulation times are prohibitive.
- **MOSEK** 10 has various work limits on the computational aspects of the different formulations.
- For example, $(R_\lambda)$ and $(R_\mu)$ are never attempted if matrix dimensions exceed certain thresholds.

# Practicability of a reformulation

- Performing a reformulation should be fast.
- Especially in a commercial solver setting, excessive reformulation times are prohibitive.
- **MOSEK** 10 has various work limits on the computational aspects of the different formulations.
- For example, $(R_\lambda)$ and $(R_\mu)$ are never attempted if matrix dimensions exceed certain thresholds.

# Practicability of a reformulation

```
MOSEK Version 10.0.13(BETA) (Build date: 2022-4-22 16:00:07)
Copyright (c) MOSEK ApS, Denmark WWW: mosek.com
Platform: Linux/64-X86

Reading started.
Reading terminated. Time: 0.00

Read summary
  Type                       : QO (quadratic optimization problem)
  Objective sense            : minimize
  Scalar variables           : 343
  Matrix variables           : 0
  Scalar constraints         : 0
  Affine conic constraints : 0
  Disjunctive constraints    : 0
  Cones                      : 0
  Integers                   : 343
  Time                       : 0.0

Problem
  Name                     :
  Objective sense          : minimize
  Type                     : QO (quadratic optimization problem)
  Constraints              : 0
  Affine conic cons.       : 0
  Disjunctive cons.        : 0
  Cones                    : 0
  Scalar variables         : 343
  Matrix variables         : 0
  Integer variables        : 343

Optimizer started.
```

# Practicability of a reformulation

- Performing a reformulation should be fast.
- Especially in a commercial solver setting, excessive reformulation times are prohibitive.
- **MOSEK** 10 has various work limits on the computational aspects of the different formulations.
- For example, $(R_\lambda)$ and $(R_\mu)$ are never attempted if matrix dimensions exceed certain thresholds.

# Practicability of a reformulation

- Performing a reformulation should be fast.
- Especially in a commercial solver setting, excessive reformulation times are prohibitive.
- **MOSEK** 10 has various work limits on the computational aspects of the different formulations.
- For example, $(R_\lambda)$ and $(R_\mu)$ are never attempted if matrix dimensions exceed certain thresholds.

# Practicability of solving (RSDP)

The computationally most expensive reformulation is $(R_S)$, even when imposing McCormick inequalities only on

$$L_q = \{(i,j) \mid \exists k : q_{ij}^k \neq 0\}.$$

- On the instance set from [Billionnet et al., 2016], we get the following reformulation times:

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S$ |
|---|---|---|---|---|---|
| time (sec.) | geo. mean | 0.0048 | 0.0030 | 0.0136 | 0.7254 |
| | shifted | 0.0066 | 0.0035 | 0.0188 | 1.9145 |

- On other test sets, it is sometimes out-of-reach to solve (RSDP) within hours of computational time.

- In practice we have to resort to some approximation.

# Practicability of solving (RSDP)

The computationally most expensive reformulation is $(R_S)$, even when imposing McCormick inequalities only on

$$L_q = \{(i,j) \mid \exists k : q_{ij}^k \neq 0\}.$$

- On the instance set from [Billionnet et al., 2016], we get the following reformulation times:

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S$ |
|---|---|---|---|---|---|
| time (sec.) | geo. mean | 0.0048 | 0.0030 | 0.0136 | 0.7254 |
| | shifted | 0.0066 | 0.0035 | 0.0188 | 1.9145 |

- On other test sets, it is sometimes out-of-reach to solve (RSDP) within hours of computational time.

- In practice we have to resort to some approximation.

# Practicability of solving (RSDP)

The computationally most expensive reformulation is ($R_S$), even when imposing McCormick inequalities only on

$$L_q = \{(i,j) \mid \exists k : q_{ij}^k \neq 0\}.$$

- On the instance set from [Billionnet et al., 2016], we get the following reformulation times:

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S$ |
|---|---|---|---|---|---|
| time (sec.) | geo. mean | 0.0048 | 0.0030 | 0.0136 | 0.7254 |
| | shifted | 0.0066 | 0.0035 | 0.0188 | 1.9145 |

- On other test sets, it is sometimes out-of-reach to solve (RSDP) within hours of computational time.

- In practice we have to resort to some approximation.

# Practicability of solving (RSDP) (cont.)

Solution: Separate McCormick inequalities in rounds, using some simple violation criteria.

- May result in a series of smaller SDPs being solved.
- Call this ($R_S^c$).

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S$ | $R_S^c$ |
|---|---|---|---|---|---|---|
| time (sec.) | geo. mean | 0.0048 | 0.0030 | 0.0136 | 0.7254 | 0.1013 |
| | shifted | 0.0066 | 0.0035 | 0.0188 | 1.9145 | 0.1601 |
| gap (%) | geo. mean | 225.84 | 43.19 | 21.00 | 15.34 | 11.24 |
| | shifted | 226.07 | 43.50 | 22.38 | 15.67 | 14.15 |

- Seems to work reasonably in practice.

# Practicability of solving (RSDP) (cont.)

Solution: Separate McCormick inequalities in rounds, using some simple violation criteria.

- May result in a series of smaller SDPs being solved.
- Call this ($R_S^c$).

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S$ | $R_S^c$ |
|---|---|---|---|---|---|---|
| time (sec.) | geo. mean | 0.0048 | 0.0030 | 0.0136 | 0.7254 | 0.1013 |
| | shifted | 0.0066 | 0.0035 | 0.0188 | 1.9145 | 0.1601 |
| gap (%) | geo. mean | 225.84 | 43.19 | 21.00 | 15.34 | 11.24 |
| | shifted | 226.07 | 43.50 | 22.38 | 15.67 | 14.15 |

- Seems to work reasonably in practice.

# Practicability of solving (RSDP) (cont.)

Solution: Separate McCormick inequalities in rounds, using some simple violation criteria.

- May result in a series of smaller SDPs being solved.
- Call this ($R_S^c$).

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S$ | $R_S^c$ |
|---|---|---|---|---|---|---|
| time (sec.) | geo. mean | 0.0048 | 0.0030 | 0.0136 | 0.7254 | 0.1013 |
| | shifted | 0.0066 | 0.0035 | 0.0188 | 1.9145 | 0.1601 |
| gap (%) | geo. mean | 225.84 | 43.19 | 21.00 | 15.34 | 11.24 |
| | shifted | 226.07 | 43.50 | 22.38 | 15.67 | 14.15 |

- Seems to work reasonably in practice.

# Practicability of solving (RSDP) (cont.)

Solution: Separate McCormick inequalities in rounds, using some simple violation criteria.

- May result in a series of smaller SDPs being solved.
- Call this $(R_S^c)$.

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S$ | $R_S^c$ |
|---|---|---|---|---|---|---|
| time (sec.) | geo. mean | 0.0048 | 0.0030 | 0.0136 | 0.7254 | 0.1013 |
| | shifted | 0.0066 | 0.0035 | 0.0188 | 1.9145 | 0.1601 |
| gap (%) | geo. mean | 225.84 | 43.19 | 21.00 | 15.34 | 11.24 |
| | shifted | 226.07 | 43.50 | 22.38 | 15.67 | 14.15 |

- Seems to work reasonably in practice.

# Computational experiments

- $(R_Q)$, $(R_\lambda)$, $(R_\mu)$ and $(R_S^c)$ have been implemented in **MOSEK** 10, user param

  `MSK_IPAR_MIO_QCQO_REFOMRULATION_METHOD`.

- Takes care of practicability aspects as above (work limits, ...).

- Note that everything is transformed to MISOCP form:

$$x^T Qx = \|Fx\|_2^2 \text{ with } Q = F^T F.$$

- Special interest:
  - Interplay between bound and performance?
  - SDP-based methods?

- All runs single-threaded, time limit 2h, solving to optimality.

# Computational experiments

- $(R_Q)$, $(R_\lambda)$, $(R_\mu)$ and $(R_S^c)$ have been implemented in **MOSEK** 10, user param

  `MSK_IPAR_MIO_QCQO_REFOMRULATION_METHOD`.

- Takes care of practicability aspects as above (work limits, ...).
- Note that everything is transformed to MISOCP form:

$$x^T Q x = \|Fx\|_2^2 \text{ with } Q = F^T F.$$

- Special interest:
  - Interplay between bound and performance?
  - SDP-based methods?
- All runs single-threaded, time limit 2h, solving to optimality.

# Computational experiments

- $(R_Q)$, $(R_\lambda)$, $(R_\mu)$ and $(R_S^c)$ have been implemented in **MOSEK** 10, user param

  `MSK_IPAR_MIO_QCQO_REFOMRULATION_METHOD`.

- Takes care of practicability aspects as above (work limits, ...).

- Note that everything is transformed to MISOCP form:

$$x^T Q x = \|Fx\|_2^2 \text{ with } Q = F^T F.$$

- Special interest:
  - Interplay between bound and performance?
  - SDP-based methods?

- All runs single-threaded, time limit 2h, solving to optimality.

# Computational experiments

- $(R_Q)$, $(R_\lambda)$, $(R_\mu)$ and $(R_S^c)$ have been implemented in **MOSEK** 10, user param

  MSK_IPAR_MIO_QCQO_REFOMRULATION_METHOD.

- Takes care of practicability aspects as above (work limits, ...).
- Note that everything is transformed to MISOCP form:

$$x^T Qx = \|Fx\|_2^2 \text{ with } Q = F^T F.$$

- Special interest:
  - Interplay between bound and performance?
  - SDP-based methods?
- All runs single-threaded, time limit 2h, solving to optimality.

# Instances from [Billionnet et al., 2016]

- 280 **non-convex** randomly generated MIQCQP instances
- no binary variables
- some have $m = 0$, some $m > 0$, but structure somewhat homogeneous
- reformulation times always below 1 sec.

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|
| solved (193) | | 104 | 134 | 174 | 188 | - | - |
| t (sec.) | geo. mean | 353.05 | 222.88 | 69.69 | 38.20 | 31.67 | 151.12 |
| | shifted | 512.73 | 362.5 | 136.82 | 79.67 | 70.35 | 275.73 |
| wins | | 21 | 11 | 74 | 127 | - | - |

- Best dual bound also wins: 173 / 193
- 😄

# Instances from [Billionnet et al., 2016]

- 280 **non-convex** randomly generated MIQCQP instances
- no binary variables
- some have $m = 0$, some $m > 0$, but structure somewhat homogeneous
- reformulation times always below 1 sec.

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|
| | solved (193) | 104 | 134 | 174 | 188 | - | - |
| t (sec.) | geo. mean | 353.05 | 222.88 | 69.69 | 38.20 | 31.67 | 151.12 |
| | shifted | 512.73 | 362.5 | 136.82 | 79.67 | 70.35 | 275.73 |
| | wins | 21 | 11 | 74 | 127 | - | - |

- Best dual bound also wins: 173 / 193
- 😄

# Instances from [Billionnet et al., 2016]

- 280 **non-convex** randomly generated MIQCQP instances
- no binary variables
- some have $m = 0$, some $m > 0$, but structure somewhat homogeneous
- reformulation times always below 1 sec.

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|
| | solved (193) | 104 | 134 | 174 | 188 | - | - |
| t (sec.) | geo. mean | 353.05 | 222.88 | 69.69 | 38.20 | 31.67 | 151.12 |
| | shifted | 512.73 | 362.5 | 136.82 | 79.67 | 70.35 | 275.73 |
| | wins | 21 | 11 | 74 | 127 | - | - |

- Best dual bound also wins: 173 / 193
- 😄

# BiqMac instances

- 340 **non-convex** binary quadratic and Max-cut instances
- pure BQPs
- reformulation times always below 5.5 sec.

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|
| | solved (286) | 212 | 167 | 230 | 272 | - | - |
| t (sec.) | geo. mean | 18.22 | 254.08 | 50.56 | 17.43 | 6.54 | 12.56 |
| | shifted | 76.37 | 499.51 | 115.69 | 47.28 | 22.43 | 50.14 |
| | wins | 202 | 74 | 110 | 176 | - | - |

- Best dual bound also wins: 250 / 286
- 😄

# BiqMac instances

- 340 **non-convex** binary quadratic and Max-cut instances
- pure BQPs
- reformulation times always below 5.5 sec.

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|
| | solved (286) | 212 | 167 | 230 | 272 | - | - |
| t (sec.) | geo. mean | 18.22 | 254.08 | 50.56 | 17.43 | 6.54 | 12.56 |
| | shifted | 76.37 | 499.51 | 115.69 | 47.28 | 22.43 | 50.14 |
| | wins | 202 | 74 | 110 | 176 | - | - |

- Best dual bound also wins: 250 / 286
- 😄

# BiqMac instances

- 340 **non-convex** binary quadratic and Max-cut instances
- pure BQPs
- reformulation times always below 5.5 sec.

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|
| | solved (286) | 212 | 167 | 230 | 272 | - | - |
| t (sec.) | geo. mean | 18.22 | 254.08 | 50.56 | 17.43 | 6.54 | 12.56 |
| | shifted | 76.37 | 499.51 | 115.69 | 47.28 | 22.43 | 50.14 |
| | wins | 202 | 74 | 110 | 176 | - | - |

- Best dual bound also wins: 250 / 286
- 😄

# QPLIB non-convex

- 288 **non-convex** instances, 143 of which allow for some reformulation, **94** of which allow for all reformulations
- $\Longrightarrow$ work limits of some method act on 49 instances
- quite heterogeneous, mostly binary and few general integer variables
- reformulation times always below 5 sec.
- Further divide the 94 instances **32** into "borderline convex" instances (all smallest eigenvalues $\approx 0$):

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|
| solved (31) | | 1 | 29 | 27 | 25 | - | - |
| t (sec.) | geo. mean | 7030.89 | 10.23 | 99.85 | 78.17 | 8.07 | 10.23 |
| | shifted | 7030.99 | 29.16 | 134.36 | 169.72 | 22.77 | 29.16 |
| wins | | 0 | 24 | 4 | 12 | - | - |

# QPLIB non-convex

- 288 **non-convex** instances, 143 of which allow for some reformulation, **94** of which allow for all reformulations
- $\implies$ work limits of some method act on 49 instances
- quite heterogeneous, mostly binary and few general integer variables
- reformulation times always below 5 sec.
- Further divide the 94 instances **32** into "borderline convex" instances (all smallest eigenvalues $\approx 0$):

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|
| | solved (31) | 1 | 29 | 27 | 25 | - | - |
| t (sec.) | geo. mean | 7030.89 | 10.23 | 99.85 | 78.17 | 8.07 | 10.23 |
| | shifted | 7030.99 | 29.16 | 134.36 | 169.72 | 22.77 | 29.16 |
| | wins | 0 | 24 | 4 | 12 | - | - |

# QPLIB non-convex

- 288 **non-convex** instances, 143 of which allow for some reformulation, **94** of which allow for all reformulations
- $\implies$ work limits of some method act on 49 instances
- quite heterogeneous, mostly binary and few general integer variables
- reformulation times always below 5 sec.
- Further divide the 94 instances **32** into “borderline convex” instances (all smallest eigenvalues $\approx 0$):

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|
| | solved (31) | 1 | 29 | 27 | 25 | - | - |
| t (sec.) | geo. mean | 7030.89 | 10.23 | 99.85 | 78.17 | 8.07 | 10.23 |
| | shifted | 7030.99 | 29.16 | 134.36 | 169.72 | 22.77 | 29.16 |
| | wins | 0 | 24 | 4 | 12 | - | - |

# QPLIB non-convex (cont.)

- ... and **62** "strictly non-convex" instances:

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|
| | solved (22) | 18 | 11 | 15 | 13 | - | - |
| t (sec.) | geo. mean | 1113.18 | 2375.68 | 1466.29 | 1318.42 | 544.82 | 819.48 |
| | shifted | 1146.72 | 2403.35 | 1495.98 | 1361.53 | 572.06 | 846.94 |
| | wins | 12 | 0 | 6 | 6 | - | - |

- Best dual bound also wins:
  - borderline convex: 25 / 31
  - strictly non-convex 12 / 22, 6 cases where $R_Q$ wins despite worse dual bound (technology shift!)
- ☺

# QPLIB non-convex (cont.)

- … and **62** "strictly non-convex" instances:

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|
| | solved (22) | 18 | 11 | 15 | 13 | - | - |
| t (sec.) | geo. mean | 1113.18 | 2375.68 | 1466.29 | 1318.42 | 544.82 | 819.48 |
| | shifted | 1146.72 | 2403.35 | 1495.98 | 1361.53 | 572.06 | 846.94 |
| | wins | 12 | 0 | 6 | 6 | - | - |

- Best dual bound also wins:
  - borderline convex: 25 / 31
  - strictly non-convex 12 / 22, 6 cases where $R_Q$ wins despite worse dual bound (technology shift!)
- ☺

# QPLIB non-convex (cont.)

- ... and **62** "strictly non-convex" instances:

| | | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|
| | solved (22) | 18 | 11 | 15 | 13 | - | - |
| t (sec.) | geo. mean | 1113.18 | 2375.68 | 1466.29 | 1318.42 | 544.82 | 819.48 |
| | shifted | 1146.72 | 2403.35 | 1495.98 | 1361.53 | 572.06 | 846.94 |
| | wins | 12 | 0 | 6 | 6 | - | - |

- Best dual bound also wins:
  - borderline convex: 25 / 31
  - strictly non-convex 12 / 22, 6 cases where $R_Q$ wins despite worse dual bound (technology shift!)
- ☺

# Binary least squares

- 51 **convex** instances with random data
- pure BQPs
- reformulation times always below 0.1 sec.

| | | no reform. | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|---|
| | solved (51) | 43 | 23 | 47 | 51 | 50 | - | - |
| t (sec.) | geo. mean | 81.47 | 385.19 | 33.74 | 18.51 | 23.66 | 18.12 | 33.74 |
| | shifted | 146.65 | 588.86 | 80.70 | 47.73 | 56.19 | 46.58 | 80.70 |
| | wins | 1 | 0 | 11 | 46 | 19 | - | - |

- Best dual bound also wins: 19 / 51
- ☺

# Binary least squares

- 51 **convex** instances with random data
- pure BQPs
- reformulation times always below 0.1 sec.

| | | no reform. | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|---|
| | solved (51) | 43 | 23 | 47 | 51 | 50 | - | - |
| t (sec.) | geo. mean | 81.47 | 385.19 | 33.74 | 18.51 | 23.66 | 18.12 | 33.74 |
| | shifted | 146.65 | 588.86 | 80.70 | 47.73 | 56.19 | 46.58 | 80.70 |
| | wins | 1 | 0 | 11 | 46 | 19 | - | - |

- Best dual bound also wins: 19 / 51
- ☺

# Binary least squares

- 51 **convex** instances with random data
- pure BQPs
- reformulation times always below 0.1 sec.

| | | no reform. | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|---|
| | solved (51) | 43 | 23 | 47 | 51 | 50 | - | - |
| t (sec.) | geo. mean | 81.47 | 385.19 | 33.74 | 18.51 | 23.66 | 18.12 | 33.74 |
| | shifted | 146.65 | 588.86 | 80.70 | 47.73 | 56.19 | 46.58 | 80.70 |
| | wins | 1 | 0 | 11 | 46 | 19 | - | - |

- Best dual bound also wins: 19 / 51
- ☺

# Public **convex** instances

- 28 convex instances from public sources (QPLIB, Google groups, ...)
- quite heterogeneous
- reformulation times always below 4 sec.
- 7 borderline convex instances (use $R_\lambda$!), 21 remaining:

| | | no reform. | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|---|
| | solved (18) | 7 | 14 | 7 | 13 | 11 | - | - |
| t (sec.) | geo. mean | 609.63 | 201.87 | 438.35 | 125.9 | 282.4 | 48.08 | 134.13 |
| | shifted | 953.32 | 424.08 | 752.64 | 244.39 | 550.94 | 115.94 | 286.29 |
| | wins | 0 | 6 | 1 | 7 | 3 | - | - |

- Best dual bound also wins: 16 / 18, 2 cases where $R_Q$ wins despite worse dual bound
- 😄

# Public **convex** instances

- 28 convex instances from public sources (QPLIB, Google groups, ...)
- quite heterogeneous
- reformulation times always below 4 sec.
- 7 borderline convex instances (use $R_\lambda$!), 21 remaining:

| | | no reform. | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|---|
| | solved (18) | 7 | 14 | 7 | 13 | 11 | - | - |
| t (sec.) | geo. mean | 609.63 | 201.87 | 438.35 | 125.9 | 282.4 | 48.08 | 134.13 |
| | shifted | 953.32 | 424.08 | 752.64 | 244.39 | 550.94 | 115.94 | 286.29 |
| | wins | 0 | 6 | 1 | 7 | 3 | - | - |

- Best dual bound also wins: 16 / 18, 2 cases where $R_Q$ wins despite worse dual bound
- 😄

# Public **convex** instances

- 28 convex instances from public sources (QPLIB, Google groups, ...)
- quite heterogeneous
- reformulation times always below 4 sec.
- 7 borderline convex instances (use $R_\lambda$!), 21 remaining:

| | | no reform. | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | no-sdp virt. best |
|---|---|---|---|---|---|---|---|---|
| | solved (18) | 7 | 14 | 7 | 13 | 11 | - | - |
| t (sec.) | geo. mean | 609.63 | 201.87 | 438.35 | 125.9 | 282.4 | 48.08 | 134.13 |
| | shifted | 953.32 | 424.08 | 752.64 | 244.39 | 550.94 | 115.94 | 286.29 |
| | wins | 0 | 6 | 1 | 7 | 3 | - | - |

- Best dual bound also wins: 16 / 18, 2 cases where $R_Q$ wins despite worse dual bound
- 😄

# What we have learned

- Which method works good/best depends on the problem class, **but also on the data**!
- On borderline convex models use $R_\lambda$ (i.e., a numerical perturbation).
- Reformulations also interesting for already convex models.
- An SDP-based method can be the method of choice.
- A good dual bound is important, but not only.

# What we have learned

- Which method works good/best depends on the problem class, **but also on the data**!
- On borderline convex models use $R_\lambda$ (i.e., a numerical perturbation).
- Reformulations also interesting for already convex models.
- An SDP-based method can be the method of choice.
- A good dual bound is important, but not only.

# What we have learned

- Which method works good/best depends on the problem class, **but also on the data**!
- On borderline convex models use $R_\lambda$ (i.e., a numerical perturbation).
- Reformulations also interesting for already convex models.
- An SDP-based method can be the method of choice.
- A good dual bound is important, but not only.

# What we have learned

- Which method works good/best depends on the problem class, **but also on the data**!
- On borderline convex models use $R_\lambda$ (i.e., a numerical perturbation).
- Reformulations also interesting for already convex models.
- An SDP-based method can be the method of choice.
- A good dual bound is important, but not only.

# Automatically choosing a reformulation?

- The best method for a given application may be established experimentally.

- **MOSEK** 10 also has some heuristic for automatically choosing a reformulation:

| | no reform. | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | heur. |
|---|---|---|---|---|---|---|---|
| [Billionnet et al., 2016] | - | 512.73 | 362.5 | 136.82 | 79.67 | 70.35 | 79.61 |
| BiqMac | - | 76.37 | 499.51 | 115.69 | 47.28 | 22.43 | 96.8 |
| QPLIB non-convex | - | 1146.72 | 2403.35 | 1495.98 | 1361.53 | 572.06 | 1000.2 |
| BLS | 146.65 | 588.86 | 80.70 | 47.73 | 56.19 | 46.58 | 47.83 |
| Public convex | 953.32 | 424.08 | 752.64 | 244.39 | 550.94 | 115.94 | 218.33 |

- ... but a more sophisticated method for choosing is desirable, see also [Bonami et al., 2022].

# Automatically choosing a reformulation?

- The best method for a given application may be established experimentally.

- **MOSEK** 10 also has some heuristic for automatically choosing a reformulation:

| | no reform. | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | heur. |
|---|---|---|---|---|---|---|---|
| [Billionnet et al., 2016] | - | 512.73 | 362.5 | 136.82 | 79.67 | 70.35 | 79.61 |
| BiqMac | - | 76.37 | 499.51 | 115.69 | 47.28 | 22.43 | 96.8 |
| QPLIB non-convex | - | 1146.72 | 2403.35 | 1495.98 | 1361.53 | 572.06 | 1000.2 |
| BLS | 146.65 | 588.86 | 80.70 | 47.73 | 56.19 | 46.58 | 47.83 |
| Public convex | 953.32 | 424.08 | 752.64 | 244.39 | 550.94 | 115.94 | 218.33 |

- ... but a more sophisticated method for choosing is desirable, see also [Bonami et al., 2022].

# Automatically choosing a reformulation?

- The best method for a given application may be established experimentally.

- **MOSEK** 10 also has some heuristic for automatically choosing a reformulation:

| | no reform. | $R_Q$ | $R_\lambda$ | $R_\mu$ | $R_S^c$ | virt. best | heur. |
|---|---|---|---|---|---|---|---|
| [Billionnet et al., 2016] | - | 512.73 | 362.5 | 136.82 | 79.67 | 70.35 | 79.61 |
| BiqMac | - | 76.37 | 499.51 | 115.69 | 47.28 | 22.43 | 96.8 |
| QPLIB non-convex | - | 1146.72 | 2403.35 | 1495.98 | 1361.53 | 572.06 | 1000.2 |
| BLS | 146.65 | 588.86 | 80.70 | 47.73 | 56.19 | 46.58 | 47.83 |
| Public convex | 953.32 | 424.08 | 752.64 | 244.39 | 550.94 | 115.94 | 218.33 |

- ... but a more sophisticated method for choosing is desirable, see also [Bonami et al., 2022].

# References I

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

[Bonami et al., 2022] Bonami, P., Lodi, A., and Zarpellon, G. (2022).
A classifier to decide on the linearization of mixed-integer quadratic problems in cplex.
*Operations Research*.

[Dong and Lou, 2018] Dong, H. and Lou, Y. (2018).
Compact disjunctive approximations to nonconvex quadratically constrained programs.
Technical report.

# References II

[Furini and Traversi, 2019] Furini, F. and Traversi, E. (2019). Theoretical and computational study of several linearisation techniques for binary quadratic problems. *Annals of Operations Research*, 279(1):387–411.

[Glover and Wolsey, 1974] Glover, F. and Wolsey, L. (1974). Converting the 0-1 polynomial programming problem to a 0-1 linear program. *Oper. Res.*, 22(1):180–182.

[Hammer and Rubin, 1970] Hammer, P. L. and Rubin, A. (1970). Some remarks on quadratic programming with 0-1 variables. *RAIRO*, 3:67–79.