A piecewise-quadratic convexification for exactly solving box-constrained quadratic programs

Amélie Lambert[1] and Daniel Porumbel[1]

[1]CEDRIC-CNAM, 292 rue saint Martin, 75141 Paris Cedex 03

Abstract This work aims at finding the global minimum of a quadratic function f with box constrained variables. For this goal, we construct a piecewise-quadratic convex relaxation of f defined as the maximum of $k \geq 1$ convex functions. We show that when $k \to \infty$ the optimal solution of this relaxation converges to an optimal solution of the SDP relaxation of the initial problem. The resulting convexification is tighter that the one produced by previous related methods that use $k = 1$. Its integration into a spatial branch-and-bound algorithm brings a second advantage : compared to previous related methods, it can refine the bound at each node by computing new convex functions specifically tailored to act on the considered node. Numerical results suggest that our algorithm can outperform different solvers or other methods based on quadratic convex relaxations.

Keywords Quadratic Programming, piecewise-quadratic underestimator, cutting-quadrics

1 Introduction

We consider a non-convex Quadratic Box-constrained Program (QBP) of the form :

$$(QBP) \begin{cases} \min f(x) \equiv \langle Q, xx^\top \rangle + c^\top x \\ \qquad\qquad \ell_i \leq x_i \leq u_i \quad \forall i \in \mathcal{I}, \end{cases} \quad (1)$$

where $\langle A, B \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{ij}$, $\mathcal{I} = \{1, \ldots, n\}$, $(Q, c, \ell, u) \in \mathcal{S}_n \times \mathbb{R}^n \times \mathbb{R}^n_+ \times \mathbb{R}^n$, and $\mathcal{S}_n$ is the set of symmetric matrices of order n. Without loss of generality, we assume that the box constraints (1) take the form $x_i \in [0, 1]$. (QBP) is a fundamental $\mathcal{NP}$-hard global optimization problem [13]. The most classical methods for solving it include branch-and-bound algorithms that use various convex relaxations to determine lower bounds, *e.g.*, either linear, quadratic convex or semi-definite relaxations (see [9, 7]). For many of these cases, the quadratic function can be expressed in an extended space of variables, introducing new variables Y_{ij} with $i, j \in \mathcal{I}$ that are meant to satisfy $Y_{ij} = x_i x_j$. This traditional approach was first used for the linear relaxation, leading to a branch-and-bound based on the relaxation of the non-convex equalities $Y_{ij} = x_i x_j$ (see for instance [15, 19, 21]).

We also express the problem in this extended (x, Y) space, but we go beyond the linear relaxation. We propose the following model which is equivalent to (QBP) for any semidefinite positive (SDP) matrix $S_0 \succeq 0$.

$$\min \langle S_0, xx^\top \rangle + c^\top x + \langle Q - S_0, Y \rangle \quad (2a)$$
$$Y = xx^\top \quad (2b)$$
$$\ell_i \leq x_i \leq u_i \qquad i \in \mathcal{I} \quad (2c)$$
$$Y \in \mathcal{S}_n \quad (2d)$$

A well-known SDP relaxation of (QBP) is obtained by lifting x to a symmetric matrix $X = xx^T$ and by relaxing the non-convex constraints $X = xx^T$ into $X - xx^T \succeq 0$. After linking X and x using the McCormick constraints (3a)-(3d), we obtain the model below, also called the "Shor's plus RLT" relaxation of (QBP) [1].

$$(SDP) \begin{cases} \min f(X, x) \equiv \langle Q, X \rangle + c^T x \\ \quad X_{ij} \leq u_j x_i + \ell_i x_j - \ell_i u_j, \ \ i, j \in \mathcal{I} \ \ (3a) \\ \quad X_{ij} \leq \ell_j x_i + u_i x_j - u_i \ell_j, \ \ i, j \in \mathcal{I} \ \ (3b) \\ \quad X_{ij} \geq u_j x_i + u_i x_j - u_i u_j, \ \ i, j \in \mathcal{I} \ \ (3c) \\ \quad X_{ij} \geq \ell_j x_i + \ell_i x_j - \ell_i \ell_j, \ \ i, j \in \mathcal{I} \ \ (3d) \\ \quad \begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0 \quad (3e) \\ \quad x \in \mathbb{R}^n \quad X \in \mathcal{S}_n \quad (3f) \end{cases}$$

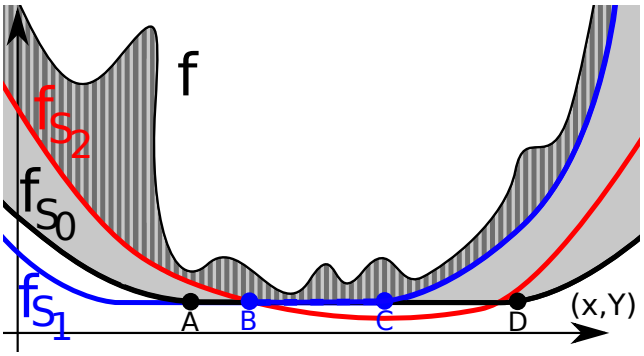

FIGURE 1 – Convex function f_{S_0} in black may reach its minimum over many solutions of the (x, Y) space (see the long flat segment [A,D]), because S_0 may have a large null space, often of dimension close to $\frac{n}{2}$. The resulting convexification in light-gray is weaker than the piecewise-quadratic convexification from the striped area (max of f_{S_0}, f_{S_1} and f_{S_2}) that has the same minimum; this latter convex function reaches its minimum over a segment [B,C] shorter than [A,D].

By solving the above (SDP) in a preprocessing step, the method `MIQCR` [5] determines a positive semidefinite matrix S_0 that maximizes the lower bound associated to the following convex relaxation of (2a)–(2d) : minimize $f_{S_0}(x, Y)$ over all (x, Y) that satisfy the McCormick constraints [15] (instead of $Y = xx^T$), where f_{S_0} is the quadratic surface (quadric) given by :

$$f_S(x, Y) = \langle S, xx^\top \rangle + c^\top x + \langle Q - S, Y \rangle \qquad (4)$$

The main idea of our work is to replace this unique function f_{S_0} with multiple functions f_{S_k}, with $k = 0, 1, 2, \ldots$; we then minimize the function $f^*(x, Y) = \max_k f_{S_k}(x, Y)$ over all (x, Y). Since each function f_{S_k} is quadratic and convex, f^* is a piecewise-quadratic convex understimator (see Figure 1). We generate the functions f_k one by one as in a cutting-planes approach; each function f_k aims at cutting the current optimal solution (x, Y) by making it sub-optimal.

Figure 1 shows how this approach based on multiple quadratic cuts ($k > 1$) may generate a tighter convexification than `MIQCR` ($k = 1$), even if it does not improve the bound computed at the root node of the branch-and-bound. The new algorithm can also provide a better bound than `MIQCR` at any non-root node, because it can integrate cuts specifically generated to penalize solutions associated to any given node. Notice that it would have been impractical for `MIQCR` to solve (SDP) at each node.

The paper is organized as follows. Section 2 introduces a parameterized family of relaxations of (QBP) that are piecewise-quadratic and convex; we will show that the optimal value of this relaxation reaches the value of (SDP). Section 3 introduces an iterative `Cutting Quadrics Algorithm (CQA)` that can be seen as an extension of the cutting-planes algorithm that uses quadratic surfaces (or quadrics) instead of hyperplanes. In Section 4, this algorithm is integrated within a branch-and-bound to solve (QBP) to global optimality. Finally, Section 5 presents experimental results on the *boxqp* instances, suggesting that our new approach is faster than state-of-the-art solvers, and is able to significantly reduce the number of nodes in comparison to the basic `MIQCR`.

2 A family of convex piecewise-quadratic relaxations

Given a set $\mathcal{K} = \{S_k \succeq \mathbf{0}, k = 0, 1, \ldots, p\}$ of SDP matrices, the multi-cut version of (2a)–(2d) takes the form below, forming a family of equivalent formulations of (QBP) indexed by $\mathcal{K}$:

$$(P_\mathcal{K}) \begin{cases} \min t & \\ \quad t \geq \langle S_k, xx^\top \rangle + c^\top x + \langle Q - S_k, Y \rangle, \ S_k \in \mathcal{K} & \text{(5a)} \\ \quad Y = xx^\top & \text{(5b)} \\ \quad \ell_i \leq x_i \leq u_i \qquad i \in \mathcal{I} & \text{(5c)} \\ \quad Y \in \mathcal{S}_n, \ t \in \mathbb{R} & \text{(5d)} \end{cases}$$

Like in the mono-cut version (2a)–(2d) of the above program, the only non-convexity of $(P_\mathcal{K})$ comes from Constraints (5b) that we can classically relax using the McCormick envelopes (6c)–(6f). We obtain $(\overline{P}_\mathcal{K})$ a family of convex relaxations of (QBP) indexed by a set $\mathcal{K}$ as above :

$$(\overline{P}_\mathcal{K}) \begin{cases} \min t & \text{(6a)} \\ \quad t \geq \langle S_k, xx^\top \rangle + c^\top x + \langle Q - S_k, Y \rangle, \ S_k \in \mathcal{K} & \text{(6b)} \\ \quad Y_{ij} \leq u_j x_i + \ell_i x_j - \ell_i u_j, \qquad i, j \in \mathcal{I} & \text{(6c)} \\ \quad Y_{ij} \leq \ell_j x_i + u_i x_j - u_i \ell_j, \qquad i, j \in \mathcal{I} & \text{(6d)} \\ \quad Y_{ij} \geq u_j x_i + u_i x_j - u_i u_j, \qquad i, j \in \mathcal{I} & \text{(6e)} \\ \quad Y_{ij} \geq \ell_j x_i + \ell_i x_j - \ell_i \ell_j, \qquad i, j \in \mathcal{I} & \text{(6f)} \\ \quad \ell_i \leq x_i \leq u_i, \qquad i \in \mathcal{I} & \text{(6g)} \\ \quad Y \in \mathcal{S}_n, \ t \in \mathbb{R} & \text{(6h)} \end{cases}$$

Clearly, for any set $\mathcal{K}$, the problem $(\overline{P}_\mathcal{K})$ is a relaxation of (QBP), since for any solution $\bar{x}$ of (QBP) of value t, the solution $(\bar{x}, \bar{x}\bar{x}^\top, t)$ is feasible for $(\overline{P}_\mathcal{K})$ with a value of t. Moreover, since

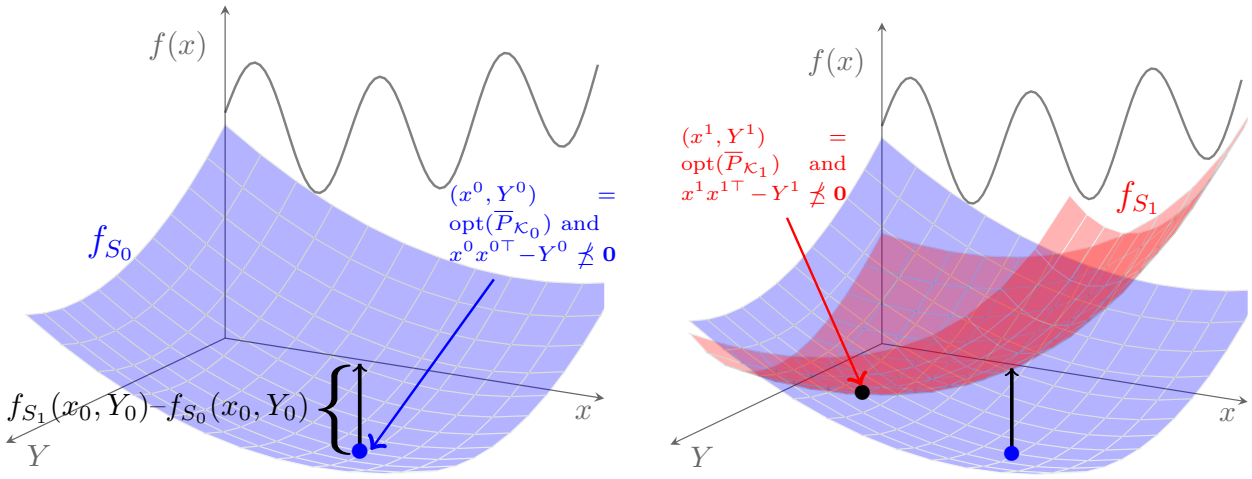

FIGURE 2 – The `Cutting Quadrics Algorithm (CQA)` starts from an initial convex function f_{S_0} (blue surface) of problem $(\overline{P}_{\mathcal{K}_0})$ whose optimal solution is (x^0, Y^0), see the small blue disk. Then it generates matrix S_1 to impose penalty $f_{S_1}(x^0, Y^0)$ on (x^0, Y^0), as marked by the black vertical arrow in both figures. Thus, after adding function f_{S_1} (red surface) to $(\overline{P}_{\mathcal{K}_0})$, the optimal solution of $(\overline{P}_{\mathcal{K}_1})$ moves to (x^1, Y^1).

all matrices $S_k \in \mathcal{K}$ are SDP, each constraint (6b) define a quadratic convex set, and so, $(\overline{P}_{\mathcal{K}})$ is a convex problem. Now, given an integer p, we pose the problem (LB_p) of determining the best set of matrices $\mathcal{K} = \{S_0, S_1, \dots S_p\}$, in the sense of leading to the tightest lower bound of (QBP) :

$$(LB_p) \left\{ \max_{S_0, S_1, \dots S_k \succeq \mathbf{0}} v\left(\overline{P}_{\{S_0, S_1, \dots S_k\}}\right) \right.$$

This connects our work with previous convex relaxations [4, 5, 11]; when we restrict $(\overline{P_{\mathcal{K}}})$ to $\mathcal{K} = \{S_0\}$, we obtain the original `MIQCR` method from [5]. It is proven in [11], that, the optimal solution of (LB_1) can be deduced from the optimal dual solution of (SDP). It is, moreover, proven in [11] that, if strong duality holds for (SDP), the optimal value of (LB_1) equals the optimal value of (SDP). We now state Proposition 1

Proposition 1 $v(LB_\infty) = v(LB_p) = \dots = v(LB_0) = v(SDP)$.

Proof. The last equality (for $\mathcal{K} = \{S_0\}$), follows from [11, Theorem 3.1]. Moreover, by construction, we obviously have $v(LB_\infty) \geq v(LB_p) \geq \dots \geq v(LB_1) = v(SDP)$. We now show $v(LB_\infty) \leq v(SDP)$. Take a feasible solution $(\overline{X}, \overline{x})$ of (SDP) of objective value $f(\overline{X}, \overline{x}) = \langle Q, \overline{X} \rangle + c^T \overline{x}$. The solution $(Y = \overline{X}, x = \overline{x})$ is feasible for $(\overline{P_{\mathcal{K}}})$ with a value of $\max_{k \in \mathcal{K}} \langle S_k, \overline{x}\overline{x}^\top - \overline{X} \rangle + c^\top \overline{x} + \langle Q, \overline{X} \rangle \leq f(\overline{X}, \overline{x})$; this was obtained from $\langle S_k, \overline{x}\overline{x}^\top - \overline{X} \rangle \leq 0$, which is true for any $S_k \succeq 0$ given that $\overline{x}\overline{x}^\top - \overline{X} \preceq 0$ by virtue of (3e). □

3 The cutting-quadrics algorithm

The `Cutting Quadrics Algorithm (CQA)` starts from an initial relaxation $(\overline{P}_{\mathcal{K}_0})$ associated to an initial set $\mathcal{K}_0$ of SDP matrices (that may contain the matrix determined by `MIQCR`). At each iteration k, `CQA` calls a convex solver to determine the optimal solution (x^k, Y^k, t^k) of $(\overline{P}_{\mathcal{K}_k})$. It then constructs a matrix $S_{k+1} \succeq 0$ such that $f_{S_{k+1}}(x^k, Y^k) > t^k$ making current solution (x^k, Y^k, t^k) infeasible for $(\overline{P}_{\mathcal{K}_{k+1}})$, i.e., for the new program (6a)-(6h) enriched with $f_{S_{k+1}}$. The next iteration has to move to a new solution, as illustrated in Figure 2. We thus generalize the well-known cutting-planes algorithm by generating convex quadratic cuts instead of separating hyperplanes.

We define $S_{k+1} = r^k \cdot v_{max} v_{max}^\top$, where v_{max} is the eigenvector of $(x^k x^{k\top} - Y^k)$ of maximum eigenvalue. In practice, we can also define $S_{k+1} = r^k \sum v_i v_i^T$, where the sum is carried out over all eigenvectors v_i having a positive eigenvalue.

The overall solution method is summed up in Algorithm 1, and we prove its convergence in Theorem 1. Note that we start for set $\mathcal{K}_0$ that will at least integrate the null matrix $\mathbf{0_n}$, that corresponds to the standard linearization of $f(x)$.

Algorithm 1: The Cutting-Quadrics Algorithm

Input : variable bounds ℓ and u, precision parameters ϵ and δ, (optional) initial matrices $\mathcal{K}_0$
Output: The best lower bound on (LB_∞)

$\mathcal{K}_0 \leftarrow \mathcal{K}_0 \cup \{\mathbf{0_n}\}$ // $\mathcal{K}_0$ may contain the optimal matrix used by MIQCR
$(x^0, Y^0, t^0) \leftarrow \text{Solve}(\overline{P}_{\mathcal{K}_0})$ // variable t^0 is actually the optimum obj. value
$k \leftarrow 0$
while $(x^k x^{k\top} - Y^k \not\preceq 0)$ // In practice we use $\lambda_{\max}(x^k x^{k\top} - Y^k) > \delta$
do
  $r^k = 100 + k$ // A penalty parameter; we keep $r^k = 100$ in practice
  $v_{max} \leftarrow$the eigenvector of $x_k^* x_k^{*\top} - Y_k^*$ of maximum eigenvalue
  $S_{k+1} \leftarrow r^k \cdot v_{max} v_{max}^\top$ // Or $S_{k+1} = r \sum v_i v_i^T$, where the sum is carried out over all the eigenvectors v_i associated to a positive eigenvalue
  $\mathcal{K}_{k+1} = \mathcal{K}_k \cup S_{k+1}$ // add a new quadratic cut associated to S_{k+1}
  $(x^{k+1}, Y^{k+1}, t^{k+1}) \leftarrow \text{Solve}(\overline{P}_{\mathcal{K}_{k+1}})$
  $k \leftarrow k + 1$
  if $t^k - t^{k-1} < \epsilon$ **then**
    break // Bound progress too small (only in practice)

return t^k as the optimal solution of (QBP) if $Y^k = x^k x^{k\top}$, or as a lower bound otherwise

Theorem 1 *When $k \to \infty$, the value of the solutions t^k generated by Algorithm 1 converge to the optimal value of (SDP).*

Proof. We use three steps : *i*) the new quadric added at iteration $k+1$ separates the current optimal solution (x^k, Y^k, t^k) of iteration k, which implies that the sequence of optimal values t^k increases monotonically, *ii*) the maximum eigenvalue of $x^k x^{k\top} - Y^k$ converges to 0, and *iii*) t^k reaches the optimal value of (SDP) when $k \to \infty$.
(i) Let (x^k, Y^k, t^k) the optimal solution to $(\overline{P}_{\mathcal{K}_k})$ of iteration k of objective value t^k. We first prove that the new quadric $f_{S_{k+1}}$ cuts (x^k, Y^k, t^k), or equivalently that $f_{S_{k+1}}(x^k, Y^k) > f_{S_k}(x^k, Y^k) = t^k$. Let v be the eigenvector of $(x^k x^{k\top} - Y^k)$ of maximum eigenvalue λ_{max}. By definition, we have $S_{k+1} = r^{k+1} \cdot vv^\top$ and we obtain :

$$\begin{aligned}
&f_{S_{k+1}}(x^k, Y^k) - f_{S_k}(x^k, Y^k) \\
&= \langle S_{k+1}, x^k x^{k\top} \rangle + c^\top x^k + \langle Q - S_{k+1}, Y^k \rangle \\
&\quad - \langle S_k, x^k x^{k\top} \rangle - c^\top x^k - \langle Q - S_k, Y^k \rangle \\
&= \langle S_{k+1} - S_k, x^k x^{k\top} - Y^k \rangle \\
&= r^{k+1} \lambda_{max} - \langle S_k, x^k x^{k\top} - Y^k \rangle \\
&\geq r^{k+1} \lambda_{max} - r^k \lambda_{max},
\end{aligned}$$

where for the last inequality we used that S_k has the form $S_k = r^k \cdot uu^\top$ (where u is an eigenvector of $x^{k-1} x^{k-1\top} - Y^{k-1}$) coupled with the following well-known property [17, §3.2] :

$$\lambda_{max} = \max_{u \in \mathbb{R}^n || u || = 1} \langle uu^\top, x^k x^{k\top} - Y^k \rangle$$

This proves $f_{S_{k+1}}(x^k, Y^k) > f_{S_k}(x^k, Y^k)$, because $r^{k+1} > r^k$ as imposed by the first line of the **while** loop.
(ii) We will now prove that for any $\delta > 0$ no matter how small, there exist some $\bar{k}$ such that $\lambda_{max}\left(x^k x^{k\top} - Y^k\right) < \delta$ for any $k \geq \bar{k}$. Suppose for the sake of contradiction that this is not the case. This means that the algorithm generates an infinite number of matrices $(x^k x^{k\top} - Y^k) \in \mathcal{S}_n$ that all satisfy $\lambda_{max}\left(x^k x^{k\top} - Y^k\right) \geq \delta$. The Bolzano-Weierstrass theorem states that any infinite sequence in a bounded set (recall that (x, Y) belong to the unit hypercube) contains a convergent subsequence. This means that there exists a subsequence (k_i) such that $(x^{k_i} x^{k_i\top} - Y^{k_i})$ converges to some fixed point $\overline{xx}^\top - \overline{Y}$ when $i \to \infty$. By assumption, this convergence point satisfies $\lambda_{max}\left(\overline{xx}^\top - \overline{Y}\right) \geq \delta$. For any infinitesimal ϵ, there exists some k_ϵ such that for any $k_i > k_\epsilon$, any element in matrix $(x^{k_i} x^{k_i\top} - Y^{k_i})$ is at a distance smaller than ϵ from the corresponding element of $(\overline{xx}^\top - \overline{Y})$, *i.e.*, it stays in a neighborhood N_ϵ of $(\overline{xx}^\top - \overline{Y})$ of ∞-norm below ϵ. *But here comes the contradiction.* For a sufficiently small ϵ, we have $\lambda_{max}(xx^\top - Y) > \delta' > 0 \; \forall (x, Y) \in N_\epsilon$ for some δ' arbitrarily close to δ. Thus, for a sufficiently large

$\hat{k}$ and $r^{\hat{k}} = 100 + k$, $f_{S_{\hat{k}}}(x, Y)$ includes a penalty of $r^{\hat{k}}\delta'$ that is large enough to make any $(x, Y) \in N_\epsilon$ sub-optimal. No iteration after $\hat{k}$ can stay in N_ϵ.
(iii) We now prove that $t^k \to v(SDP)$.

Let us first take any convergence point $(\overline{x}, \overline{Y}, \overline{t})$ of Algorithm 1 that has to satisfy $(\overline{x}\overline{x}^\top - \overline{Y}) \preceq 0$ and use it to build a feasible solution of (SDP) such that $\overline{t} \geq v(SDP)$. Since $(\overline{x}\overline{x}^\top - \overline{Y}) \preceq 0$, the solution $(x = \overline{x}, X = \overline{Y})$ is feasible for (SDP) with a value of $c^\top \overline{x} + \langle Q, \overline{Y}\rangle \geq v(SDP)$. The optimal value $\overline{t}$ can be written as below (regardless of how exactly $\mathcal{K}$ was constructed) :

$$\begin{aligned}\overline{t} =& \max_{S\in\mathcal{K}} \langle S, \overline{x}\overline{x}^\top - \overline{Y}\rangle + c^\top \overline{x} + \langle Q, \overline{Y}\rangle \\ =& c^\top \overline{x} + \langle Q, \overline{Y}\rangle \geq v(SDP)\end{aligned}$$

We used $\max_{S\in\mathcal{K}} \langle S, \overline{x}\overline{x}^\top - \overline{Y}\rangle = \langle \mathbf{0_n}, \overline{x}\overline{x}^\top - \overline{Y}\rangle = 0$ which is true because $\mathbf{0} \in \mathcal{K}$ and because $\langle S, \overline{x}\overline{x}^\top - \overline{Y}\rangle \leq 0$ for any SDP matrix $S \in \mathcal{K}$.

This proves that $\overline{t} \geq v(SDP)$ and we now explain why we can not have $\overline{t} > v(SDP)$. Take the optimal solution (x, X) of (SDP) and notice that its value in (6a)-(6f) is

$$\begin{aligned}&\max_{S\in\mathcal{K}} \langle S, xx^\top - X\rangle + c^\top x + \langle Q, X\rangle \\ =&\max_{S\in\mathcal{K}} \langle S, xx^\top - X\rangle + v(SDP) \leq v(SDP),\end{aligned}$$

where we simply used $\langle S, xx^\top - X\rangle \leq 0$, which is true by virtue of $S \succeq \mathbf{0}$ and $xx^\top - X \preceq \mathbf{0}$. □

The proof of Theorem 1 implies the following Corollary.

Corollary 1 *Algorithm 1 solves problem (SDP). In particular, if $(\overline{x}, \overline{Y}, \overline{t})$ is a convergence point of Algorithm 1, $(\overline{Y}, \overline{x})$ is an optimal solution to (SDP) with a value of $\overline{t} = v(SDP)$.*

Theorem 1 thus state that Algorithm 1 generates the same lower bound as `MIQCR`, *i.e.*, v(SDP). However, as mentioned previously, the additional quadrics tighten the convexification by reducing the set of optimal equivalent solutions (like in Figure 2). This increases the potential of the new convexification when integrated in a branch-and-bound algorithm.

Recall Algorithm 1 can start from any set $\mathcal{K}_0$. To improve its convergence, we describe hereafter several ways to populate $\mathcal{K}_0$. The first one consists in adding to $\mathcal{K}_0$ the optimal matrix calculated by `MIQCR`, which is $S_0 = Q + \Phi^1 + \Phi^2 - \Phi^3 - \Phi^4$, where Φ^1, Φ^2, Φ^3, Φ^4 are the symmetric matrices built from the optimal dual variables associated with Constraints (3a)–(3d) of (SDP). This approach has the drawback of requiring solving a large SDP problem. A faster way is the following : take the matrix Q and increase its diagonal by the opposite of the smallest eigenvalue of Q, *i.e.*, $S_0 = Q - \lambda_{\min} \cdot I_n \succeq 0$. Another choice relies on extracting the convex part of Q, by constructing the matrix $S_0 = \sum \lambda_i v_i v_i^T$ where the sum is carried over all non-negative eigenvalues λ_i of Q associated to eigenvectors v_i. Preliminary tests show that the latter approach is still very fast and produces better results than the second one. Note moreover that the size of $\mathcal{K}_0$ is not limited, so it is possible to include all these matrices in the initial set $\mathcal{K}_0$.

4 A branch-and-bound algorithm to optimally solve (QBP)

A classical way to solve (QBP) is to use a branch-and-bound algorithm (see [3] for a complete description) where the bounding step is based on `CQA`. We here provide more details on our implementation.

The variable selection strategy
Let $\epsilon_b > 0$ be the precision parameter that state if an equality is satisfied, and (x^k, Y^k) be the solution of $(\overline{P}_{\mathcal{K}_k})$ at the current node, two cases are possible :

1. If $-\epsilon_b \leq \left(Y^k - x^k x^{k\top}\right)_{ij} \leq \epsilon_b$ for all $i, j \in I$, then (x, Y) is the optimal solution of the considered branch.
2. Else, we first try to determine i^* by solving problem below restricted to the diagonal (to $i^* = j^*$). If this fails, we solve the same problem below with no restriction on i^* and j^* :

$$(i^*, j^*) = \text{argmax}|(x_i x_j - Y_{ij})|$$

The branching rules
The feasible interval $[\ell_{i^*}, u_{i^*}]$ of the selected variable x_{i^*} is always split in half : the two child nodes are $x_{i^*} \in \left[\ell_{i^*}, \frac{\ell_{i^*}+u_{i^*}}{2}\right]$ and $x_{i^*} \in \left[\frac{\ell_{i^*}+u_{i^*}}{2}, u_{i^*}\right]$.

The node selection strategy
For selecting the next sub-problem we use the

"best-first" selection strategy, *i.e.*, we select the node with the highest evaluated lower bound.

Upper bounding
We use a rather basic coordinate descent heuristic to determine upper bounds. We start from the solution x^k of $(\overline{P}_{\mathcal{K}_k})$ at the current node (ignoring all Y^k components), that is also feasible for the original (QBP). Our coordinate descent begins by iterating over all $i \in [1..n]$; for each i, we fix all variables to their current values except for x_i^k. We then solve an optimization sub-problem with only one decision variable, namely x_i^k. After scanning all variables once, we can repeat the process as long as there is at least one variable i for which we detect a possible improvement.

Lower bounding
At each node of the B&B, we evaluate $(\overline{P}_{\mathcal{K}_k})$ from scratch, i.e. without inheriting any quadratic cuts produced by `CQA` at previous nodes. In fact, we only keep the matrices of set $\mathcal{K}_0 \cup \mathbf{0_n}$ (first step of Algorithm 1). We stop executing `CQA` on a given node as soon as the progress of the lower bound $t^{k+1} - t^k$ becomes smaller than a fixed value δ. We will evaluate two algorithm versions :

`CQBB-1` use a value of $\delta = 0.01$,

`CQBB-2` for this version, we actually force at maximum one iteration per node.

Regarding `CQBB-2`, notice that to generate a first S_1 we need a first solution (x^0, Y^0). To accelerate the node evaluation, we proposed a *presolving initial iteration* to generate an initial solution (x^0, Y^0) without fully solving $(\overline{P}_{\mathcal{K}_0})$. When we first evaluate a new node, we fix all values of x and Y to that of the optimal value of the parent node, with the exception of the variables that no longer satisfy the new McCormiks envelopes. We thus evaluate the resulting simplified problem with fewer variables more rapidly than fully solving $(\overline{P}_{\mathcal{K}_0})$.

5 Numerical results

We evaluate our global optimization algorithm `CQBB` on 81 purely continuous quadratic instances with box constraints called *boxqp* [8, 20]. The sizes of the instances vary from $n = 20$ to 90 and the densities of matrix Q from 20%, to 100%. An instance with n variables, a density d%, and whose instance number with the same characteristics is k is named *spar-n-d-k*.

Experimental environment :
Our experiments were carried out on a server with 2 CPU Intel Xeon of 2.3 GHz, each of them having 32 thread, and $8 * 16$ GB of RAM with a Linux operating system. We will compare to the original `MIQCR` method [11] and to the standard solvers `Cplex 12.9` [14], `Baron 21.1.13` [18], and `Gurobi 9.1.1` [16].

For both `CQBB-1` and `CQBB-2`, we start with the initial set $\mathcal{K}_0 = \{S_0\}$, where S_0 is obtained by solving (SDP) heuristically by calling the solver `Mosek` [2] together with the Conic Bundle library [12] within a lagrangian duality framework as described in [6]. We only use $r^k = 100$ to compute all S_k with $k \geq 1$, and we solve the convex relaxations $(\overline{P}_{\mathcal{K}_k})$ at each `CQA` iteration using the C interface to the solver `Mosek`. Finally, we set the total time limit to 1 hour for all methods and the relative optimality gap of the branch-and-bound to $\epsilon_b = 10^{-4}$. We define $S_{k+1} = r^k \sum v_i v_i^T$, summing over all eigenvectors v_i associated to an eigenvalue no smaller than 0.01.

To compare the performances of the solvers, we use a performance profile of the CPU times (see [10]). The basic idea is the following : for each instance i and each solver s, we denote by t_{is} the time for solving instance i by solver s, and we define the *performance ratio* as $r_{is} = \frac{t_{is}}{\min\limits_s t_{is}}$. Let N be the total number of instances considered, an overall assessment of the performance of solver s for a given ρ is given by $P(r_{is} \leq \rho) = \frac{1}{N} *$ number of instances i such that $r_{is} \leq \rho$. In Figure 3, we present the performance profile of the CPU times for all the compared algorithms for the *boxqp* instances. Our new approach `CQBB-2` outperforms the original `MIQCR` method and compares well with other solvers in terms of the number of instances solved. In fact, `Cplex` solves 73 instances, `Baron` 74 solves instances, `CQBB-1` solves 77 instances, `MIQCR` and `Gurobi` solve 78 instances, and `CQBB-2` solves 80 instances, out of 81 within the time limit of 1 hour.

If the performance profile starts for $\rho = 1$ with a smaller value $P(r_{is} \leq \rho)$ for `CQBB-2` than for the commercial solvers, this is because the smallest instances are solved more rapidly by `Gurobi`

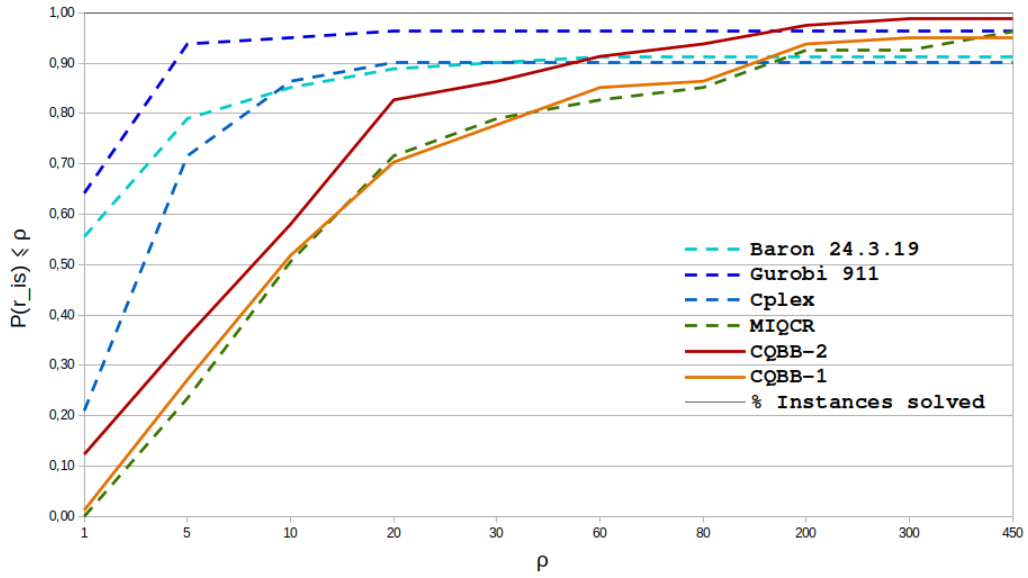

FIGURE 3 – Performance profile of the total CPU time for the *boxqp* instances with $n = 20$ to 90 with a time limit of 1 hour.

or `Cplex`. However, the focus of our algorithm is on the largest and denser instances; and with regards to them, `CQBB-2` dominates the compared methods, by solving more instances in the long run.

Table 1 reports the total number of generated nodes for the *boxqp* instances with $n \in [30..90]$ with a time limit of 1 hour; we also removed the easier instances that require less than 30 nodes for all methods. The number of nodes is significantly reduced for method `CQBB-2` in comparison to `MIQCR` (by a factor of 1.4 on average), and is further reduced with method `CQBB-1`, i.e. when we allow more `CQA` iterations per node.

6 Conclusions

We have presented a generic approach to solve box-constrained quadratic programs to global optimality. The main idea is to combine the strength of quadratic convex relaxations with the cutting-planes logic. Indeed, instead of considering a unique under-estimator of the objective function, we propose a family of convexifications indexed by an arbitrary number of quadrics.

We have proposed an original algorithm to generate these quadrics one by one. The quadric generated at each iteration is actually a quadratic cut that separates the current optimal solution, acting similarly to a hyper-plane of the well known cutting-planes method. We proved that this iterative algorithm converges to the optimal value of a tight semidefinite relaxation of (QBP).

To solve (QBP) to global optimality, we have implemented a spatial branch-and-bound algorithm that relies on the bound computed by `CQA` at each node. Numerical results suggest that the resulting method outperforms standard solvers on the largest instances. Since `CQA` can refine the bound at each node of the branching tree, we almost systematically reduce the total number of B&B needed nodes compared to `MIQCR`.

Many improvements are still possible to make the method reach its full potential, both in theory or in practice. For instance, from a practical point of view, one idea would be to aggregate the matrices of the set $\mathcal{K}$ at each iteration of the algorithm, to solve more rapidly the convex quadratic relaxation. The method can quite easily extend to integrate binary variables, linear or convex constraints, or to minimize the maximum of several non-convex functions (a problem of a different nature, beyond the power of `MIQCR`).

References

[1] K. M. Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 43(2) :471–484, 2009.

Instance	MIQCR	CQBB 2	CQBB-1
Spar-030-060-1	43	31	25
Spar-030-070-1	85	71	51
Spar-030-070-3	39	31	9
Spar-030-080-1	53	69	65
Spar-040-040-1	511	271	231
Spar-040-050-1	31	25	25
Spar-040-050-2	45	35	33
Spar-040-060-1	299	227	181
Spar-040-100-3	247	211	181
Spar-050-030-3	37	11	11
Spar-050-040-2	35	17	17
Spar-050-050-2	79	67	61
Spar-050-050-3	103	133	127
Spar-060-020-3	65	33	33
Spar-070-025-1	87	37	37
Spar-070-025-2	225	127	125
Spar-070-025-3	243	143	117
Spar-070-050-1	197	155	147
Spar-070-075-1	75	63	57
Spar-070-075-2	1901	1551	1223
Spar-070-075-3	877	779	727
Spar-080-025-1	255	425	425
Spar-080-025-2	857	471	463
Spar-080-025-3	171	149	123
Spar-080-050-2	47	77	77
Spar-080-050-3	293	203	203
Spar-080-075-1	123	107	105
Spar-080-075-2	353	335	315
Spar-080-075-3	1221	1031	953
Spar-090-025-1	3375	1707	1699
Spar-090-025-2	3229	1561	1547
Spar-090-025-3	675	405	419
Spar-090-050-1	1655	1473	1383
Spar-090-050-3	531	423	429
Spar-090-075-3	1319	1103	981

TABLE 1 – Number of nodes per instance for the *boxqp* instances ($n = 30$ to 90).

[2] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.2.*, 2019.

[3] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22 :1–131, 2013.

[4] 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.

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

[6] 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.

[7] P. Bonami, O. Günlük, and J. Linderoth. Globally solving nonconvex quadratic programming problems with box constraints via integer programming methods. *Mathematical Programming Computation*, 10(3) :333–382, 2018.

[8] S. Burer and D. Vandenbussche. Globally solving box-constrained nonconvex quadratic programs with semidefinite-based finite branch-and-bound. *Comput Optim Appl*, 43 :181–195, 2009.

[9] J. Chen and S. Burer. Globally solving nonconvex quadratic programming problems via completely positive programming. *Mathematical Programming Computation*, 4(1) :33–52, 2012.

[10] D. Dolan and J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91 :201–213, 1986.

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

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

[13] R. Horst, P. M. Pardalos, and N. V. Thoai. Introduction to global optimization. *Kluwer, Dordrecht*, 2000.

[14] IBM. IBM CPLEX 12.9 Reference Manual. "`https://www.ibm.com/docs/en/icos/12.9.0?topic=cplex-callable-library-c-api-reference-manual`", 2020.

[15] G.P. McCormick. Computability of global solutions to factorable non-convex programs : Part i - convex underestimating problems. *Mathematical Programming*, 10(1) :147–175, 1976.

[16] GUROBI optimization. *GUROBI 9.1*, 2021.

[17] Daniel Porumbel. Demystifying the characterization of sdp matrices in mathematical programming, 2022. arXiv prepint nr. 2210.13072.

[18] N.V. Sahinidis and M. Tawarmalani. Baron 9.0.4 : Global optimization of mixed-integer nonlinear programs. *User's Manual*, 2010.

[19] H.D. Sherali and W.P. Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, volume 31. Springer Science & Business Media, 2013.

[20] D. Vandenbussche and G. Nemhauser. A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Mathematical Programming*, 102(3) :259–275, 2005.

[21] Y. Yajima and T. Fujie. A polyhedral approach for nonconvex quadratic programming problems with box constraints. *Journal of Global Optimization*, 13(2) :151–170, 1998.