Global solution of non-convex quadratically constrained quadratic programs

Sourour Elloumi, Amélie Lambert

Global solution of non-convex quadratically constrained quadratic programs

Sourour Elloumi[1] and Amélie Lambert[2]

1 CEDRIC-ENSTA, 828 Boulevard des Maréchaux, 91120 Palaiseau, France
{sourour.elloumi}@ensta-paristech.fr
2 CEDRIC-CNAM, 292 rue saint Martin, F-75141 Paris Cedex 03, France
amelie.lambert@cnam.fr

Abstract. The class of mixed-integer quadratically constrained quadratic programs (QCQP) consists of minimizing a quadratic function under quadratic constraints where the variables could be integer or continuous. On a previous paper we introduced a method called `MIQCR` for solving QC-QPs with the following restriction : all quadratic sub-functions of purely continuous variables are already convex. In this paper, we propose an extension of `MIQCR` which applies to any QCQP. Let (P) be a QCQP. Our approach to solve (P) is first to build an equivalent mixed-integer quadratic problem (P^*). This equivalent problem (P^*) has a quadratic convex objective function, linear constraints, and additional variables y that are meant to satisfy the additional quadratic constraints $y = xx^T$, where x are the initial variables of problem (P). We then propose to solve (P^*) by a branch-and-bound algorithm based on the relaxation of the additional quadratic constraints and of the integrality constraints. This type of branching is known as spatial branch-and-bound. Computational experiences are carried out on a total of 325 instances. The results show that the solution time of most of the considered instances is improved by our method in comparison with the recent implementation of `QuadProgBB`, and with the solvers `Cplex`, `Couenne`, `Scip`, `BARON` and `GloMIQO`.

Key words: General mixed-integer quadratic programming, Global optimization, Spatial branch-and-bound, Quadratic convex relaxation, Experiments

1 Introduction

We consider the problem of optimizing a quadratic function subject to quadratic and bound constraints:

$$(P)\begin{cases}\min f_0(x) \equiv \langle Q_0, xx^T\rangle + c_0^T x & \\ \text{s.t.} & \\ \quad f_r(x) \equiv \langle Q_r, xx^T\rangle + c_r^T x \le b_r & r = 1,\dots,m \\ \quad \ell \le x \le u & \\ \quad x_i \in \mathbb{N} & \forall i \in J \\ \quad x_i \in \mathbb{R} & \forall i \in I\backslash J\end{cases}$$

with $\langle A, B\rangle = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}b_{ij}$, and where $I = \{1,\dots,n\}$, $J \subset I$, $\forall\ r = 0,\dots,m$ $(Q_r, c_r) \in \mathcal{S}_n \times \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $u \in \mathbb{R}^n$. Without loss of generality we suppose that $l \in \mathbb{R}^n_+$. We denote by $\mathcal{S}_n$ the set of $n\times n$ symmetric matrices, by $\mathcal{S}_n^+$ the set of positive semidefinite matrices of $\mathcal{S}_n$ and $M \succeq 0$ means that $M \in \mathcal{S}_n^+$. We also denote by $\mathbf{0}_n$ the zero $n \times n$ matrix and by I^2 the cartesian product of a set I by itself.

We assume the feasible domain of (P) to be non-empty. Problem (P) trivially contains the case where there are quadratic equalities, since an equality can be replaced by two inequalities. It also contains the case of linear constraints since a linear equality is a quadratic constraint with a zero quadratic part.

Problem (P) is a Mixed-Integer Quadratically Constrained Program (MIQCP) and belongs to the class of $\mathcal{NP}$-hard problems [37]. It arises in many application areas such as graph theory [38], market prices computation [44], or the pooling problem introduced by Haverly [7, 40]. Several applications of the box-constrained case, namely, when $J = \emptyset$ and the only constraints are $\ell \le x \le u$ were mentioned by Moré and Toraldo in [49]. Moreover, (P) generalizes several difficult problems such as binary programming, fractional programming, or polynomial programming. This is due to the fact that these problems can be reformulated Redinto a MIQCP. Binary quadratic programming has itself a large set of applications. We refer the reader to recent surveys on more general mixed integer non linearproblems [25].

Exact solution methods for solving (P) generally use the well-known branch-and-bound algorithm. Key operations of this algorithm are called bounding and branching. Bounding often relies on the computation of a lower bound (for a minimization problem) by solving a relaxed problem of the initial one. There are several bounding strategies. Among others, three strategies are classically used: approximation of the convex envelope of each quadratic term [45], outer approximation of convex terms [21, 30, 31], or computation of a bound thanks to semidefinite relaxations [6, 52]. Branching strategies depend on the nature of the variables. For integer variables, branching is commonly done by recursively dividing the solution set into two subsets in such a way that a current fractional solution is discarded. This represents a hope to improve the bounds further obtained by the relaxation in the two subsets. For continuous variables, branching is done by considering a variable x_i whose current interval is $[\ell_i, u_i]$, choosing some value $\bar{x}_i$ in $]\ell_i, u_i[$, and dividing the solution set into two subsets, one with interval $[\ell_i, \bar{x}_i]$ and the other with interval $[\bar{x}_i, u_i]$ for variable x_i. Doing this does not directly discard undesired points but may change the structure of the relaxed problem. This branching is called spatial branch-and-bound (sBB) in the global optimization literature and is due to Falk and Soland [32]. Finally, a sBB algorithm can be improved by computing feasible local solutions.

Several versions of the sBB algorithm were proposed [9, 55, 56, 54, 61, 62]. Three classical relaxations can be used to perform a sBB. The first and most common version uses a linear relaxation of (P) tightened by cuts such as the reformulation-linearization technique (RLT) [53]. Some softwares, implementing the methods described above, are available for solving (P). See for instance `Couenne 0.5` [11], `Baron` [51], `Scip 3.2.1` [1, 58]. Another frequently used technique to obtain convex relaxations of (P) is semi-definite programming, see for example [5, 52]. The last way is based on a non-linear convex relaxation and is called αBB [2–4, 24, 33–35, 59, 60]. It is implemented into software `GloMIQO 2` [46–48]

`Cplex 12.6.2` [20, 22, 42] and the recent solver `QuadProgBB` [27] also handle (P), but are limited to the case where all constraints are linear. The solver `QuadprogBB` uses a finite branch-and-bound scheme, in which branching is based on the first-order KKT conditions. Moreover, polyhedral-semidefinite relaxations are solved at each node of the branch-and-bound tree. In particular, their relaxations are derived from completely positive and doubly nonnegative programs. In the last version of `Cplex 12.6.2`, a sBB algorithm is performed based on two different quadratic and convex relaxations of (P) built as follows: in the first one, (P) is reformulated as a separable problem where the diagonal Hessian matrix is chosen positive semi-definite, in the second one, a convex reformulation is built based on the factorized eigenvector space. Moreover, `Cplex 12.6.2` exploits the performances of Cplex for binary quadratic programs to solve box-constrained problems [22].

In this paper, we present a new sBB algorithm for solving (P) to ϵ-global optimality. In previous papers [15–17], we introduced quadratic convex reformulation methods to solve (P) with the following restriction : all quadratic sub-functions of purely continuous variables are already convex. More precisely, let $f(a, b)$ be a quadratic function of integer variables a, and of continuous variables b. Function $f(a, b)$ is a sum of monomials, and can be viewed as the sum of 2 functions: $f^1(a, b)$ that is composed of the mixed-integer monomials, and $f^2(b)$ that corresponds to the monomials with only continuous variables. We call $f^2(b)$ the sub-function of purely continuous variables. Our contribution in this paper is to handle the general quadratic case without any restriction over the pure-continuous sub-functions. One originality of our approach is that it is based on an equivalent quadratic reformulation of (P) which is computed thanks to semi-definite programming, in a pre-processing step. Then, a quadratic convex relaxation of the equivalent formulation is used as a bounding strategy within a sBB. To evaluate our algorithm, we present computational experiences on 325 instances coming from benchmarks of the literature. We compare our algorithm with the recent implementation of `QuadProgBB`, and with the solvers `Cplex 12.6.2` [42], `Couenne 0.5` [11], `Scip 3.2.1` [58], `BARON 17.3.31` and `GloMIQO 2` [48]. We show on performance profiles that for these instances, our method outperforms the others for many large and high density instances.

The outline of the paper is the following. In Section 2, we review quadratic convex reformulation methods and situate the current paper compared to our previous ones. In Section 3, we present our new quadratic reformulation of (P) and we show that this reformulation is an improvement of the complete linearization standardly used within a sBB. Then, in Section 4, we describe the main features of our sBB based on the relaxation of the quadratic constraints $y_{ij} = x_i x_j$ and of the integrality constraints. In Section 5, we present our computational results. Section 6 draws a conclusion.

2 Review of Quadratic Convex Reformulations

Problem (P) presents two difficulties: non-convexity of functions f_r and the integrality of some of its variables. It is qualified as convex when all functions $f_0, \ldots, f_m$ are convex [50]. When (P) is convex and there are no integer variables $(J = \emptyset)$, we have a convex problem that can be solved in polynomial time. Thanks to the later property, convex problems with integer variables can be solved by a branch-and-bound based on continuous relaxation, as frequently done in quadratic convex solvers. Without these convexity assumptions, problem (P), even with only continuous variables, is significantly harder to solve.

In this paper, we present an algorithm based on a reformulation. By this, we mean an algorithm that works in two phases. In the first phase, an equivalent formulation to the initial problem is designed. The key idea is that one relaxation of the equivalent formulation is a convex problem. Finally, the equivalent formulation can be solved to ϵ-global optimality by a spatial branch-and-bound process based on this convex relaxation. This is the second phase of the algorithm.

In the presence of integer variables, many reformulation methods rely on the reformulation of a quadratic program into a MILP. This is called linearization. A very well-known linearization for quadratic programs with binary variables is due to Fortet [36] and consists in replacing any product of two binary variables x_i and x_j by an additional variable y_{ij}, together with a set of linear constraints enforcing the equality $y_{ij} = x_i x_j$. The idea of linearization can be extended to quadratic programs with general bounded integer variables with a tricky binary expansion of the general integers [14]. The obtained MILP is then solved by branch-and-bound or branch-and-cut. Linearization is also used for continuous variables. An equivalent problem is built where the only non-linearity is in the equality $y_{ij} = x_i x_j$. This equality is then relaxed in each node and enforced by a spatial branch-and-bound.

Another alternative to linearization consists in reformulating a non-convex quadratic program with binary variables into a convex quadratic program with the same variables. This idea appeared in Hammer and Rubin [39] where the authors use the equality $x_i^2 = x_i$ which holds for any binary variable x_i, and the smallest (resp. largest) eigenvalue in order to shift the diagonal terms of the Hessian matrix of the objective function and obtain an equivalent convex (resp. concave) function. Another simple quadratic convex reformulation was proposed in [12]. This reformulation, also based on $x_i^2 = x_i$, transforms every non-convex product $x_i x_j$ into the convex function $\frac{1}{2}((x_i + x_j)^2 - x_i - x_j)$ or to the concave function $\frac{1}{2}(-(x_i - x_j)^2 + x_i + x_j)$. These simple quadratic convex reformulations, while being very easy to implement, often lead to inefficient methods because their continuous relaxation bound may be very poor.

Considering the problem of minimizing a quadratic function of binary variables, Billionnet and Elloumi [13] introduced the question of how to change the diagonal terms of the Hessian matrix in such a way that (i) the obtained function is convex, and (ii) the continuous relaxation bound of this function is as tight as possible. The authors prove that, in this regard, "optimal" diagonal terms can be obtained from the dual solution of a semidefinite programming relaxation of the initial problem. With these optimal diagonal terms, the continuous relaxation bound is equal to the semidefinite programming relaxation bound. The reformulation idea and the solution method were then extended in [19] to the case of binary quadratic programs with linear equalities and the acronym `QCR` for Quadratic Convex Reformulation was born. Here, the SDP relaxation whose solution gives the best reformulation contains the so-called RLT constraints, obtained by multiplying the linear inequalities by the variables in order to get stronger relaxations.

Further extentions were designed in Billionnet et al. [15] for the case of bounded integer variables but still with linear constraints. In this case, a binary expansion of the initial variables is performed, together with the addition of new variables y_{ij} that represent the product of two general integer variables x_i and x_j. These additional variables allow to widen the family of potential reformulations since any perturbation of each element of the Hessian matrix is now considered. In this extension, a quadratic program with linear constraints and bounded integer variables is considered. The reformulation phase is based on a stronger SDP relaxation, sometimes called "Shor+RLT" in the literature [6]. The reformulated problem is a convex quadratic problem with continuous and binary variables, which is again solved by a quadratic convex programming solver. We present an extension to the case where the initial quadratic problem contains continuous variables in Billionnet et al. [15]. But, we had the following restriction: in the objective function, all quadratic sub-functions of purely continuous variables are already convex. This extension was called `MIQCR` (Mixed Integer Quadratic Convex Reformulation).

Another extension is obtained in [17] to the case of programs with quadratic inequalities, with the same kind of restriction on the quadratic sub-functions of purely continuous variables. The quadratic convex reformulation is presented in a new setting which includes linearization as a particular case. More precisely, the equivalent problem has additional variables y_{ij}, additional quadratic constraints $y_{ij} = x_i x_j$, a convex objective function and a set of valid inequalities.

These quadratic constraints are linearized by the addition of a large number of variables (binary expansion) and constraints. The obtained equivalent problem is a mixed-integer quadratic program with a convex objective function and linear constraints. It can be solved by a mixed-integer quadratic convex solver. However, in the presence of convex purely continuous sub-functions, the "Shor+RLT" relaxation cannot yet be used for computing the reformulation, hence, in this case we use a weaker SDP relaxation than in the current paper.

For the linearly constrained case, we introduced in [16], a specific branch-and-bound algorithm to solve the equivalent formulation. Constraints $y_{ij} = x_i x_j$ are not linearized but are rather enforced within a branch-and-bound process based on the relaxation of these quadratic constraints only, i.e. we keep the integrality constraints.

In this paper, contrarily to the reformulation proposed in [15, 17], and following the ideas of [16], we keep in our reformulation the non-convex quadratic constraints $y_{ij} = x_i x_j$. To solve the reformulated problem, we design a sBB based on the relaxation of constraints $y_{ij} = x_i x_j$ and of the integrality constraints. In contrast to [17], not only we obtain convex relaxations of smaller size, but we can also handle general quadratic problems.

3 A quadratic reformulation of (P)

We consider any set of positive semi-definite matrices $S_0, \ldots, S_m$. We also lift the problem to a higher space by introducing variables y_{ij} that are meant to satisfy $y_{ij} = x_i x_j$. We build the following equivalent quadratic formulation to (P):

$$(P_{S_0,\ldots,S_m}) \begin{cases} \min \quad f_{0,S_0}(x,Y) \equiv \langle S_0, xx^T \rangle + c_0^T x + \langle Q_0 - S_0, Y \rangle \\ \text{s.t.} \\ f_{r,S_r}(x,Y) \equiv \langle S_r, xx^T \rangle + c_r^T x + \langle Q_r - S_r, Y \rangle \le b_r & r = 1, \ldots, m \quad (1) \\ y_{ij} \le u_j x_i + l_i x_j - u_j l_i & (i,j) \in I^2,\ i \le j \quad (2) \\ y_{ij} \le u_i x_j + l_j x_i - u_i l_j & (i,j) \in I^2,\ i \le j \quad (3) \\ y_{ij} \ge u_j x_i + u_i x_j - u_i u_j & (i,j) \in I^2,\ i \le j \quad (4) \\ y_{ij} \ge l_j x_i + l_i x_j - l_i l_j & (i,j) \in I^2,\ i \le j \quad (5) \\ y_{ii} \ge x_i & i \in J \quad (6) \\ y_{ij} = y_{ji} & (i,j) \in I^2,\ i < j \quad (7) \\ y_{ij} = x_i x_j & (i,j) \in I^2 \quad (8) \\ x_i \in \mathbb{N} & \forall i \in J \quad (9) \end{cases}$$

To build $(P_{S_0,\ldots,S_m})$, we introduce n^2 new variables Y to model the products $x_i x_j$ (Constraints (8)). Then, we formulate $f_r(x)$ as a sum of a quadratic function of the x variables and a linear function of the Y variables. It holds that $f_{r,S_r}(x,Y)$ is equal to $f_r(x)$ if $y_{ij} = x_i x_j$. We also add the well-known McCormick inequalities (2) - (5) [45], and Constraints (6) that come from $x_i^2 \ge x_i$, a valid inequality for general integer variables to tighten the relaxation. Because matrices $S_0, \ldots S_m$ are positive semidefinite, the reformulated problem $(P_{S_0,\ldots,S_m})$

has the property that when Constraints (8) and (9) are relaxed, it is a convex problem. We call this convex relaxation $(\overline{P}_{S_0,\dots,S_m})$.

We then consider the problem of finding the best set of positive semi-definite matrices $S_0,\dots,S_m$, in the sense that the optimal solution value of $(\overline{P}_{S_0,\dots,S_m})$ is as large as possible. This amounts to solving the following problem (OPT_S):

$$(OPT_S)\left\{\max_{S_0,\dots,S_m\succeq 0} v(\overline{P}_{S_0,\dots,S_m})\right.$$

where $v(P)$ is the optimal value of problem (P). The following theorem shows that $v(OPT_S)$ is equal to the optimal value of an SDP program which is a semi-definite relaxation of (P).

Theorem 1. *Let (SDP) be the following semi-definite program:*

$$(SDP)\left\{\begin{array}{lll}
\min f(X,x) = \langle Q_0, X\rangle + c_0^T x & & \\
s.t. & & \\
\quad \langle Q_r, X\rangle + c_r^T x \le b_r & r = \{1,\dots,m\} & (10)\\
\quad X_{ij} - u_j x_i - l_i x_j + u_j l_i \le 0 & (i,j)\in I^2,\ i\le j & (11)\\
\quad X_{ij} - u_i x_j - l_j x_i + u_i l_j \le 0 & (i,j)\in I^2,\ i\le j & (12)\\
\quad -X_{ij} + u_j x_i + u_i x_j - u_i u_j \le 0 & (i,j)\in I^2,\ i\le j & (13)\\
\quad -X_{ij} + l_j x_i + l_i x_j - l_i l_j \le 0 & (i,j)\in I^2,\ i\le j & (14)\\
\quad -X_{ii} + x_i \le 0 & i\in J & (15)\\
\quad \begin{pmatrix} 1 & x \\ x^T & X\end{pmatrix} \succeq 0 & & (16)\\
\quad x\in\mathbb{R}^n \quad X\in\mathcal{S}_n & & (17)
\end{array}\right.$$

It holds that $v(OPT_S) = v(SDP)$. Besides, the following positive semi-definite matrices allow to build an optimal solution $(S_0^*,\dots,S_m^*)$ of (OPT_S):

i) $\forall r = 1,\dots,m,\ S_r^* = \mathbf{0}_n$

ii) $S_0^* = Q_0 + \sum_{r=1}^{m} \alpha_r^* Q_r + \Phi^*$, *where α^* is the vector of optimal dual variables associated with Constraints (10). Matrix Φ^* is computed as:*

$\diamond\ \Phi_{ii}^* = \Phi_{ii}^{1*} + \Phi_{ii}^{2*} - \Phi_{ii}^{3*} - \Phi_{ii}^{4*} - \varphi_i^*,$
$\diamond\ \Phi_{ij}^* = \Phi_{ij}^{1*} + \Phi_{ij}^{2*} - \Phi_{ij}^{3*} - \Phi_{ij}^{4*},$

where Φ^{1*}, Φ^{2*}, Φ^{3*}, Φ^{4*} are the symmetric matrices built from the optimal dual variables associated with Constraints (11)–(14), and φ^* is the vector of dual variables associated with Constraints (15).

Proof. This proof is similar to Proof of Theorem 1 in [17], but we now consider the case where the lower bounds ℓ_i over the variables x can be any non negative reals, while in [17], ℓ_i was considered as 0. To prove Theorem 1, we show that $v(OPT_S) = v(SDP)$ by showing that $v(OPT_S) \leq v(SDP)$ and then $v(OPT_S) \geq v(SDP)$.

$\diamond$ To prove that $v(OPT_S) \leq v(SDP)$, we show that $v(\overline{P}_{\bar{S}_0,\ldots,\bar{S}_m}) \leq v(SDP)$ for any $\bar{S}_0,\ldots,\bar{S}_m \in \mathcal{S}_n^+$, which in turn implies that $v(OPT_S) \leq v(SDP)$ since the right hand side is constant. For this, we show that if $(\bar{x},\bar{X})$ is feasible for (SDP), then $(x,Y) := (\bar{x},\bar{X})$ is *i)* feasible for $(\overline{P}_{\bar{S}_0,\ldots,\bar{S}_m})$ and *ii)* its objective value is less or equal than $v(SDP)$. Since $(\overline{P}_{\bar{S}_0,\ldots,\bar{S}_m})$ is a minimization problem, $v(\overline{P}_{\bar{S}_0,\ldots,\bar{S}_m}) \leq v(SDP)$ follows.

i) We prove that (x,Y) is feasible to $(\overline{P}_{\bar{S}_0,\ldots,\bar{S}_m})$. Constraints(2)-(7) are obviously satisfied from Constraints (11)–(15) and (17). We now prove that Constraints (1) are satisfied:

$$\begin{aligned}\langle \bar{S}_r, xx^T\rangle + c_r^T x + \langle Q_r - \bar{S}_r, Y\rangle &= \langle \bar{S}_r, \bar{x}\bar{x}^T\rangle + c_r^T\bar{x} + \langle Q_r - \bar{S}_r, \bar{X}\rangle \\ &= \langle \bar{S}_r, \bar{x}\bar{x}^T - \bar{X}\rangle + c_r^T\bar{x} + \langle Q_r, \bar{X}\rangle \\ &\leq b_r \qquad \text{from Constraints (10), since} \\ &\qquad\qquad \bar{S}_r \succeq 0\text{, and Constraint (16).}\end{aligned}$$

ii) Let us compare the objective values. For this, we prove that $\langle \bar{S}_0, \bar{x}\bar{x}^T\rangle + c_0^T\bar{x} + \langle Q_0 - \bar{S}_0, \bar{X}\rangle - \langle Q_0, \bar{X}\rangle - c_0^T\bar{x} \leq 0$ or that $\langle \bar{S}_0, \bar{x}\bar{x}^T - \bar{X}\rangle \leq 0$. This last inequality follows from $\bar{S}_0 \succeq 0$ and Constraint (16).

$\diamond$ Let us secondly prove that $v(OPT_S) \geq v(SDP)$ or equivalently $v(OPT_S) \geq v(DSDP)$ where $(DSDP)$ is the dual of (SDP). The following problem $(DSDP)$ is the dual of (SDP):

$$(DSDP)\begin{cases}\max g(\alpha,\varPhi) = -\sum\limits_{r=1}^{m}\alpha_r b_r + \langle \varPhi^1+\varPhi^2, ul^T\rangle - \langle \varPhi^3, uu^T\rangle - \langle \varPhi^4, ll^T\rangle \\ \text{s.t.} \\ Q_0 + \sum\limits_{r=1}^{m}\alpha_r Q_r + \varPhi \succeq 0 & (18) \\ c_0 + \sum\limits_{r=1}^{m}\alpha_r c_r - (\varPhi^1+\varPhi^2-2\varPhi^3)^T u - (\varPhi^1+\varPhi^2-2\varPhi^4)^T l + \varphi \geq 0 & (19) \\ \varPhi = \varPhi^1+\varPhi^2-\varPhi^3-\varPhi^4 - diag(\varphi) & (20) \\ \alpha\in\mathbb{R}_+^m,\ \varPhi\in\mathcal{S}_n,\ \varPhi^1,\varPhi^2,\varPhi^3,\varPhi^4\in\mathcal{S}_n,\ \varPhi_{ij}^r \geq 0\,\forall(r,i,j),\ \varphi\in\mathbb{R}_+^n\end{cases}$$

where $\alpha\in\mathbb{R}_+^m$ are the dual variables associated to constraints (10), and $\varPhi^i$, $i=1,\ldots,4$ are the positive semidefinite matrices built from the dual variables θ associated with constraints (11), (12), (13), (14), respectively. For instance, if θ^1 is the dual variable associated to constraint (11), then $\varPhi^1 = \frac{\theta^1+\theta^{1T}}{2}$. φ are the dual variables associated to constraints (15). As mentioned in Constraint (20) we have $\Phi = \Phi^1 + \Phi^2 - \Phi^3 - \Phi^4 - diag(\varphi)$.

Let $(\bar{\alpha}, \bar{\Phi}^1, \bar{\Phi}^2, \bar{\Phi}^3, \bar{\Phi}^4, \bar{\varphi})$ be a feasible solution to $(DSDP)$ and let $\bar{\Phi} = \bar{\Phi}^1 + \bar{\Phi}^2 - \bar{\Phi}^3 - \bar{\Phi}^4 - diag(\bar{\varphi})$, then we build the following positive semidefinite matrices:

$$\bar{S}_r = \mathbf{0}_n \qquad r = 1, \ldots, m$$
$$\bar{S}_0 = Q_0 + \sum_{r=1}^{m} \bar{\alpha}_r Q_r + \bar{\Phi}$$

by Constraint (18), $(\bar{S}_0, \ldots, \bar{S}_m)$ form a feasible solution to (OPT_S). The objective value of this solution is equal to $v(\overline{P}_{\bar{S}_0,\ldots,\bar{S}_m})$.

We now prove that $v(\overline{P}_{\bar{S}_0,\ldots,\bar{S}_m}) \geq v(DSDP)$. For this, we prove that for any feasible solution $(\bar{x}, \bar{Y})$ to $(\overline{P}_{\bar{S}_0,\ldots,\bar{S}_m})$, the associated objective value is not smaller than $g(\bar{\alpha}, \bar{\Phi})$. Denote by Δ the difference between the objective values, i.e., $\Delta = \langle \bar{S}_0, \bar{x}\bar{x}^T \rangle + c_0^T \bar{x} + \langle Q_0 - \bar{S}_0, \bar{Y} \rangle - g(\bar{\alpha}, \bar{\Phi})$. We below prove that $\Delta \geq 0$.

$$\begin{aligned}
\Delta &= \langle \bar{S}_0, \bar{x}\bar{x}^T \rangle + c_0^T \bar{x} + \langle Q_0 - \bar{S}_0, \bar{Y} \rangle + \sum_{r=1}^{m} \bar{\alpha}_r b_r - \langle \Phi^1 + \Phi^2, ul^T \rangle + \langle \Phi^3, uu^T \rangle + \langle \Phi^4, ll^T \rangle \\
&\geq c_0^T \bar{x} - \langle \sum_{r=1}^{m} \bar{\alpha}_r Q_r + \bar{\Phi}, \bar{Y} \rangle + \sum_{r=1}^{m} \bar{\alpha}_r b_r - \langle \Phi^1 + \Phi^2, ul^T \rangle + \langle \Phi^3, uu^T \rangle + \langle \Phi^4, ll^T \rangle \\
&\qquad\qquad \text{since } \bar{S}_0 \succeq 0, \text{ and } Q_0 - \bar{S}_0 = -(\sum_{r=1}^{m} \bar{\alpha}_r Q_r + \bar{\Phi}) \\
&= c_0^T \bar{x} + \sum_{r=1}^{m} \bar{\alpha}_r (b_r - \langle Q_r, \bar{Y} \rangle) - \langle \bar{\Phi}, \bar{Y} \rangle - \langle \Phi^1 + \Phi^2, ul^T \rangle + \langle \Phi^3, uu^T \rangle + \langle \Phi^4, ll^T \rangle \\
&\geq c_0^T \bar{x} + \sum_{r=1}^{m} \bar{\alpha}_r c_r^T \bar{x} - \langle \bar{\Phi}, \bar{Y} \rangle - \langle \Phi^1 + \Phi^2, ul^T \rangle + \langle \Phi^3, uu^T \rangle + \langle \Phi^4, ll^T \rangle
\end{aligned}$$

as $c_r^T \bar{x} + \langle Q_r, \bar{Y} \rangle \leq b_r$ and $\bar{\alpha}_r \geq 0$. Moreover, by Constraint (20) we get:

$$\begin{aligned}
\Delta &\geq c_0^T \bar{x} + \sum_{r=1}^{m} \bar{\alpha}_r c_r^T \bar{x} - \langle \bar{\Phi}^1 + \bar{\Phi}^2 - \bar{\Phi}^3 - \bar{\Phi}^4 - diag(\bar{\varphi}), \bar{Y} \rangle - \langle \Phi^1 + \Phi^2, ul^T \rangle + \langle \Phi^3, uu^T \rangle + \langle \Phi^4, ll^T \rangle \\
&= c_0^T \bar{x} + \sum_{r=1}^{m} \bar{\alpha}_r c_r^T \bar{x} - \langle \bar{\Phi}^1, \bar{Y} + ul^T \rangle - \langle \bar{\Phi}^2, \bar{Y} + ul^T \rangle + \langle \bar{\Phi}^3, \bar{Y} + uu^T \rangle + \langle \bar{\Phi}^4, \bar{Y} + ll^T \rangle + \langle diag(\bar{\varphi}), \bar{Y} \rangle
\end{aligned}$$

By Constraints (2)–(6), and since all the coefficients of $\bar{\Phi}^1, \bar{\Phi}^2, \bar{\Phi}^3, \bar{\Phi}^4$, and $\bar{\varphi}$ are non-negative, we get:

$$\begin{aligned}
\Delta &\geq c_0^T\bar{x} + \sum_{r=1}^{m}\bar{\alpha}_r c_r^T\bar{x} - \langle\bar{\Phi}^1, \bar{x}(u^T + l^T)\rangle - \langle\bar{\Phi}^2, \bar{x}(u^T + l^T)\rangle + \langle\bar{\Phi}^3, 2\bar{x}u^T\rangle + \langle\bar{\Phi}^4, 2\bar{x}l^T\rangle + \bar{\varphi}^T\bar{x} \\
&= \Big(c_0 + \sum_{r=1}^{m}\bar{\alpha}_r c_r - (\bar{\Phi}^1 + \bar{\Phi}^2 - 2\bar{\Phi}^3)^T u - (\bar{\Phi}^1 + \bar{\Phi}^2 - 2\bar{\Phi}^4)^T l + \bar{\varphi}\Big)^T \bar{x} \\
&\geq 0 \qquad \text{since } \bar{x} \geq 0 \text{ and by Constraint (19).}
\end{aligned}$$

□

To sum up, we reformulate (P) as the equivalent following problem:

$$(P^*)\begin{cases} \min \quad f_{0,S_0^*}(x,Y) = \langle S_0^*, xx^T\rangle + c_0^T x + \langle Q_0 - S_0^*, Y\rangle \\ \text{s.t.} \\ \quad f_r(x,Y) = \langle Q_r, Y\rangle + c_r^T x \leq b_r \quad r = 1,\dots,m \\ \quad (2) - (9) \end{cases}$$

From (P^*), we build a quadratic convex relaxation of (P^*) by dropping Constraints (8) and (9). We call this relaxation $(\overline{P}^*)$. The optimal value of $(\overline{P}^*)$ is equal to the optimal value of (SDP) which is known to provide a tight bound [6].

One can note the generality of our algorithm which is completely independent from the equivalent convex formulation (P^*) that we used in this paper. In fact, it works with any set of positive semi-definite matrices $S_0,\dots,S_m$. As mentioned in the Introduction, sBB algorithms developed to solve (P) are classically based on complete linearization of (P) which corresponds to reformulation $(\overline{P}_{S_0,\dots,S_m})$ where we set all matrices to $\mathbf{0}_n$. From this remark, we can deduce Corollary 1.

Corollary 1. *Take the following feasible solution to* (OPT_S) *that amounts to the complete linearization of* (P)*:*

$$\bar{S}_r = \mathbf{0}_n \qquad r = 0,\dots,m$$

By definition, we have $v(\overline{P}_{\bar{S}_0,\dots,\bar{S}_m}) \leq v(\overline{P}^*)$. *In other words, the bound obtained by the complete linearization is weaker than the bound we get with the solution of* $(\overline{P}^*)$.

As an illustration, we solve the following small instance obtained from the instance in [8] by a shift on the bounds of the variables:

$$(P_{Ex})\begin{cases} \min \quad x_1^2 + 6x_1x_2 - 2x_1x_4 + 10x_2x_3 + 20x_3x_4 - 60x_1 - 160x_2 - 300x_3 - 180x_4 + 3500 \\ \text{s.t.} \\ \quad 14x_1x_2 - 12x_1x_4 + 8x_2x_3 - 16x_2x_4 + 6x_3x_4 - 20x_1 - 60x_2 - 140x_3 + 220x_4 \leq 17 \\ \quad x_i \in [0, 20] \qquad i = 1,\dots,4 \end{cases}$$

From the optimal solution of the associated semidefinite relaxation (SDP_{Ex}), we deduce the bound -3300 and the following equivalent reformulation:

$$(P^*_{Ex}) \begin{cases} \min \quad 1.996x_1^2 + 8.008x_2^2 + 15.006x_3^2 + 8.993x_4^2 + 6x_1x_2 - 2x_1x_4 + 10x_2x_3 + 20x_3x_4 \\ \qquad -60x_1 - 160x_2 - 300x_3 - 180x_4 - 0.996y_{11} - 8.008y_{22} - 15.006y_{33} - 8.993y_{44} + 3500 \\ \text{s.t.} \\ \quad 14y_{12} - 12y_{14} + 8y_{23} - 16y_{24} + 6y_{34} - 20x_1 - 60x_2 - 140x_3 + 220x_4 \le 17 \\ \quad (2) - (8) \end{cases}$$

As expected, at the root node of the sBB, we obtain again the optimal value -3300 from the quadratic convex relaxation $(\overline{P}^*_{Ex})$. The sBB further proves, at the root node, that -3300 is also the optimal solution value of (P^*_{Ex}) and thus of (P_{Ex}). Observe that the root bound computed in [8] is -3400. Finally, the complete linearization bound is -3900.

4 Main features of our spatial branch-and-bound

A classical way to solve MIQCPs is to use a sBB algorithm. A complete description of this algorithm can be found for instance in [10]. In this paper, we solve (P^*) by a sBB where the bounding step is based on $(\overline{P}^*)$. In the following we give some details on our implementation of `MIQCR_BB`.

The variable selection strategy

Let $(\bar{x}, \bar{Y})$ be the solution of the relaxed problem at the current node, three cases are possible:

1. If $(\bar{x}, \bar{Y})$ satisfies Constraints (8) and (9) with an accuracy ϵ_{const}, then $(\bar{x}, \bar{Y})$ is the optimal solution of the considered branch. The branch is pruned.
2. Else, if Constraints (8) are not satisfied, we select an index i^* satisfying:
$$(i^*, j^*) = \text{argmax}|s^*_{0ij}(\bar{x}_i\bar{x}_j - \bar{y}_{ij})|$$
where s^*_{0ij} is the (i,j)-th element of matrix S^*_0.
3. Else, we select the first index $i^* \in J$, such that $\bar{x}_{i^*} \notin \mathbb{N}$

The branching rules

Let $(\bar{x}, \bar{Y})$ be the solution of $(\overline{P}^*)$ at the current node, x_{i^*} the selected variable with a current value $\bar{x}_{i^*}$. We branch as described below:

1. *If x_{i^*} is an integer variable (i.e. $i^* \in J$)*:
   i) Branch 1: $x_{i^*} \le \lfloor \bar{x_{i^*}} \rfloor$, i.e. $u_{i^*} = \lfloor \bar{x_{i^*}} \rfloor$.
   ii) Branch 2: $x_{i^*} \ge \lceil \bar{x_{i^*}} \rceil$, i.e. $l_{i^*} = \lfloor \bar{x_{i^*}} \rfloor$.

2. *If x_{i^*} is a continuous variable (i.e. $i^* \in I \backslash J$)*, let γ be a parameter in $[0, 1]$, and let $v_{i^*} = (1-\gamma)\frac{u_{i^*}+l_{i^*}}{2} + \gamma \bar{x_{i^*}}$:
   i) Branch 1: $x_{i^*} \leq v_{i^*}$, i.e. $u_{i^*} = v_{i^*}$.
   ii) Branch 2: $x_{i^*} \geq v_{i^*}$, i.e. $l_{i^*} = v_{i^*}$.

The node selection strategy

We implement two classical strategies for selecting the next subproblem: the "depth-first" and the "best-first" selection strategy. In our experiments of Section 5, we use the most efficient for the considered class of instances.

Computation of feasible solutions and bound propagation

At each node of our algorithm, we compute a feasible solution to (P) using one of the two following strategies:

1. From a current solution $(\bar{x}, \bar{Y})$ of $(\overline{P}^*)$, if $\bar{x}$ satisfies the initial constraints, then it is a feasible solution to (P) and $f_0(\bar{x})$ can be used as an upper bound.

2. We alternatively use the local search of standard solvers to compute feasible local solutions to (P). More precisely, we use the local search of `Cplex 12.6.2` [42] for unconstrained and linearly constrained problems, and the local search of `Scip 3.2.1` [1] for quadratically constrained problems.

We also use bound propagation for upper and lower bounds of the x variables. for this, we implement the propagation of quadratic constraints described in [29].

5 Computational results

Considered instances

We evaluate our algorithm on several sets of instances. The first set is composed of 90 pure-continuous quadratic instances with box constraints called *boxqp* or *spar* coming from [26, 57]. Then, we perform experiments on 100 instances of quadratically constrained quadratic programs of [17] available at [43]. For those, we consider two classes of instances: the class QCP_5 of pure-continuous instances and the class $IQCP_5$ of pure-integer instances. Finally, we consider the 135 instances of quadratically constrained quadratic programs from [8] called *unitbox*.

Experimental environment

Our experiments were carried out on a server with 2 CPU Intel Xeon each of them having 12 cores and 2 threads of 2.5 GHz and $4 * 16$ GB of RAM using a

Linux operating system. For all algorithms, we use the multi-threading version of `Cplex 12.6.2` with up to 48 threads.

For method `MIQCR_BB`, we used the solver CSDP [23] together with the Conic Bundle algorithm [41] for solving semi-definite programs (SDP), as described in [18]. We used the C interface of the solver `Cplex 12.6.2` for solving the quadratic convex problem ($\overline{P}^*$) at each node of the search tree of `MIQCR_BB`. For computing feasible local solutions, we use the local search of `Cplex 12.6.2` for class *boxqp*, and the local search of `Scip 3.2.1` [1] for classes QCP_5, $IQCP_5$, and *unitbox*.

Parameters of `MIQCR_BB`

We set the parameters as follows:

– Phase 1: Parameters `axtol, aytol` of CSDP [23] are set to 10^{-5}. The precision of the *Conic Bundle* [41] is set to 10^{-5}.
– Phase 2: We initialize γ to 0.25 and accuracies as follows:
  - relative mipgap of the branch-and-bound: $\epsilon = 10^{-5}$ for classes *boxqp*, QCP_5 and $IQCP_5$ and 10^{-4} for class *unitbox*.
  - absolute accuracy for the constraints violation: $\epsilon_{const} = 10^{-4}$,
  - absolute accuracy for considering a value as zero or as an integer: $\epsilon_{zero} = 10^{-6}$,
  - for `Cplex 12.6.2` [42], the relative mipgap is 10^{-5}, the absolute gap is 0.99, and the parameter `varsel` is set to 0.

Solvers used for comparison

– `QuadProgBB` [27] (MIQCP solver) that performs a finite branch-and-bound, based on the same semidefinite relaxation as `MIQCR_BB`, but in `QuadProgBB` the branch-and-bound enforces the first-order KKT conditions. Here, it runs with `Cplex 12.6.2` [42] and `matlab`, and the relative mipgap is set to 10^{-5}.
– `Cplex 12.6.2` [42] (MIQP solver). This version of Cplex implements the recent advances introduced in [22], that exploit the performances of Cplex for binary quadratic programming to solve box-constrained continuous problems. The relative mipgap is set to 10^{-5}.
– `Couenne 0.5` [11] (MINLP solver) that uses a complete linearization as convex relaxation into a SBB procedure. The solver `Couenne 0.5` runs with `Cplex 12.6.2`, and the relative mipgap is set to 10^{-5}.
– `GloMIQO 2` [48] (MIQCP solver) that mixes several algorithmic components for solving MIQCPs. After a reformulation of the instance, it generates tight convex relaxations, in particular by detecting special structures. As we do not have the license of `GloMIQO 2`, for the results of this method, we take the results of the paper [48]. We observe that these experiments were carried out on a server with a very similar configuration to our server. However, the relative mipgap (10^{-4}) is different from the relative mipgap (10^{-5}) of the other solvers for the *boxqp* class.
– `Scip 3.2.1` [58] (MINLP solver) that uses a linear outer approximation as convex relaxation into a sBB. The relative mipgap is set to 10^{-5} for classes QCP_5 and $IQCP_5$ and to 10^{-4} for class *unitbox*.

– `BARON 17.3.31` [51] (MINLP solver) that uses linear relaxations combined with domain reduction strategies within a sBB. The relative mipgap is set to 10^{-5} for classes *boxqp*, and QCP_5 and $IQCP_5$ and to 10^{-4} for class *unitbox*.

5.1 Results for the *boxqp* instances

This set of 90 instances was proposed in [57] and extended in [26]. They were generated as follows: nonzeros of Q_0 and c_0 are integers uniformly generated over the interval $[-50, 50]$. The sizes of the instances of [57] are $n = 20, 30, 40, 50$, and 60 and the densities vary from 20% to 100%. For the instances of [26] the sizes are $n = 70, 80, 90$, and 100 and the densities are 25%, 50%, and 75%. In these instances $J = \emptyset$, $r = 0$, and for all $i \in I$, $l_i = 0$, $u_i = 1$. An instance with n variables, a density of d%, and whose instance number is k is named *spar-n-d-k*.

In these experiments, we set the time limit to 1 hour, and we set the node selection strategy to "best first". In Figure 1, we present the performance profile [28] of the CPU times for methods `MIQCR_BB`, `QuadprogBB`, `Cplex`, `Couenne`, `BARON` and `GloMIQO` over the 90 *boxqp* instances. In this profile we can see that `MIQCR_BB` outperforms the other algorithms both in terms of the total CPU time and of the number of instances solved. We do not report results for the solver `Scip` as it was less efficient than `Couenne`. More precisely, it solves 17 instances out of 90 within the time limit.

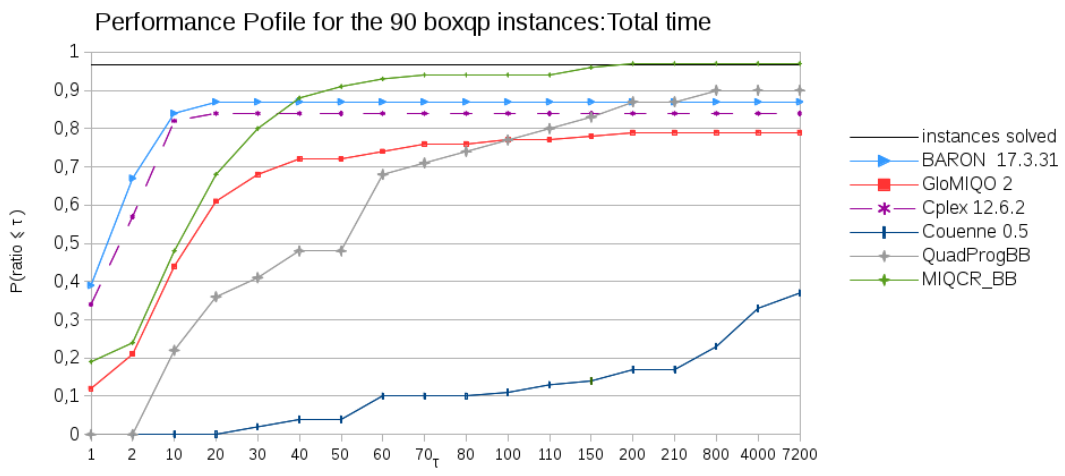

Fig. 1. Performance profile of the total time for the *boxqp* instances with $n = 20$ to 100 with a time limit of 1 hour.

In Tables 1 and 2 we present a more detailed comparison. Each line corresponds to one instance. We report in Column *Gap* the initial gap that is equal to

$|\frac{Opt-Bound}{Opt}|*100$ where *Opt* is the best known solution of the instance and *Bound* the optimal value of the relaxation at the root node of the branch-and-bound, in Column *Time* the CPU time in seconds, where `-` means that the instance is unsolved within the time limit, and in Column *Nodes* the number of nodes visited. For these instances, the initial gaps of methods `MIQCR_BB` and `QuadProgBB` are the same, while the initial gap of `Cplex` is about 105 times larger. We did not report the reformulation time, but we observe that in `MIQCR_BB` the reformulation phase represents about 32% of the total solution time, and it represents about 3.6% for `QuadProgBB`.

For these instances, `Couenne` is not able to solve instances with more than 60 variables within the time limit while the other methods solve instances with up to 100 variables within one hour. We observe that the complete linearization used in `Couenne` is outperformed by `MIQCR_BB` as suggested by Corollary 1. Finally, over the 90 considered instances, `Couenne` solves 33 instances, `GloMIQO` solves 71 instances (with a relative gap of 10^{-4} instead of 10^{-5} for other methods), `Cplex` solves 76 instances, `BARON` solves 78 instances, `QuadProgBB` solves 81 instances, and `MIQCR_BB` solves 87 instances within the time limit. In particular `MIQCR_BB` is more performant for large instances with high density.

In Table 3, we report the results of `MIQCR_BB` and `Cplex` for the 9 instances of *boxqp* with $n = 125$. We observe that `MIQCR_BB` is able to solve one more instance, but `Cplex` is faster over the 2 instances solved by both methods.

5.2 Instances of quadratically constrained quadratic programs : $IQCP_5$ and QCP_5

We use class $IQCP_5$ from [17] and we build class QCP_5 by continous relaxation of the variables of $IQCP_5$. Each instance consists of minimizing a quadratic function subject to 5 quadratic inequality constraints. Instances of $IQCP_5$ were randomly generated as follows:

- $\ell_i = 0$ and $u_i = 20$, for all $i \in I = \{1, \dots, n\}$.
- The coefficients of Q_0 are integers uniformly distributed in the interval $[-5, 5]$ with a density of 75%, and $c_0 = 0$
- The coefficients of Q_r are integers uniformly distributed in the interval $[0, 10]$ with a density of 25%, and $c_r = 0$
- $b_r = \lfloor 0.1 * (\sum_{i=1}^{n}\sum_{j=1}^{n} q_{rij} u_i u_j)\rfloor$.

We use the instances with $n =$ 10, 20, 30, 40 or 50 for each class, and for each n we have 10 instances. We have a total of 100 instances. The instances are named *QCP5-n-k* or *IQCP5-n-k* where n is the number of variables, and k is the instance number.

In these experiments, we set the time limit to 1 hour, and we set the node selection strategy to "best first" for QCP_5 and to "depth first" for $IQCP_5$. In Figure 2, we present the performance profile of the CPU times for methods

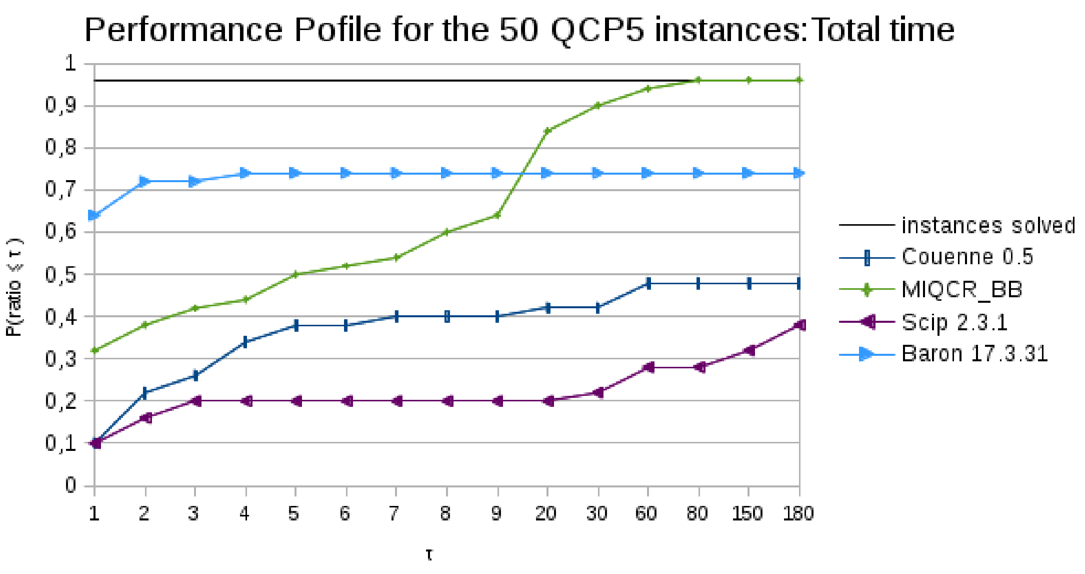

Fig. 2. Performance profile of the total time for the QCP_5 instances with $n = 10$ to 50 with a time limit of 1 hour.

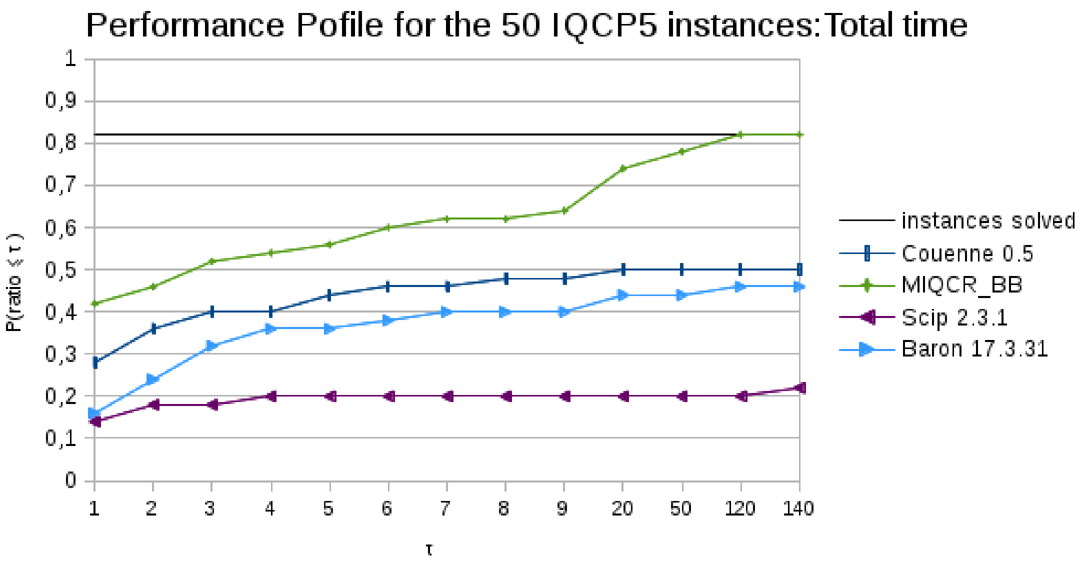

Fig. 3. Performance profile of the total time for the $IQCP_5$ instances with $n = 10$ to 50 with a time limit of 1 hour.

MIQCR_BB, Couenne, Scip, and BARON over the 50 QCP_5 instances, and in Figure 3 for the 50 $IQCP_5$ instances. We recall that the difference between classes QCP_5 and $IQCP_5$ is only the type of the variables. In QCP_5 all variables are continuous, while in $IQCP_5$ they are all integers. In these profiles, for both classes, we observe that MIQCR_BB outperforms the other algorithms.

We present in Tables 4–7 a more detailed comparison of the methods. Each line corresponds to one instance. We report in Column *Sdp T.* the reformulation CPU time, and in Column *sBB T.* the time spent for the sBB. In Table 4, we can observe that BARON is the fastest solver for the smallest instances and MIQCR_BB is the fastest solver for the largest instances. Moreover, for instances with $n = 40$, BARON solves 7 instances over 10 within the time limit, while MIQCR_BB solves the 10 instances. In Table 6, we observe that that Couenne is fastest for the smallest instances ($n = 10$ or 20), and here again, MIQCR_BB is the fastest solver for the largest instances ($n = 30$ or 40). Indeed, MIQCR_BB solves all these instances, while Couenne solves 5 out of 20, BARON solves 3 out of 20, and Scip is not able to solve any instances of these sizes.

We observe that the initial gap with reformulation (P^*) ($S_0 = S_0^*$: MIQCR_BB) is on average about 15 times smaller than the initial gap with the complete linearization ($S_0 = \mathbf{0}_n$: Couenne) for these instances. It is important to notice that the sizes of the two reformulations are the same. However, the price of this better initial gap is the solution of a semi-definite relaxation. In fact this solution time represents about 38% of the total time on average over the 100 instances.

The results for the 10 generated instances of each class QCP_5 and $IQCP_5$ with $n = 50$ are reported in Tables 5 and 7. MIQCR_BB is able to solve 8 instances of class QCP_5 and 4 of class $IQCP_5$ while Couenne, Scip, and BARON cannot solve any one. We also tested mixed-integer instances built from $IQCP_5$ where we relax the integrality constraints of half of the variables. Since the results reveal a similar trend as for instances of classes QCP_5 and $IQCP_5$, we did not report the specific results in this section.

5.3 Results for the *unitbox* instances

Each instance from [8] consists in minimizing a quadratic function of n continuous variables in the interval $[0, 1]$, subject to m quadratic inequalities. For the considered instances, n varies from 8 to 50, and m from 8 to 100. An instance is denoted by *unitbox-n-m-k-d* where n is the number of variables, m is the number of quadratic constraints, k is the instance number, and d is the density in %. We set the time limit to 2 hours, and the node selection strategy to "best first".

In Figure 4, we present the performance profile of the CPU times for methods MIQCR_BB, GloMIQO, Scip, and BARON over the 135 *unitbox* instances. We observe that MIQCR_BB outperforms the other methods. Here again, the results for GloMIQO are taken from [48].

In Tables 8-10, we present a more detailed comparison of the methods. Each line corresponds to one instance. We observe that Scip solves 88 instances, GloMIQO and BARON solve 109 instances, and MIQCR_BB solves 119 instances out

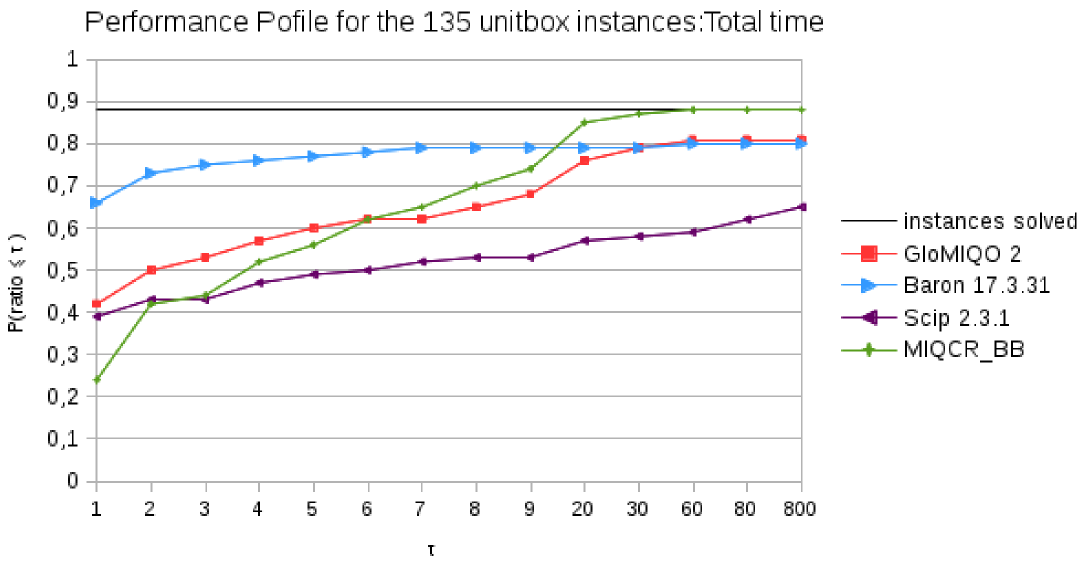

Fig. 4. Performance profile of the total time for the *unitbox* instances with $n = 8$ to 50 with a time limit of 2 hours.

of 135 within the time limit. Observe that `GloMIQO` or `BARON` are faster on most of the sparse instances, while `MIQCR_BB` is again faster on large and dense instances.

6 Conclusion

We consider the general problem (P) of minimizing a quadratic function subject to quadratic constraints where the variables can be integer or continuous. In this paper, we extend the quadratic convex reformulation method to the solution of (P). Our previous versions of this method could handle these programs with the restriction that all quadratic sub-functions of purely continuous variables are already convex.

We start by a reformulation step which solves a semidefinite program (SDP) in order to build an equivalent quadratic program. From this equivalent program, we compute a strong quadratic convex relaxation which captures the tightness of (SDP). In previous versions, in the presence of continuous variables, the reformulated problem was computed thanks to a weaker semidefinite relaxation of (P), but it could be solved by standard branch-and-bound for mixed-integer quadratic convex programs. In this paper, to handle the continuous variable case, we build a tighter convex relaxation, but we can no longer rely on standard branch-and-bound. We thus develop an appropriate spatial branch-and-bound to solve the reformulated problem where the bounding step solves our strong quadratic convex relaxation. Our whole method can be viewed as an improvement of the classic spatial branch-and-bound based on complete linearization.

We report computational results on 325 instances. These results show that the method allows us to solve almost all the continuous box-constrained instances with up to 100 variables and some of the instances with 125 variables. Among the considered instances with 5 inequality constraints, the method can handle instances with up to 50 integer or continous variables in less than 1 hours of computation time. Finally, for the *unitbox* instances, our method solves 119 instances out of 135 within a time limit of 2 hours, which is, to the best of our knowledge, the best result for this class of problems.

Acknowledgement: This research was partially funded by the Fondation Mathématique Jacques Hadamard (FMJH) through the Gaspard Monge Program for Optimization and operations research (PGMO). The authors are gratefull to Alain Billionnet for all our collaborations and for a carefull reading of this manuscript.

References

1. T. Achterberg. Scip : solving constraint integer programs. *Mathematical Programming Computation*, (1):1–41, 2009.
2. Claire S Adjiman, Ioannis P Androulakis, and Christodoulos A Floudas. A global optimization method, αbb, for general twice-differentiable constrained nlpsii. implementation and computational results. *Computers & Chemical Engineering*, 22(9):1159–1179, 1998.
3. C.S. Adjiman, S. Dallwig, C.A. Floudas, and A. Neumaier. A global optimization method, αbb, for general twice-differentiable constrained nlpsi. theoretical advances. *Computers and Chemical Engineering*, 22(9):1137–1158, 1998.
4. I.P. Androulakis, C.D. Maranas, and C.A. Floudas. abb : A global optimization method for general con- strained nonconvex problems. *Journal of Global Optimization*, 7:337–363, 1995.
5. M.F. Anjos and J.B. Lasserre. Handbook of semidefinite, conic and polynomial optimization: Theory, algorithms, software and applications. *International Series in Operational Research and Management Science*, 166, 2012.
6. 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.
7. C. Audet, J. Brimberg, P. Hansen, S. Le Digabel, and N. Mladenović. Pooling problem: Alternate formulations and solution methods. *Management science*, 50(6):761–776, 2004.
8. X. Bao, N. V. Sahinidis, and M. Tawarmalani. Multiterm polyhedral relaxations for nonconvex, quadratically constrained quadratic programs. *Optimization Methods Software*, 24(4-5):485–504, 2009.
9. X. Bao, N.V. Sahinidis, and M. Tawarmalani. Semidefinite relaxations for quadratically constrained quadratic programming: A review and comparisons. *Mathematical Programming*, 129:129–157, 2011.
10. P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 2013.

11. P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Waechter. Branching and bounds tightening techniques for non-convex minlp. *Optimization Methods and Software*, 4–5(24):597–634, 2009.
12. A. Billionnet. *Optimisation discrète, De la modélisation à la résolution par des logiciels de programmation mathématique.* Dunod, 2007.
13. A. Billionnet and S. Elloumi. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Mathematical Programming*, 109(1):55–68, 2007.
14. A. Billionnet, S. Elloumi, and A. Lambert. Linear reformulations of integer quadratic programs. In *MCO 2008, september 8-10*, pages 43–51, 2008.
15. 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.
16. 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.
17. A. Billionnet, S. Elloumi, and A. Lambert. Exact quadratic convex reformulations of mixed-integer quadratically constrained problems. *Mathematical Programming*, 158(1):235–266, 2016.
18. A. Billionnet, S. Elloumi, A. Lambert, and A. Wiegele. Using a conic bundle method to accelerate both phases of a quadratic convex reformulation. *To appear, Informs Journal On Computing*, pages 1–15, 2016.
19. A. Billionnet, S. Elloumi, and M. C. Plateau. Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: The QCR method. *Discrete Applied Mathematics*, 157(6):1185 – 1197, 2009. Reformulation Techniques and Mathematical Programming.
20. C. Bliek, P. Bonami, and A. Lodi. Solving Mixed-Integer Quadratic Programming problems with IBM-CPLEX: a progress report. In *Proceedings of the Twenty-Sixth RAMP Symposium, Hosei University, Tokyo, October 16-17*, 2014.
21. P. Bonami, L. Biegler, A. Conn, G. Cornuéjols, I. Grossmann, C. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Waechter. An Algorithmic Framework for Convex Mixed Integer Nonlinear Programs. *Discrete Optimization*, 5(2):186–204, 2008.
22. P. Bonami, O. Günlük, and J. Linderoth. Solving box-constrained nonconvex quadratic programs. *optimization online*, 2016.
23. B. Borchers. CSDP, A C Library for Semidefinite Programming. *Optimization Methods and Software*, 11(1):613–623, 1999.
24. Fani Boukouvala, Ruth Misener, and Christodoulos A Floudas. Global optimization advances in mixed-integer nonlinear programming, minlp, and constrained derivative-free optimization, cdfo. *European Journal of Operational Research*, 252(3):701–727, 2016.
25. S. Burer and A. Letchford. Non-convex mixed-integer nonlinear programming: a survey. *Surveys in Oper. Res. and Mgmt. Sci.*, 17(2):97–106, 2012.
26. 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.
27. J. Chen and S. Burer. Globally solving nonconvex quadratic programming problems via completely positive programming. *Mathematical Programming Computation*, 4(1):33–52, 2012.
28. D. Dolan and J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 1986.

29. F. Domes and A. Neumaier. Constraint propagation on quadratic constraints. *Constraints*, 15(3):404–429, 2010.
30. M. A. Duran and I. E. Grossmann. A mixed-integer nonlinear programming algorithm for process systems synthesis. *AIChE J*, 32(4):592–606, 1986.
31. M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36:307–339, 1986.
32. J.E. Falk and R.M. SolandErkut. An algorithm for separable nonconvex programming problems. *Management Science*, 15:550–560, 1969.
33. CA Floudas and CE Gounaris. A review of recent advances in global optimization. *Journal of Global Optimization*, 45(1):3–38, 2009.
34. Christodoulos A Floudas and V Visweswaran. A global optimization algorithm (gop) for certain classes of nonconvex nlpsi. theory. *Computers & chemical engineering*, 14(12):1397–1417, 1990.
35. Christodoulos A Floudas and Vishy Visweswaran. Primal-relaxed dual global optimization approach. *Journal of Optimization Theory and Applications*, 78(2):187–225, 1993.
36. R. Fortet. L'algèbre de Boole et ses Applications en Recherche Opérationnelle. *Cahiers du Centre d'Etudes de Recherche Opérationnelle*, 4:5–36, 1959.
37. M.R. Garey and D.S. Johnson. Computers and Intractability: A guide to the theory of NP-Completness. *W.H. Freeman, San Francisco, CA*, 1979.
38. W.W. Hager and J.T. Hungerford. Continuous quadratic programming formulations of optimization problems on graphs. *European Journal of Operational Research*, 240(2):328 – 337, 2015.
39. P. L. Hammer and A.A. Rubin. Some remarks on quadratic programming with 0-1 variables. *Revue Française d'Informatique et de Recherche Opérationnelle*, 4:67–79, 1970.
40. C.A. Haverly. Studies of the behaviour of recursion for the pooling problem. *ACM SIGMAP Bulletin*, 26, 1978.
41. C. Helmberg. *Conic Bundle v0.3.10*, 2011.
42. IBM-ILOG. *IBM ILOG CPLEX 12.6.2 Reference Manual*, 2015.
43. A. Lambert. IQCP/MIQCP: Library of Integer and Mixed-Integer Quadratic Quadratically Constrained Programs. "`http://cedric.cnam.fr/~lamberta/Library/iqcp_miqcp.html`", 2013.
44. M. Madani and M. Van Vyve. Computationally efficient MIP formulation and algorithms for european day-ahead electricity market auctions. *European Journal of Operational Research*, 242(2):580 – 593, 2015.
45. G.P. McCormick. Computability of global solutions to factorable non-convex programs: Part i - convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976.
46. R. Misener and C. A. Floudas. Global optimization of mixed-integer quadratically-constrained quadratic programs (MIQCQP) through piecewise-linear and edge-concave relaxations. *Math. Program. B*, 136(1):155–182, 2012. `http://www.optimization-online.org/DB_HTML/2011/11/3240.html`.
47. R. Misener and C. A. Floudas. GloMIQO: Global mixed-integer quadratic optimizer. *Journal of Global Optimization*, 57(1):3–50, 2013.
48. R. Misener, J. B. Smadbeck, and C. A. Floudas. Dynamically generated cutting planes for mixed-integer quadratically constrained quadratic programs and their incorporation into GloMIQO 2. *Optimization Methods and Software*, 30(1):215–249, 2015.
49. J.J. Moré and G. Toraldo. Algorithms for bound constrained quadratic programming problems. *Numerische Mathematik*, 55(4):377–400, 1989.

50. M. Kilinç P. Bonami and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. In *Mixed integer nonlinear programming*, pages 1–39. Springer New York, 2012.
51. N.V. Sahinidis and M. Tawarmalani. Baron 9.0.4: Global optimization of mixed-integer nonlinear programs. *User's Manual*, 2010.
52. A. Saxena, P. Bonami, and J. Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: projected formulations. *Mathematical Programming*, 130:359–413, 2011.
53. H. D. Sherali and W. P. Adams. A hierarchy of relaxation between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal Discrete Mathematics*, 3:411–430, 1990.
54. M. Tawarmalani and N. V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical programming*, 99(3):563–591, 2004.
55. M. Tawarmalani and N.V. Sahinidis. Convexification and global optimization in continuous and mixed-integer nonlinear programming. *Kluwer Academic Publishing, Dordrecht, The Netherlands*, 2002.
56. M. Tawarmalani and N.V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005.
57. D. Vandenbussche and G. Nemhauser. A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Mathematical Programming*, 102(3):259–275, 2005.
58. S. Vigerske and A. Gleixner. Scip: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. *optimization online*, 2016.
59. V Visweswaran and CA Floudas. New properties and computational improvement of the gop algorithm for problems with quadratic objective functions and constraints. *Journal of Global Optimization*, 3(4):439–462, 1993.
60. V Visweswaran and CA Floudast. A global optimization algorithm (gop) for certain classes of nonconvex nlpsii. application of theory and test problems. *Computers & chemical engineering*, 14(12):1419–1434, 1990.
61. K. Zorn and N. V. Sahinidis. Computational experience with applications of bilinear cutting planes. *Industrial & Engineering Chemistry Research*, 52(22):7514–7525, 2013.
62. K. Zorn and N. V. Sahinidis. Global optimization of general non-convex problems with intermediate bilinear substructures. *Optimization Methods and Software*, 29(3):442–462, 2014.