An efficient Compact Quadratic Convex Reformulation for general integer quadratic programs

Alain Billionnet, Sourour Elloumi
CEDRIC-ENSIIE, 1, square de la résistance, F-91025 Evry cedex, France {alain.billionnet@ensiie.fr, sourour.elloumi@ensiie.fr}

Amélie Lambert
CEDRIC-CNAM, 292 rue saint Martin, F-75141 Paris Cedex 03, France amelie.lambert@cnam.fr

We address the exact solution of general integer quadratic programs with linear constraints. These programs constitute a particular case of mixed-integer quadratic programs for which we introduce in [3] a general solution method based on quadratic convex reformulation, that we called `MIQCR`. This reformulation consists in designing an equivalent quadratic program with a convex objective function. The problem reformulated by `MIQCR` has a relatively important size that penalizes its solution time. In this paper, we propose a convex reformulation less general than `MIQCR` because it is limited to the general integer case, but that has a significantly smaller size. We call this approach Compact Quadratic Convex Reformulation (`CQCR`). We evaluate `CQCR` from the computational point of view. We perform our experiments on instances of general integer quadratic programs with one equality constraint. We show that `CQCR` is much faster than `MIQCR` and than the general non-linear solver BARON [25] to solve these instances. Then, we consider the particular class of binary quadratic programs. We compare `MIQCR` and `CQCR` on instances of the Constrained Task Assignment Problem. These experiments show that `CQCR` can solve instances that `MIQCR` and other existing methods fail to solve.

Key words: Quadratic Programming; Integer Programming; Exact Convex Reformulation; Computational experiments

---

1 Introduction

Consider the following linearly-constrained integer quadratic program:

$$(QP)\begin{cases} \min_x & f(x) = \sum_{i=1}^n \sum_{j=1}^n q_{ij} x_i x_j + \sum_{i=1}^n c_i x_i \\ s.t. & \sum_{i=1}^n a_{ri} x_i = b_r \quad r \in R \quad (1) \\ & x_i \le u_i \quad i \in I \quad (2) \\ & x_i \ge 0 \quad i \in I \quad (3) \\ & x_i \in \mathbb{N} \quad i \in I \quad (4) \end{cases}$$

where $A = (a_{ij}) \in \mathbf{M}_{m,n}$ (set of $m \times n$ integer matrices), $b \in \mathbb{N}^m$, $I = \{i \,:\, i = 1, \ldots, n\}$, $R = \{r \,:\, r = 1, \ldots, m\}$, $u_i \in \mathbb{N}\,(i \in I)$, $Q = (q_{ij}) \in \mathbf{S}_n$ (space of symmetric matrices of order n), and $c \in \mathbb{R}^n$. We shall suppose the feasible domain of (QP) non-empty. We consider here an equality constrained program. If some inequality constraints must be considered, we suppose that they have been reformulated as equality constraints by adding integer and upper bounded slack variables. This is always possible because all coefficients a_{ri} and b_r are integer, and because variables are nonnegative and upper bounded.

(QP) is a hard optimization problem [14]. It can be viewed as a generalization of Integer Linear Programming where the main additional difficulty is the non-convexity of the objective function (unless matrix Q is positive semi-definite). Many applications in operations research and industrial engineering involve discrete variables in their formulation. Some of these applications can be formulated as (QP). For instance, (QP) is used in [12] for the unit commitment problem and for the Markowitz mean-variance model, in [13] for the chaotic mapping of complete multipartite graphs, in [7] for the material cutting, and in [17] for the capacity planning.

Problems such as (QP) are often solved by branch-and-cut procedures. These algorithms are based on a bound that can be generally computed polynomially. These bounds can, for instance, be a convex approximation of (QP). This is the case in general mixed-integer non-linear algorithms that are based on global optimization techniques [1, 11, 19, 26, 29]. We briefly recall here the method presented in [26] and implemented through the mixed-integer non-linear solver BARON [25]. This algorithm is a polyhedral branch-and-cut procedure that facilitates the reliable use of nonlinear convex relaxations in global optimization. It exploits convexity in order to generate polyhedral cutting planes and relaxations for multivariate non-convex problems. The mixed-integer non-linear solver BARON is able to solve an important number of instances from `globallib` [15] and `minlplib` [23].

We also recall here the Mixed Integer Quadratic Convex Reformulation (`MIQCR`) that was introduced in [3]. Here, for convex reformulation, we use the definition of Audet and al. [2], as we build an equivalent problem to (QP) that has a quadratic and convex objective function. This approach solves general mixed-integer quadratic problems and obviously can handle (QP). The idea of `MIQCR` is to design a problem equivalent to (QP) with a convex objective function. This equivalent problem is computed thanks to the solution of a semi-definite relaxation of (QP). The semi-definite relaxation and the reformulated problem involve an important number of additional variables and constraints. In this paper, we propose a Compact Quadratic Convex Reformulation (`CQCR`), based on the same ideas as `MIQCR`, that handles general integer quadratic programs and that leads to a reformulated problem and a semi-definite relaxation with smaller sizes.

From a theoretical point of view, our new approach `CQCR` uses a reformulated problem which bound obtained by continuous relaxation is weaker than the one of `MIQCR`. However, from the computational point of view, `CQCR` is much faster than `MIQCR` on instances of the class EIQP (Equality Integer Quadratic Problem) [3, 21]. This reduced solution time concerns both the semi-definite relaxation and the reformulated problem. We also compare these two approaches on instances of the Constrained Task Assignment Problem (CTAP), a particular case of (QP) with binary variables.

The outline of the paper is the following. In Section 2, we recall the `MIQCR` approach applied to (QP). In Section 3, we present our new compact reformulation `CQCR`. Then, in Section 4, we report our computational evaluation of `CQCR`. Section 5 is a conclusion.

2 MIQCR applied to (QP)

When applied to (QP), MIQCR consists in reformulating it into the following parameterized problem $(QP_{\alpha,\beta})$ [3, 4]:

$$
(QP_{\alpha,\beta}) \begin{cases}
\min\limits_{x,y,z,t} & f_{\alpha,\beta}(x,y) & & \\
s.t. & (1)(2)(3) & & \\
 & x_i = \sum\limits_{k=0}^{\lfloor log(u_i)\rfloor} 2^k t_{ik} & i \in I & (5) \\
 & z_{ijk} \leq u_j t_{ik} & (i,k) \in E, j \in I & (6) \\
 & z_{ijk} \leq x_j & (i,k) \in E, j \in I & (7) \\
 & z_{ijk} \geq x_j - u_j(1-t_{ik}) & (i,k) \in E, j \in I & (8) \\
 & z_{ijk} \geq 0 & (i,k) \in E, j \in I & (9) \\
 & y_{ij} = \sum\limits_{k=0}^{\lfloor log(u_i)\rfloor} 2^k z_{ijk} & (i,j) \in I^2 & (10) \\
 & y_{ij} \geq u_i x_j + u_j x_i - u_i u_j & (i,j) \in I^2 & (11) \\
 & y_{ii} \geq x_i & i \in I & (12) \\
 & y_{ij} = y_{ji} & (i,j) \in I^2, i \leq j & (13) \\
 & t_{ik} \in \{0,1\} & (i,k) \in E & (14)
\end{cases}
$$

where

$$f_{\alpha,\beta}(x,y) = f(x) + \sum_{i=1}^{n}\sum_{j=1}^{n}\beta_{ij}(x_i x_j - y_{ij}) + \alpha\sum_{r=1}^{m}(\sum_{i=1}^{n} a_{ri}x_i - b_r)^2$$

with $\alpha \in \mathbb{R}$, $\beta \in \mathbf{S}_n$, $E = \{(i,k) \,:\, i = 1,\ldots,n,\ k = 0,\ldots \lfloor log(u_i)\rfloor\}$.

In Constraints (5), we make a binary decomposition of variables x_i by use of 0-1 variables t_{ik}. Hence, any product of variables $x_i x_j$ can be written as $\sum\limits_{k=0}^{\lfloor log(u_i)\rfloor} 2^k t_{ik} x_j$. We linearize the last expression by use of variables z_{ijk} and Constraints (6)-(9) that enforce the equality $z_{ijk} = t_{ik}x_j$, when t_{ik} is 0 or 1. Variables y_{ij} satisfy $y_{ij} = x_i x_j$ by Constraints (10), and their use allows us to avoid putting variables z_{ijk} and t_{ik} in the objective function. Moreover, Constraints (11)-(13) are valid inequalities that tighten the formulation.

This linearization adds an important number of variables and constraints. More precisely, if we denote by $N = |E| = \sum\limits_{i=1}^{n}(\lfloor log(u_i)\rfloor + 1)$ the number of t variables, $(QP_{\alpha,\beta})$ has $O(nN)$ variables and linear constraints.

Parameters α and β are interesting only when the reformulated function $f_{\alpha,\beta}(x,y)$ is convex. In this case, the continuous relaxation of $(QP_{\alpha,\beta})$ is a convex optimization problem, and general mathematical programming solvers such as Cplex [18] can solve $(QP_{\alpha,\beta})$ through a Branch and Bound based on continuous relaxation. In [3], we state the problem of looking for parameters α and β such that the continuous relaxation bound of $(QP_{\alpha,\beta})$ is maximized. These best parameters can be computed as the dual solution of the following semi-definite relaxation of (QP), (SDP), that has $O(n^2)$ variables and constraints:

$$
(SDP) \begin{cases}
\min\limits_{X,x} \quad f(X,x) = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} q_{ij}X_{ij} + \sum\limits_{i=1}^{n} c_i x_i \\
s.t. \quad (1) \\
\quad \sum\limits_{r=1}^{m}(\sum\limits_{i=1}^{n}(\sum\limits_{j=1}^{n} a_{ri}a_{rj}X_{ij} - 2a_{ri}b_r x_i)) = -\sum\limits_{r=1}^{m} b_r^2 & (15) \\
\quad X_{ij} \leq u_j x_i & (i,j) \in I^2 \quad (16) \\
\quad X_{ij} \leq u_i x_j & (i,j) \in I^2 \quad (17) \\
\quad X_{ij} \geq u_j x_i + u_i x_j - u_i u_j & (i,j) \in I^2 \quad (18) \\
\quad X_{ij} \geq 0 & (i,j) \in I^2 \quad (19) \\
\quad X_{ii} \geq x_i & i \in I \quad (20) \\
\quad \begin{pmatrix} 1 & x \\ x^T & X \end{pmatrix} \succeq 0 & (21) \\
\quad x \in \mathbb{R}^n \quad X \in \mathbf{S}^n & (22)
\end{cases}
$$

In MIQCR, we perturb the Q matrix of $f(x)$ using a scalar parameter α and a matrix parameter β. More precisely, we consider the perturbed matrix $Q_{\alpha,\beta} = Q + \alpha AA^T + \beta$. To get the equivalent function $f_{\alpha,\beta}(x,y)$, we use the additional variables y_{ij} and we subtract the linear terms $\beta_{ij}y_{ij}$ while adding linear constraints enforcing $y_{ij} = x_i x_j$. However, in order to make any matrix positive semi definite, it is sufficient to perturb its diagonal terms. We can thus consider the perturbed matrix $Q_{\alpha,\lambda} = Q + \alpha AA^T + diag(\lambda)$, where $diag(\lambda)$ is a diagonal matrix with the elements of vector λ on the diagonal. We denote by $f_{\alpha,\lambda}(x,y)$ the associated function perturbed by the scalar parameter α and the vector parameter λ.

3 A Compact Quadratic Convex Reformulation (CQCR)

In this section, following the same reasoning steps as in MIQCR, we propose a convex reformulation of (QP) that leads to a reformulated program with a reduced size. The main starting idea is to perturb only the diagonal entries of Q, as described above, and thus to linearize only the squared variables x_i^2. For given parameters α and λ, let $(CQP_{\alpha,\lambda})$ be the following program:

$$
(CQP_{\alpha,\lambda}) \begin{cases}
\min\limits_{x,v,z,t} & f_{\alpha,\lambda}(x,v) = f(x) + \alpha\sum\limits_{r=1}^{m}(\sum\limits_{i=1}^{n} a_{ri}x_i - b_r)^2 + \sum\limits_{i=1}^{n}\lambda_i(x_i^2 - v_i) \\
s.t. & (1)(2)(3) \\
& x_i = \sum\limits_{k=0}^{\lfloor log(u_i)\rfloor} 2^k t_{ik} \qquad i \in I \qquad (23) \\
& z_{ik} \leq u_i t_{ik} \qquad (i,k) \in E \qquad (24) \\
& z_{ik} \leq x_i \qquad (i,k) \in E \qquad (25) \\
& z_{ik} \geq x_i - u_i(1 - t_{ik}) \qquad (i,k) \in E \qquad (26) \\
& z_{ik} \geq 0 \qquad (i,k) \in E \qquad (27) \\
& v_i = \sum\limits_{k=0}^{\lfloor log(u_i)\rfloor} 2^k z_{ik} \qquad i \in I \qquad (28) \\
& v_i \geq 2u_i x_i - u_i^2 \qquad i \in I \qquad (29) \\
& v_i \geq x_i \qquad i \in I \qquad (30) \\
& t_{ik} \in \{0,1\} \qquad (i,k) \in E \qquad (31)
\end{cases}
$$

Constraints (23) are identical to Constraints (5) of $(QP_{\alpha,\beta})$: they make a binary decomposition of x_i through the 0-1 variables t_{ik}. Then, $x_i^2 = \sum\limits_{k=0}^{\lfloor log(u_i)\rfloor} 2^k t_{ik} x_i$ can be written as $\sum\limits_{k=0}^{\lfloor log(u_i)\rfloor} 2^k z_{ik}$ using variables z_{ik} and Constraints (24)-(27) to get the equality $z_{ik} = t_{ik}x_i$. Finally, Constraints (28) ensure $v_i = x_i^2$. Constraint (29)-(30) strengthen the formulation. Hence, problem $(CQP_{\alpha,\lambda})$ is equivalent to (QP).

The advantage of this reformulation lies in the size of $(CQP_{\alpha,\lambda})$ that is of $O(N)$ variables and constraints, and is then about n times smaller than that of $(QP_{\alpha,\beta})$.

As in `MIQCR`, we are interested in the optimal convex reformulation within the new reformulation scheme, i.e. we look for parameters α and λ such that $f_{\alpha,\lambda}(x,v)$ is convex and the bound obtained by continuous relaxation of $(CQP_{\alpha,\lambda})$ is as large as possible. The following theorem provides a computation method for optimal parameters α^* and λ^*.

Theorem 1 *Let (SDP') be the following program:*

$$
(SDP') \begin{cases}
\min\limits_{X,x} & f(X,x) = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} q_{ij}X_{ij} + \sum\limits_{i=1}^{n} c_i x_i \\
s.t. & (1) \\
& (15)(20)(21)(22) \\
& X_{ii} \leq u_i x_i \qquad i \in I \qquad (32) \\
& X_{ii} \geq 2u_i x_i - u_i^2 \qquad i \in I \qquad (33) \\
& X_{ii} \geq 0 \qquad i \in I \qquad (34)
\end{cases}
$$

An optimal solution (α^*, λ^*) can be deduced from the optimal values of the dual variables of (SDP'). The optimal parameter α^* is the optimal value of the dual variable associated with Constraint (15). The optimal parameters λ^* are computed as $\lambda^* = \lambda^{1*} - \lambda^{2*} - \lambda^{3*} - \lambda^{4*}$, where λ^{1*}, λ^{2*}, λ^{3*}, and λ^{4*} are the optimal values of the dual variables associated with Constraints (32), (33), (34), and (20), respectively.

A proof can be deduced from [3] or [21]. We give here a sketch of the proof.
Sketch of proof.

1. Recall that we are searching for an optimal convex reformulation within our scheme. We are thus interested in the optimal value of $(\overline{CQP}_{\alpha,\lambda})$, where $(\overline{CQP}_{\alpha,\lambda})$ is the continuous relaxation of $(CQP_{\alpha,\lambda})$ (i.e. relaxation of Constraints (31))

   We then prove that the following program $(P_{\alpha,\lambda})$ is equivalent to $(\overline{CQP}_{\alpha,\lambda})$:

$$(P_{\alpha,\lambda}) \begin{cases} \min\limits_{x,v} & f_{\alpha,\lambda}(x,v) \\ s.c. & (1) \\ & (29)(30) \\ & v_i \geq 0 \qquad i \in I \quad (35) \\ & v_i \leq u_i x_i \qquad i \in I \quad (36) \\ & x \in \mathbb{R}^n, v \in \mathbb{R}^n \qquad (37) \end{cases}$$

   $(P_{\alpha,\lambda})$ is much smaller than $(\overline{CQP}_{\alpha,\lambda})$ since it does not contain the z and t variables, but it however has the same optimal value as $(\overline{CQP}_{\alpha,\lambda})$.

   We are now searching for the optimal parameters α^* and λ^* that both make $f_{\alpha^*,\lambda^*}(x,v)$ convex, and maximize the optimal value of $(\overline{CQP}_{\alpha,\lambda})$. This problem amounts to solve the following problem:

$$(CP): \quad \max_{\substack{\alpha\in\mathbb{R}, \quad \lambda\in\mathbb{R}^n \\ Q_{\alpha,\lambda}\succeq 0}} \{v(P_{\alpha,\lambda})\}$$

   where $v(P_{\alpha,\lambda})$ is the optimal solution value of $(P_{\alpha,\lambda})$ and $Q_{\alpha,\lambda}$ is the Hessian matrix of $f_{\alpha,\lambda}(x,v)$.

2. We then prove that $v(CP) = v(SDP')$

   (a) We prove that $v(CP) \leq v(SDP')$. More precisely, for any feasible solution (α, λ) to (CP), we show that $v(P_{\alpha,\lambda}) \leq v(SDP')$. For this, from a feasible solution

$(\bar{X}, \bar{x})$ of (SDP'), we deduce a feasible solution (x, v) to $(P_{\alpha,\lambda})$, that satisfies $f_{\alpha,\lambda}(x, v) \leq f(\bar{X}, \bar{x})$.

(b) We prove that $v(CP) \geq v(SDP')$, or equivalently that $v(CP) \geq v(DSDP')$, where $(DSDP')$ is the dual of (SDP'). For this, from any feasible solution to $(DSDP')$, we build a feasible solution to (CP), with a larger objective function value.

□

Problems (SDP) and (SDP') that allow to compute optimal parameter values for `MIQCR` and `CQCR`, respectively, are two different semi-definite relaxations of (QP). They differ from each other by their number of constraints. Observe that Constraints (32)-(34) of (SDP') represent the particular case $j = i$ in Constraints (16)-(19) of (SDP). Hence, (SDP') is a weaker semi-definite relaxation than (SDP), but it has $O(n)$ constraints while (SDP) has $O(n^2)$ constraints without counting the Constraints (1) and (21).

As in `MIQCR`, where the optimal value of the continuous relaxation of (QP_{α^*,β^*}) equals the optimal value of (SDP), the optimal value of the continuous relaxation of $(CQP_{\alpha^*,\lambda^*})$ is here equal to the optimal value of (SDP').

The binary variables case:

In this case $u_i = 1$ and Constraints (20),(32)-(34) amount to:

$$\begin{cases} X_{ii} \geq x_i & (20') \\ X_{ii} \leq x_i & (32') \\ X_{ii} \geq 2x_i - 1 & (33') \\ X_{ii} \geq 0 & (34') \end{cases}$$

Constraints (20') and (32') imply $X_{ii} = x_i$. Consequently, Constraints (33') and (34') become $0 \leq X_{ii} \leq 1$ that are redundant with the combination of Constraint $X_{ii} = x_i$ and (21). Thus, Constraints (20),(32)-(34) can be replaced by $X_{ii} = x_i$. Similarly, in the reformulated problem $(CQP_{\alpha,\lambda})$, Constraints (23)-(31) amount to $x_i = v_i = z_{i0} = t_{i0}$ and $t_{i0} \in \{0, 1\}$. All this is in coherence with the identity $x_i^2 = x_i$ for binary variables. We claim that `CQCR` is equivalent to `QCR` [5] for equality constrained binary quadratic programming, that is a method specially designed for this class of problem. However, `CQCR` constitutes an improvement of `QCR`, in terms of continuous relaxation bound, for inequality constrained binary quadratic programming. Indeed, using integer slack variables, it allows to transform each inequality into an equality and to consider these new equality constraints in the convexification process.

As a conclusion of this section, we present the exact solution algorithm for general integer non-convex quadratic programs (QP) based on `CQCR` and described in Algorithm 1.

Algorithm 1 Solution algorithm to (QP) based on `CQCR`

step 1: Solve (SDP').

step 2: Deduce α^* and λ^*.

step 3: Solve $(CQP_{\alpha^*,\lambda^*})$ with a standard mixed-integer quadratic solver.

As already mentioned above, `CQCR` has the same main steps as `MIQCR`. It is based on the solution of a semi-definite relaxation followed by the solution of a reformulated problem. On the one hand, `CQCR` relies on a weaker semi-definite relaxation than `MIQCR`. On the other hand, both the semi-definite problem (SDP'), and the reformulated problem $(CQP_{\alpha^*,\lambda^*})$ are about n times smaller in `CQCR` than in `MIQCR`.

In the following section, we present experiments that give a measurement of the global efficiency of `CQCR` compared to `MIQCR` and to the general mixed-integer non-linear solver BARON.

4 Computational results

In this section, we perform our experiments on instances of general integer quadratic programs with one equality constraint. Then, we consider the particular class of binary quadratic programs on instances of the Constrained Task Assignment Problem.

Experimental environment:

Our experiments were carried out on a PC with an Intel core $i7$ processor of 1.73 GHz and 6 GB of RAM using a Linux operating system for `CQCR` and `MIQCR`, and a Windows operating system for BARON. We used the solver CSDP [6] for the semi-definite programs. We used the solver Cplex version 12 [18] for solving the reformulated problems of `CQCR` and `MIQCR`, and for the solver BARON.

4.1 Experiments on the Equality Integer Quadratic Problem $(EIQP)$

Instances description:

Our experiments concern the Equality Integer Quadratic Problem $(EIQP)$ that consists of minimizing a quadratic function subject to one linear equality constraint:

$$(EIQP)\begin{cases} \min\limits_{x} & x^TQx + c^Tx \\ s.t. & \sum\limits_{i=1}^{n} a_i x_i = b \\ & 0 \le x_i \le u_i \quad i \in I \\ & x_i \in \mathbb{N} \quad\quad\; i \in I \end{cases}$$

We generate three classes of instances $(EIQP_1)$, $(EIQP_2)$ and $(EIQP_3)$. These instances are available online [8].

Instances from class $(EIQP_1)$ and $(EIQP_2)$ were already used in [3, 21], and instances of class $(EIQP_3)$ were already used in [21]. These instances are randomly generated as follows:

$(EIQP_1)$:

- The coefficients of Q and c are integers uniformly distributed in the interval $[-100, 100]$. More precisely, for any $i \le j$, a number ν is generated in $[-100, 100]$, and then we set $q_{ij} = q_{ji} = \nu$. Q is hence a full dense symmetric matrix with integer coefficient in $[-100, 100]$.
- The a_i coefficients are integers uniformly distributed in the interval $[1, 50]$.
- $b = 15 * \sum\limits_{i=1}^{n} a_i$
- $u_i = 30$, $i \in I$.

$(EIQP_2)$:

- The coefficients of Q and c are randomly generated as for $(EIQP_1)$.
- The a_i coefficients are integers uniformly distributed in the interval $[1, 100]$.
- $b = 20 * \sum\limits_{i=1}^{n} a_i$

- $u_i = 50$, $i \in I$.

$(EIQP_3)$:

- The coefficients of Q and c are randomly generated as for $(EIQP_1)$.
- The a_i and b are randomly generated as for $(EIQP_2)$.
- $u_i = 70$, $i \in I$.

For classes $(EIQP_1)$, $(EIQP_2)$, and $(EIQP_3)$, and for each $n = 20$, 30, or 40, we generate 5 instances obtaining a total of 45 instances. Each of these instances has at least one solution since $x_i = b/\sum_{i=1}^{n} a_i$ for all i is feasible.

Experimental results:

For these instances, we first compare the solution time of the whole process of methods `CQCR` and `MIQCR` with the solution time of the general mixed-integer non-linear solver BARON. Then, we compare `CQCR` and `MIQCR` on several criterias : the initial gap, the SDP solution time, the solution time after convex reformulation, and the number of nodes visited for each approach.

The results are presented in Tables 1 and 2.

Legends of Table 1:

- *Name*: $EIQP_k$_`n`_`i`, for $k = \{1, 2, 3\}$, where `k` is the class of the instance, `n` is the number of variables, and `i` the number of the instance.
- *Opt*: The optimal solution value of the instance.
- BARON, `MIQCR`, or `CQCR`: CPU time (in seconds) required by all the process for `CQCR` and `MIQCR`, i.e. solution time of the semi-definite relaxation + solution time of the reformulated problem, and CPU time (in seconds) required by BARON for solving the instance. If the optimum is not found within 2 hours of CPU time, we present the final gap of BAR0N (g%), where $g = \frac{\text{upperbound}-\text{lowerbound}}{\text{upperbound}} * 100$.

Legends of Table 2:

- *Name*: $EIQP_k$_n_i, for $k = \{1, 2, 3\}$, where k is the class of the instance, n is the number of variables, and i the number of the instance.
- *ig (initial gap)*: $\left|\frac{Opt - l}{Opt}\right| * 100$ where l is the optimal value of the continuous relaxation at the root node.
- *T CSDP*: CPU time (in seconds) required by CSDP for solving the semidefinite relaxation.
- *T Cplex*: CPU time (in seconds) required by Cplex for solving the convex integer quadratic program after convex reformulation.
- *Nodes*: Number of nodes visited during the Branch and Cut algorithm

Table 1 focuses on the comparison of solution times. We observe that both MIQCR and CQCR are able to solve all the instances of this class of problems in less that 2 hours of CPU time, whereas BARON solves only 27 of the 45 instances considered. It is then clear that for these classes of general integer quadratic instances MIQCR and CQCR are better suited than BARON. Moreover, the total solution time is smaller for CQCR in comparison to MIQCR. The average total solution time is divided for $(EIQP_1)$ (resp. $(EIQP_2)$ and $(EIQP_3)$) by a factor 320 (resp. 339 and 81) with CQCR in comparison to MIQCR.

An additional comparison between approaches MIQCR and CQCR is presented in Table 2. As mentioned in Section 3, CQCR leads to a reformulated problem with a weaker continuous relaxation bound than MIQCR. For class $(EIQP_1)$ (resp. $(EIQP_2)$ and $(EIQP_3)$) the average gap of MIQCR is 18 (resp. 95 and 33) times smaller than that of CQCR.

However, the computation time of the solution of the SDP relaxation by the CSDP solver is significantly smaller for CQCR. Indeed, for $(EIQP_1)$ (resp. $(EIQP_2)$ and $(EIQP_3)$) the average *CSDP* solution time is divided by a factor 763 (resp. 1245 and 1101) in comparison to MIQCR.

If we focus on the computation time after convex reformulation, that is to say the solution time of the integer quadratic convex problem by the solver Cplex, the results reveal a similar trend than for the SDP solution time. Indeed, the average *Cplex* solution time over all the instances is divided for $(EIQP_1)$ (resp. $(EIQP_2)$ and $(EIQP_3)$) by a factor 131 (resp. 124 and 19) for CQCR in comparison to MIQCR.

Table 1: Solution times or final gaps after 2 hours for the 45 instances of class ($EIQP$) with BARON, MIQCR and CQCR

Name	*Opt*	BARON	MIQCR	CQCR
$EIQP_1$_20_1	-5311070	5.07	55.56	1.25
$EIQP_1$_20_2	-5098379	2.78	40.37	1.28
$EIQP_1$_20_3	-4554397	14.19	56.23	1.27
$EIQP_1$_20_4	-5614860	1.23	35.30	0.26
$EIQP_1$_20_5	-4354396	12.41	49.82	1.28
Average		**7.14**	**47.46**	**1.07**
$EIQP_1$_30_1	-10210390	1026.93	444.86	1.66
$EIQP_1$_30_2	-11243370	46.91	321.71	1.75
$EIQP_1$_30_3	-9862120	527.87	589.59	2.70
$EIQP_1$_30_4	-10720488	2552.91	382.59	1.65
$EIQP_1$_30_5	-10835084	1965.84	494.92	2.69
Average		**1224.09**	**446.74**	**2.09**
$EIQP_1$_40_1	-20907112	98.34	4730.20	2.37
$EIQP_1$_40_2	-21274411	(3.49 %)	2243.17	3.23
$EIQP_1$_40_3	-17033610	(11.56 %)	1861.70	2.26
$EIQP_1$_40_4	-18268074	(5.23 %)	2718.84	6.23
$EIQP_1$_40_5	-17373411	(30.54 %)	2751.04	6.28
Average		**98.34 (1)**	**2860.99**	**4.07**
$EIQP_2$_20_1	-9321876	153.00	183.08	1.25
$EIQP_2$_20_2	-9013418	107.03	57.07	0.24
$EIQP_2$_20_3	-15337225	2.70	55.13	1.25
$EIQP_2$_20_4	-11863777	19.86	109.12	2.26
$EIQP_2$_20_5	-12095004	22.70	51.69	1.26
Average		**61.06**	**91.22**	**1.25**
$EIQP_2$_30_1	-23592535	3550.27	642.13	3.66
$EIQP_2$_30_2	-25924713	216.01	867.61	2.69
$EIQP_2$_30_3	-21938906	7188.62	910.34	2.63
$EIQP_2$_30_4	-29913305	193.46	827.72	3.65
$EIQP_2$_30_5	-22422891	(10.80 %)	668.45	3.65
Average		**2787.09 (4)**	**783.25**	**3.26**
$EIQP_2$_40_1	-42548497	(23.86 %)	2600.88	7.29
$EIQP_2$_40_2	-35957464	(33.91 %)	7529.19	8.30
$EIQP_2$_40_3	-40116963	(27.40 %)	3142.81	9.33
$EIQP_2$_40_4	-51306080	186.50	5599.46	2.35
$EIQP_2$_40_5	-38090192	(31.90 %)	5432.43	7.28
Average		**186.50 (1)**	**4860.95**	**6.91**
$EIQP_3$_20_1	-13226046	61.04	158.70	0.27
$EIQP_3$_20_2	-16400092	74.26	40.53	0.27
$EIQP_3$_20_3	-13372984	78.05	73.88	1.27
$EIQP_3$_20_4	-9904855	1610.04	137.40	3.25
$EIQP_3$_20_5	-10903367	183.35	52.23	1.27
Average		**401.35**	**92.55**	**1.27**
$EIQP_3$_30_1	-24412436	1003.52	366.65	4.62
$EIQP_3$_30_2	-25640775	(26.99 %)	669.68	8.62
$EIQP_3$_30_3	-23342586	(17.38 %)	505.95	4.65
$EIQP_3$_30_4	-29843855	(11.50 %)	914.40	3.65
$EIQP_3$_30_5	-26911633	(17.55 %)	1083.74	12.65
Average		**1003.52 (1)**	**708.08**	**6.84**
$EIQP_3$_40_1	-50352748	(39.58 %)	2691.66	10.27
$EIQP_3$_40_2	-46862608	(62.49 %)	2676.46	10.29
$EIQP_3$_40_3	-51680153	(57.19 %)	2584.19	7.33
$EIQP_3$_40_4	-49068049	(58.06 %)	4002.21	159.29
$EIQP_3$_40_5	-36454613	(127.24 %)	6428.02	88.23
Average		**- (0)**	**3676.51**	**55.08**

(i) : i instances out of 5 were solved within the time limit

Table 2: Comparison of `MIQCR` and `CQCR` on the 45 instances of class $(EIQP)$

	MIQCR				CQCR			
Name	*ig (%)*	*T CSDP*	*T Cplex*	*Nodes*	*ig (%)*	*T CSDP*	*T Cplex*	*Nodes*
$EIQP_1$_20_1	0.09	35.56	20.00	2657	1.24	0.25	1.00	4647
$EIQP_1$_20_2	0.13	30.37	10.00	368	0.24	0.28	1.00	918
$EIQP_1$_20_3	0.06	33.23	23.00	1	1.38	0.27	1.00	1111
$EIQP_1$_20_4	0	33.30	2.00	0	0.21	0.26	0	56
$EIQP_1$_20_5	0.15	35.82	14.00	165	2.30	0.28	1.00	1057
Average	**0.09**	**33.66**	**13.80**	**638**	**1.07**	**0.27**	**0.80**	**1557**
$EIQP_1$_30_1	0.04	352.86	92.00	51	1.36	0.66	1.00	1416
$EIQP_1$_30_2	0.00	312.71	9.00	0	0.49	0.75	1.00	50
$EIQP_1$_30_3	0.04	426.59	163.00	1125	1.55	0.70	2.00	2901
$EIQP_1$_30_4	0.05	359.59	23.00	61	1.10	0.65	1.00	929
$EIQP_1$_30_5	0.09	342.92	152.00	612	1.09	0.69	2.00	3076
Average	**0.05**	**358.94**	**87.80**	**370**	**1.12**	**0.69**	**1.40**	**1674**
$EIQP_1$_40_1	0.04	1904.20	2826.00	1377	0.50	1.37	1.00	2095
$EIQP_1$_40_2	0.04	1861.17	382.00	1253	1.54	1.23	2.00	2650
$EIQP_1$_40_3	0	1832.70	29.00	0	1.34	1.26	1.00	812
$EIQP_1$_40_4	0.04	2277.84	441.00	3750	1.70	1.23	5.00	8658
$EIQP_1$_40_5	0.21	2056.04	695.00	3481	2.19	1.28	5.00	11894
Average	**0.07**	**1986.39**	**874.60**	**1972**	**1.46**	**1.27**	**2.80**	**5222**
$EIQP_2$_20_1	0.14	41.08	142.00	4027	3.10	0.25	1.00	4359
$EIQP_2$_20_2	0.00	51.07	6.00	8	3.89	0.24	0	498
$EIQP_2$_20_3	0.03	33.13	22.00	559	1.35	0.25	1.00	1454
$EIQP_2$_20_4	0.19	57.12	52.00	2502	2.67	0.26	2.00	4473
$EIQP_2$_20_5	0.08	43.69	8.00	48	2.96	0.26	1.00	1277
Average	**0.09**	**45.22**	**46.00**	**1429**	**2.79**	**0.25**	**1.00**	**2412**
$EIQP_2$_30_1	0.25	507.13	135.00	1089	3.06	0.66	3.00	6111
$EIQP_2$_30_2	0.11	312.61	555.00	2144	0.65	0.69	2.00	4904
$EIQP_2$_30_3	0.01	357.34	553.00	1190	2.32	0.63	2.00	8452
$EIQP_2$_30_4	0.05	335.72	492.00	4140	0.78	0.65	3.00	6002
$EIQP_2$_30_5	0.10	462.45	206.00	1560	3.20	0.65	3.00	6419
Average	**0.10**	**395.05**	**388.20**	**2025**	**2.00**	**0.66**	**2.60**	**6378**
$EIQP_2$_40_1	0.02	2125.88	475.00	986	1.18	1.29	6.00	9086
$EIQP_2$_40_2	0.05	4838.19	2691.00	6568	1.20	1.30	7.00	13363
$EIQP_2$_40_3	0.05	2086.81	1056.00	5730	1.17	1.33	8.00	13346
$EIQP_2$_40_4	0.00	5377.46	222.00	0	0.55	1.35	1.00	86
$EIQP_2$_40_5	0.04	4932.43	500.00	2695	2.53	1.28	6.00	10845
Average	**0.03**	**3872.15**	**988.80**	**3196**	**1.33**	**1.31**	**5.60**	**9345**
$EIQP_3$_20_1	0	158.70	0	8	2.94	0.27	0	1488
$EIQP_3$_20_2	0.03	36.53	4.00	8	4.31	0.27	0	1714
$EIQP_3$_20_3	0.16	43.88	30.00	522	5.01	0.27	1.00	2995
$EIQP_3$_20_4	1.60	75.40	62.00	4412	11.27	0.25	3.00	7323
$EIQP_3$_20_5	0.07	47.23	5.00	51	7.49	0.27	1.00	1922
Average	**0.37**	**72.35**	**20.20**	**1000**	**6.20**	**0.27**	**1.00**	**3088**
$EIQP_3$_30_1	0.02	350.65	16.00	11	6.85	0.62	4.00	7516
$EIQP_3$_30_2	0.23	410.68	259.00	5304	7.47	0.62	8.00	19398
$EIQP_3$_30_3	0.05	394.95	111.00	3829	3.60	0.65	4.00	7581
$EIQP_3$_30_4	0.04	845.40	69.00	936	3.06	0.65	3.00	4982
$EIQP_3$_30_5	0.31	926.74	157.00	4297	6.77	0.65	12.00	29532
Average	**0.13**	**585.68**	**122.40**	**2875**	**5.55**	**0.64**	**6.20**	**13802**
$EIQP_3$_40_1	0.01	2276.66	415.00	79	2.09	1.27	9.00	12784
$EIQP_3$_40_2	0	2311.46	365.00	950	2.75	1.29	9.00	11036
$EIQP_3$_40_3	0.04	1910.19	674.00	2191	2.30	1.33	6.00	6012
$EIQP_3$_40_4	0.03	2322.21	1680.00	6604	6.08	1.29	158.00	245712
$EIQP_3$_40_5	0.48	4753.02	1675.00	15584	7.98	1.23	87.00	121551
Average	**0.11**	**2714.71**	**961.80**	**5082**	**1.28**	**4.24**	**53.80**	**79419**

Hence, although `MIQCR` provides a much better bound, `CQCR` is more effective as it solves faster all the 45 considered instances.

4.2 Experiments on the Constrained Task Assignment Problem (CTAP)

Description of the problem:

The Constrained Task Assignment Problem (CTAP) consists in finding an assignment of tasks (facilities) to processors (locations) such that the memory constraints are satisfied, and such that the total execution and communication cost is minimized. Problem CTAP is a special case of the Generalized Quadratic Assignment Problem (GQAP). This problem describes a broad class of binary programming problems, where M pair-wise related entities must be assigned to N destinations constrained by the destinations' ability to accommodate them. The GQAP has numerous applications, including facility design, scheduling and network design.

Several authors proposed algorithms specialized for solving the GQAP, as in [16, 22]. The exact algorithm of Hahn and al. [16] is an algorithm based on a Reformulation Linearization Technique [27] dual ascent procedure. The heuristic of Mateus and al. [22] is based on the meta-heuristic GRASP [10], with path-relinking [20, 24].

We now describe more formally problem CTAP:

- A set of n tasks
- A set of m processors
- The execution cost e_{ik} of task i on processor k
- The communication cost c_{ij} between tasks i and j if they are assigned to different processors
- The memory requirement s_i of task i
- The total available memory n_k of processor k. The sum of memory requirements of the tasks assigned to processor k must not exceed n_k.

A natural mathematical formulation of CTAP can be considered by taking the variable vector $x = (x_{ik})$, $i = 1, \ldots, n, k = 1, \ldots, m$ where x_{ik} is equal to 1 if task i is allocated to processor k and is equal to 0 otherwise.

Table 3: Four configurations of the CTAP instances

	Config A	Config B	Config C	Config D
int_e	[0,100]	[0,10]	[0,100]	[0,0]
int_c	[0,100]	[0,100]	[0,10]	[0,100]

Let $c_0 = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c_{ij}$. The CTAP can be formulated by the following binary quadratic program [9]:

$$
(CTAP) \begin{cases} \min_x \quad f(x) = c_0 + \sum_{i=1}^{n}\sum_{k=i}^{m} e_{ik}x_{ik} - \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=1}^{m} c_{ij}x_{ik}x_{jk} & \\ s.t. \quad \sum_{k=1}^{m} x_{ik} = 1 & i = 1,\ldots,n \quad (38) \\ \sum_{i=1}^{n} s_i x_{ik} \leq n_k & k = 1,\ldots,m \quad (39) \\ x \in \{0,1\}^{n\times m} & i \in I \quad (40) \end{cases}
$$

Instances description:

We used the instances that were produced in [9] and are available online [28]. In these instances, 4 configurations are considered. For each configuration, two classes of instances are randomly generated: a class with a complete communication graph, called *tassc*, and a second class where the density of the communication graph is 50%, called *tass*. This gives a total of 8 types of instances. For each type, 5 instances of size 10 tasks and 3 processors, 5 instances of size 20 tasks and 5 processors, and 5 instances of size 24 tasks and 8 processors are generated. We obtain a total of 120 instances. Note that several instances of size 24 tasks and 8 processors are not feasible, this is why we report here the results of the 24 feasible instances over the 40 initially generated.

Table 3 describes the 4 configurations. The execution costs e_{ik} are integers generated in the interval int_e, and the communication costs c_{ij} are integers generated in the interval int_c. For all the configurations, the sizes of the tasks s_i are integers generated in the interval $[1, 10]$, and the capacities of the processors n_k are integers in the interval $[S/m, 2 * S/m]$ where $S = \sum_{i=1}^{n} s_i$ is the sum of all the task sizes. In this way, we are sure that the problem

$(CTAP)$ has at least one fractional solution $\bar{x}$ where $\forall(i,k)$, $\bar{x}_{ik} = \frac{1}{m}$. The inequality constraints (39) are reformulated as equalities by use of slack variables e_k that are integers in the interval $[0, n_k]$.

Experimental results

Here we compare MIQCR with our new approach CQCR.

Legends of Table 4:

- *Name*: pnmgi, where p is the problem name, n the number of tasks, m the number of processors, g the kind of generation as explained above, and i the instance letter.
- *Opt*: The optimal solution value of the instance.
- MIQCR or CQCR: CPU time (in seconds) required by all the process, i.e. solution time of the semi-definite relaxation + solution time of the reformulated problem.

Legends of Tables 5-7:

- *Name*: pnmgi, where p is the problem name, n the number of tasks, m the number of processors, g the kind of generation as explained above, and i the instance letter.
- *Opt*: The optimal solution value of the instance.
- *Sol*: The best solution value found within the time limit.
- *T CSDP*: CPU time (in seconds) required by CSDP for solving the semidefinite relaxation. More precisely, the solution time for solving (SDP) in MIQCR, and the solution time for solving (SDP') in CQCR.
- *ig (initial gap)*: $\left|\dfrac{Opt - l}{Opt}\right| * 100$ where, in MIQCR, l is the optimal value of the continuous relaxation of (QP_{α^*,β^*}), and, in CQCR, l is the optimal value of the continuous relaxation of $(CQP_{\alpha^*,\lambda^*})$.
- *T Cplex*: CPU time (in seconds) required for solving the convex reformulations of (QP). More precisely, the solution time of Cplex for solving (QP_{α^*,β^*}) in MIQCR, and the solution time for solving $(CQP_{\alpha^*,\lambda^*})$ in CQCR. The time limit is fixed to 2 hours in

both cases. If the optimum is not found within 2 hours of CPU time, we report the final gap of CQCR (g%), where $g = \left|\frac{Opt - lb}{Opt}\right| * 100$, where lb is the best bound found after 2 hours of CPU time, and Opt is the optimal solution value of the instance [28].

- *Nodes*: number of nodes visited during each Branch and Bound procedure (MIQCR or CQCR).

Only CQCR is able to handle instances of size larger than 10 tasks and 3 processors. This is why for classes of problems *tass* and *tassc* 2005 and 2408, we do not present the results of MIQCR. In these cases MIQCR is limited by the huge size of its semidefinite relaxation that cannot be handled by CSDP.

The comparison between the solution time of MIQCR and CQCR is presented in Table 4 for classes of problems *tass* and *tassc* of size 10 tasks and 3 processors. We observe that CQCR is much faster than MIQCR. Indeed, the average solution time of CQCR is divided by a factor of 1891 (resp. by a factor 6579) for class (*tass*)1003 (resp. (*tassc*)1003) in comparison to MIQCR.

An additional comparison between MIQCR and CQCR for classes of problems (*tass*) and (*tassc*) 1003 is presented in Tables 5. First, as expected, we observe that MIQCR gives a better continuous relaxation bound than CQCR. Indeed, the average gap over all the instances is multiplied by a factor of about 4 for CQCR in comparison to MIQCR. We observe that the semidefinite relaxation of CQCR is much faster to solve than the MIQCR one. The average semidefinite solution time over all the instances is improved for CQCR by a factor of about 6574 in comparison to MIQCR. We now focus on the computation time after convex reformulation, that is to say the solution time to solve the quadratic and convex reformulated program of MIQCR and CQCR by Cplex. The average solution time of CQCR is improved by a factor of about 76 in comparison to MIQCR.

Results of classes *tass* and *tassc* of size 20 tasks and 5 processors and of size 24 tasks and 8 processors are presented in Tables 6 and 7, respectively. First, we observe that CQCR is able to solve all the instances of size 20 tasks and 5 processors in less than 2 hours of CPU time, and 6 instances over the 24 instances of size 24 tasks and 8 processors. In [16], Hahn and al. make experiences on instances *tass*2005Aa, *tassc*2005De, *tass*2408Aa and *tass*2408Ca. In their paper, with a specialized approach to solve GQAP, they solved *tass*2005Aa in 128 s. (92.55 s. with CQCR), *tassc*2005De in 14390 s. (887.63 s. with CQCR), *tass*2408Aa in 719862 s. (about 200 hours) (we obtain a final gap of 0.09% in 7200 s. for CQCR). For the instance

Table 4: Solution times of the 40 instances of classes *tass* and *tassc* 1003 with MIQCR and CQCR

	Opt	MIQCR	CQCR
*tass*1003Aa	731	476.71	0.10
*tass*1003Ab	713	469.89	0.06
*tass*1003Ac	645	391.19	0.06
*tass*1003Ad	688	422.69	1.06
*tass*1003Ae	715	398.31	0.06
Average		**431.76**	**0.27**
*tass*1003Ba	306	452.10	0.06
*tass*1003Bb	528	402.51	0.06
*tass*1003Bc	326	356.38	1.05
*tass*1003Bd	364	391.12	0.06
*tass*1003Be	324	428.81	0.06
Average		**406.18**	**0.26**
*tass*1003Ca	346	376.80	0.06
*tass*1003Cb	424	377.58	0.05
*tass*1003Cc	347	235.18	1.05
*tass*1003Cd	434	386.67	0.05
*tass*1003Ce	285	258.56	0.06
Average		**326.96**	**0.26**
*tass*1003Da	219	466.29	0.06
*tass*1003Db	402	392.82	0.05
*tass*1003Dc	297	416.33	0.05
*tass*1003Dd	445	438.40	0.05
*tass*1003De	358	405.80	0.05
Average		**423.93**	**0.05**
*tassc*1003Aa	1616	399.98	0.06
*tassc*1003Ab	1390	385.31	0.07
*tassc*1003Ac	1730	418.38	0.06
*tassc*1003Ad	1289	438.16	0.05
*tassc*1003Ae	1048	439.55	0.07
Average		**416.27**	**0.06**
*tassc*1003Ba	1299	425.97	0.07
*tassc*1003Bb	865	401.06	0.06
*tassc*1003Bc	1154	444.86	0.06
*tassc*1003Bd	834	408.11	0.06
*tassc*1003Be	812	406.86	0.06
Average		**417.37**	**0.06**
*tassc*1003Ca	455	424.39	0.06
*tassc*1003Cb	467	328.23	0.05
*tassc*1003Cc	475	344.60	0.05
*tassc*1003Cd	472	233.40	0.06
*tassc*1003Ce	350	249.19	0.06
Average		**315.96**	**0.06**
*tassc*1003Da	843	436.75	0.06
*tassc*1003Db	879	400.76	0.06
*tassc*1003Dc	1230	439.47	0.06
*tassc*1003Dd	956	453.50	0.06
*tassc*1003De	848	416.21	0.06
Average		**429.34**	**0.06**

Table 5: Comparison of `MIQCR` and `CQCR` on the 40 instances of classes *tass* and *tassc* 1003

		MIQCR				CQCR			
	opt	ig (%)	T CSDP	T Cplex	Nodes	ig (%)	T CSDP	T Cplex	Nodes
*tass*1003Aa	731	8.17	474.71	2.00	70	22.45	0.10	0	339
*tass*1003Ab	713	1.99	468.89	1.00	10	14.75	0.06	0	143
*tass*1003Ac	645	8.36	389.19	2.00	80	22.52	0.06	0	409
*tass*1003Ad	688	0.22	421.69	1.00	7	16.77	0.06	1.00	191
*tass*1003Ae	715	5.44	397.31	1.00	135	20.61	0.06	0	379
Average		**4.84**	**430.36**	**1.40**	**60**	**19.42**	**0.07**	**0.20**	**292**
*tass*1003Ba	306	10.27	450.10	2.00	40	35.68	0.06	0	242
*tass*1003Bb	528	8.80	399.51	3.00	172	31.11	0.06	0	372
*tass*1003Bc	326	0	354.38	2.00	113	31.50	0.05	1.00	656
*tass*1003Bd	364	9.84	388.12	3.00	120	37.31	0.06	0	143
*tass*1003Be	324	0	427.81	1.00	39	41.06	0.06	0	155
Average		**5.78**	**403.98**	**2.20**	**97**	**35.33**	**0.06**	**0.20**	**314**
*tass*1003Ca	346	0.01	375.80	1.00	0	4.08	0.06	0	25
*tass*1003Cb	424	1.96	376.58	1.00	0	3.60	0.05	0	0
*tass*1003Cc	347	0	235.18	0	0	0.85	0.05	1.00	1
*tass*1003Cd	434	3.72	385.67	1.00	13	4.56	0.05	0	7
*tass*1003Ce	285	0	258.56	0	0	2.62	0.06	0	18
Average		**1.14**	**326.36**	**0.60**	**3**	**3.14**	**0.06**	**0.20**	**10**
*tass*1003Da	219	0	464.29	2.00	19	45.88	0.06	0	38
*tass*1003Db	402	0	389.82	3.00	385	27.95	0.05	0	977
*tass*1003Dc	297	14.95	414.33	2.00	250	34.88	0.05	0	512
*tass*1003Dd	445	5.33	436.40	2.00	142	32.79	0.05	0	234
*tass*1003De	358	12.54	403.80	2.00	121	37.43	0.05	0	582
Average		**6.56**	**421.73**	**2.20**	**183**	**35.79**	**0.05**	**0**	**469**
*tassc*1003Aa	1616	7.62	397.98	2.00	224	15.50	0.06	0	321
*tassc*1003Ab	1390	0	385.31	0	0	13.95	0.07	0	223
*tassc*1003Ac	1730	0.80	417.38	1.00	15	7.33	0.06	0	74
*tassc*1003Ad	1289	3.90	437.16	1.00	49	12.35	0.05	0	183
*tassc*1003Ae	1048	2.51	438.55	1.00	21	12.28	0.07	0	133
Average		**2.97**	**415.27**	**1.00**	**62**	**12.28**	**0.06**	**0**	**187**
*tassc*1003Ba	1299	1.73	423.97	2.00	80	15.50	0.07	0	172
*tassc*1003Bb	865	0	399.06	2.00	199	33.61	0.06	0	600
*tassc*1003Bc	1154	11.84	441.86	3.00	276	23.47	0.06	0	351
*tassc*1003Bd	834	13.39	406.11	2.00	206	37.61	0.06	0	496
*tassc*1003Be	812	16.48	404.86	2.00	88	32.04	0.06	0	219
Average		**8.69**	**415.17**	**2.20**	**170**	**28.44**	**0.06**	**0**	**368**
*tassc*1003Ca	455	1.78	423.39	1.00	23	4.32	0.06	0	30
*tassc*1003Cb	467	1.32	327.23	1.00	3	3.54	0.05	0	19
*tassc*1003Cc	475	0.64	344.60	0	1	3.38	0.05	0	8
*tassc*1003Cd	472	0	233.40	0	0	1.24	0.06	0	1
*tassc*1003Ce	350	0	249.19	0	0	5.53	0.06	0	12
Average		**0.75**	**315.56**	**0.40**	**5**	**3.60**	**0.06**	**0**	**14**
*tassc*1003Da	843	0.73	435.75	1.00	16	18.84	0.06	0	119
*tassc*1003Db	879	4.93	398.76	2.00	20	21.50	0.06	0	248
*tassc*1003Dc	1230	11.24	436.47	3.00	305	32.06	0.06	0	715
*tassc*1003Dd	956	4.52	451.50	2.00	43	17.55	0.06	0	122
*tassc*1003De	848	19.85	414.21	2.00	313	35.45	0.06	0	684
Average		**8.25**	**427.34**	**2.00**	**139**	**25.08**	**0.06**	**0**	**378**

Table 6: Solution of the 40 instances of classes *tass* and *tassc* 2005 with `CQCR`

		CQCR			
	Opt	*ig (%)*	*T CSDP*	*T Cplex*	*Nodes*
*tass*2005Aa	3059	15.90	0.55	92.00	156137
*tass*2005Ab	2954	15.74	0.62	65.00	128105
*tass*2005Ac	3012	28.52	0.60	441.00	940313
*tass*2005Ad	3174	19.44	0.58	418.00	777018
*tass*2005Ae	3054	28.46	0.64	807.00	1613380
Average		**21.61**	**0.60**	**364.60**	**722991**
*tass*2005Ba	2442	24.96	0.59	2565.00	4098384
*tass*2005Bb	2088	26.77	0.67	219.00	387686
*tass*2005Bc	1986	45.51	0.67	466.00	987663
*tass*2005Bd	2449	35.44	0.67	1273.00	2640252
*tass*2005Be	2453	22.84	0.58	106.00	191263
Average		**31.10**	**0.64**	**925.80**	**1661050**
*tass*2005Ca	783	3.50	0.59	2.00	934
*tass*2005Cb	636	6.96	0.57	3.00	6962
*tass*2005Cc	772	4.96	0.58	2.00	3671
*tass*2005Cd	682	2.56	0.58	0	47
*tass*2005Ce	732	3.17	0.56	1.00	495
Average		**0.58**	**4.23**	**1.60**	**2422**
*tass*2005Da	2413	27.89	0.61	2210.00	3727651
*tass*2005Db	2316	30.95	0.59	636.00	1085785
*tass*2005Dc	1965	45.04	0.61	313.00	624536
*tass*2005Dd	2211	38.91	0.62	786.00	1467529
*tass*2005De	2302	30.22	0.57	408.00	668291
Average		**0.60**	**34.60**	**870.60**	**1514758**
*tassc*2005Aa	6412	20.39	0.66	641.00	1465539
*tassc*2005Ab	6260	10.50	0.58	11.00	22654
*tassc*2005Ac	6491	13.30	0.65	14.00	35426
*tassc*2005Ad	6267	16.17	0.63	78.00	147410
*tassc*2005Ae	6194	12.92	0.64	36.00	76297
Average		**14.65**	**0.63**	**156.00**	**349465**
*tassc*2005Ba	5420	21.53	0.68	178.00	372343
*tassc*2005Bb	5370	20.69	0.61	129.00	242784
*tassc*2005Bc	5645	23.06	0.59	7096.00	13911347
*tassc*2005Bd	5420	21.64	0.60	257.00	488492
*tassc*2005Be	5836	28.63	0.61	539.00	1161577
Average		**23.11**	**0.62**	**1639.80**	**3235309**
*tassc*2005Ca	1181	7.26	0.58	34.00	52581
*tassc*2005Cb	1017	6.85	0.63	1.00	2994
*tassc*2005Cc	1197	9.38	0.62	23.00	43629
*tassc*2005Cd	1038	4.84	0.58	1.00	1167
*tassc*2005Ce	1166	5.58	0.58	5.00	10861
Average		**6.78**	**0.60**	**12.80**	**22246**
*tassc*2005Da	5139	17.22	0.58	19.00	36744
*tassc*2005Db	5519	20.97	0.61	882.00	1534662
*tassc*2005Dc	5907	13.31	0.58	289.00	499546
*tassc*2005Dd	5494	20.16	0.66	399.00	801283
*tassc*2005De	5435	25.48	0.63	887.00	1731446
Average		**19.43**	**0.61**	**495.20**	**920736**

Table 7: Solution of the 24 instances of classes *tass* and *tassc* 2408 with `CQCR`

		CQCR				
	Opt	*Sol*	*ig (%)*	*T CSDP*	*T Cplex*	*Nodes*
*tass*2408Aa	5643	5648	19.60	1.64	(0.09%)	3495146
*tass*2408Ab	5339	5339	20.21	1.63	(0.66 %)	3824791
*tass*2408Ac	4896	4919	21.95	1.50	(0.47 %)	4231497
*tass*2408Ae	5416	5416	22.12	1.64	3710	3435829
Average			**20.97**	**1.60**	**3710 (1)**	**3435829 (1)**
*tass*2408Ba	4654	4673	21.09	1.70	(0.41 %)	3906481
*tass*2408Bc	4173	4204	24.06	1.70	(0.74 %)	3947823
*tass*2408Be	4487	4487	25.50	1.68	(0 %)	4196913
Average			**23.55**	**1.69**	**- (0)**	**- (0)**
*tass*2408Ca	957	957	7.77	1.80	8.00	7938
*tass*2408Cc	1016	1016	7.20	1.92	2.00	1195
*tass*2408Ce	960	960	5.64	1.77	17.00	16700
Average			**6.87**	**1.83**	**9.00**	**8611**
*tass*2408Db	4743	4744	22.95	1.56	(0.02%)	4772083
*tass*2408Dc	4036	4068	30.41	1.62	(0.79%)	5874322
*tass*2408Dd	4169	4203	23.91	1.62	(0.82 %)	4157146
*tass*2408De	3963	3987	27.36	1.71	(0.61 %)	4440719
Average			**26.16**	**1.63**	**- (0)**	**- (0)**
*tassc*2408Ae	10359	10464	18.26	1.62	(1.01%)	4253030
Average			**18.26**	**1.62**	**- (0)**	**- (0)**
*tassc*2408Bc	10341	10372	12.53	1.62	(0.30%)	3360046
*tassc*2408Bd	10226	10274	11.29	1.62	(0.47%)	3145183
Average			**11.91**	**1.62**	**- (0)**	**- (0)**
*tassc*2408Cc	1641	1641	4.47	1.62	252.00	174279
*tassc*2408Cd	1520	1520	3.75	1.79	3.00	1918
Average			**4.11**	**1.71**	**127.5**	**88098**
*tassc*2408Da	10557	10562	13.79	1.75	(0.05 %)	3979150
*tassc*2408Db	10427	10516	19.10	1.63	(0.85 %)	4555744
*tassc*2408Dc	9202	9202	18.42	1.58	(0%)	4717833
*tassc*2408Dd	9312	9312	19.08	1.62	(0 %)	4621036
*tassc*2408De	9268	9363	18.86	1.70	(1.03 %)	4307644
Average			**17.85**	**1.66**	**- (0)**	**- (0)**

`(i)` : `i` instances out of 5 were solved within the time limit *tass*2408Ca, they spend 6.6 s. to obtain a solution value 1028 at 7.4% of the optimum, while we solve the instance in 9.8 s. with `CQCR`.

5 Conclusion

In this paper we have presented an efficient Compact Quadratic Convex Reformulation to solve general integer quadratic programs. This convex reformulation, called `CQCR`, consists in designing a new quadratic problem that is equivalent to the initial problem and that has a convex objective function. This reformulation is computed thanks to a semi-definite relaxation of the initial problem. `CQCR` is inspired of ideas of a more general quadratic convex reformulation, called `MIQCR`, that handles general mixed-integer quadratic programs. A drawback of `MIQCR` is the important size of both its semidefinite relaxation and its reformulated program. Our compact reformulation, `CQCR`, leads to a semidefinite relaxation and a reformulated problem both having much smaller sizes. However, the continuous relaxation value of `CQCR` is weaker than that of `MIQCR`. We evaluate `CQCR` from the computational point of view. We perform our experiences on two classes of instances. The first one concerns general integer programs with one linear equality constraint. We show that `CQCR` is significantly faster than `MIQCR` to solve the considered instances. The second class concerns binary quadratic programming, and more precisely the Constrained Task Assignment Problem (CTAP). Our results show that `CQCR` is a better approach in terms of computational time and is up to solve almost the considered instances in less than 2 hours of CPU time.

References

[1] Audet, C., Hansen, P., Savard, G.: Essays and Surveys in Global Optimization. GERAD 25th Anniversary Series, Springer, New York, (2005)

[2] Audet, C., Hansen, P., Jaumard, B., Savard, G.:Links Between Linear Bilevel and Mixed 0-1 Programming Problems. *Journal of Optimization Theory and Applications.* **93**(2) : 273-300, (1997)

[3] Billionnet, A., Elloumi, S., Lambert, A.: Extending the QCR method to the case of general mixed integer program. *Mathematical Programming.* Available online DOI: 10.1007/s10107-010-0381-7 (2010)

[4] Billionnet, A., Elloumi, S., Lambert, A.: Linear Reformulations of Integer Quadratic Programs. *MCO 2008, september 8-10*, 43-51 (2008)

[5] Billionnet, A., Elloumi, S., Plateau, M.-C.: 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)

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

[7] Cui, Y.: Dynamic programming algorithms for the optimal cutting of equal rectangles. Appl.Math. Model. **29**, 1040-1053 (2005)

[8] EIQP : `http://cedric.cnam.fr/~lambe_a1/siqp/Library/index.php`

[9] Elloumi, S. Contribution à la résolution des programmes non linéaires en variables 0-1, application aux problèmes de placement de tâches dans les systèmes distribués. Thèse de doctorat en informatique, Conservatoire National des Arts et Métiers, (1991)

[10] Feo, T.A., Resende, M.G.C.: A probabilistic heuristic for a computationally difficult set covering problem. *Oper. Res. Lett.* **8**, 67-71 (1989)

[11] Floudas, C.A.: Deterministic Global Optimization. Kluwer Academic Publishing, Dordrecht, The Netherlands, (2000)

[12] Frangioni, A., Gentile, C.: Perspective cuts for a class of convex 0-1 mixed integer programs. Mathematical Programing. **106**, 225-236 (2006)

[13] Fu, H.L., Shiue, L., Cheng, X., Du, D.Z., Kim, J.M.: Quadratic Integer Programming with Application in the Chaotic Mappings of Complete Multipartite Graphs. J. Optim. Theory Appl. **110** (3), 545-556 (2001)

[14] Garey, M.R., Johnson, D.S.: Computers and Intractability: A guide to the theory of NP-Completness. W.H. Freeman, San Francisco, CA, 1979

[15] globallib: `http://www.gamsworld.org/global/globallib/globalstat.htm`

[16] Hahn, P., Kim, B.J., Guignard, M. Smith, J., Zhu, Y.R.: An algorithm for the generalized quadratic assignment problem. *Computational Optimization and Applications.* **40** (3), 351-372, (2008). Available online DOI:10.1007/s10589-007-9093-1.

[17] Hua, Z.S., Banerjee, P.: Aggregate line capacity design for PWB assembly systems. Int. J.Prod. Res. **38**(11), 2417-2441 (2000)

[18] IBM-ILOG, IBM ILOG CPLEX 12.1 Reference Manual. IBM ILOG CPLEX, (2010)

[19] Liberti, L., Maculan, N.: Global Optimization: From Theory to Implementation, Chapter: Nonconvex Optimization and Its Applications. Springer, New York, (2006)

[20] Laguna, M., Mart, R.: GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS J. Comput.* **11**, 44-52 (1999)

[21] Lambert, A. Résolution de programmes quadratiques en nombres entiers . Thèse de doctorat en informatique, Conservatoire National des Arts et Métiers. (ref. CEDRIC 1838) (2009)

[22] Mateus, G.R., Resende, M.G.C., Silva, R.M.A.: GRASP with path-relinking for the generalized quadratic assignment problem. *Journal of heuristics*. (2010). Available online DOI: 10.1007/s10732-010-9144-0.

[23] minlplib: `http://www.gamsworld.org/minlp/minlplib.htm`

[24] Resende, M.G.C., Ribeiro, C.C.: GRASP with path-relinking: Recent advances and applications. *Ibaraki, T., Nonobe, K., Yagiura, M. (eds.) Metaheuristics: Progress as Real Problem Solvers*, 29-63. Springer-Verlag, Berlin (2005)

[25] Sahinidis, N.V., Tawarmalani, M.: BARON 9.0.4: Global Optimization of Mixed-Integer Nonlinear Programs, User's Manual. (2010) Available at `http://www.gams.com/dd/docs/solvers/baron.pdf`

[26] Sahinidis, N.V., Tawarmalani, M.:A polyhedral branch-and-cut approach to global optimization.Mathematical Programming. **103**(2),225-249, (2005).

[27] Sherali, H.D., Adams, W.P.: A tight linearization and an algorithm for zero-one quadratic programming problems. Management Science. **32**(10), 1274-90 (1986)

[28] TAPLib : `http://cedric.cnam.fr/oc/TAP/TAP.html`

[29] Tawarmalani, M., Sahinidis, N.V.: Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming. Kluwer Academic Publishing, Dordrecht, The Netherlands, (2002)