Computational Optimization and Applications manuscript No.
(will be inserted by the editor)

On the Solution of Nonconvex Cardinality Boolean Quadratic Programming problems. A computational study

Ricardo M. Lima · Ignacio E. Grossmann

Received: date / Accepted: date

Abstract This paper addresses the solution of a cardinality boolean quadratic programming problem using three different approaches. The first transforms the original problem into six Mixed-Integer Linear Programming (MILP) formulations. The second approach takes one of the MILP formulations and relies on the specific features of an MILP solver, namely using starting incumbents, polishing, and callbacks. The last involves the direct solution of the original problem by solvers that can accomodate the nonlinear combinatorial problem. Particular emphasis is placed on the definition of the MILP reformulations and their comparison with the other approaches. The results indicate that the data of the problem has a strong influence on the performance of the different approaches, and that there are clear-cut approaches that are better for some instances of the data. A detailed analysis of the results is made to identify the most effective approaches for specific instances of the data.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n. PCOFUND-GA-2009-246542 and from the Fundação para a Ciência e Tecnologia, Portugal, under grant agreement n. DFRH/WIIA/67/2011.

Ricardo M. Lima
Computer, Electrical and Mathematical Sciences & Engineering Division,
King Abdullah University of Science and Technology (KAUST),
Thuwal 23955-6900, Kingdom of Saudi Arabia.
E-mail: ricardo.lima@kaust.edu.sa
Tel.: +966 12 808 0434

Ignacio E. Grossmann
Department of Chemical Engineering,
Carnegie Mellon University,
5000 Forbes Avenue,
Pittsburgh, PA 15213, USA.
E-mail: grossmann@cmu.edu

Keywords Integer programming · Quadratic programming · Computing science

1 Introduction

Quadratic Programming problems have been receiving increasing attention due to theoretical developments and software implementations. In this work, we focus our attention on a specific type of Boolean Quadratic Programming (BQP) problems, and we study different formulations to solve them using off-the-shelf software. We address the solution of BQP problems using different Mixed-Integer Linear Programming (MILP) reformulations and Linear Programming based Branch & Bound (B&B) solvers[1] [2, p. 98], and also using nonlinear formulations through global optimization based solvers. The main goals are to evaluate the practical options for solving a particular type of BQP problem using commonly available software, and to provide a broad set of results that can guide practitioners to choose the best combination of formulation and solver.

Important applications of BQP include a class of facility location problems, the so-called Quadratic Assignment Programming (QAP) [3], task allocation [4], and molecular conformation [5].

BQP problems are a class of problems that involve only binary variables and a bilinear term in the objective function. The general formulation of a BQP is given by

$$\min\{c^T x + x^T Q x : Ax \leq b, x \in \{0,1\}^n\}, \tag{1}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $Q \in \mathbb{R}^{n\times n}$, and $A \in \mathbb{R}^{m\times n}$. BQP problems without linear constraints, called unconstrained BQP, with $Q \in \mathbb{R}^{n\times n}$, as well as BQP belong to the class of NP-hard problems [6], see Caprara [7] for a further discussion on the complexity of BQP problems. These problems are considered classic problems in combinatorial optimization, see for example Padberg [8] for the characterization of the polytope of an unconstrained BQP.

In this work, the focus is on the computational solution of (P1), a specific class of BQP:

$$\min\{c^T x + x^T Q x : x \in \hat{B}_{n,M}\}, \tag{P1}$$

where

$$\hat{B}_{n,M} = \left\{ x : \sum_{1\leq i\leq n} x_i = M \right\} \cap B^n$$
$$B^n = \{0,1\}^n,$$

and $c \in \mathbb{R}^n$, $Q \in \mathbb{R}^{n\times n}$, Q is not necessarily a positive-semidefinite matrix, and $0 < M < n$. Problem (P1) is a nonconvex nonlinear combinatorial problem involving a quadratic term in the objective function

[1] The term Linear Programming based Branch & Bound (B&B) refers to implicit enumeration algorithms that use linear programming relaxations to calculate bounds. This term is used to distinguish from recent semidefinite based B&B algorithms [1].

and a cardinality constraint. The formulation (P1) is known as the Cardinality BQP (CBQP) problem. This problem or variants with small differences are known under different names. For example, changing the direction of the optimization to maximization, this problem is known as the heaviest k-subgraph problem, the densest k-subgraph problem [9], p-dispersion-sum problem [10, 11], or the k-cluster problem [12]. Another problem related with the CBQP is the Cardinality-Constrained Quadratic Optimization (CCQO) problem, which is a QP problem with a constraint on the maximum number of non-zero variables in the solution. However, while in the CBQP all variables are binary, in the CCQO the variables are continuous. Bertsimas and Shioda [13] proposed a B&B algorithm to solve a reformulation of CCQO problems that shows some computational advances when compared with CPLEX. An annotated review of combinatorial problems subject to cardinality constraints is given by Bruglieri et al. [14], where some applications are described and the problems are characterized.

Typical applications of CBQP include problems in edge-weighted graphs (see Billionnet [9] for a detailed description), and facility location problems [10, 11]. Facility location problems are a typical application for the p-dispersion-sum problem, where the goal is to maximize the distance between the location of facilities in order to minimize competition, or cover large areas in the case of telecommunication problems. In chemistry, the CBQP formulation was applied to solve problems related with molecular conformation [5], while in the Process Systems Engineering literature, a CBQP formulation was proposed to describe the energy states of particles in doped semiconductors and solved via semidefinite relaxations [15].

In the last decades, significant theory has been developed to characterize the convex hull of the BQP and CBQP leading to the development of strong valid inequalities and several MILP formulations. On the other hand, in the recent years MILP solvers have implemented algorithms to deal with Mixed-Integer Quadratic Programming (MIQP) problems that can cope with nonconvex CBQP; examples include CPLEX, GUROBI, XPRESS and SCIP [16]. In addition, MILP solvers encompass several features involving callbacks to user routines, hot starts, and meta-heuristics to find integer solutions, which help to build tailored strategies to solve difficult problems. Based on these solver options, we have developed two methods integrated with an MILP formulation. Other options to solve CBQP problems include more general solvers that can rigorously address nonlinear combinatorial problems, for example α-BB [17], LindoGlobal [18], Couenne [19], BARON [20], or ANTIGONE [21].

This work is motivated by the emergence of different approaches and software that can solve CBQP problems. Thereby, our goal is to identify the best approach. Three options are evaluated:

1. The reformulation of the CBQP problem into an MILP problem. Some well-known formulations are used and based on those some new variants are tested. These MILP formulations are solved with different MILP solvers.

2. The application of solution strategies using advanced features of an MILP solver, including specific MILP solver options and callback routines to generate cuts, which we implement to solve the MILP reformulations.
3. The direct solution of the CBQP problem by a set of solvers that can address this type of problems without requiring a reformulation from the user. Two types of solvers are used: 1) typical MILP solvers that can handle CBQP problems; and 2) general global optimization solvers that can handle nonlinear combinatorial problems.

In the next three sections, each of these approaches is discussed in detail.

2 Reformulations of BQP problems

In general, the solution of a BQP using an MILP solution approach involves the reformulation of the nonlinear term in the objective function and the construction of a valid MILP formulation. A well known MILP reformulation of the unconstrained BQP is given by

$$\begin{aligned} \min \ & \sum_i (c_i + q_{ii}) x_i + \sum_{i<j\leq n} (q_{ij} + q_{ji}) z_{ij} \\ \text{subject to } & z_{ij} \geq x_i + x_j - 1, \qquad \forall\, i < j \\ & z_{ij} \leq x_i, \qquad \forall\, i < j \\ & z_{ij} \leq x_j, \qquad \forall\, i < j \\ & z_{ij} \geq 0, \qquad \forall\, i < j \\ & z_{ij} \leq 1, \qquad \forall\, i < j \\ & x \in \{0,1\}, \qquad \forall\, i, \end{aligned} \tag{P2}$$

where the variables z are continuous. (P2) is a classic formulation that can be found in several references, see [7, 8]. This formulation is based on the reformulation technique proposed in [22] for 0-1 polynomials, where the term $x_i x_j$ is replaced by a new variable z_{ij}, while $x_i = x_i^2$, for $x_i \in \{0,1\}$, and the following inequalities are added:

$$z_{ij} \geq x_i + x_j - 1, \qquad \forall\, i < j \tag{1}$$

$$z_{ij} \leq x_i, \qquad \forall\, i < j \tag{2}$$

$$z_{ij} \leq x_j, \qquad \forall\, i < j. \tag{3}$$

The inequalities above can also be derived from the logic proposition [23]:

$$x_i \wedge x_j \Leftrightarrow z_{ij}, \qquad i < j, \tag{4}$$

or derived using the Reformulation-Linearization Technique (RLT) of Sherali and Adams (1998) from the following equations:

$$z_{ij} \geq \max\left\{x_i^l x_j + x_i x_j^l - x_i^l x_j^l, x_i^u x_j + x_i x_j^u - x_i^u x_j^u\right\} \tag{5}$$

$$z_{ij} \leq \min\left\{x_i^u x_j + x_i x_j^l - x_i^u x_j^l, x_i^l x_j + x_i x_j^u - x_i^l x_j^u\right\}, \tag{6}$$

where $x^l = 0$ and $x^u = 1$ are the lower and upper bound, respectively, of the variables. Following the same reasoning, the CBQP problem can be transformed into

$$\begin{aligned}
\min\ & W = \sum_i q_{ii} x_i + \sum_i \sum_{j>i} \left(q_{ij} + q_{ji}\right) z_{ij} \\
\text{subject to } & \sum_i x_i = M \\
& z_{ij} \geq x_i + x_j - 1, \qquad \forall\, i < j \\
& z_{ij} \leq x_i, \qquad \forall\, i < j \\
& z_{ij} \leq x_j, \qquad \forall\, i < j \\
& z_{ij} \geq 0, \qquad \forall\, i < j \\
& z_{ij} \leq 1, \qquad \forall\, i < j \\
& x_i \in \{0,1\}, \qquad \forall\, i.
\end{aligned} \tag{P3}$$

Note that we replace $(c_i + q_{ii})$ by q_{ii}, since in the case studies addressed the parameters c_i and q_{ii} are both generated randomly. The derivation of the objective function of (P3) is described in Appendix A. This formulation has been analyzed by several authors. From the theoretical point of view, the properties of the convex hull of the BQP and CBQP have been studied using polyhedral theory and convex analysis, where the main goal is to find good lower bounds for the minimization problem. In general, the three main approaches to finding the lower bounds are based on semidefinite relaxations [24], Lagrangian relaxation, and reformulation techniques. A good survey on the two last approaches is given by Pisinger [25], and additional references are given for the semidefinite relaxation and reformulation techniques in the following sections.

Significant results valid for the BQP polytope are also valid for the CBQP [26]. Padberg [8] is a classic reference on the characterization of the convex hull of the BQP problem and has proposed four families of facets: triangle, clique, cut, and generalized cut inequalities. These inequalities were later studied by Boros and Hammer [27], who established new relations between different types of inequalities. Barahona et al. [6] and Pardalos and Rodgers [28] addressed the solution of the BQP using algorithmic techniques within a Linear Programming based B&B framework. The first proposed a branch and cut algorithm where cutting planes are applied at each node, while Pardalos and Rodgers [28] have studied a specific preprocessing technique to fix the variables. Based on the inequalities developed by Padberg [8], Sherali et al. [29] proposed a new class of facets for the BQP. Gueye and Michelon [30] and later Gueye and Michelon [31] proposed a reformulation framework where the objective is to derive reduced formulations,

in terms of the number of variables, of the classic formulation (P2). Hansen and Meyer [32] studied reduced derivations that aim at obtaining a good relaxation without compromising the computational performance. Following the same objective, Liberti [33] proposed an efficient, compact reformulation for BQP subject to assignment constraints. Burer and Letchford [34] extended the results described for the BQP in the characterization of box-constrained quadratic programming problems.

Mehrotra [26] studied the specific case of the following BQP:

$$\max\{c^T x + x^T Q x : x \in \hat{B}_{n,M}\}, \tag{7}$$

and discussed the derivation of some inequalities based on the inequalities of the Knapsack Constrained Boolean Quadratic Programming (KCBQP) polytope [35]. Mehrotra [26] proposed two types of inequalities: the tree inequalities and the star inequality and discussed the additional valid inequalities: long-cycle and long-tree inequalities. In addition, Mehrotra [26] developed a cutting plane algorithm using the tree and star inequalities, but discarded the utilization of the long-cycle and long-tree inequalities due to difficulties in the solution of the separation problems. Faye and Trinh [36, 37] extended the work of Padberg [8] on the BQP by considering the constraint:

$$\sum_{1 \leq i \leq n} x_i = M, \tag{8}$$

and proved that the family of clique inequality facets of the linear BQP also induce facets of the CBQP. The ultimate goal of the works mentioned above is the identification of valid inequalities that are facets of the polytope of the BQP and CBQP. Some of these works were concerned with the characterization of the problem and the derivation of facet inducing inequalities, without concern for the practical implication of the resulting size of the problems and the impact on the computational performance. The importance of having a tighter relaxation for a given MILP problem is well known within a B&B framework. However, the size of the LP problem to be solved at each node may also have a significant impact on the total computational time required to solve the corresponding MILP problem to optimality. An essential reference on the theoretical and computational aspect is the work of Billionnet [9], who has studied the efficiency of several MILP formulations for specific types of CBQP arising from weighted-edge graphs problems.

2.1 Formulations

In this section, we introduce the formulations used for the MILP solution approach. These formulations are based on the classic formulation presented, and on the combination of different valid inequalities proposed in the literature. From these inequalities, those that lead to smaller problems were selected, and the ones that require the solution of separation subproblems are not considered. Mehrotra [26] and Gueye and Michelon [30] report that this type of problem and the associated separation subproblems of some

inequalities may lead to difficult subproblems, restricting the scope of branch and cut algorithms. Our goal is to study and derive a relatively efficient MILP formulation from the computational point of view. By relatively efficient, we mean a formulation with a compromise between the quality of the continuous relaxation and the size of the formulation that outperforms the others. Six different MILP formulations are described next.

The first formulation is the simplest reformulation that leads to the smallest representation of the CBQP problem:

$$\begin{aligned} \min\ & W = \sum_i q_{ii} x_i + \sum_i \sum_{j>i} \left(q_{ij} + q_{ji}\right) z_{ij} \\ \text{subject to}\ & \sum_i x_i = M \\ & z_{ij} \geq x_i + x_j - 1, \qquad \forall\, i < j \\ & z_{ij} \geq 0, \qquad \forall\, i < j \\ & z_{ij} \leq 1, \qquad \forall\, i < j \\ & x_i \in \{0, 1\}, \qquad \forall\, i. \end{aligned} \tag{F0}$$

Formulation (F0) is only valid for problems with $q_{ij} + q_{ji} \geq 0$. For $q_{ij} + q_{ji} < 0$, this formulation is not equivalent to the original problem to be solved, because with $q_{ii} > 0$ it may lead to solutions with $x_i = 0$ and $z_{ij} = 1$, or for $q_{ij} + q_{ji} < 0$ and $q_{ii} < 0$ and $q_{jj} > 0$ it leads to solutions with $x_i = 1$, $x_j = 0$, and $z_{ij} = 1$, which are not valid solutions of the original problem. This formulation is presented for reference and it is not further used due to the lack of generalization for all q_{ij}.

The formulation (F1) is based on the classic reformulation, which by adding the inequalities $z_{ij} \leq x_i$ and $z_{ij} \leq x_j$ to the previous formulation, results in formulation (P3).

$$\text{Same as (P3).} \tag{F1}$$

The formulation (F2) is based on the classic formulation (F1), and it includes the following constraint:

$$\sum_{i<j} z_{ij} + \sum_{i>j} z_{ji} = (M-1)\, x_j, \qquad \forall\, j, \tag{9}$$

which is adapted from the star inequality proposed by Mehrotra [26]. The difference between (9) and the star inequality is that (9) considers $=$ instead of $\leq$.

$$
\begin{aligned}
\min\ & W = \sum_i q_{ii} x_i + \sum_i \sum_{j>i} \left(q_{ij} + q_{ji}\right) z_{ij} \\
\text{subject to}\ & \sum_i x_i = M \\
& z_{ij} \geq x_i + x_j - 1, \qquad \forall\, i < j \\
& z_{ij} \leq x_i, \qquad \forall\, i < j \\
& z_{ij} \leq x_j, \qquad \forall\, i < j \\
& \sum_{i<j} z_{ij} + \sum_{i>j} z_{ji} = (M-1)\, x_j, \qquad \forall j \\
& z_{ij} \geq 0, \qquad \forall\, i < j \\
& z_{ij} \leq 1, \qquad \forall\, i < j \\
& x_i \in \{0,1\}, \qquad \forall\, i.
\end{aligned}
\tag{F2}
$$

Formulation (F2) may appear to be an equivalent formulation of the one used by Macambira and de Souza [38] for the weighted maximal b-clique problem; however, it is not. The formulation of those authors is similar to (F2), but it does not include the constraint $\sum_i x_i = M$, and therefore, one may be tempted to drop that constraint from (F2), because of the presence of (9). However, the formulations are different. The formulation from Macambira and de Souza [38] is maximizing, while here it is minimizing. Therefore, if that constraint is removed from (F2), then for $q_{ij} \geq 0\ \forall\, i, j$ the solution is $x_i = 0,\ z_{ij} = 0\ \forall\, i, j$ and $W = 0$, which is not a feasible solution if the constraint $\sum_i x_i = M$ is considered. Therefore, they are not equivalent, and the constraint $\sum_i x_i = M$ cannot be dropped.

The formulation (F3) is obtained by eliminating the inequality $z_{ij} \geq x_i + x_j - 1$ from the formulation (F2). Billionnet [9] used a similar formulation for the heaviest k-subgraph problem with only non-negative q_{ij}. The underlying rationale is that a more compact formulation is obtained at the cost of a potentially weaker lower bound,

$$
\begin{aligned}
\min\ & W = \sum_i q_{ii} x_i + \sum_i \sum_{j>i} \left(q_{ij} + q_{ji}\right) z_{ij} \\
\text{subject to}\ & \sum_i x_i = M \\
& z_{ij} \leq x_i, \qquad \forall\, i < j \\
& z_{ij} \leq x_j, \qquad \forall\, i < j \\
& \sum_{i<j} z_{ij} + \sum_{i>j} z_{ji} = (M-1)\, x_j, \qquad \forall\, j \\
& z_{ij} \geq 0, \qquad \forall\, i < j \\
& z_{ij} \leq 1, \qquad \forall\, i < j \\
& x_i \in \{0,1\}, \qquad \forall\, i.
\end{aligned}
\tag{F3}
$$

This formulation is still valid to represent the original CBQP problem due to constraint (9), see Appendix A.

The formulation (F4) is based on (F3) but adding the triangular inequalities proposed by Padberg [8] for sets of three variables indexed by i, j, k.

$$
\begin{aligned}
\min\ & W = \sum_i q_{ii} x_i + \sum_i \sum_{j>i} \left(q_{ij} + q_{ji}\right) z_{ij} \\
\text{subject to}\ & \sum_i x_i = M \\
& z_{ij} \leq x_i, \qquad \forall\, i < j \\
& z_{ij} \leq x_j, \qquad \forall\, i < j \\
& \sum_{i<j} z_{ij} + \sum_{i>j} z_{ji} = (M-1)\, x_j, \qquad \forall\, j \\
& z_{ij} \geq z_{ik} + z_{jk} - x_k, \qquad \forall\, i < j < k \\
& z_{ik} \geq z_{ij} + z_{jk} - x_j, \qquad \forall\, i < j < k \\
& z_{jk} \geq z_{ij} + z_{ik} - x_i, \qquad \forall\, i < j < k \\
& z_{ij} \geq 0, \qquad \forall\, i < j \\
& z_{ij} \leq 1, \qquad \forall\, i < j \\
& x_i \in \{0, 1\}, \qquad \forall\, i.
\end{aligned}
\tag{F4}
$$

The triangular inequalities were initially proposed for the BQP problem, but it was proved that they are valid for the CBQP and that they lead to a tighter relaxation. The triangular inequalities are known to generate a large number of constraints, and therefore, in an alternative strategy to be presented later, the dynamic generation and addition of these inequalities to formulation (F3) is considered. In addition to [8], the triangular inequalities are also used by [30].

Formulation (F5) is a more compact reformulation of (F3) without the constraints $z_{ij} \leq x_j$, and $z_{ij} \leq x_i$. These two constraints enforce upper bounds on z_{ij}, which in (F5) are enforced by constraint (9).

$$
\begin{aligned}
\min\ & W = \sum_i q_{ii} x_i + \sum_i \sum_{j>i} \left(q_{ij} + q_{ji}\right) z_{ij} \\
\text{subject to}\ & \sum_i x_i = M \\
& \sum_{i<j} z_{ij} + \sum_{i>j} z_{ji} = (M-1)\, x_j, \qquad \forall\, j \\
& z_{ij} \geq 0, \qquad \forall\, i < j \\
& z_{ij} \leq 1, \qquad \forall\, i < j \\
& x_i \in \{0, 1\}, \qquad \forall\, i.
\end{aligned}
\tag{F5}
$$

The last formulation is based on one formulation proposed in [9]. The derivation of the formulation (F6) is based on the linearization technique proposed by Glover [39], which was applied to a CBQP problem by Billionnet [9]. However, the formulation proposed by Billionnet [9] is applied to a maximization

problem with non-negative q_{ij}, but in this work it is extended to deal with the problem (P1):

$$\begin{aligned} \min\ & W = \sum_i q_{ii} x_i + \tfrac{1}{2} \sum_i t_i + \tfrac{1}{2} \sum_i U_i x_i \\ \text{subject to}\ & \sum_i x_i = M \\ & t_i \geq \sum_{j<i} \left(q_{ji} + q_{ij}\right) x_j + \sum_{j>i} \left(q_{ij} + q_{ji}\right) x_j - U_i, \qquad \forall\, i \\ & t_i \geq \left(L_i - U_i\right) x_i, \qquad \forall\, i \\ & x_i \in \{0,1\}, \qquad \forall\, i. \end{aligned} \tag{F6}$$

In the formulation above, L_i and U_i are constants that represent bounds, and t_i is a new variable indexed only by i. The derivation of this formulation is presented in Appendix B.

The formulations (F1)-(F6) encompass a set of valid MILP representations of the same original problem, ranging from compact formulations to formulations expected to have a tight continuous relaxation.

2.2 Quality of the formulations

In this section, a theoretical analysis of the quality of the relaxations of the MILP formulations is presented. Here, W_n denotes the value of the objective function of the continuous relaxation of the formulation (Fn).

Proposition 1 *The following inequalities hold:*

$$W_0 \leq W_1 \leq W_2, \tag{10}$$

$$W_3 \leq W_2, \tag{11}$$

$$W_3 \leq W_4, \tag{12}$$

$$W_5 \leq W_3 \leq W_2. \tag{13}$$

Proof: The lower bound provided by (F1) will be at least as large or equal than (F0), due to the fact that (F1) considers the two additional constraints: $z_{ij} \leq x_i$ and $z_{ij} \leq x_j$. The formulation (F2) has one more constraint than (F1). The formulation (F3) is equal to the formulation (F2) without the constraint $z_{ij} \geq x_i + x_j - 1$. This inequality may be redundant for some instances, which may transform the formulation (F3) into an efficient approach for those cases. The formulation (F4) is based on formulation (F3), and additionally it includes the triangular inequalities, therefore its continuous relaxation is stronger than (F3). The formulation (F5) results from eliminating some constraints from the formulation (F3) in order to obtain a more compact formulation. □

Based on Proposition 1, it is not possible to establish a general rule that groups all the inequalities in one expression. For example, at this point no relation between W_1 and W_3 can be established. In addition,

formulation (F6) cannot be compared with the other formulations. The computational results will help to understand some of the additional relations between the quality of the relaxations.

3 Solution methods using specific features of an MILP solver

The second approach implemented to solve the CBQP problem is based on the MILP formulation (F3) and takes advantage of specific features of CPLEX. Based on this, we propose two methods using the following procedures:

1. A sequence of three problems is solved with the total maximum CPU time of 7200 seconds for the three problems. The first solves the MILP formulation (F3) with the objective of finding the best possible integer solution in a short time. This is accomplished by invoking the heuristic available within CPLEX to polish MILP solutions [40]. Note that CPLEX has other strategies to give priority to find integer solutions, for example through the option mipempashis or options related with heuristics. In the second step, the continuous relaxation of formulation (F3) is solved, and the violated triangluar inequalities are identified. In the third step, the formulation (F3) is complemented with the triangular inequalities that were violated in the second step, and the problem is solved using the integer solution from the first step as starting incumbent. This strategy is denominated as the sequential solution. Note that the maximum time set to solve the first problem depends on the maximum time set for the overall method, it depends on the time required to find an integer solution (the MILP polish options needs one before it can start), and on the performance of the MILP polish algorithm in the first problem.
2. The second method relies on using CPLEX with callbacks during the B&B algorithm to generate cuts that cut off solutions of the continuous relaxation. This approach is implemented using the BCH tool from GAMS [41]. In this implementation the formulation (F3) is used, and in the cut generation routine the triangular inequalities that are violated are added to the continuous relaxation solved within the B&B algorithm within the MILP solver. Two variants of this method are considered: 1) add all violated constraints identified in the user cut routine to the problem, and 2) add only a subset of the violated constraints identified in the user cut routine. The former has the advantage of building a tighter relaxation at the beginning than the latter, but at the price of increasing the size of the problem considerably. While the second strategy yields a slower improvement of the relaxation, it includes smaller subproblems.

4 Direct solution of CBQP problems

Problem P1 is a pure combinatorial problem that can be solved through an MILP reformulation as shown in Section 2, by using a general solver that can deal with nonlinear combinatorial problems, or it can be

solved using the recent algorithms available within the MILP solvers. The common available MILP solvers, such as CPLEX, GUROBI, and XPRESS can solve MIQP problems with the following formulation:

$$\min\{c^T x + x^T Q x : x : Ax \leq b, x \in \{0,1\}^n\}, \tag{14}$$

with nonconvex objective function, i.e. when the matrix Q is not positive-semidefinite. CPLEX and GUROBI have two different approaches: 1) reformulation of the nonlinear terms in the objective function, which is similar to the reformulations presented; and 2) convexification of the objective function by perturbation of the matrix Q.

4.1 Convexification of nonconvex BQP problems

In this work, we apply the term convexification to denote the exact reformulation that transforms a nonconvex objective function into an equivalent convex objective function. The exact procedure used by the MILP solvers for this reformulation is not described in detail in the open literature. However, there is a short description in Bliek et al. [42] of the approach used by CPLEX, and several authors have proposed convexification procedures. Billionnet and Elloumi [24] show that the objective function of the problem

$$\min\{c^T x + x^T Q x : x \in \{0,1\}^n\} \tag{MIQP_F}$$

can be transformed into a convex function, $f_u(x) = c^T x + x^T Q x + Penalty(u, x)$, where $Penalty(u, x)$ is a penalty that depends on the vector u and x, which under specific conditions leads to a convex problem:

$$\min\{f_u(x) : x \in [0,1]^n\} \tag{QP_Fu}$$

that yields a valid lower bound of the former problem. Their objective was to derive a convex MIQP problem that could be used within a B&B based algorithm.

The underlying rationale is the perturbation of the objective function $f(x) = c^T x + x^T Q x$ with a vector u and a matrix $D = \mathrm{diag}(u)$ such that a new convex objective function is defined: $f_u(x) = (c+u)^T x + x^T (Q - D)x$. The function f_u can be rewritten as $f_u(x) = f(x) + \sum_{i=1}^{n} u_i \left(x_i - x_i^2\right)$, which clearly shows that $f_u(x) = f(x)$ when $x \in \{0,1\}^n$. The problem $\min\{f_u(x) : x \in [0,1]^n\}$ is convex if by a suitable choice of a vector u, $(Q - D)$ results in a positive-semidefinite matrix. Note that the function f_u depends on the value of the vector u. Therefore, given that $f_u(x) = f(x)$ when $x \in \{0,1\}^n$, solving the nonconvex problem (MIQP_F) presented before is equivalent to solving the following problem

$$\min\{f_u(x) : x \in \{0,1\}^n\} \tag{MIQP_Fu}$$

within a B&B framework where the continuous relaxation is given by the problem (QP_Fu). However, the determination of a particular vector u such that $(Q - D) \succeq 0$ is essential to build the convex problem,

as well as to set a strong lower bound from the relaxation. This vector can be determined based on the calculation of the minimum eigenvalue of matrix Q, or through the solution of a semidefinite programming (SDP) problem [43]. These authors proposed an interesting approach based on duality theory and the solution of an SDP problem solved in a preprocessing step. The solution of the SDP problem also determines a vector u, which is used to formulate a convex BQP, which in turn results in improved lower bounds. In addition, Billionnet and Plateau [43] move the cardinality constraint to the objective function in order to further help to obtain a better convexification.

Recently, there have been several contributions to the development of particular B&B algorithms to solve BQP problems. In these algorithms, SDP problems are used as relaxations of the original problem and provide tight bounds on the objective function value [1, 12, 44, 45]. Rendl et al. [45] present a useful summary of the expected computational performance of these B&B algorithms and other methods to solve BQP problems as a function of the characteristics of the problems.

5 Computational experiments

The computational experiments aim at evaluating the value of the three approaches to solving the CBQP problem:

1. MILP formulations - application of MILP solvers to the MILP reformulations (F1) to (F6) (Sections 1 and 2);
2. Solution methods using specific features of an MILP solver (Section 3);
3. Direct solution of the CBQP problem - solution of Problem (P1) without any user reformulation (Section 4).

Regarding the first approach, the MILP formulations described before are characterized by the size of the model and by the quality of the continuous relaxation in terms of the value of the objective function. However, with the combination of these two characteristics, it is not trivial to infer which formulation will have a better performance with a B&B based MILP solver. Furthermore, current MILP solvers have implemented several algorithms at the level of pre-processing techniques, cutting planes, heuristics, parallelization, and tree-search, which make it difficult to estimate the performance of an MILP formulation (see Rothberg [40], Lima and Grossmann [46], Bixby and Rothberg [47] for reviews on some MILP solver features). The performance of the MILP formulations is analyzed using three different MILP solvers: CPLEX 12.6.1.0, GUROBI 6.0.0, and XPRESS 27.01.02.

The second approach implements two tailored methods based on the MILP formulation (F3). These two methods are applied to one case study that the MILP solvers had difficulties to close the optimality gap.

The last approach relies on the direct solution of the CBQP problem (P1) by the solvers CPLEX 12.6.1.0, GUROBI 6.0.0, XPRESS 27.01.2, BARON 14.4.0, ANTIGONE 1.1, and SCIP 3.1. This ap-

proach is characterized by the fact that these solvers can address the CBQP problem (P1) without requiring a reformulation from the user. CPLEX 12.6.1.0 and GUROBI 6.0.0 can solve the CBQP problem (P1) by automatically linearizing the nonlinear terms of the objective function or by convexifying the objective function by perturbing the matrix Q. In our experiments [2], we test both direct approaches, and the following labels are used to identify the results:

- CPLEX 12.6.1.0 - MILP - automatic reformulation of problem (P1) into an MILP;
- CPLEX 12.6.1.0 - MIQP - convexification of the objective function;
- GUROBI 6.0.0 - MILP - automatic reformulation of problem (P1) into an MILP;
- GUROBI 6.0.0 - MIQP - convexification of the objective function.

Ten case studies are considered by combining five sizes of the CBQP problem (P1), $n = \{50, 75, 100, 200, 300\}$, and two values for the constant M involved in the cardinality constraint, $M = \{n/5, n/1.25\}$. The computational experiments are performed using several instances of the matrix $Q \in \mathbb{R}^{n \times n}$ for each case study characterized by the pair (n, M). The instances are based on different coefficients of the matrix Q, with each coefficient of the matrix generated using a random number generation function with a uniform distribution for a given lower bound, LB, upper bound, UB, and sparsity density, SD. 16 families of matrices Q are built for $Q(LB, UB, SD) = \{(-100, 100, SD), (-1, 1, SD), (0, 1, SD), (0, 100, SD), SD = \{0.10, 0.50, 0.75, 1.00\}\}$, and for each family, five matrices are generated resulting in 80 instances of the matrix Q spanning the interval [-100, 100]. More information on these matrices can be found in the Supplementary Results document.

The results obtained are presented using performance profiles based on relative computational times [48], and relative and absolute optimality gap profiles [49]. In the classification of the performance profiles, relative means the result of the solver (or formulation) in relation to the best solver (or formulation) in the test. On the one hand, the term relative optimality gap profiles means that the profiles are obtained by dividing the optimality gap of the solver by the minimum optimality gap in the test. On the other hand, absolute optimality gap profile means that the profiles represent only the optimality gap of the solver. See Appendix C for further details.

These profiles provide a graphical methodology to compare the computational performance of different solvers, and formulations. For each optimization run the stopping criterion is the maximum CPU time of 7200 seconds or the optimality gap of 0.0%. For the second approach using the first solution method, the total CPU time limit is 7200 seconds, and the time limit for the first problem solved is 1000 seconds. The nodes of the computer cluster used have CPUs Quad sockets AMD Opteron 6376, with 16 cores/socket 2.3GHz with 256GB RAM. All models are implemented in the modeling system GAMS [41].

[2] CPLEX options: qtolin = 1 - automatic linearization of the nonlinear terms; qtolin =0 - convexification of the objective function. GUROBI options: preqlinearize = 1, premiqpmethod = 1 - automatic linearization of the nonlinear terms; preqlinearize = 0, premiqpmethod = 0 - convexification of the objective function.

Note that the solvers used in this work are in continuous improvement. Therefore, from the beginning of this study until the time the work is published, a new release of a solver may have been available, which probably could lead to a different performance and comparison with other solvers. Thus, in order to assess for the CBQP problem the value of the algorithmic improvements implemented in CPLEX 12.6.1.0. and the recent advances in MILP and MIQP solvers, we try to evaluate two very different versions of CPLEX, 7.1 and 12.6.1.0.

CPLEX 7.1 is one of the earliest versions that we have access, and is implemented in GAMS 20.1 built in October 31, 2001. These experiments will also allow us to evaluate the value of good MILP formulations for the CBQP problem, and compare them with the CPLEX MILP algorithm developments. These comparisons are presented in sub-section 5.4 of the computational results.

5.1 MILP formulations

The results obtained with the MILP reformulations are divided by two values of M, namely $M = n/5$ and $M = n/1.25$. This organization is chosen due to the impact of this parameter on the relative quality of the linear relaxations of the MILP formulations as it will be discussed next.

5.1.1 BQP with $M = n/5$

The characteristics of the MILP formulations used, in terms of the size of the formulations, and quality of the continuous relaxation are presented in Tables 1 and 2. The first table provides data on the size of the formulations for $n = \{50, 300\}$, where formulation (F4) stands out due to the large number of constraints and non-zeros, and formulation (F6) by the compact size.

Table 2 presents the optimal values for the continuous relaxation of each MILP formulation for a selected number of instances studied, for the sake of saving space. However, in the Supplementary Results document are given the values for all the instances.

Table 1 Size of the MILP formulations for $n = \{50, 300\}$.

	$n = 50$				$n = 300$			
Formulation	0-1 Var	Cont. Var	NC	Non-zeros	0-1 Var	Cont. Var	NC	Non-zeros
F1	50	1,226	3,677	9,901	300	44,851	134,552	322,848
F2	50	1,226	3,727	12,401	300	44,851	134,852	412,848
F3	50	1,226	2,502	8,726	300	44,851	90,002	278,298
F4	50	1,226	60,077	241,476	300	44,851	13,410,452	53,686,351
F5	50	1,226	52	3,826	300	44,851	302	98,898
F6	50	51	102	2,751	300	301	602	18,933

NC - number of constraints, Non-zeros - total number of coefficients of the matrices and vectors of the formulations with non-zero values.

The optimal values presented in Table 2 complement the results from Section 2.2, providing additional computational information about the lower bounds of the formulations. These computational results help to identify the inequalities from Section 2.2 that are strict, and provide insights on when they are strict as a function of the sparsity density, and as a function of the coefficients of matrix Q.

In general, formulation (F4) is the tightest formulation. The exception occurs when the sparsity density is low and the minimum coefficient of Q is zero, see the column corresponding to $Q(0,\, 1,\, 0.10)$. The optimal values obtained for $SD = 0.10$ show that if all $q_{ij} \geq 0$, then the optimal solution of all continuous relaxations are equal to zero, and they are equal to the solution of the MILP problem. This is explained by the fact that 90% of the coefficients of the matrix Q are zero, and thus, the nonzero variables x_i forced by the cardinality constraint, and the variables z_{ij} correspond to the coefficients with $q_{ij} = 0$. If $SD = 0.10$ and $-1 \leq q_{ij} \leq 1$, then $W_1 = W_2 = W_3$, $W_5 < W_2$, $W_2 < W_4$, and $W_6 \leq W_1$. In this case there are z_{ij} that do not tend to zero and the constraint $\sum_{i<j} z_{ij} + \sum_{i>j} z_{ji} = (M-1)\, x_j$ is redundant.

In the other extreme, with $SD = 1.00$, we have $W_1 < W_2 = W_3$ and $W_1 < W_5 < W_6 < W_2 < W_4$. The relation $W_2 = W_3$ shows that the constraint $z_{ij} \geq x_i + x_j - 1$ is redundant. For $q_{ij} \geq 0$, x_i tends to zero (limited by the cardinality constraint) and thus $x_i + x_j \leq 1$, which makes the equation $z_{ij} \geq x_i + x_j - 1$ redundant. Thus in formulation (F1), no lower bound is imposed on z_{ij} and then $z_{ij} = 0,\ \forall\, i, j$. In formulations (F2) and (F3), equation $\sum_{i<j} z_{ij} + \sum_{i>j} z_{ji} = (M-1)\, x_j$ enforces that some z_{ij} must be greater than zero, and therefore, a tighter relaxation is obtained, leading to $W_1 < W_2 = W_3$. For $SD = 1.00$ and $-1 \leq q_{ij} \leq 1$, a similar reasoning can be followed since a lower bound on z_{ij} is redundant. Formulation (F5) does not enforce the upper bounds on z_{ij} through the constraints $z_{ij} \leq x_j$ and $z_{ij} \leq x_i$, and therefore, a lower optimal value can be obtained leading to a weaker formulation than (F2) and (F3). In the Supplementary Results document, it is shown that the relations described above between the optimal values of the continuous relaxations of the formulations are valid for all the instances.

Table 2 Optimal value for the continuous linear relaxation of each MILP formulation, for $n = 50$, $M = n/5$. The results correspond to one instance.

	$SD = 0.10$		$SD = 0.50$		$SD = 1.00$	
	$Q(-1, 1, SD)$	$Q(0, 1, SD)$	$Q(-1, 1, SD)$	$Q(0, 1, SD)$	$Q(-1, 1, SD)$	$Q(0, 1, SD)$
W_1	-13.21	0.00	-47.45	0.00	-82.00	1.96
W_2	-13.21	0.00	-35.35	0.51	-53.63	22.29
W_3	-13.21	0.00	-35.35	0.51	-53.63	22.29
W_4	-10.61	0.00	-22.84	4.30	-37.75	30.50
W_5	-37.32	0.00	-54.86	0.00	-76.45	11.82
W_6	-17.81	0.00	-41.39	0.00	-64.28	18.24
W^*	-10.61	0.00	-20.62	5.91	-36.03	31.38

SD - Sparsity density of matrix Q. W_n - Optimal value of the continuous relaxation of Fn.
W^* - Optimal value for the MILP problem for each instance.

Figures 1 and 2 show the performance obtained for each formulation for $n = \{50, 75, 100, 200, 300\}$, $M = n/5$, and for the 80 instances based on the random matrices Q. In these profiles, each formulation includes the performance of the three MILP solvers, which roughly provides an indication of the performance of the formulation independently of the solver. These profiles indicate that formulation (F3) has the highest probabilities, 59.6% and 56.2%, of being the fastest formulation to solve any given problem of the size $n = \{50, 75\}$, respectively. For $n = 50$, formulation (F3) can solve 100% of the problems to optimality, while for $n = 75$ this number drops to 91.3% of the problems. For $n = \{50, 75\}$, the results show a clear difference between the most efficient formulation, (F3), and the most inefficient formulations, (F4) and (F5). (F4) is the tightest formulation with the best relaxation, but with the worst performance, which is explained by the size of the MILP formulation that forces the MILP solvers to deal with large LP problems at each node. Therefore, we did not consider formulation (F4) for the problems with $n > 75$.[3]

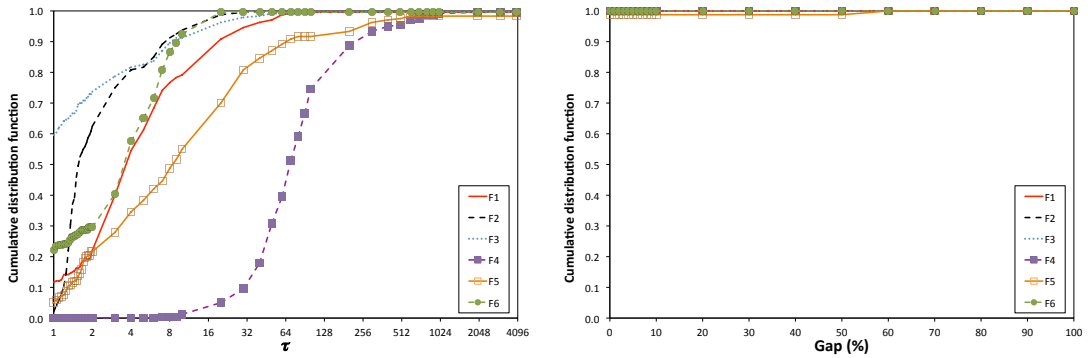
Increasing the size of the problems, for $n = 100$ formulation (F1) has the highest probability of being the fastest to solve the problems to optimality, 15.8%, and it can solve to optimality 25% of the problems. Note that the problems solved are those with $SD = 0.10$. However, none of the instances with $n > 100$ are solved to optimality, independently of the formulation and solver, see Figures 1(g), 1(h), 2(a), and 2(b). For $n = \{200, 300\}$, the difference between the performance of the formulations decreases, but formulation (F6) has a slightly advantage over the other formulations.

The overall performance of each solver over the 80 instances and the 6 MILP formulations is shown in Figure 3. The profiles show that GUROBI 6.0.0 has the best probability of solving any problem to optimality for $n = \{50, 100\}$, while CPLEX 12.6.1.0 can perform better for $n = 75$. The performance of the solvers decreases substantially by increasing the number of variables of the problem, within the CPU time limit set.

5.1.2 BQP with $M = n/1.25$

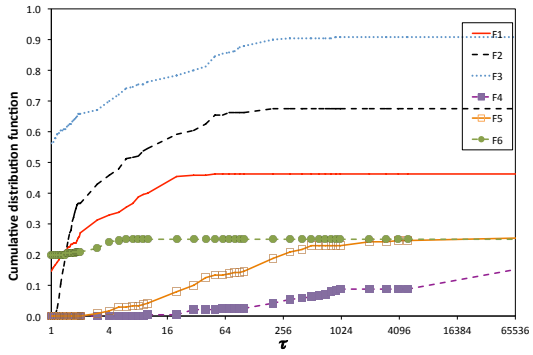
The optimal values of the continuous relaxations for each MILP formulation are presented in Table 3, for one instance of each matrix $Q(LB, UB, SD)$. These computational results show that the formulation (F4) is the tightest in all cases. Furthermore, they provide additional insights about the comparison of the quality of some formulations, for which no strong relation was possible to infer before. For example, for (F1) and (F3), which for $M = n/5$ the computational results show that $W_1 \leq W_3$, now for $M = n/1.25$ the results indicate that $W_3 \leq W_1$. Furthermore, the relations between (F2), (F3), and (F6) are represented by strict inequalities, $W_3 < W_6 < W_2$.

[3] For the larger problems with formulation (F4), the MILP solvers are not able to solve the relaxation at the root node within the time limit of 7200 seconds. One way to address this issue is to instruct the MILP solver to use at the root node only the Barrier solver using the parallelization option and a large number of threads, e.g. 40 threads. With this approach the time to solve the relaxation decreases substantially and the solver can start the tree-search, however, the solver still has to handle a large LP problem at each node.

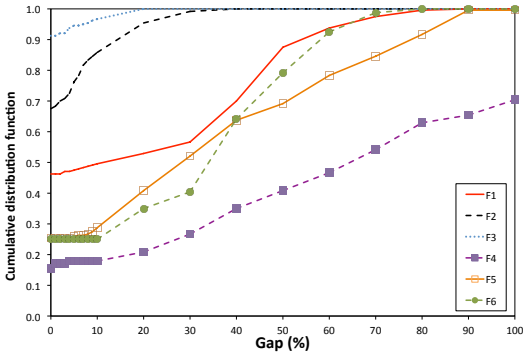


(a) Relative CPU time (s), $n = 50$.

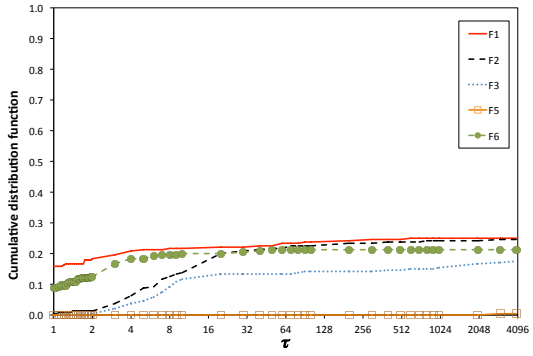
(b) Absolute gap (%), $n = 50$.



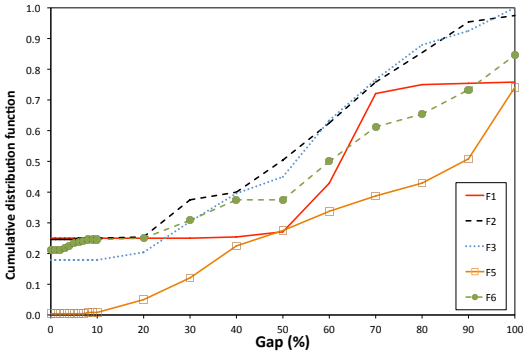
(c) Relative CPU time (s), $n = 75$.



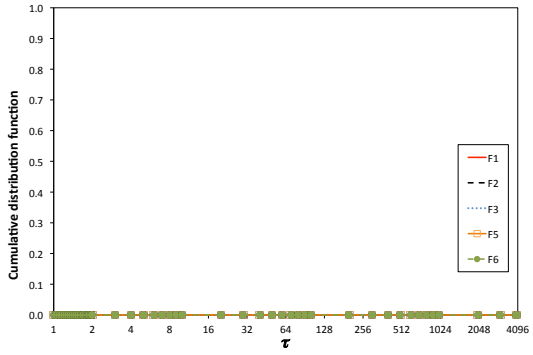
(d) Absolute gap (%), $n = 75$.



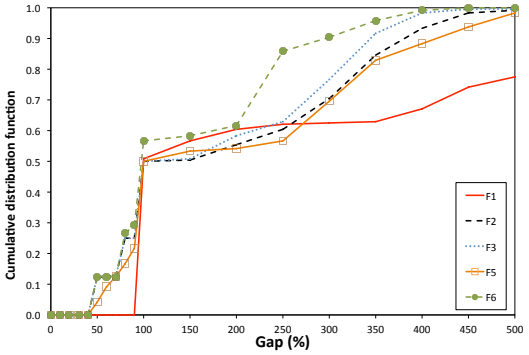
(e) Relative CPU time (s), $n = 100$.



(f) Absolute gap (%), $n = 100$.

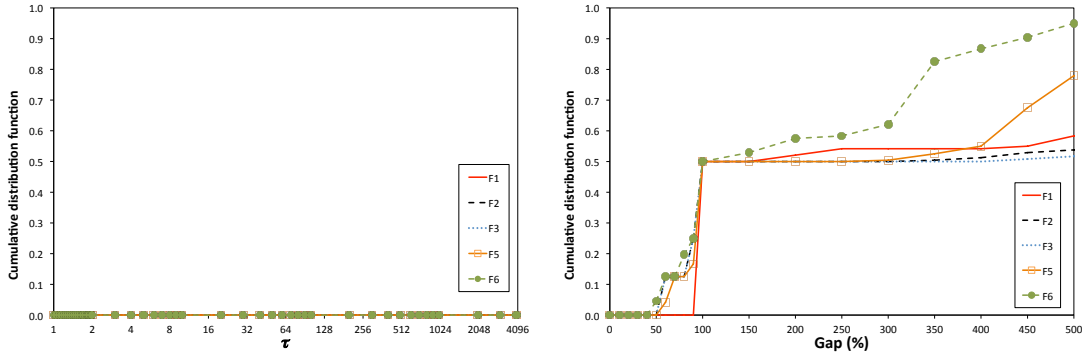


(g) Relative CPU time (s), $n = 200$.



(h) Absolute gap (%), $n = 200$.

Figure 1 Performance profiles based on the CPU time (left) and absolute gap (right) of each MILP formulation. Results obtained with the solvers CPLEX 12.6.1.0, GUROBI 6.0.0, and XPRESS 27.01.02. $n = \{50, 75, 100, 200\}$, $M = n/5$. Stopping criteria: optimality gap of 0.0% or 7200 seconds. 240 instances.



(a) Relative CPU time (s), $n = 300$.

(b) Absolute gap (%), $n = 300$.

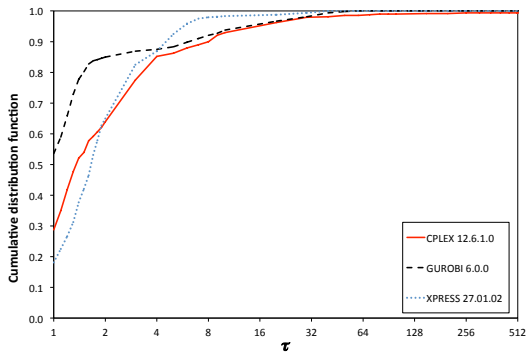
Figure 2 Performance profiles based on the CPU time (left) and absolute gap (right) of each MILP formulation. Results obtained with the solvers CPLEX 12.6.1.0, GUROBI 6.0.0, and XPRESS 27.01.02. $n = 300$, $M = n/5$.Stopping criteria: optimality gap of 0.0% or 7200 seconds. 240 instances.

Analyzing first the results obtained with the formulations (F1), (F2), (F3), and (F5), the inequality $z_{ij} \geq x_i + x_j - 1$ seems to be very effective in practice, which is supported by comparing formulations (F1) and (F2) that include the cut with formulation (F3) that does not include the cut. For $M = n/1.25$, the continuous relaxation is forced to set some $x_i \geq 0.5$, due to the cardinality constraint. Thus, it is easy to see that there exists a pair i, j such that $x_i + x_j \geq 1$, and therefore, the inequality $z_{ij} \geq x_i + x_j - 1$ enforces valid bounds on z_{ij}, which provides a tighter relaxation. The relative value of the relaxation of formulation (F5) also decreases for $M = n/1.25$ when compared with (F1), which further shows the importance of the constraint $z_{ij} \geq x_i + x_j - 1$. In the Supplementary Results document, it is shown that the relations described above between the optimal values of the continuous relaxations of the formulations are valid for all the instances.

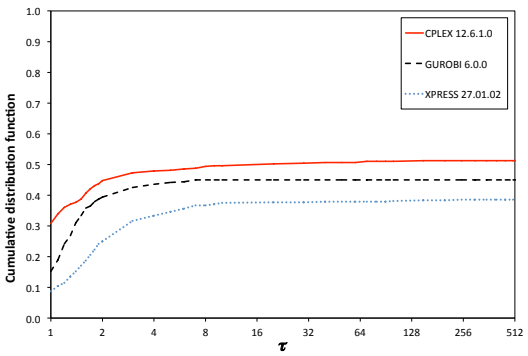
Table 3 Optimal value for the continuous linear relaxation of each MILP formulation, for $n = 50$, $M = n/1.25$. The results correspond to one instance.

	$SD = 0.10$		$SD = 0.50$		$SD = 1.00$	
	$Q(-1, 1, SD)$	$Q(0, 1, SD)$	$Q(-1, 1, SD)$	$Q(0, 1, SD)$	$Q(-1, 1, SD)$	$Q(0, 1, SD)$
W_1	-20.74	58.91	-28.83	340.24	-84.76	712.97
W_2	-20.74	58.91	-21.43	350.42	-79.27	747.10
W_3	-48.99	5.01	-138.72	265.41	-220.01	674.78
W_4	-20.58	59.07	-15.72	355.04	-74.44	749.64
W_5	-61.23	0.03	-237.53	172.98	-386.12	595.52
W_6	-26.83	55.36	-40.92	339.31	-105.75	737.37
W^*	-20.58	59.07	-15.13	355.04	-74.44	749.64

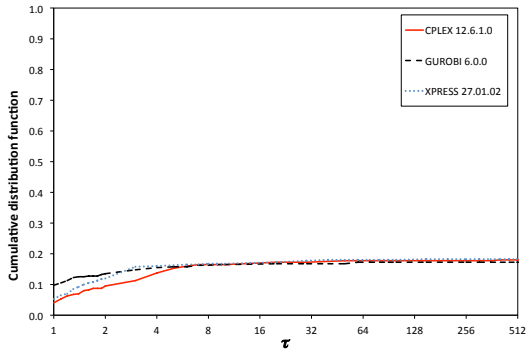
SD - Sparsity density of matrix Q. W_n - Optimal value of the continuous relaxation n.
W^* - Optimal value for the MILP problem for each instance.



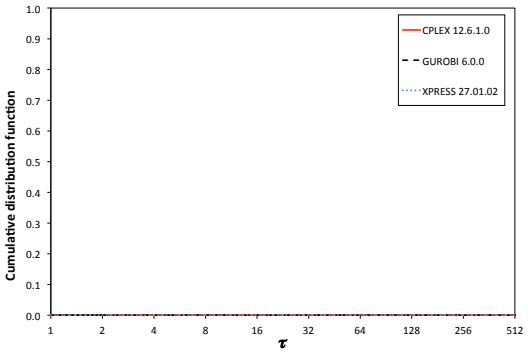
(a) $n = 50$.



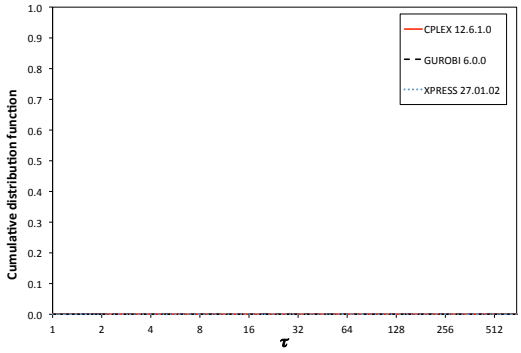
(b) $n = 75$.



(c) $n = 100$.



(d) $n = 200$.



(e) $n = 300$.

Figure 3 Performance profiles based on the CPU time of each solver. Results obtained for the MILP reformulations. $n = \{50, 75, 100, 200, 300\}$, $M = n/5$.Stopping criteria: optimality gap of 0.0% or 7200 seconds. 400 instances.

Figure 4 and 5 show the results obtained with the MILP formulations for $n = \{50, 75, 100, 200, 300\}$, taking into account the set of instances for the matrix Q and the 3 MILP solvers. For this family of instances for $n = 50$, the formulation (F6) has 98.8% of probability of being the best formulation to solve any given problem. In contrast, for $n = 50$, formulation (F3) has the probability of being the best formulation of only 0%, which is the totally opposite trend with the highest probability of 59.6% obtained for $M = n/5$.

For the cases with $n = \{75, 100\}$ and $M = n/1.25$, formulation (F2) has the best performance, see Figures 4(c) and 4(d) . Note that with $n = 75$ and $M = n/1.25$, formulation (F2) can solve 100% of the problems to optimality. Whereas with $n = 75$ and $M = n/5$, the overall maximum number of problems solved to optimality is 25.4%, compare Figures 1(d) and 4(d).

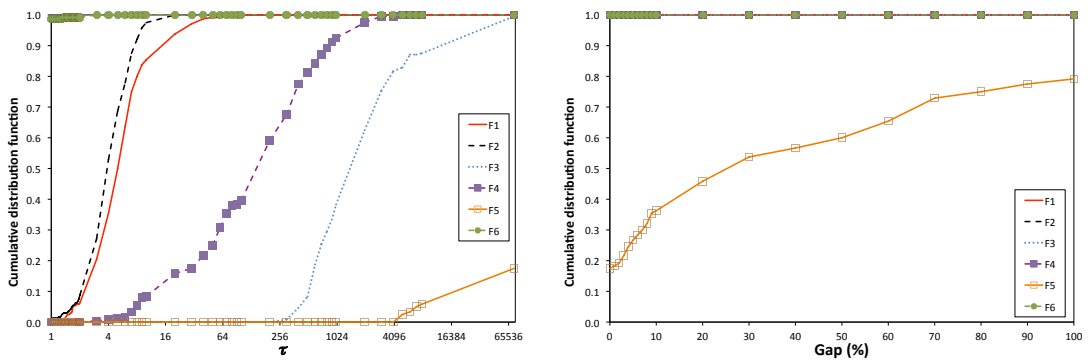
The comparison between the MILP solvers applied to the MILP reformulations show that GUROBI 6.0.0 has the best performance for $n = 50$, and CPLEX 12.6.1.0 for $n = \{75, 100\}$, see detailed results in the Supplementary Results document.

5.2 Solution methods using specific features of an MILP solver

The second approach implemented to solve the CBQP problem is based on the formulation (F3) and CPLEX 12.6.1.0. The main goal is to evaluate the applicability of the two methods described in Section 3 to solve some of the instances that the solvers had more difficulty to close the optimality gap, namely the case with $n = 100$, and $M = n/5$, with $SD = \{0.50, 0.75, 1.00\}$. A detailed analysis shows that formulation (F3) has the best performance with CPLEX 12.6.1 for those instances, see the Supplementary Results document.

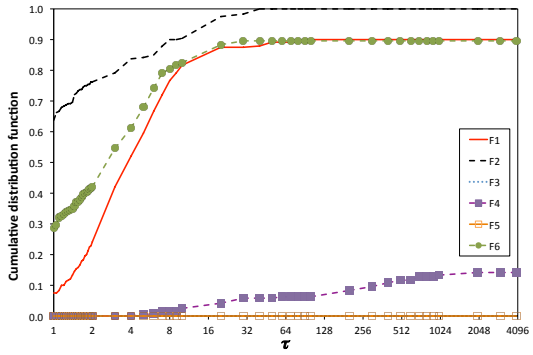
The results obtained with these methods are presented in Figure 6. The first method is represented by the label SEQ, and the two variations of the second by BCH and BCH2. The performance profiles in the first and second sub-figures are based on the relative and absolute gap, respectively, obtained at the end of the optimization run, and in the third sub-figure are based on the CPU times of the optimization runs that close the gap. These profiles show that the MILP approach and the three solution methods tested are not able to close the optimality gap within the CPU time limit of 7200 seconds. However, the performance profiles based on the gap indicate that the method that solves a sequence of problems (SEQ), and the strategy that uses the callbacks with the partial addition of the cuts (BCH2) can improve the MILP approach for $n = 100$. For the larger problems, the MILP formulation and the solution methods tested reached the CPU time limit with large gaps.

For the problems with $n = 100$ and $SD = \{0.50, 0.75, 1.00\}$, GUROBI 6.0.0 and XPRESS 27.01.02 do not have a better performance than CPLEX 12.6.1.0, see the Supplementary Results document. Therefore, these two tailored methods are also more efficient than GUROBI 6.0.0, and XPRESS 27.01.02 when applied to these MILP reformulations.

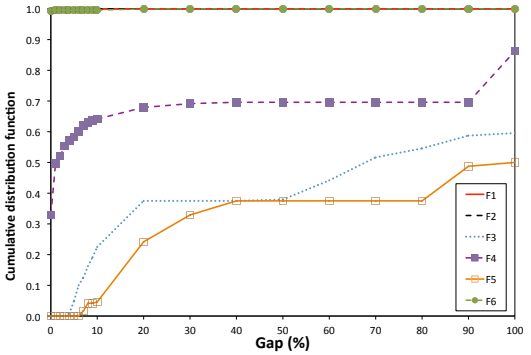


(a) Relative CPU time (s), $n = 50$.

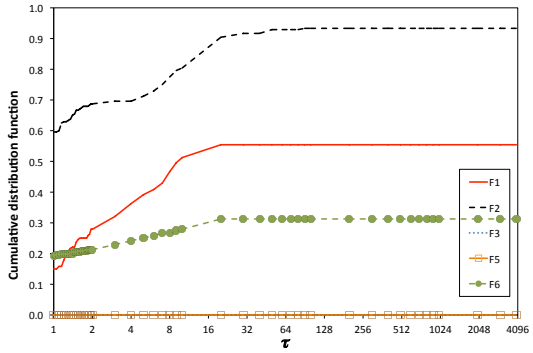
(b) Absolute gap (%), $n = 50$.



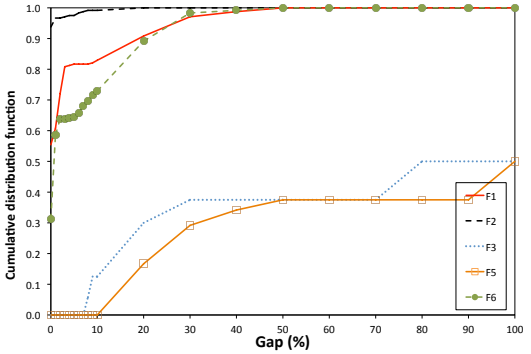
(c) Relative CPU time (s), $n = 75$.



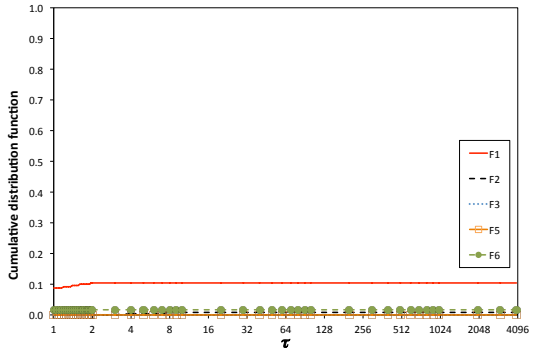
(d) Absolute gap (%), $n = 75$.



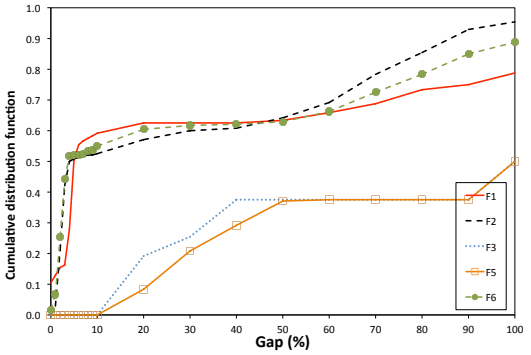
(e) Relative CPU time (s), $n = 100$.



(f) Absolute gap (%), $n = 100$.

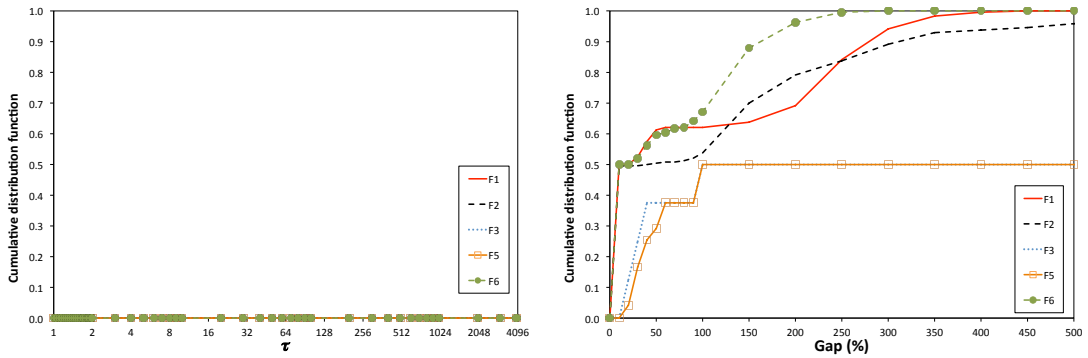


(g) Relative CPU time (s), $n = 200$.



(h) Absolute gap (%), $n = 200$.

Figure 4 Performance profiles based on the CPU time (left) and absolute gap (right) of each MILP formulation. Results obtained with the solvers CPLEX 12.6.1.0, GUROBI 6.0.0, and XPRESS 27.01.02. $n = \{50, 75, 100, 200\}$, $M = n/1.25$. Stopping criteria: optimality gap of 0.0% or 7200 seconds. 240 instances.



(a) Relative CPU time (s), $n = 300$.

(b) Absolute gap (%), $n = 300$.

Figure 5 Performance profiles based on the CPU time (left) and absolute gap (right) of each MILP formulation. Results obtained with the solvers CPLEX 12.6.1.0, GUROBI 6.0.0, and XPRESS 27.01.02. $n = 300$, $M = n/1.25$. Stopping criteria: optimality gap of 0.0% or 7200 seconds. 240 instances.

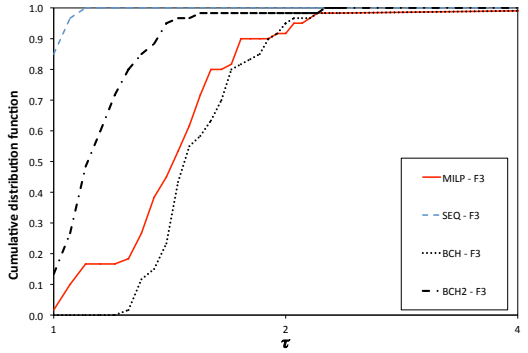
In the Supplementary Results document, we present additional results for these methods applied to smaller instances, but using a lower CPU time limit. There we show the results for $n = 75$, and $M = n/5$, with $SD = \{0.50, 0.75, 1.00\}$, with the time limit of 1000 seconds. The results show that the methods proposed have some advantage over the use of the formulation (F3).

The results presented before show that formulation (F3) has a good performance for problems with $n = \{50, 75\}$ and $M = n/5$, but it has difficulties to solve problems with $n = 100$ and $M = n/5$, within the maximum CPU time set of 7200 seconds. When a smaller CPU time is set, then formulation (F3) has difficulties to solve the problems with $n = 75$. The two methods proposed here may help to find better solutions at the frontier of the capacity of formulation (F3) to solve the problems to optimality. However, these methods are limited by the applicability of the triangular inequalities to large problems. These inequalities increase significantly the size of the continuous linear relaxation, and therefore, for large problems they lead to large continuous relaxation problems that limit the performance of the MILP solvers. We try to overcome this limitation by just adding a fraction of the cuts generated in the method labeled as BCH2.

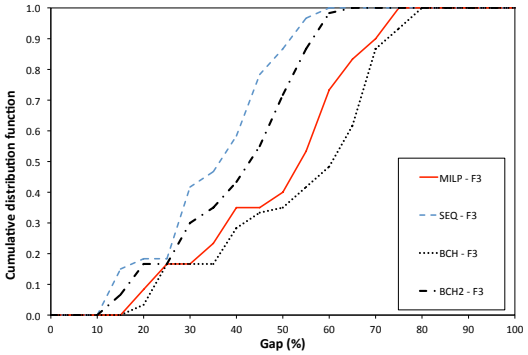
Note that the implementation of the BCH tool in GAMS does not allow to use multiple threads. This restriction provides stability to the performance of the solver and GAMS, but simultaneously may limit the performance of using callbacks in the second method.

5.3 Direct solution of the CBQP problem

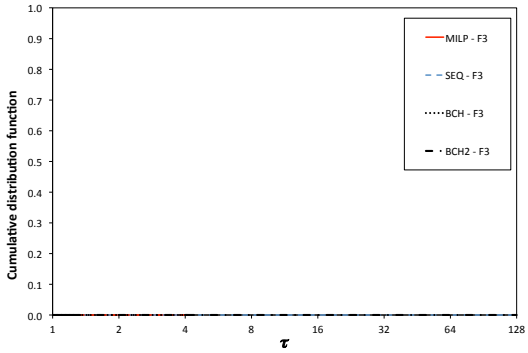
The CBQP problem (P1) can be solved directly by the solvers CPLEX 12.6.1.0, GUROBI 6.0.0, XPRESS 27.01.02, BARON 14.4.0, ANTIGONE 1.1, and SCIP 3.1 without requiring any user reformulation into



(a) Relative gap (%), $n = 100$.



(b) Absolute gap (%), $n = 100$.



(c) Relative CPU time (s), $n = 100$.

Figure 6 Performance profiles based on the relative and absolute gap (top) and CPU time bottom for the MILP formulations and tailored MILP approaches obtained woth CPLEX 12.6.1.0 for the instances with $n = 100$, $M = n/5$, $SD = \{0.50, 0.75, 1.00\}$. Stopping criteria: optimality gap of 0.0% or 7200 seconds. 60 instances.

an MILP problem. However, CPLEX 12.6.1.0, GUROBI 6.0.0, XPRESS 27.01.02 may internally reformulate the problem (P1) into an MILP problem.

The results obtained with these solvers over the 80 instances with $M = n/5$ for the cases with $n = \{50, 75, 100, 200, 300\}$ are presented in Figures 7 and 8. The profiles show that CPLEX 12.6.1.0 - MIQP has a clear advantage over the other solvers and methods for $n = \{50, 75\}$, see Figures 7(a) to 7(d). For $n = 100$, CPLEX 12.6.1.0 - MILP has a probability of being the fastest to close the gap of 25%, which corresponds to the problems with $SD = 0.10$, see Figure 7(e). For $SD = 0.10$, CPLEX 12.6.1.0 - MILP is a better option than CPLEX 12.6.1.0 - MIQP. However, for $n = 100$ as well as for $n = \{200, 300\}$, CPLEX 12.6.1.0 - MIQP has a better distribution of the absolute gaps, see Figures 7(f), 7(h), and 8(b).

Among the general purpose global optimization codes, for the smaller problems there are some differences between their performance but not significant, check for example Figure 7(a). While for the

larger problems BARON 14.4.0 performs similar to ANTIGONE 1.1 and CPLEX 12.6.1.0 - MILP, for example analyze their performance in Figure 7(e), and the overlap of their profiles for the absolute gap below 100% in Figures 7(h) and 8(b).

The results obtained with these solvers over the 80 instances with $M = n/1.25$ for the cases with $n = \{50, 75, 100, 200, 300\}$ are available in the Supplementary Results document.

5.3.1 Direct solution vs. MILP reformulations

In this section, the results obtained for each MILP formulation are compared with the direct solution using CPLEX 12.6.1.0, GUROBI 6.0.0, and XPRESS 27.01.02.

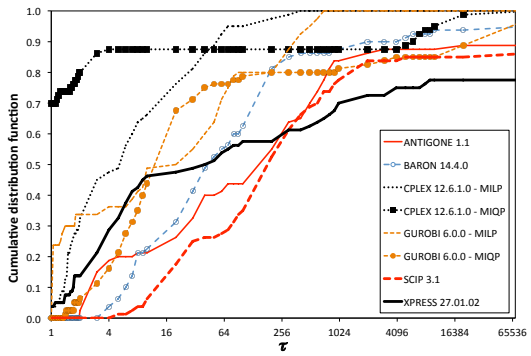
Figures 9 to 11 show that CPLEX 12.6.1.0 - MIQP has a high probability of being the best solver for any given problem with $SD = \{0.50, 0.75, 1.00\}$, while with GUROBI 6.0.0 the formulation (F3) is the winning approach to solve the problems for $n = \{50, 75\}$, and with XPRESS 27.01.02 the formulation (F3) is again the approach with the highest probability for $n = \{50, 75\}$. For the larger problems, there is not a clear winner approach for these two last solvers.

The initial values of the relaxations for one particular instance are presented in Figure 12 for the MILP formulations, and for the CBQP problem solved directly. For this instance formulation (F4) is the tightest formulation, and CPLEX 12.6.1.0 - MIQP has the second best lower bound at the root node. Note that there are several formulations that have the lower bound equal to 0.0. The tight bound obtained by CPLEX 12.6.1.0 - MIQP plus the fact that during the B&B it deals with a much smaller problem than the MILP formulations, contributes to the relative good performance of the solver as shown in Figures 7, 8, and 9.

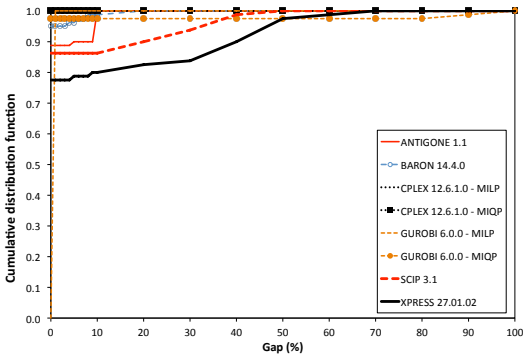
5.4 Contributions of algorithms and hardware developments

In this section, the contribution of software developments within one MILP solver, and the impact of using more than one CPU thread for the MILP and MIQP approaches is discussed. Here, the only solver used is CPLEX and two different versions are considered, 7.1 and 12.6.1.0. The first version does not have an MIQP solver available, nor the option to use more than one thread. In terms of hardware, the focus is on the number of threads used within the B&B search. The formulations (F1) and (F3) are solved for the case with $n = 50$, $M = n/5$, and formulations (F2), (F3), and (F6) for the case with $n = 50$, $M = n/1.25$.

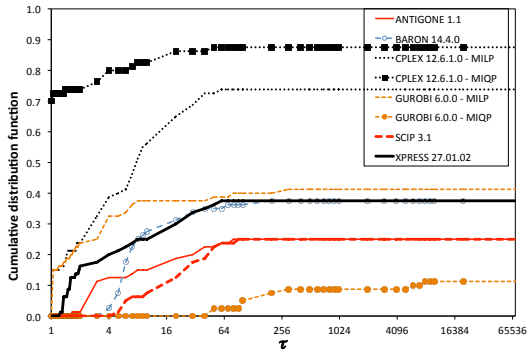
The results show that for $n = 50$, $M = n/5$, using the best formulation, (F3), with CPLEX 7.1 remains a good option when compared to using the version 12.6.1.0 with a weak formulation, (F1), see Table 4. In addition, with the formulation (F1), CPLEX 7.1 has a better performance than CPLEX 12.6.1.0 with only one thread, which is surprising.



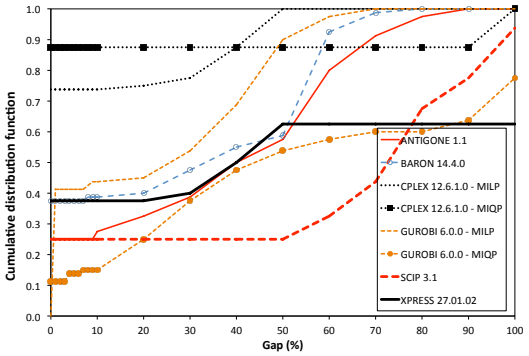
(a) Relative CPU time (s), $n = 50$.



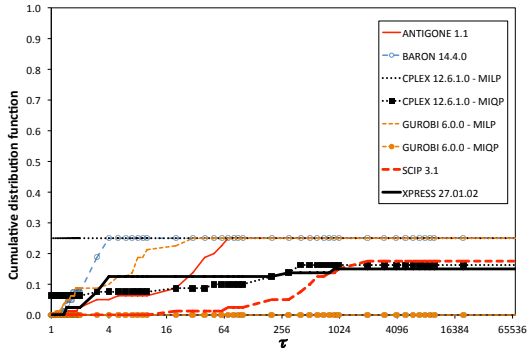
(b) Absolute gap (%), $n = 50$.



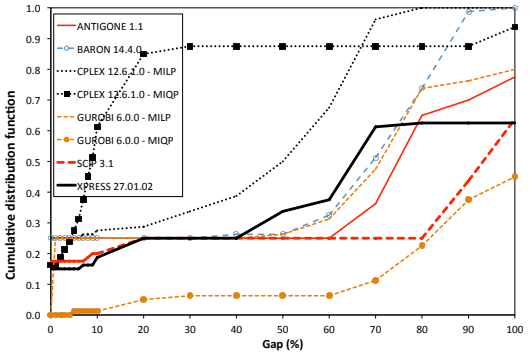
(c) Relative CPU time (s), $n = 75$.



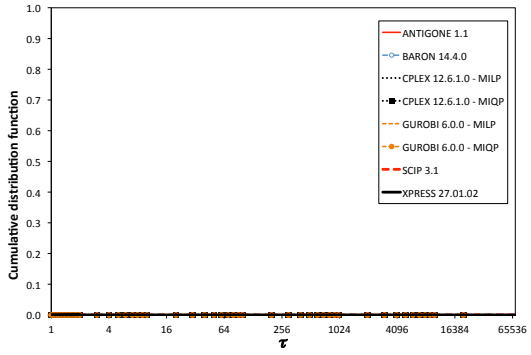
(d) Absolute gap (%), $n = 75$.



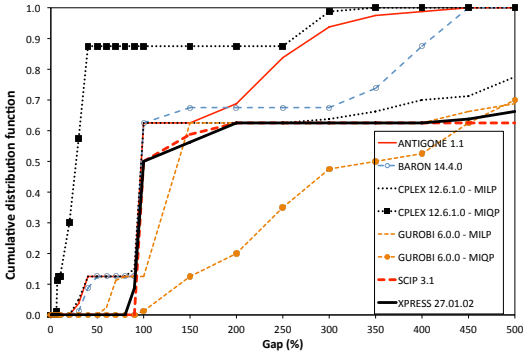
(e) Relative CPU time (s), $n = 100$.



(f) Absolute gap (%), $n = 100$.

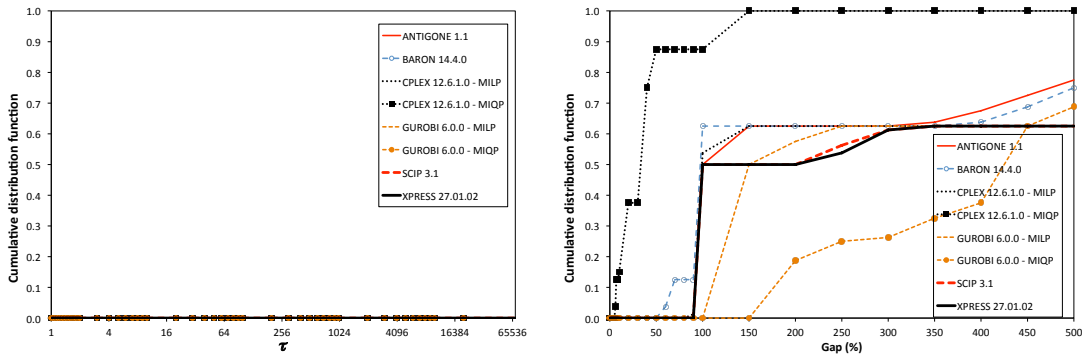


(g) Relative CPU time (s), $n = 200$.



(h) Absolute gap (%), $n = 200$.

Figure 7 Performance profiles for the solvers applied directly to the CBQP problem. $n = \{50, 75, 100, 200\}$, $M = n/5$. Stopping criteria: optimality gap of 0.0% or 7200 seconds. 80 instances.



(a) Relative CPU time (s), $n = 300$.

(b) Absolute gap (%), $n = 300$.

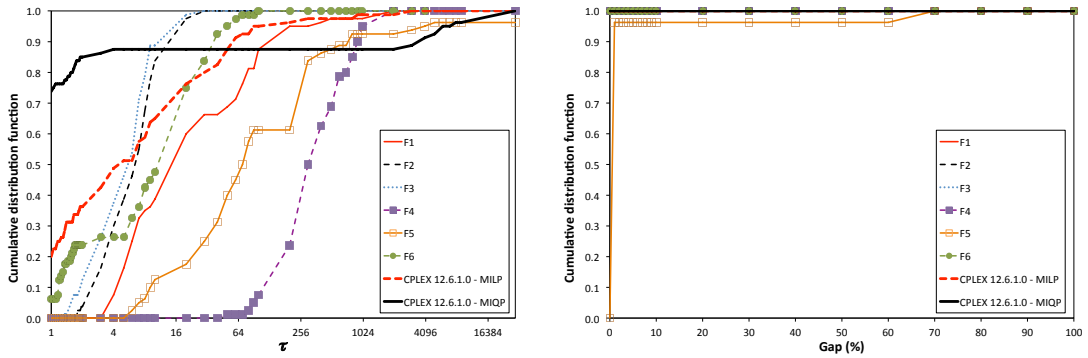
Figure 8 Performance profiles for the solvers applied directly to the CBQP problem. $n = 300$, $M = n/5$. Stopping criteria: optimality gap of 0.0% or 7200 seconds. 80 instances.

For $n = 50$, $M = n/1.25$, the trend is similar, but with the worst formulation, (F3), CPLEX 12.6.1.0 with 8 threads still has difficulties to close the gap of some instances, and the more efficient formulations are still competitive when solved with CPLEX 7.1. For example, with formulation (F6), CPLEX 7.1 on average needs 3 seconds to solve a problem, while with formulation (F3), CPLEX 12.6.1.0 with 8 threads presents an average gap of 4.2% and average CPU time of 660 seconds, see Table 5.

These results demonstrate that strong formulations are worthwhile, and may outperform algorithmic advances implemented in CPLEX 12.6.1.0. The performance of CPLEX applied to the MILP reformulations as well as when applied directly, with both the MILP and MIQP approaches, suggests that for this problem, the number of threads used has a significant impact over the CPU time required to close the gap.

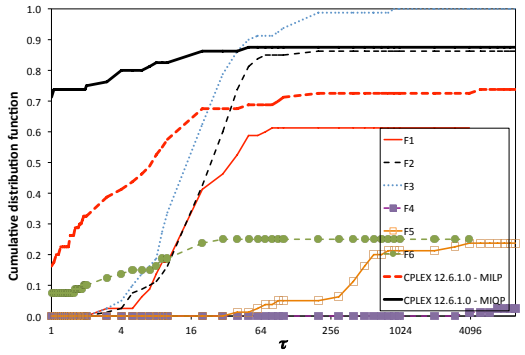
Table 4 Comparison between CPLEX 7.1 and CPLEX 12.6.1.0. Average values for 80 instances for $n = 50$, $M = n/5$. Stopping criteria: optimality gap of 0.0% or 7200 seconds.

	CPLEX 7.1		CPLEX 12.6.1.0 - 1 Thread		12.6.1.0 - 8 threads	
Formulation	Gap (%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)
F1	0.0	354	0.0	597	0.0	100
F3	0.0	75	0.0	38	0.0	7
Direct solution						
MILP			0.0	189	0.0	33
MIQP			12.5	136	0.0	70

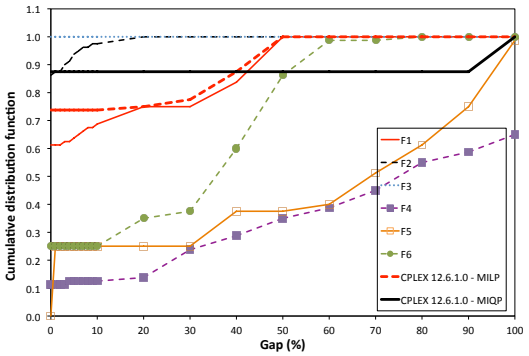


(a) Relative CPU time (s), $n = 50$.

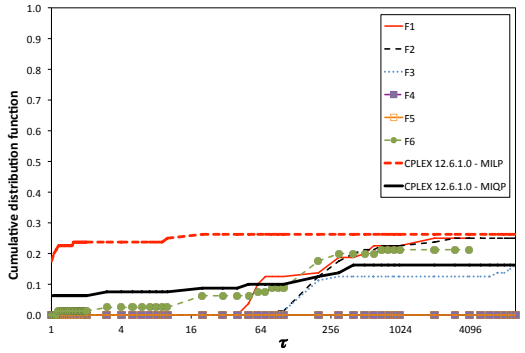
(b) Absolute gap (%), $n = 50$.



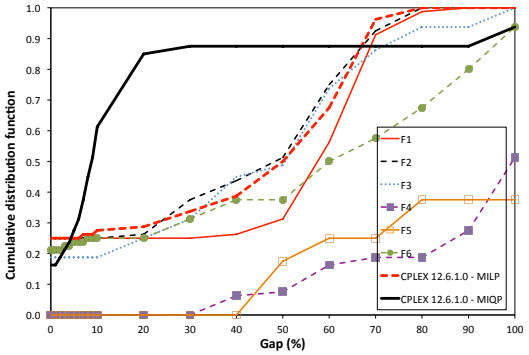
(c) Relative CPU time (s), $n = 75$.



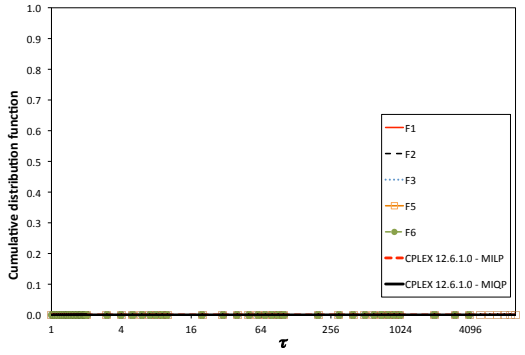
(d) Absolute gap (%), $n = 75$.



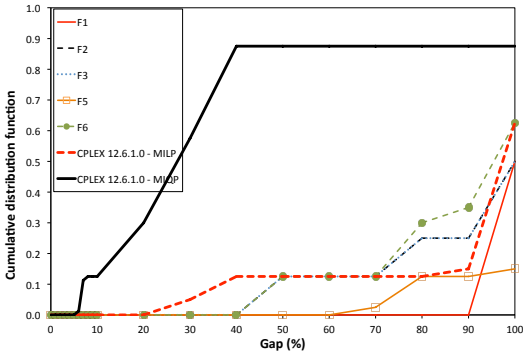
(e) Relative CPU time (s), $n = 100$.



(f) Absolute gap (%), $n = 100$.

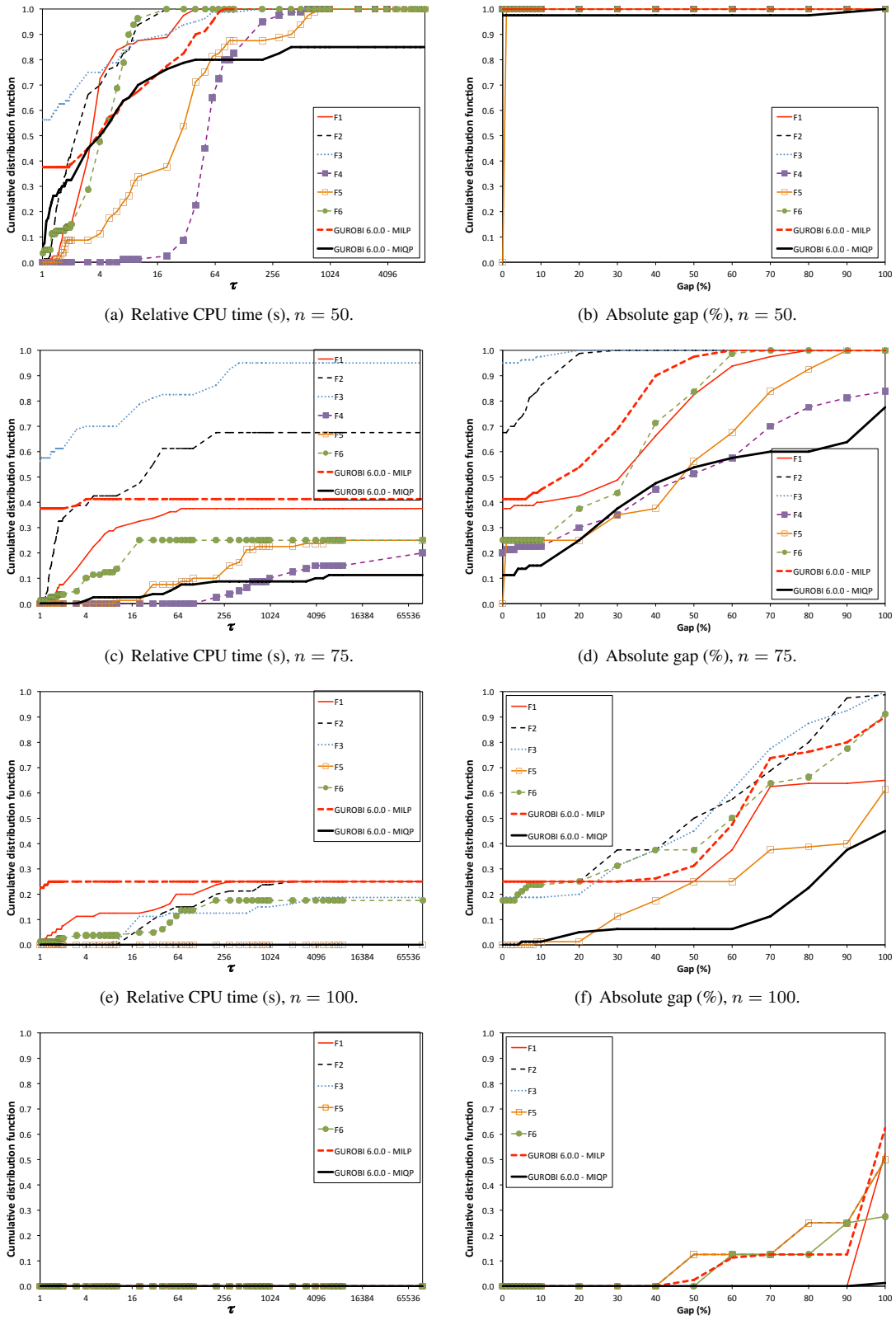


(g) Relative CPU time (s), $n = 200$.



(h) Absolute gap (%), $n = 200$.

Figure 9 Performance profiles based on the CPU time (left) and absolute gap (right) for CPLEX 12.6.1.0. $n = \{50, 75, 100, 200\}$, $M = n/5$. Stopping criteria: optimality gap of 0.0% or 7200 seconds. 80 instances.



(a) Relative CPU time (s), $n = 50$.

(b) Absolute gap (%), $n = 50$.

(c) Relative CPU time (s), $n = 75$.

(d) Absolute gap (%), $n = 75$.

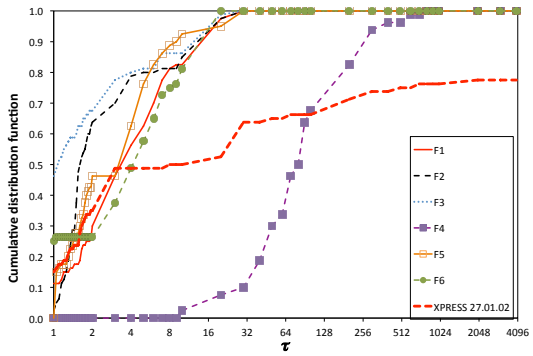
(e) Relative CPU time (s), $n = 100$.

(f) Absolute gap (%), $n = 100$.

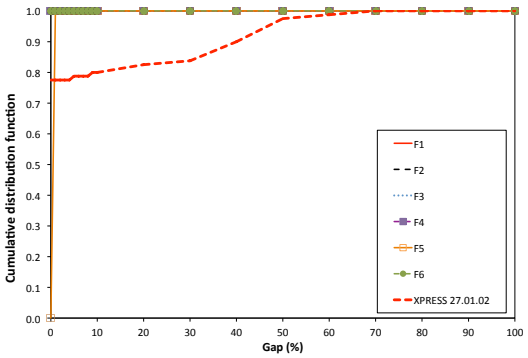
(g) Relative CPU time (s), $n = 200$.

(h) Absolute gap (%), $n = 200$.

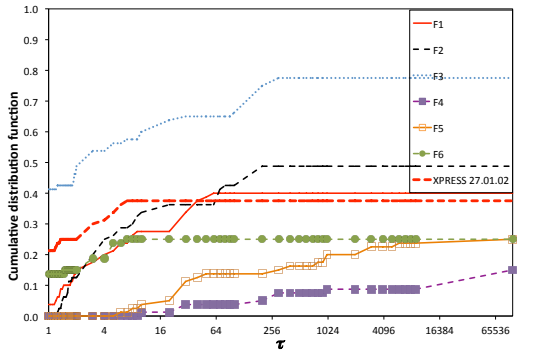
Figure 10 Performance profiles based on the CPU time (left) and absolute gap (right) for GUROBI 6.0.0. $n = \{50, 75, 100, 200\}$, $M = n/5$. Stopping criteria: optimality gap of 0.0% or 7200 seconds. 80 instances.



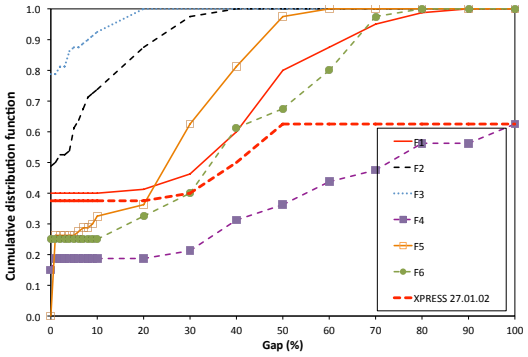
(a) Relative CPU time (s), $n = 50$.



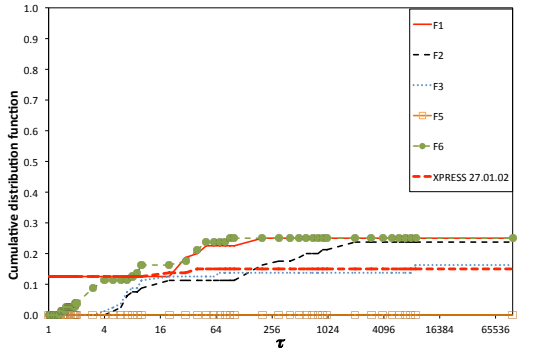
(b) Absolute gap (%), $n = 50$.



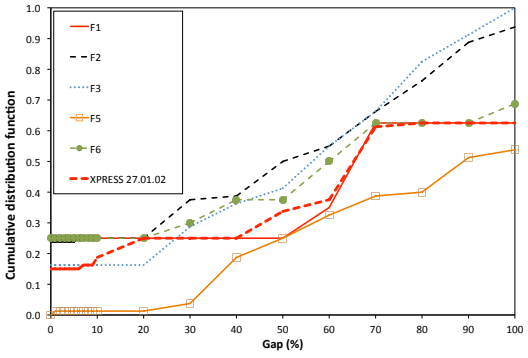
(c) Relative CPU time (s), $n = 75$.



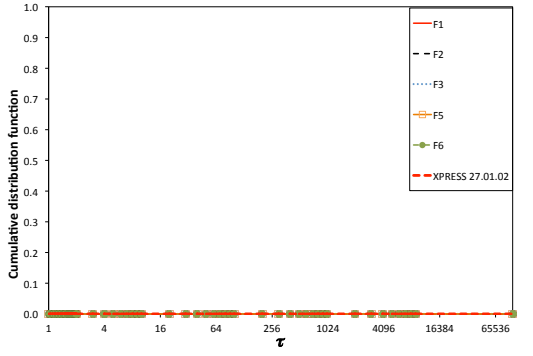
(d) Absolute gap (%), $n = 75$.



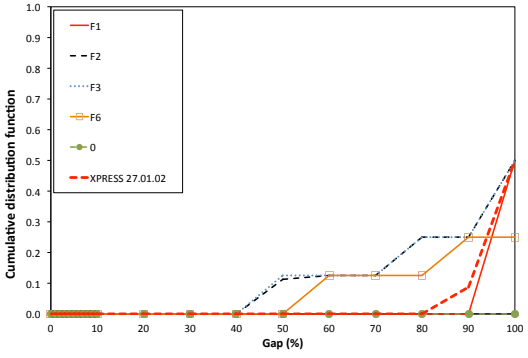
(e) Relative CPU time (s), $n = 100$.



(f) Absolute gap (%), $n = 100$.



(g) Relative CPU time (s), $n = 200$.



(h) Absolute gap (%), $n = 200$.

Figure 11 Performance profiles based on the CPU time (left) and absolute gap (right) for XPRESS 27.01.02. $n = \{50, 75, 100, 200\}$, $M = n/5$. Stopping criteria: optimality gap of 0.0% or 7200 seconds. 80 instances.

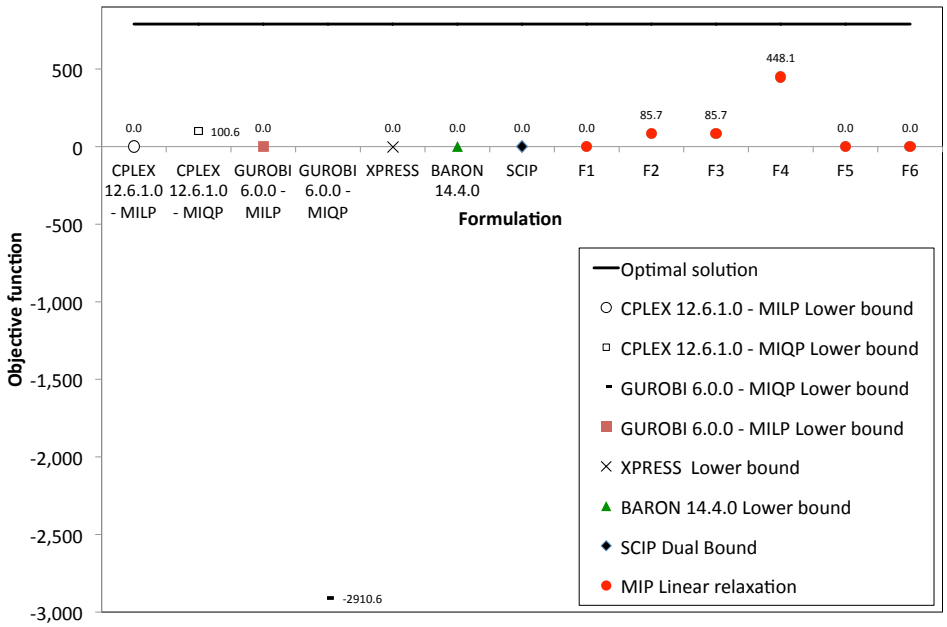


Figure 12 Optimal values of the continuous relaxations of the MILP formulations and at the root node of the solvers used. $n = 50$, $M = n/5$, $Q(0, 100, 0.5)$.

Table 5 Comparison between CPLEX 7.1 and CPLEX 12.6.1.0. Average values for 80 instances for $n = 50, M = n/1.25$. Stopping criteria: optimality gap of 0.0% or 7200 seconds.

	CPLEX 7.1		CPLEX 12.6.1.0 - 1 Thread		12.6.1.0 - 8 threads	
Formulation	Gap (%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)
F2	0.0	6	0.0	3	0.0	2
F3	13.5	6794	4.5	4611	4.2	660
F6	0.0	3	0.0	2	0.0	1
Direct solution						
MILP			0.0	18	0.0	4
MIQP			0.0	0.3	0.0	0.5

5.5 Remarks for the overall results

1. Based on the techniques studied and on the results obtained, we provide the following recommendations for the solution of CBQP problems:
   - Use a straightforward approach (Section 4) by applying CPLEX to the original model (P1) with the convexification of the objective function active and use multiple threads. This should be a good option for non-sparse matrices Q, and it does not require any reformulation.
   - Test formulations (F3) or (F2) with an MILP solver. Tune the solver to the problem at hand, see for example Lima [50] or Lima and Grossmann [46].

- Implement a method based on specific features of the solvers to improve the performance of the MILP formulations. For example, test the addition of the triangular inequalities through callbacks.
- Test a concurrent optimization approach using multiple MILP formulations and solvers to cover all the relevant features of the data.

2. This work has a set of supplementary results available that provide additional information about the performance of the solvers and formulations. These results include:
   (a) Additional results for Sections 5.1, 5.2, 5.3, and 5.4.
   (b) Analysis of the variability of the results introduced by the different instances of matrix Q. Those results, show for example that for matrices Q with $SD = 0.10$: a) CPLEX 12.6.1.0 - MILP is the best approach; b) GUROBI 6.0.0 - MILP is the best approach; and c) with XPRESS27.01.02, the formulation (F6) performs better. While for matrices Q with $SD = 1.00$: a) CPLEX 12.6.1.0 - MIQP is clearly the best approach; b) with GUROBI, formulation (F3) is much more efficient than any other approach; and c) with XPRESS, formulation (F3) is also more effective than any other approach.

6 Conclusions

In this paper, three different approaches to solve CBQP problems are presented, and the corresponding computational results for 10 case studies and 80 instances are discussed. The size of the original problems studied range from 50 to 300 binary variables. The size of these problems may seem moderate, but in fact, we can only solve to optimality the smaller problems, with 50 and 75 binary variables, and some problems with 100 binary variables, if matrices Q with low density are considered.

In the first approach, six different MILP formulations are implemented and studied from the theoretical and computational point of view. From the set of MILP formulations, there is a subset of formulations that are comparable in terms of quality of the lower bound, and another subset of formulations where the quality of the lower bound is inferred by computational results. In the section of the computational experiments, we provide an extensive quantitative analysis of the performance of the solvers and formulations, and therefore, here we reduce our conclusions to a qualitative analysis. The results show that the parameter M involved in the cardinality constraint, and the sparsity of the matrix Q have a significant impact on the lower bound of the MILP formulations, and consequently on the relative computational performance of the formulations. Overall, formulation (F3) was found to be the most efficient for $M = n/5$ and original problems with 50 and 75 binary variables, and (F2) for $M = n/1.25$ and original problems with 50, 75, and 100 binary variables. For $M = n/1.25$ and original problems with 200 binary variables the formulation (F1) outperforms the other formulations for some instances.

The second approach based on specific features of CPLEX 12.6.1.0 can improve the quality of the solution, in terms of optimality gap, for problems with 100 binary variables. For problems with 75 binary variables and with a smaller computational budget, the methods of the second approach can solve

MILP problems that the first approach based on applying CPLEX 12.6.1.0 to the MILP formulations cannot solve. However, the performance is still not as good as using CPLEX 12.6.1.0 with the automatic convexification of the objective function.

In general, CPLEX 12.6.1.0 with the automatic convexification of the objective function is more efficient than the best MILP reformulations. The results also show that using GUROBI 6.0.0, and XPRESS 27.01.02, the reformulations (F2) and (F3) are clearly more efficient than solving the CBQP problem directly with these solvers. However, the automatic convexification approach implemented in CPLEX 12.6.1.0 is not useful for matrices Q with low densities. The remaining solvers used, BARON 14.4.0, ANTIGONE 1.1 and SCIP 3.1 rely on convexification procedures that do not seem efficient in handling the nonlinearities in the CBQP problem.

The comparison of the results obtained with the last version of CPLEX 12.6.1.0 with 1 and 8 threads, and with version 7.1 shows the importance of a proper formulation and the positive impact that the number of threads used may have on the required computational time.

Acknowledgements The first author acknowledges the support of the Center for Uncertainty Quantification in Computational Science & Engineering. This research used resources in the King Abdullah University of Science and Technology Scientific Computing Center, supported by the Information Technology, Research Computing Team. The authors would like to thank the Center for Advanced Process Decision-making at Carnegie Mellon University for their financial support.

Appendix A. A note on formulation (F3)

The validity of formulation (F3) can be established as follows. The constraint $z_{ij} \geq x_i + x_j - 1$ is not considered in (F3), which forces $z_{ij} = 1$ when $x_i = 1$ and $x_j = 1$. In formulation (F3), this relation is enforced by constraint (9). To demonstrate this, consider that $M = 3$, $x_j = 1$, $x_k = 1$, $x_l = 1$, $x_i = 0, \forall\, i \neq j, i \neq k, i \neq l$, and $j < k < l$ then the constraint in (9) results in

$$\sum_{s<j} z_{sj} + \sum_{s>j} z_{js} = 2, \tag{A.1}$$

$$\sum_{s<k} z_{sk} + \sum_{s>k} z_{ks} = 2, \tag{A.2}$$

$$\sum_{s<l} z_{sl} + \sum_{s>l} z_{ls} = 2, \tag{A.3}$$

$$\sum_{s<i} z_{is} + \sum_{s>i} z_{is} = 0, \qquad \forall\, i \neq j, i \neq k, i \neq l \tag{A.4}$$

where in each (A.1), (A.2), and (A.3) there are only two positive variables z, because the fourth set of equations forces all the other variables to zero, leading to $z_{jk} = 1$, $z_{jl} = 1$, $z_{kl} = 1$, which is a solution of the original problem. Note that in (A.4), we have $x_i = 0$ and then the $rhs = 0$.

Appendix B. Formulation (F6)

Formulation (F6) compared with the other formulations has some additional elements that require some clarification. Therefore, in this appendix, we derive formulation (F6), and we explain the meaning of the variable t_i, and the parameters L_i and U_i. The derivation of the formulation is based on the linearization technique proposed in [39], and on one of the formulation derived in [9].

We start by defining the CBQP problem of interest as

$$\begin{aligned} \min \ & \sum_i q_{ii} x_i + \sum_i \sum_{j>i} (q_{ij} + q_{ji}) x_i x_j \\ \text{subject to } & \sum_i x_i = M \\ & x_i \in \{0,1\}, \qquad \forall\, i. \end{aligned} \tag{A.5}$$

Following Glover [39] we define a new variable z_i that is only indexed by i as

$$z_i = x_i \left[\sum_{j<i} (q_{ij} + q_{ji}) x_j + \sum_{j>i} (q_{ij} + q_{ji}) x_j \right], \qquad \forall\, i, \tag{A.6}$$

which is used to replace the quadratic term in the objective function of (A.5). Next the linearization technique is applied to the product between x_i and the term within the square brackets in (A.6). For this linearization we need to define the lower and upper bounds for the term inside of the square brackets in (A.6):

$$L_i \leq \sum_{j<i} (q_{ij} + q_{ji}) x_j + \sum_{j>i} (q_{ij} + q_{ji}) x_j \leq U_i, \qquad \forall\, i. \tag{A.7}$$

In the context of the problem (A.5), because of the cardinality constraint, L_i is equal to the sum over j of the $M-1$ smallest coefficients $q_{ij} + q_{ji}$, and U_i is equal to the sum over j of the $M-1$ greatest coefficients $q_{ij} + q_{ji}$, see [9] for additional details. Using the linearization technique from Glover [39] applied to (A.6), problem (A.5) can be re-written as

$$\begin{aligned} \min \ & \sum_i q_{ii} x_i + \tfrac{1}{2} \sum_i z_i \\ \text{subject to } & \sum_i x_i = M \\ & z_i \geq \sum_{j<i} (q_{ij} + q_{ji}) x_j + \sum_{j>i} (q_{ij} + q_{ji}) x_j - U_i (1 - x_i), \qquad \forall\, i \\ & z_i \geq L_i x_i, \qquad \forall\, i \\ & x_i \in \{0,1\}, \qquad \forall\, i. \end{aligned} \tag{A.8}$$

Note that the fraction $1/2$ in the objective function of (A.8) is needed to prevent the double accounting of $(q_{ij} + q_{ji})$ in z_i and z_j when $x_i = 1$ and $x_j = 1$. Note also that because we are minimizing, we do not

need to include in (A.8) the other two constraints derived in the linearization technique proposed in [39]:

$$z_i \leq \sum_{j<i} (q_{ij} + q_{ji}) x_j + \sum_{j>i} (q_{ij} + q_{ji}) x_j - L_i(1 - x_i), \qquad \forall\, i \tag{A.9}$$

$$z_i \leq U_i x_i, \qquad \forall\, i. \tag{A.10}$$

Defining $t_i = z_i - U_i x_i$ and replacing z_i by $t_i + U_i x_i$ in problem (A.8), this problem can be re-written as

$$\begin{aligned} \min\ & W = \sum_i q_{ii} x_i + \tfrac{1}{2} \sum_i t_i + \tfrac{1}{2} \sum_i U_i x_i \\ \text{subject to}\ & \sum_i x_i = M \\ & t_i \geq \sum_{j<i} (q_{ji} + q_{ij})\, x_j + \sum_{j>i} (q_{ij} + q_{ji})\, x_j - U_i, \qquad \forall\, i \\ & t_i \geq (L_i - U_i)\, x_i, \qquad \forall\, i \\ & x_i \in \{0, 1\}, \qquad \forall\, i, \end{aligned} \tag{F6}$$

which leads to formulation (F6).

Appendix C. Performance profiles

The performance profiles presented are based on the work of Dolan and Moré [48], who have proposed an alternative graphical comparison methodology to support the benchmarking of optimization software. This type of profiles are used in several benchmark tests, and they are an alternative to other comparison techniques based on averages and standard deviations, or other statistical indicators.

The performance profiles represent for a given set of results, P, the cumulative distribution function, $p_s(\tau)$, of the performance ratios between the computational time of the instance p solved with the solver s and the minimum of the computational times taken by the other solvers for the same instance. Thus the performance ratio is given by

$$\rho_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : 1 \leq s \leq n_s\}}\,, \quad \forall\, p, s \tag{A.11}$$

and the cumulative performance profile is defined as

$$p_s(\tau) = \frac{1}{n_p} \text{ size } \{p \in P : \rho_{p,s} \leq \tau\}\ , \quad \forall\, s \tag{A.12}$$

where $t_{p,s}$ is the computational time that solver s needed to solve problem p, n_s is the number of solvers, n_p is the number of instances, and τ denotes the ratio of the time factor of interest. The performance ratios with $\tau = 1$ correspond to the solvers with the minimum computational time, and therefore, $p_s(1)$ is the probability of the solver s to have the best performance on any given problem [48].

As an example consider the performance profiles based on the CPU time in Figure 1, where the profiles for the MILP formulations are presented. These performance profiles are built over the formulations, and not over the solvers. This means that each formulation corresponds to an index s (denoted by solver in the nomenclature above), and thus $n_s = 6$, and $n_p = 80\,\text{instances} \times 3\,\text{MILP solvers} = 240$. Each instance of these 240 is then solved for each formulation. For each instance the minimum computational time of the six formulations is calculated, and then the computational time of each formulation is divided by the minimum time, this is the performance ratio. Based on the calculation of the performance ratios for all instances, the cumulative profile is then calculated for a set of fixed ratios of time factors. $p_s(1)$ is the probability of the formulation s to be the best formulation to solve any given problem independently of the solver, and of the parameters of the 80 instances defined for matrix Q. For $\tau = 4$, $p_s(4)$ is the probability of any formulation to be the best within a time factor of 4 of the best formulation. The performance profiles provide also information about the formulations that fail to solve to optimality a set of problems. This is given by $1 - p_s(\tau)$, and thus, the profiles that do not meet $p_s(\tau) = 1$, mean that there is a probability that they will not solve a subset of problems to optimality within the maximum CPU time set. Figure 13 illustrates a performance profile based on the computational times.

The performance profiles based on the absolute gap [49] are easier to understand and construct than the profiles based on the computational times describe above. A cumulative performance profile using absolute gaps for each solver s is defined as

$$p_s(Gap(\%)) = \frac{1}{n_p} \text{size}\left\{p \in P : \gamma_{p,s} \leq Gap(\%)\right\} \ , \quad \forall\, s \tag{A.13}$$

where $Gap(\%)$ is the optimality gap of interest, and $\gamma_{p,s}$ is the optimality gap that solver s obtained for problem p. As an example consider the performance profiles based on the optimality gap in Figure 1, where the profiles for the MILP formulations are presented. These performance profiles are built over the formulations, and not over the solvers. This means that each formulation corresponds to an index s (denoted by solver in the nomenclature above), and thus $n_s = 6$, and $n_p = 80\,\text{instances} \times 3\,\text{MILP solvers} = 240$. For each formulation the cumulative profile of the optimality gaps of the 240 optimization runs are determined. $p_s(0)$ is the probability of the formulation s to solve any given problem to optimality independently of the solver, and of the parameters of the 80 instances defined for matrix Q. Figure 14 depicts an absolute profile for optimality gaps.

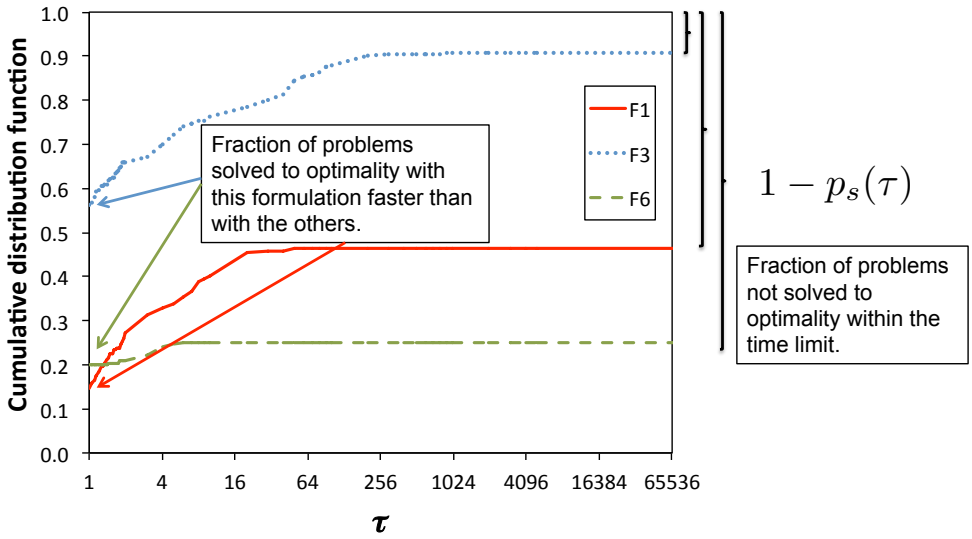


Figure 13 Example of performance profile built over the CPU time.

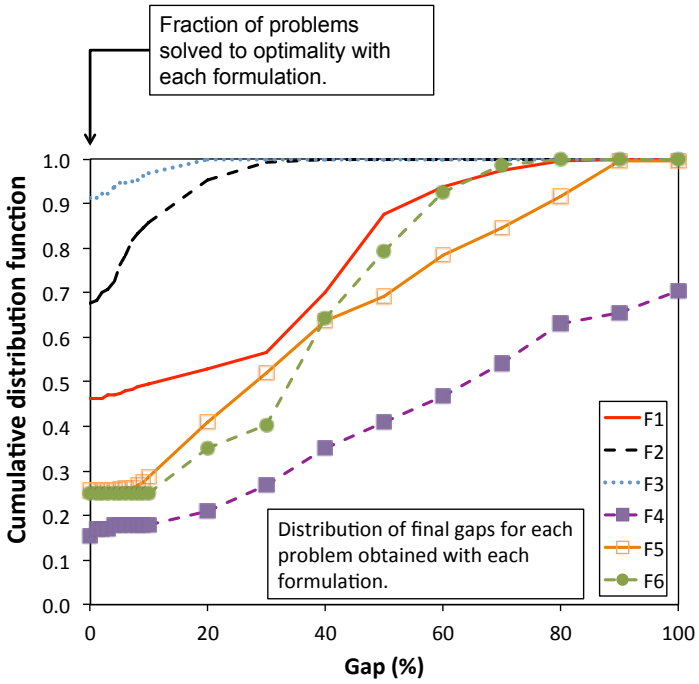


Figure 14 Example of a performance profile based on the absolute gap.

References

1. N. Krislock, J. Malick, and F. Roupin. Improved semidefinite bounding procedure for solving max-cut problems to optimality. *Math. Program., Ser. A*, 143:61–86, 2014.

2. L.A. Wolsey. *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998.
3. E. M. Loiola, N. M. M. Abreu, P.O. Boaventura-Netto, P. Hahn, and T. Querido. A survey for the quadratic assignment problem. *Eur. J. Oper. Res.*, 176(2):657 – 690, 2007.
4. A. Billionnet, M. C. Costa, and A. Sutter. An efficient algorithm for a task allocation problem. *J. ACM*, 39(3):502–518, 1992.
5. A.T. Phillips and J.B. Rosen. A quadratic assignment formulation of the molecular conformation problem. *J. Global Optim*, 4(2):229–241, 1994.
6. F. Barahona, M. Junger, and G. Reinelt. Experiments in quadratic 0-1 programming. *Math. Program.*, 44(2):127 – 137, 1989.
7. A. Caprara. Constrained 0-1 quadratic programming: Basic approaches and extensions. *Eur. J. Oper. Res.*, 187(3):1494–1503, 2008.
8. M. Padberg. The boolean quadric polytope - some characteristics, facets and relatives. *Math. Program.*, 45(1):139 – 172, 1989.
9. A. Billionnet. Different formulations for solving the heaviest k-subgraph problem. *INFOR*, 43(3): 171 – 186, 2005.
10. D Pisinger. Upper bounds and exact algorithms for p-dispersion problems. *Comput. Oper. Res.*, 33 (5):1380–1398, 2006.
11. Rafael Marti, Micael Gallego, and Abraham Duarte. A branch and bound algorithm for the maximum diversity problem. *Eur. J. Oper. Res.*, 200(1):36–44, 2010.
12. J. Malick and F. Roupin. Solving k-cluster problems to optimality with semidefinite programming. *Math. Program.*, 136(2, SI):279–300, 2012.
13. D. Bertsimas and R. Shioda. Algorithm for cardinality-constrained quadratic optimization. *Comput. Optim. Appl.*, 43(1):1–22, 2009.
14. M. Bruglieri, M. Ehrgott, H.W. Hamacher, and F. Maffioli. An annotated bibliography of combinatorial optimization problems with fixed cardinality constraints. *Discrete Appl. Math.*, 154(9):1344 – 1357, 2006.
15. R. Porn, O. Nissfolk, F. Jansson, and T. Westerlund. The Coulomb Glass - Modeling and Computational Experience with a Large Scale 0-1 QP Problem. In Pistikopoulos, E.N. and Georgiadis, M.C. and Kokossis, A.C., editor, *21st European Symposium on Computer Aided Process Engineering*, volume 29, pages 658–662, 2011.
16. T. Achterberg. SCIP solving constraint integer programs. *Math. Program.*, 1(1):1–41, 2009.
17. I.P. Androulakis, C.D. Maranas, and C.A. Floudas. αBB: A global optimization method for general constrained nonconvex problems. *J. Global Optim*, 7(4):337–363, 1995.
18. Inc. Lindo systems. *LINDOGlobal*, 2007. Available at `http://www.gams.com/dd/docs/solvers/baron.pdf`.

19. P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Waechter. Branching and bounds tightening techniques for non-convex MINLP. *Optim. Method. Softw.*, 24(4-5):597–634, 2009.
20. M. Tawarmalani and N.V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Math. Program.*, 103:225–249, 2005.
21. Ruth Misener and Christodoulos A. Floudas. ANTIGONE: Algorithms for continuous/integer global optimization of nonlinear equations. *J. Global Optim*, 59(2-3):503–526, July 2014.
22. F. Glover and E. Woolsey. Converting 0-1 polynomial programming problem to a 0-1 linear program. *Oper. Res.*, 2(1):180–182, 1974.
23. R. Raman and I. E. Grossmann. Relation between MILP modeling and logical inference for chemical process synthesis. *Comput. Chem. Eng.*, 15(2):73 – 84, 1991.
24. A. Billionnet and S. Elloumi. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Math. Program.*, 109(1):55 – 68, 2007.
25. D. Pisinger. The quadratic knapsack problem - a survey. *Discrete Appl. Math.*, 155(5):623–648, 2007.
26. A. Mehrotra. Cardinality constrained Boolean quadratic polytope. *Discrete Appl. Math.*, 79(1-3): 137 – 154, 1997.
27. E. Boros and P. L. Hammer. Cut-polytopes, boolean quadric polytopes and nonnegative quadratic pseudo-boolean functions. *Math. Oper. Res.*, 18(1):245 – 253, 1993.
28. P. M. Pardalos and G. P. Rodgers. Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing*, 45(2):131 – 144, 1990.
29. H. D. Sherali, Y. H. Lee, and W. P. Adams. A simultaneous lifting strategy for identifying new classes of facets for the boolean quadric polytope. *Oper. Res. Lett.*, 17(1):19 – 26, 1995.
30. S. Gueye and P. Michelon. "Miniaturized" linearizations for quadratic 0/1 problems. *Ann. Oper. Res.*, 140(1):235 – 261, 2005.
31. S. Gueye and P. Michelon. A linearization framework for unconstrained quadratic (0-1) problems. *Discrete Appl. Math.*, 157(6):1255 – 1266, 2009.
32. P. Hansen and C. Meyer. Improved compact linearizations for the unconstrained quadratic 0-1 minimization problem. *Discrete Applied Mathematics*, 157:1267–1290, 2009.
33. L. Liberti. Compact linearization for binary quadratic problems. *4OR*, 5(3):231–245, 2007.
34. S. Burer and A.N. Letchford. On nonconvex quadratic programming with box constraints. *SIAM J. Optimiz.*, 20(2):1073 – 1089, 2009.
35. E. L. Johnson, A. Mehrotra, and G. L. Nemhauser. Min-cut clustering. *Math. Program.*, 62(1):133 – 151, 1993.
36. A. Faye and Q. Trinh. Polyhedral results for a constrained quadratic 0-1 problem. Technical Report CEDRIC-03-511, CEDRIC laboratory, CNAM-Paris, France, 2003.
37. A. Faye and Q. A. Trinh. A polyhedral approach for a constrained quadratic 0-1 problem. *Discrete Appl. Math.*, 149(1-3):87 – 100, 2005.

38. E.M. Macambira and C.C. de Souza. The edge-weighted clique problem: Valid inequalities, facets and polyhedral computations. *Eur. J. Oper. Res.*, 123(2):346–371, 2000.
39. F. Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22:455–60, 1975.
40. E. Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS J. Comput.*, 19(4):534–541, 2007.
41. A. Brooke, D. Kendrick, A. Meeraus, and R. Raman. *GAMS - A user's guide*, 1998.
42. 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*, pages 171–180, Hosei University, Tokyo, Japan, 2014.
43. S. Billionnet, A.and 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 Appl. Math.*, 157(6):1185–1197, 2009.
44. T.P. Dinh, N.N. Canh, T. Le, and H. An. An efficient combined DCA and B&B using DC/SDP relaxation for globally solving binary quadratic programs. *J. Global Optim*, 48(4):595 – 632, 2010.
45. F. Rendl, G. Rinaldi, and A. Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Math. Program., Ser. A*, 121:307–335, 2010.
46. R. M. Lima and I. E. Grossmann. *Chemical Engineering Greetings to Prof. Sauro Pierucci*, chapter Computational advances in solving Mixed Integer Linear Programming problems, pages 151–160. AIDIC, 2011.
47. R. Bixby and E. Rothberg. Progress in computational mixed integer programming- a look back from the other side of the tipping point. *Ann. Oper. Res.*, 49(1):37–41, 2007.
48. E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91(2):201–213, 2002.
49. M. R. Bussieck, S. P. Dirkse, and S. Vigerske. Paver 2.0: an open source environment for automated performance analysis of benchmarking data. *J. Global Optim*, 59(2-3):259–275, 2014.
50. R. M. Lima. IBM ILOG CPLEX What is inside of the box?, 2010. Accessed on October 22, 2015, http://egon.cheme.cmu.edu/ewocp/docs/rlima_cplex_ewo_dec2010.pdf.