On the Solution of Nonconvex Cardinality Boolean Quadratic Programming problems

Ricardo M. Lima[a], Ignacio E. Grossmann[b]
[a] Unidade de Modelação e Optimização de Sistemas Energéticos, Laboratório Nacional de Energia e Geologia, Lisboa, Portugal
[b] Department of Chemical Engineering, Carnegie Mellon University, PA, USA
{ricardo.lima@lneg.pt, grossmann@andrew.cmu.edu}

Abstract

This paper addresses the solution of a nonlinear boolean quadratic programming problem using three different approaches. The first uses a classic linearization technique to transform the original problem into a Mixed Integer Linear Programming (MILP) problem for which multiple formulations are studied. The second uses the capabilities of current MILP solvers to deal with Mixed Integer Quadratic Programming (MIQP) problems, and the third relies on the utilization of optimization solvers available that can deal with nonlinear combinatorial problems. Two additional strategies relying on the MILP formulations are evaluated. Special emphasis is placed on the definition of computationally efficient MILP reformulations and their comparison with other approaches. The results indicate that the most efficient approach relies on solving the problem as a MIQP problem using a specific MILP solver from the set of solvers tested, while the most efficient MILP formulations are still a better option than solving the problem as a MIQP using the remaining MILP solvers tested, and their performance is considerably superior to the application of general purpose solvers.

Key words: Mixed Integer Programming, Boolean Quadratic Polytope, Nonconvex 0-1 Quadratic Programming
History:

---

1. Introduction

Boolean Quadratic Programming (BQP) problems are a specific 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 $c, b \in \Re^n$, $Q \in \Re^{n \times n}$, and $A \in \Re^{n \times n}$. BQP, as well as the unconstrained BQP, belongs to the class of NP-hard problems and is considered a classic problem in combinatorial optimization, see for example Padberg (1989) for the characterization of the polytope of an unconstrained BQP. Important real world applications of BQP include a class of problems on facilities location, the so called Quadratic Assignment Programming (QAP) (Loiola et al., 2007), tasks allocation (Billionnet et al., 1992), and molecular conformation (Phillips and Rosen, 1994).

In this work the focus is on the computational solution of a specific class of Cardinality BQP (CBQP) problems defined as:

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

where

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

and $c \in \Re^n$, $Q \in \Re^{n \times n}$ and Q is not necessarily a positive semidefinite matrix. This is a nonconvex nonlinear combinatorial problem involving a quadratic term in the objective function and a cardinality constraint. The formulation (P1) is a specific case of the BQP. An annotated review on CBQP is given by Bruglieri et al. (2006), where some applications are described and the problem is characterized. Typical applications include problems in edge-weighted graphs (see Billionnet (2005) for a detailed description), and facility location problems (Bruglieri et al., 2006).

The objective of this work is to evaluate the practical options available to solve CBQP problems using the common available software. 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 Mixed Integer Programming (MILP) formulations, which will be described later in this work. On the other hand, in the recent years MILP solvers have implemented algorithms to deal with Mixed Integer Quadratic Programming (MIQP) problems that are able to cope with nonconvex CBQP; examples include CPLEX, GUROBI, XPRESS and SCIP (Achterberg, 2009). In addition, nowadays there are several solvers available such as BARON (Tawarmalani and Sahinidis, 2005), or GloMiQO (Misener and Floudas, 2012), which can rigorously address nonlinear combinatorial problems. Therefore, based on the many options available to solve CBQP involving common available software, our goal is to identify the best approach, and therefore, three options are evaluated: 1) the solution by an MILP reformulation using an MILP solver, where efficient formulations are sought; 2) the utilization of MIQP solvers; and 3) using a set of solvers that can address this type of problems. In addition, two tailored strategies based on the MILP reformulation are evaluated: 1) solving a sequence of MILP problems with the goal of identifying violated

inequalities and at a latter step add them to the model; and 2) implementation of a callback to a user cut generation routine within a commercial MILP solver.

1.1. Reformulations of BQP problems

In general, the solution of a BQP using an MILP solution approach involves the linearization of the nonlinear term in the objective function and the construction of a valid MILP formulation. Note that in this work the term BQP is used to refer to the unconstrained version of the BQP. A well known MILP reformulation of the BQP is given by

$$\begin{array}{rl} \min & \mathbf{cx} + \sum\limits_{1 \le i < j \le n} (q_{ij} + q_{ji}) z_{ij} \\ \text{subject to} & z_{ij} \ge x_i + x_j - 1, \qquad \forall i < j \\ & z_{ij} \le x_i, \qquad \forall i < j \\ & z_{ij} \le x_j, \qquad \forall i < j \\ & z_{ij} \ge 0, \qquad \forall 1 \le i < j \\ & x \in B^n \end{array} , \tag{P2}$$

where the variables z are continuous and the constraint $z_{ij} \le 1$ is redundant. This formulation is built on the linearization technique proposed by Glover and Woolsey (1974) for 0-1 polynomials, whereas the term $x_i x_j$ is replaced by a new variable z_{ij} and the following inequalities are added:

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

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

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

The inequalities above can also be derived from the logical proposition (Raman and Grossmann, 1991):

$$x_i \wedge x_j \Leftrightarrow z_{ij}, \qquad 1 \le i < j \le n, \tag{4}$$

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

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

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

where the superscripts l and u represent the lower and upper bound, respectively, of the variables. Following the same reasoning the nonlinear CBQP can be transformed into the linear CBQP:

$$\begin{aligned} \min \quad & W = \sum_i c_i x_i + \sum_i \sum_{j>i} (q_{ij} + q_{ji})\, z_{ij} \\ \text{subject to} \quad & \sum_i x_i = M \\ & z_{ij} \geq x_i + x_j - 1, \qquad \forall i < j \\ & z_{ij} \leq x_i, \qquad \forall i < j \\ & z_{ij} \leq x_j, \qquad \forall i < j \\ & z_{ij} \geq 0 \qquad \forall 1 \leq i < j \\ & x_i \in \{0,1\}, \forall i. \end{aligned} \tag{P4}$$

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. An important remark is that important results valid for the BQP polytope are also valid for the CBQP (Mehrotra, 1997). Padberg (1989) is a classic reference on the characterization of the convex hull of the BQP problem, and has proposed three families of facets: triangle, clique, cut, and generalized cut inequalities. These inequalities were latter studied by Boros and Hammer (1993), who have established new relations between different types of inequalities. Barahona et al. (1989) and Pardalos and Rodgers (1990) addressed the solution of the BQP using algorithmic techniques within a Linear Programming Branch & Bound (BB) framework. The first proposed a branch and cut algorithm where cutting planes are applied on each node, while Pardalos and Rodgers (1990) have studied a specific preprocessing technique to fix the variables. Based on the inequalities developed by Padberg (1989), Sherali et al. (1995) proposed a new class of facets for the BQP. Gueye and Michelon (2005) and later Gueye and Michelon (2009) proposed a linearization framework where the objective is to derive reduced formulations of the classic formulation P2. Hansen and Meyer (2009) studied also the derivation of reduced derivations that aim at obtaining a good relaxation without comprimising computaional performance. Following the same objective, Liberti (2007) proposed an efficient compact linearization for BQP subject to assignment constraints. Burer and Letchford (2009) extended the results described for the BQP in the characterization of box constrained quadratic programming problem. Mehrotra (1997) studied the specific case of the following BQP:

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

where

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

and discussed the derivation of some inequalities based on the inequalities of the Knapsack Constrained Boolean Quadratic Programming (KCBQP) polytope (Johnson et al., 1993). This author 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, he has 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 on the solution of the separation problems. Faye and Trinh (2003, 2005) extended the work of Padberg (1989) over the BQP by considering the constraint:

$$\sum_{1 \leq x \leq n} x_i = M,$$

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 about the characterization of the problem and derivation of facet inducing inequalities without concerns on the practical implication of the resulting size of the problems and the impact on the computational solution. The importance of having a tighter relaxation for a given MILP problem is well known within a BB 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 global optimality. A reference on the theoretical and computational aspect is the work of Billionnet (2005), who has studied the efficiency of several MILP formulations for specific types of CBQP arising from weighted-edge graphs problems.

1.1.1. Formulations

In this section we introduce the formulations used for the 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 problems are not considered. Mehrotra (1997) reports that this type of problem and the associated separation problems of some inequalities may easily lead to additional intractable problems, and conditions the utilization of branch and cut algorithms. Our goal is to derive an efficient MILP formulation from the computational point of view, which means that the quality of the continuous relaxation may be sacrificed for the sake of a smaller formulation. Six different MILP formulations are described next.

The first formulation is the most naive linearization that leads to the smallest representation of

the CBQP problem:

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

The Formulation (F0) is only valid for problems with $q_{i'j} + q_{ji'} \geq 0$. For $q_{ij} + q_{ji} < 0$, this formulation is not equivalent to the original problem to be solved, because it leads to solutions with $x_{i'} = 0$ and $z_{i'j} = 1$, which are not valid solutions of the original problem. The following formulation is based on the classic linearization, which results by adding the inequalities $z_{ij} \leq x_i$ and $z_{ij} \leq x_j$ to the previous formulation.

$$\begin{aligned} \min \quad & W = \sum_i c_i x_i + \sum_i \sum_{j>i} \left(q_{ij} + q_{ji}\right) z_{ij} \\ \text{Subject to} \quad & \sum_i x_i = M \\ & z_{ij} \geq x_i + x_j - 1, \qquad \forall i < j \\ & z_{ij} \leq x_i, \qquad \forall i < j \\ & z_{ij} \leq x_j, \qquad \forall i < j \\ & z_{ij} \geq 0 \qquad \forall 1 \leq i < j \\ & x_i \in \{0,1\}, \forall i. \end{aligned} \tag{F1}$$

The inequalities $z_{ij} \leq x_i$ and $z_{ij} \leq x_j$ cut-off solutions with $x_{i'} = 0$ and $z_{i'j} = 1$ for $q_{i'j} + q_{ji'} < 0$, leading to a more generic formulation. The formulation (F2) is based on the classic formulation, plus it includes the following constraint:

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

which is in this work adapted from the star inequality proposed by Mehrotra (1997).

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

The formulation (F3) is built by eliminating the inequality $z_{ij} \geq x_i + x_j - 1$ from the formulation. The underlying rationale is that a more compact formulation is obtained, at the cost of a potential weaker lower bound

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

The formulation (F4) is based on (F3) but considering the triangular inequalities proposed by Padberg (1989) for sets of three variables defined by i, j, k.

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

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 tigher relaxation. Formulation (F5) is a more compact reformulation of (F3) without the constraint $z_{ij} \leq x_j$.

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

The last formulation is based on formulation (F3) with two additional equations:

$$\sum_i y_i = |I| - M$$

$$x_i = 1 - y_i, \qquad \forall i$$

leading to formulation (F6):

$$\begin{aligned}
\min \quad & W = \sum_i c_i x_i + \sum_i \sum_{j>i} \left(q_{ij} + q_{ji}\right) z_{ij} \\
\text{Subject to} \quad & \sum_i x_i = M \\
& z_{ij} \leq x_i \qquad \forall i < j \\
& z_{ij} \leq x_j \qquad \forall i < j \\
& \sum_{i<j} z_{ij} + \sum_{i>j} z_{j,i} = (M-1)\, x_j \qquad \forall j \\
& \sum_i y_i = |I| - M \\
& x_i = 1 - y_i, \qquad \forall i \\
& z_{ij} \geq 0, \qquad \forall 1 \leq i < j \\
& y_i \geq 0, \qquad \forall i \\
& x_i \in \{0, 1\}, \qquad \forall i
\end{aligned} \quad . \tag{F6}$$

The size of each formulation for $n = 50$ is summarized in Table 1 in terms of the number of equations, variables, and nonzero elements. The number of binary variables correspond to the number of variables x_i, while the number of continuous variables is associated with z_{ij}. Formulation (F0) is the most compact formulation, while formulation (F4) leads to the largest model due to the inclusion of the triangular inequalities, which are defined over $\forall i < j < k$. The formulations

Table 1: Size of the MILP formulations for $n = 50$.

Formulation	0-1 Var	Cont. Var	NEQ	NonZeros
F0	50	1,226	1,227	5,001
F1	50	1,226	3,677	9,901
F2	50	1,226	3,727	12,401
F3	50	1,226	2,502	8,726
F4	50	1,226	60,077	241,476
F5	50	1,226	1,277	6,276
F6	50	1,276	2,553	8,876

described 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.

1.1.2. Quality of the formulations

In this section a theoretical analysis of the quality of the relaxations of the MILP formulations described is presented. Here, the variable W_n represents the value of the objective function of the continuous relaxation of the formulation (Fn). The formulations (F0) and (F1) adapted for

the CBQP are well known to have a loose continuous relaxation. From these, the lower bound provided by (F1) will be at least great 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, the star inequality, that is known to be a strong valid inequality (Faye and Trinh, 2005), which leads to an improved relaxation. Therefore, the following relation is valid:

$$W_0 \leq W_1 \leq W_2.$$

The formulation (F3) is equal to the formulation (F2) minus the constraint $z_{ij} \geq x_i + x_j - 1$. It will be shown later that this inequality may be redundant for some instances, which may transform the formulation (F3) into an efficient approach for those cases. On the other hand, it may be a strong inequality, and thus, his omission leads to a weak formulation. Thus, the relation between their objective functions is the following:

$$W_3 \leq W_2.$$

The formulation (F4) is based on formulation (F3) and it includes additionally the triangular inequalities, therefore it is continuous relaxation is stronger than (F3):

$$W_3 \leq W_4.$$

The formulation (F5) results from eliminating some constraints in order to obtain a more compact formulation. Thus the objective function is bounded by W_3 and W_2:

$$W_5 \leq W_3 \leq W_2.$$

By the same reasoning the formulation (F6) leads to an objective function that is at least as tight as (F3):

$$W_3 \leq W_6.$$

1.2. Convexification of nonconvex BQP problems

Problem P1 is a pure combinatorial problem that can be solved through an MILP formulation as shown in the previous section, or it can be solved using recent algorithms available within the MILP solvers. Currently, the common available MILP solvers are able to solve MIQP problems with the following formulation:

$$\min\{c^T x + x^T Q x : x : Ax \leq b, x \in \{0,1\}^n\}, \tag{8}$$

with nonconvex objective functions, i.e. problems where the matrix Q is not positive semidefinite. Note that for relaxation problems with $x \in [0,1]^n$, the matrix Q must be positive semidefinite, in order to define a convex program that has a unique local minimum. The exact procedure used by the MILP solvers to transform the nonconvex objective function into a convex objective function is not clear. However, several authors have proposed convexification procedures. Billionnet and Elloumi (2007) show that the objective function of the problem

$$\min\{c^T x + x^T Q x : x \in \{0,1\}^n\} \qquad \text{(MIQP\_F)}$$

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

$$\min\{f_u(x) : x \in [0,1]^n\} \qquad \text{(QP\_Fu)}$$

that is then a valid lower bound of the former problem. Their objective was to derive a convex MIQP problem that could be used within a branch & bound 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 $f_u = f(x) + \sum_{i=1}^{n} u_i (x_i - x_i^2)$, 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 $(Q - D)$ results in a positive semidefinite matrix. The function f_u depends on the value of the vector u, and the matrix D is a diagonal matrix obtained from the vector u. Therefore, given that $f_u(x) = f(x)$ when $x \in \{0,1\}^n$, solving the nonconvex problem (MIQP_F) is equivalent to solve the problem

$$\min\{f_u(x) : x \in \{0,1\}^n\} \qquad \text{(MIQP\_Fu)}$$

within a branch & bound framework where the continuous relaxation is given by the problem (QP_Fu). However, the determination of a specific vector u such that $(Q - D) \succeq 0$ is essential to define the convex problem, and in addition a strong lower bound. 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) (Billionnet and Plateau, 2009). These authors proposed an interesting approach based on duality theory and on the solution of a SDP problem, solved in a preprocessing step. The solution of the SDP problem determines also a vector u, which is used to formulate a convex BQP, whereas their results show it leads to improved lower bounds.

2. Computational experiments

The computational experiments aim at evaluating the value of the first approach using different MILP formulations, and the performance of the different solvers available for directly solving this type of problem.

The MILP formulations described 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 BB based MILP solver. Furthermore, current MILP solvers have implemented several algorithms at the level of the pre-processing techniques, cutting planes, heuristics, parallelization and search over the tree, which make it difficult to estimate the performance of an MILP formulation (see Lima and Grossmann (2011); Bixby and Rothberg (2007); Rothberg (2007) for reviews on some MILP solver features). Therefore, the computational experiments were designed with the following objectives: 1) identify the best MILP formulation to handle the CBQP problem under study; 2) evaluate the performance of the MIQP solvers, and compare it with the best MILP formulations; 3) assess the utilization of MINLP solvers that were not designed specifically for nonlinear combinatorial problems, but still have the capability to solve them. The performance of the MILP formulations is analyzed using three different MILP solvers: CPLEX, GUROBI and XPRESS, while to solve the BQP as a MIQP, the same MILP solvers are used plus the solver SCIP. As an alternative the BQP problem is also solved using DICOPT, BARON and GloMIQO.

In order to assess the quality of the formulations and solvers, several cases studies based on a CBQP problem are considered by combining different sizes of the problem, $n = \{50, 75, 100\}$, and two different values for the constant M, $M = \{n/5, n/1.25\}$. A detailed computational analysis to evaluate the impact of the coefficients of the matrix Q is presented for two instances with $Q = \{U(5, 100), N(0, 1)\}$, where U(5,100) means that Q is generated using a random number generation function with an uniform distribution between 5 and 100; and N(0,1) indicates that Q is based on a normal distribution with mean 0 and standard deviation of 1. Note that with U(5,100) all coefficients of the matrix Q are positive, while for N(0,1) some coefficients may be negative. Therefore, the formulation (F0) is not used for the instances where $q_{ij} + q_{ji}$ may be negative. This analysis is made over all the solvers considered and is used to select a reduced set of approaches for further investigation.

However, to further infer the influence of Q, 20 additional instances with Q randomly generated are considered for the cases with $n = 50$ and $M = \{n/5, n/1.25\}$, and solved with CPLEX and

GUROBI using the MILP formulations and the MIQP original problem.

In this work the focus is on a single problem with different instances, in place of a battery of benchmark problems, relying on the fact that the instances selected are sufficient to show some of the computational trends involved when dealing with these type of problems. The computer used has an Intel Core i7@3.07GHz CPU, 64 bits, and 8Gb of RAM. All models are implemented in the modeling system GAMS.

The results are presented in four sections, in the first and second sections detailed computational results obtained with all solvers are discussed for $M = n/5$ and $M = n/1.25$, respectively. In the third section, the performance of the 22 instances of Q with GUROBI and CPLEX are analyzed, and in the last section the influence of the number of threads and software developments within one solver is illustrated for one case study.

2.1. BQP with $M = n/5$

The first set of test problems is based on a CBQP with the parameter M given by $M = n/5$, $n = \{50, 75, 100\}$, and for two instances with $Q = \{U(5, 100), N(0, 1)\}$. The results obtained with the MILP formulations are presented first, and next the performance of the MIQP and general solvers is analyzed.

In Table 2, the values of the objective function of the continuous relaxations of the MILP formulations for $n = 50, M = 10$ are presented. From this table is clear that formulations (F0) and (F1) are loose formulations, while formulation (F4) leads to the tightest formulation. For the first set of problems with $n = 50, M = 10$, CPLEX, GUROBI and XPRESS were able to solve the problem to global optimality using any of the formulations. However, it is clear that the

Table 2: Value of the objective function for the continuous linear relaxation of each MILP formulation, for $n = 50, M = 10$.The optimal solutions are $W = 3760.72$ and $W = -535.72$ for the cases generated with the matrices Q with U(5,100) and N(0,1), respectively.

Formulation	U(5,100)	N(0,1)
F0	327.55	-
F1	327.55	-1,362.72
F2	2,694.76	-908.49
F3	2,694.76	-908.49
F4	**3,409.89**	**-630.15**
F5	2,328.32	-1,095.62
F6	2,694.76	-908.49

U(5,100) - Uniform distribution between 5 and 100 used to generate the matrix Q. N(0,1) - Normal distribution with mean 0 and standard deviation 1 used to generate the matrix Q. Bold means best results.

formulation used has a strong influence on the CPU time required. The formulations (F3) and (F6) feature the lower CPU times for U(5,100), independently of the solver (see Table 3), while for N(0,1) these formulations are still efficient but with CPLEX the formulation (F2) has also a lower CPU time. The most inefficient formulations were (F1) and (F4) independently of the solvers and of the instances. The poor performance of (F0) and (F1) is explained by the poor initial lower bound provided by the continuous relaxation, while the good initial lower bound of (F3), (F5) and (F6) helped to achieve the lower CPU times of these formulations. The tighter formulation, (F4), as expected requires the solution of the smallest number of nodes, less than 84, to prove optimality, but it takes two orders of magnitude more time than the fastest formulations.

For the problem with 75 equations, the formulations (F3), (F5) and (F6) solved the problem within the time limit for specific instances and solvers (see Table 4). For $n = 100$, all MILP formulations with both solvers were not able to close the gap between the bounds within the time limit of one hour.

Table 3: Results obtained for each MILP formulation for the optimal solutions with $n = 50, M = 10$. The optimal solutions are $W = 3760.72$ for U(5,100) and $W = -535.72$ for N(0,1).

	U(5,100)					
	CPLEX		GUROBI		XPRESS	
Formulation	# Nodes	CPU (s)	# Nodes	CPU (s)	# Nodes	CPU (s)
F0	2,645,063	944	2,140,922	279	3,929,271	567
F1	1,653,687	1,785	1,592,331	584	1,202,453	397
F2	4,302	17	3,597	36	4,249	27
F3	**4,091**	**12**	4,040	29	4,859	19
F4	59	735	84	509	47	844
F5	13,540	23	15,873	27	30,715	32
F6	**4,091**	**12**	3,137	15	5,097	20
	N(0,1)					
F0	-	-	-	-	-	-
F1	2,570	33	18,956	48	42,499	66
F2	**1,406**	**7**	1,938	14	3,129	18
F3	1,842	8	1,664	9	3,869	16
F4	60	870	42	297	77	874
F5	17,066	22	**1,661**	**7**	6,151	15
F6	**1,842**	**7**	1,190	8	3,869	16

U(5,100) - Uniform distribution between 5 and 100 used to generate the matrix Q. N(0,1) - Normal distribution with mean 0 and standard deviation 1 used to generate the matrix Q. Versions used: CPLEX 12.4, GUROBI 4.6.1, XPRESS 22.01.15. Bold means best result.

The same problem was solved with the MIQP solvers from CPLEX, GUROBI, XPRESS, and SCIP and with a set of MINLP solvers that can handle the nonlinearities and integer variables. For the set of instances with $n = 50$ and U(5,100), CPLEX, GUROBI and XPRESS can prove global

Table 4: Results obtained for each MILP formulation for $n = 75, M = 15$. The optimal solutions are $W = 8,726.10$ and $W = -1020.70$ for U(5,100) and N(0,1), respectively.

	U(5,100)							
	CPLEX				GUROBI			
Formulation	# Nodes	W	Gap(%)	CPU (s)	# Nodes	W	Gap(%)	CPU (s)
F0	850,817	9246.17	54.04	3,636	5,526,141	8,800.39	46.2	3,609
F1	445,892	9612.77	60.47	3,600	2,455,688	8,726.10	49.7	3,604
F2	118,933	8,758.26	5.4	3,601	87,084	8,726.10	6.1	3,600
F3	**189,201**	**8,726.10**	**0.0**	**3,002**	173,591	8,726.10	0.0	3,115
F4	10	11,186.35	32.4	3,601	13	8,961.34	15.7	3,601
F5	350,637	8,726.10	0.0	3,122	1,034,674	8,774.86	7.8	3,601
F6	64,820	8,726.10	7.1	3,600	173,591	8,726.10	0.0	3,132
	N(0,1)							
F0	-	-	-	-	-	-	-	-
F1	64,533	-987.83	31.1	3,603	342,392	-998.80	58.9	3,601
F2	145,712	-1,020.70	8.1	3,601	96,395	-997.67	19.1	3,600
F3	172,803	-1,020.70	0.0	2,383	202,286	-1,020.70	0.0	3,279
F4	9	-134.48	952.2	3,601	14	-875.64	54.9	3,601
F5	506,064	-1,020.70	0.0	3,582	815,246	-1,020.70	25.7	3,601
F6	**172,803**	**-1,020.70**	**0.0**	**2,362**	202,286	-1,020.70	0.0	3,244

U(5,100) - Uniform distribution between 5 and 100 used to generate the matrix Q. N(0,1) - Normal distribution with mean 0 and standard deviation 1 used to generate the matrix Q. Versions used: CPLEX 12.4, GUROBI 4.6.1. Bold means best result.

optimality of the solutions obtained within the maximum time set of one hour for the instances with U(5,100), while BARON was able to find the global minimum, but it is not able to prove global optimality. DICOPT, GloMIQO, and SCIP returned only suboptimal solutions (see Table 6). For the instances where the coefficients q_{ij} may be negative, CPLEX and GUROBI are still efficient solvers, 1 and 6 seconds respectively, to solve the problems with $n = 50$ and N(0,1). On the other hand, BARON and SCIP can solve this problem to optimality, showing the impact of the values of the coefficients on these solvers.

Increasing the size of the model, for $n = 75$ only CPLEX was able to solve the problem using 250 and 188 seconds for the instances with U(5,100) and N(0,1), respectively. For $n = 100$ all solvers failed to prove optimality of the solutions obtained independently of the matrix Q (see Tables 7 and 8). In terms of the general purpose solvers, for the three cases of n the output log of GloMIQO indicates that the problem is linear and transfers the problem to CPLEX. After one hour, it does not return a solution from CPLEX, and the final solution returned is the integer solution from the relaxed NonLinear Programming (NLP) problem. Similarly, for the three cases of n DICOPT stops after finding that the solution of the initial relaxed NLP is integral, and returns a local solution.

Table 5: Results obtained for each MILP formulation for $n = 100, M = 20$. The best known solution for U(5,100) is $W = 16,134.42$ and the optimal solution for N(0,1) is $W = -1,642.72$.

	U(5,100)							
	CPLEX				GUROBI			
Formulation	# Nodes	W	Gap(%)	CPU (s)	# Nodes	W	Gap(%)	CPU (s)
F0	961,870	17,411.09	72.8	3,869	1,712,439	16,188.65	69.9	3,604
F1	344,628	17,634.95	74.3	3,601	827,596	16,361.04	72.9	3,604
F2	34,062	16,188.06	20.7	3,603	15,173	16,185.19	22.7	3,600
F3	51,782	16,176.60	19.2	3,602	27,560	16,222.25	21.0	3,600
F4	2	20,728.54	34.9	3,602	0	18,548.58	27.2	3,603
F5	128,323	16,134.42	19.7	3,604	140,220	16,227.30	22.6	3,600
F6	24,637	16,227.30	21.4	3,600	27,553	16,222.25	21.0	3,600
	N(0,1)							
F0	-	-	-	-	-	-	-	-
F1	7,842	-1,512.21	78.7	3,602	53,424	-1,380.25	200.4	3,600
F2	39,327	-1,613.93	72.5	3,602	15,622	-1,630.92	77.2	3,600
F3	59,581	-1,642.72	65.1	3,604	30,119	-1,634.83	69.3	3,600
F4	0	-251.39	-	5,458	0	-945.74	169.5	3,604
F5	138,261	-1,630.92	75.0	3,604	133,167	-1,642.72	78.6	3,600
F6	59,563	-1,642.72	65.1	3,603	30,205	-1,634.83	69.3	3,600

U(5,100) - Uniform distribution between 5 and 100 used to generate the matrix Q. N(0,1) - Normal distribution with mean 0 and standard deviation 1 used to generate the matrix Q. Versions used: CPLEX 12.4, GUROBI 4.6.1. Bold means best result.

Figure 1 shows the value of the objective function of the initial relaxations for the different MILP formulations, for the MIQP solvers, and BARON for the instances with $n = 50$ and U(5,100). The performance of the MILP formulations presented in Table 3 is explained by the quality of the initial relaxation at the root node and by the size of the respective formulation. The initial values of the relaxations of XPRESS and BARON suggest that these solvers are using reformulations similar to the ones used in formulations (F0) and (F1); while GUROBI has a loose formulation. The superior performance of CPLEX in the solution of the MIQP arises from the quality of the lower bound at the root node, plus the fact that during the branch and bound it deals with a much smaller problem than the MILP formulations.

In addition, two alternative approaches were implemented based on MILP formulations for $n = 75$ and U(5,100):

1. A sequence of problems is solved, where the first solves the MILP formulation (F3) with the objective of finding the best solution. This is accomplished by invoking the heuristic available within CPLEX to polish MILP solutions (Rothberg, 2007). The solution of this step is then used as the starting point in step three. In the second step the continuous relaxation of formulation (F3) is solved, and in the third step the formulation (F3) is complemented with

Table 6: Results obtained with different solvers, solving the problem using the MIQP formulation for $n = 50, M = 10$. The optimal solutions are $W = 3760.72$ for U(5,100) and $W = -535.72$ for N(0,1).

	U(5,100)				N(0,1)			
Solver	W	Gap (%)	# Nodes	CPU (s)	W	Gap (%)	# Nodes	CPU (s)
BARON	3,760.72	20.3	362,520	3,601	-535.72	0.0	9,436	525
CPLEX	**3,760.72**	**0.0**	**153,111**	**4**	**-535.72**	**0.0**	**19,671**	**1**
DICOPT	3,843.61	-	-	0.1	-503.53	-	-	0.1
GloMIQO	3,843.61	-	-	3,611	-535.72	-	-	85
GUROBI	3,760.72	0.0	1,762,117	68	-535.72	0.0	94,335	6
SCIP	3,843.61	32.1	12,346,262	3,741	-535.72	0.0	655,552	242
XPRESS	3,760.72	0.0	2,314,773	562	-504.64	51.6	981,014	3,599

Table 7: Results obtained with different solvers, solving the problem using the MIQP formulation for $n = 75, M = 15$. The optimal solutions are $W = 8,726.10$ and $W = -1020.70$ for U(5,100) and N(0,1), respectively.

	U(5,100)				N(0,1)			
Solver	W	Gap (%)	# Nodes	CPU (s)	W	Gap (%)	# Nodes	CPU (s)
BARON	8,726.10	55.6	74,164	3,600	-1,020.70	59.3	9,638	3,600
CPLEX	**8,726.10**	**0.0**	**5,838,445**	**250**	**-1,020.70**	**0.0**	**5,077,269**	**188**
DICOPT	9,142.59	-	-	0.1	-884.35	-	-	0.1
GloMIQO	9,142.59	-	-	3,600	-884.35	-	-	3,600
GUROBI	8,726.10	39.8	19,356,919	3,722	-1,020.70	30.4	20,157,785	3,699
SCIP	9,142.59	70.4	10,171,737	3,737	-886.62	93.0	3,451,412	3,650
XPRESS	9,702.62	53.3	2,538,562	4,341	-866.11	160.8	496,308	3,600

Table 8: Results obtained with different solvers, solving the problem using the MIQP formulation for $n = 100, M = 20$. The best known solution for U(5,100) is $W = 16,134.42$ and the optimal solution for N(0,1) is $W = -1,642.72$.

	U(5,100)				N(0,1)			
Solver	W	Gap (%)	# Nodes	CPU (s)	W	Gap (%)	# Nodes	CPU (s)
BARON	16,134.42	70.5	19,007	3,600	-1,642.72	118.5	2,492	3,601
CPLEX	16,134.42	1.9	53,324,748	3,601	**-1,642.72**	**0.0**	**42,062,241**	**3,593**
DICOPT	16,622.48	-	-	0.1	-1389.26	-	-	0.1
GloMIQO	16,622.48	-	-	3,600	-1389.26	-	-	3,600
GUROBI	16,222.25	81.3	10,828,371	3,640	-1,642.72	95.9	9,963,334	3,624
SCIP	16,622.48	86.9	5,990,440	3,698	-1,389.26	173.6	1,611,804	3,625
XPRESS	18,851.98	74.7	917,088	3,600	-1,257.84	283.7	219,324	3,600

U(5,100) - Uniform distribution between 5 and 100 used to generate the matrix Q. N(0,1) - Normal distribution with mean 0 and standard deviation 1 used to generate the matrix Q. Versions used: BARON 10.2.0, CPLEX 12.4, DICOPT GAMS 23.8.1, GloMIQO 1.0.0, GUROBI 4.6.1., SCIP 2.1.1, XPRESS 22.01.15. Bold means best result.

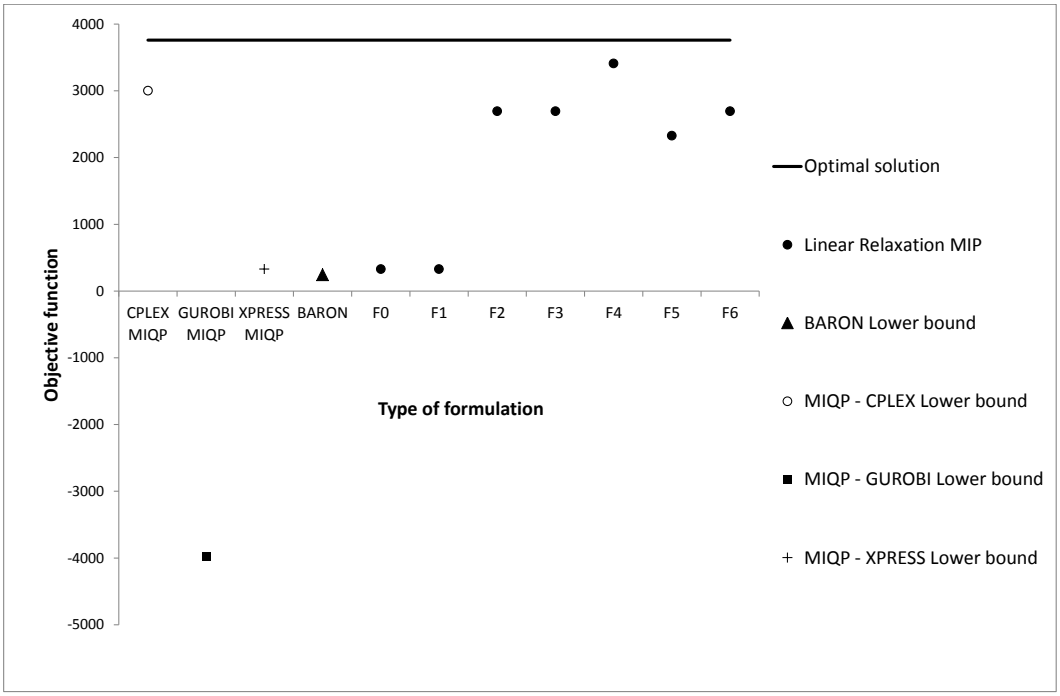

Figure 1: Comparison between the different values of the objective function of the continuous relaxations of the different formulations and the values calculated by the solvers used. $n = 50$, $M = 10$, U(5,100).

the triangular inequalities that were violated in the second step. With this approach, CPLEX can solve this problem with $n = 75$ in approximately 2000s, which means that the CPU time was reduced in 1000 seconds(a reduction of 33%) when compared with the results shown in Table 4. Notwithstanding, the performance of this strategy is still superior to using BARON, DICOPT, GloMIQO, GUROBI, SCIP and XPRESS (see Table 7), but it still requires more time than the MIQP solution with CPLEX, which took 250 seconds.

2. The second approach relies on using CPLEX with callbacks during the branch and bound algorithm to generate cuts that cut off solutions of the continuous relaxation. This approach is implemented using the BCH tool from GAMS. In this implementation the formulation (F3) is used, and in the generation cut routine the triangular inequalities that are violated are added to the continuous relaxation solved within the branch and cut algorithm within the MILP solver. With this approach the cuts generated during the root node iterations reduce the initial lower bound from 5866.5374 (gap of 33.87%) to 7538.7465 (gap of 15.01%) in 36.30 seconds (still at the root node), while using only the formulation (F3) takes approximately

339 seconds to reach the lower bound of 7543.3882. However, this approach did not present a solid performance, since it was not able to close the gap within one hour. An important remark is that the current implementation of the BCH tool in GAMS does not allow to use multiple threads, which is an important drawback, as it will be shown later.

2.2. BQP with $M = n/1.25$

The MILP formulations were also tested with $M = n/1.25$. The results show that the best MILP formulations for $M = n/5$ do not have the same performance for $M = n/1.25$. The formulations (F0), (F1) and (F2) took 944s, 1785s and 17s for $n = 50$, U(5,100) and $M = n/5$, while for $n = 50$, U(5,100), and $M = n/1.25$ took 28s, 47s and 3s, respectively. With an opposite trend, formulation (F3) took 3s in the first case and 902s in the second case (see Table 9). On the other hand, for $M = n/1.25$, formulations (F1) and (F2) have the best performance with CPLEX and GUROBI (see Tables 9, 10, 11). Note that these trends are independent of the matrix Q, since for N(0,1) the formulation (F2) for $M = n/1.25$ features also considerably lower CPU times than (F3), precisely 2s with (F2) against 821s with (F3) (see Table 9). This clearly shows that the best formulations for $M = n/5$ are not the best choice for $M = n/1.25$. This comparison is highlighted in Table 12 for $n = 50$. For $n = 50$ and U(50,100) GUROBI, performed better than CPLEX with (F2), but GUROBI with (F3), (F5) and (F6) took a large number of nodes to prove optimality. Similarly results were obtained with N(0,1). Another remark is that GUROBI was able to solve the three problems with $n = \{50, 75, 100\}$ within the time limit of one hour, independently of the matrix Q. These results suggest that the relative quality of the relaxation of each MILP formulation depends on the value of M. Figure 2 shows the initial lower bounds predicted by each formulation, where it is clear that formulation (F2) has in this case a stronger lower bound than Formulations (F3), (F5) and (F6). The explanation resides on the fact that for $M = n/5$ the solution of the continuous relaxation of (F2) has several values of x_i equal to zero or with $x_i + x_j < 1$, while for $M = n/1.25$, the solution of the relaxation has more variables x_i with the value of 1 and with a value greater than in the case $M = n/5$. The first situation is explained by the fact that this is a minimization problem and that $M \ll n$, while in the second situation $M < n$, and therefore in the latter case, some x_i must be 1 to satisfy the cardinality constraint. This situation makes the inequality $z_{ij} \geq x_i + x_j - 1$ redundant in the first situation, and an important inequality in the second case. Therefore, for $M = n/5$ its omission reduces the size of the problem, increasing the efficiency of the formulation, while for $M = n/1.25$ is a strong valid inequality, contributing for the efficiency of the formulation (F2).

Table 9: Results obtained for each MILP formulation for the optimal solutions with $n = 50, M = 40$. The optimal solutions are $W = 78,951.35$ for U(5,100) and $W = -899.16$ for N(0,1).

	U(5,100)			
	CPLEX		GUROBI	
Formulation	# Nodes	CPU (s)	# Nodes	CPU (s)
F0	53,817	28	14,451	6
F1	33,438	47	14,576	12
F2	79	3	**37**	**2**
F3	249,517	902	341,355	1,891
F4	3	382	0	46
F5	228,350	1,002	2,468,245	3,411
F6	249,517	899	243,187	1,486
	N(0,1)			
F0	-	-	-	-
F1	36	8	365	3
F2	**23**	**2**	39	4
F3	271,390	821	240,099	1,344
F4	0	216	3	91
F5	680,126	1,848	2,578,025	3,470
F6	271,390	821	288,585	1,469

U(5,100) - Uniform distribution between 5 and 100 used to generate the matrix Q. N(0,1) - Normal distribution with mean 0 and standard deviation 1 used to generate the matrix Q. Bold means best result.

Table 10: Results obtained for each MILP formulation for $n = 75, M = 60$. The optimal solutions are $W = 181,773.25$ and $W = -1,691.89$ for U(5,100) and N(0,1), respectively.

	U(5,100)							
	CPLEX				GUROBI			
Formulation	# Nodes	W	Gap(%)	CPU (s)	# Nodes	W	Gap(%)	CPU (s)
F0	1,595,505	181,817.45	1.4	3,945	3,218,639	181,773.25	0.0	2,673
F1	787,619	181,821.04	1.0	3,895	2,420,466	181,773.25	0.7	3,603
F2	465	181,773.25	0.0	28	**287**	**181,773.25**	**0.0**	**27**
F3	128,414	181,773.25	4.9	3,603	41,161	181,797.12	5.9	3,600
F4	0	187,388.48	NA	3,602	1	181,773.25	0.1	3,602
F5	127,538	181,773.25	5.0	3,604	41,355	181,773.25	6.6	3,600
F6	127,917	181,773.25	4.9	3,603	41,148	181,797.12	5.9	3,600
	N(0,1)							
F0	-	-	-	-	-	-	-	-
F1	448	-1691.89	0.0	55	7,363	-1691.89	0.0	161
F2	**250**	**-1691.89**	**0.0**	**15**	683	-1691.89	0.0	64
F3	162,531	-1691.89	180.8	3,604	42,374	-1691.89	217.3	3,600
F4	0	393.53	0.0	3,601	3	-1691.89	0.0	1,575
F5	190,046	-1691.89	194.4	3,604	41,994	-1677.34	249.7	3,600
F6	162,903	-1691.89	180.7	3,603	42,369	-1691.89	217.3	3,600

†- Time limit reached during the solution of the continuous relaxation at the root node. U(5,100) - Uniform distribution between 5 and 100 used to generate the matrix Q. N(0,1) - Normal distribution with mean 0 and standard deviation 1 used to generate the matrix Q. Bold means best result.

Table 11: Results obtained for each MILP formulation for $n = 100, M = 80$. The optimal solutions are $W = 325,188.73$ and $W = -3,122.04$ for U(5,100) and N(0,1), respectively.

	U(5,100)							
	CPLEX				GUROBI			
Formulation	# Nodes	W	Gap(%)	CPU (s)	# Nodes	W	Gap(%)	CPU (s)
F0	512,509	325,707.52	3.2	3,968	1,518,397	325,188.73	2.4	3,606
F1	177,813	325,337.81	2.8	3,747	566,469	325,188.73	2.7	3,602
F2	20,768	325,188.73	0.1	3,601	**15,332**	**325,188.73**	**0.0**	**3,557**
F3	29,488	325,274.69	7.5	3,604	10,981	325,279.27	8.2	3,600
F4	0	334,176.31	0.0	19,198	0	325,644.93	0.3	3,605
F5	36,390	325,299.08	7.6	3,601	41,167	325,188.73	8.0	3,600
F6	29,509	325,274.69	7.5	3,604	10,981	325,279.27	8.2	3,600
	N(0,1)							
F0	-	-	-	-	-	-	-	-
F1	11,851	-3,122.04	4.5	3,601	54,295	-3,122.04	22.0	3,600
F2	**13,799**	**-3,122.04**	**0.0**	**1,887**	8,840	-3,122.04	0.0	3,111
F3	34,295	-3,099.89	273.5	3,603	11,193	-3,059.44	303.5	3,600
F4	0	154.83	0.0	10,197	0	-3,072.91	9.8	3,604
F5	39,866	-3,108	284.0	3,604	41,230	-3,119.06	299.2	3,600
F6	34,255	-3,099.89	273.5	3,603	11,438	-3,059.44	302.7	3,600

†- Time elapsed during the solution of the continuous relaxation at the root node. The search over the tree did not initialize. U(5,100) - Uniform distribution between 5 and 100 used to generate the matrix Q. N(0,1) - Normal distribution with mean 0 and standard deviation 1 used to generate the matrix Q. Bold means best result.

Table 12: Results obtained with CPLEX for each MILP formulation for $n = 50$ and U(5,100) for different values of M.

	$M = 10$		$M = 40$	
Formulation	# Nodes	CPU (s)	# Nodes	CPU (s)
F0	2,645,063	944	53,817	28
F1	1,653,687	1,785	33,438	47
F2	4,302	17	**79**	**3**
F3	**4,091**	**12**	249,517	902
F4	59	735	3	382
F5	13,540	23	228,350	1,002
F6	4,091	12	249,517	899

Bold means best result.

Table 13 shows the results obtained for the MIQP formulations for different values of n and $M = n/1.25$, solved with CPLEX and GUROBI. It is clear that for this type of MIQP problems, CPLEX has a better performance than GUROBI. For example, CPLEX needs only 18 seconds to solve the problem with $n = 100$ and U(5,100), while GUROBI after one hour still has a 2.6% gap. Another remark is that with GUROBI, better results are obtained using the MILP formulation than the MIQP solver.

Table 13: Results obtained with CPLEX and GUROBI, solving the problem using the MIQP formulation for $M = n/1.25$.

		U(5,100)			N(0,1)		
Solver	n	W	Gap (%)	CPU (s)	W	Gap (%)	CPU (s)
CPLEX	50	78951.35	0.0	1	-899.16	0.0	0
GUROBI		78951.35	0.0	9	-899.16	0.0	1
CPLEX	75	181773.25	0.0	4	-1691.89	0.0	1
GUROBI		181773.24	0.0	2,252	-1691.89	0.0	48
CPLEX	100	325188.73	0.0	18	-3122.04	0.0	10
GUROBI		325188.73	2.6	3,669	-3122.04	11.8	3,636
CPLEX	200	1316710.32	0.1	3,601	-9883.11	4.4	3,601
CPLEX	300	2971613.47	0.2	3,601	-19939.46	7.7	3,601
CPLEX	400	5296802.82	0.2	3,601	-28436.53	11.0	3,601

U(5,100) - Uniform distribution between 5 and 100 used to generate the matrix Q. N(0,1) - Normal distribution with mean 0 and standard deviation 1 used to generate the matrix Q. Versions used: CPLEX 12.4, GUROBI 4.6.1.

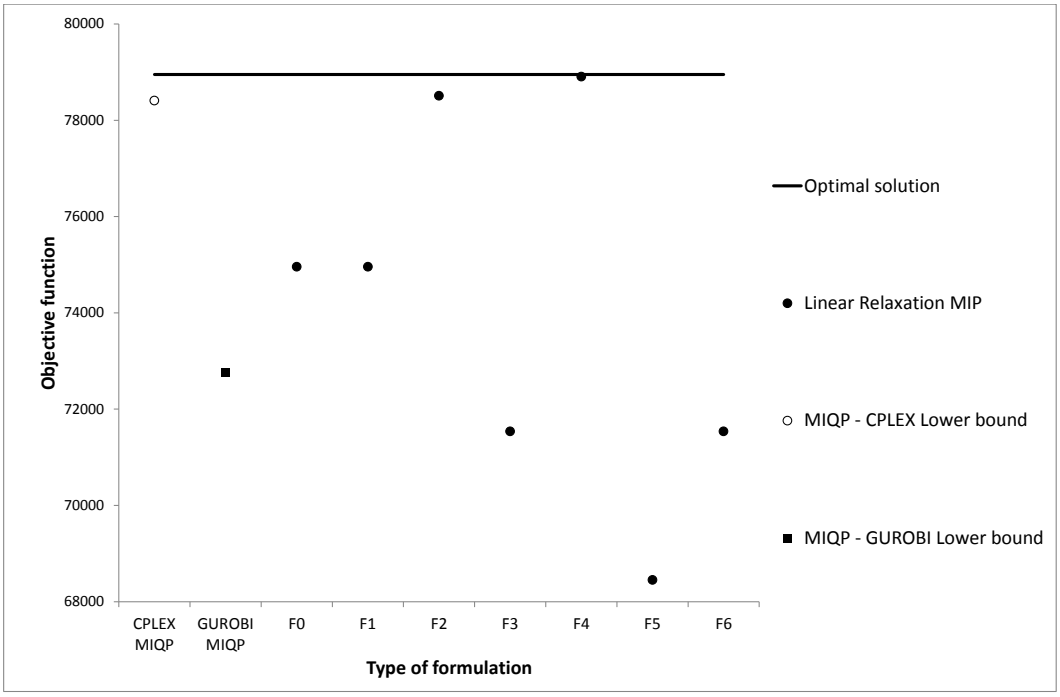

Figure 2: Comparison between the different values of the objective function of the continuous relaxations of the different formulations and the values calculated by the solvers used. $n = 50$, $M = 40$, U(5,100).

2.3. Analysis of the impact of Q for $n = 50$

The impact of the matrix Q in the performance of the solvers is evaluated for two cases with $n = 50$ and $M = \{n/5, n/1.25\}$ and for 20 additional instances with Q generated randomly using uniform and normal distributions within different domains. From the results presented in the two previous sections, CPLEX and GUROBI seem to be the most appropriate solvers to approach the problem studied. Therefore, they are used to investigate the impact of Q on the relative performance of the MILP and MIQP formulations. Figures 3 and 4 show the CPU time elapsed for the 22 instances obtained with MILP and MIQP formulations with CPLEX, while Figures 5 and 6 present the same type of results obtained with GUROBI.

The results obtained for the 22 instances confirm the analysis made on the previous sections regarding the relative quality of the MILP and MIQP formulations. First, analyzing the performance of the MILP formulations for $M = n/5$ is clear that (F1) does not have a good performance, and formulations (F3), (F5) and (F6) are the most efficient formulations (see Figures 3 and 5). As discussed before, the formulation (F2) leads to lower CPU times for most of the cases with $M = n/1.25$. These results are valid for CPLEX as well as for GUROBI.

The performance of the MIQP formulations with CPLEX is also aligned with the results shown previously, more specifically CPLEX features better CPU times for all the instances using the MIQP formulation. While with GUROBI, the results show that for 21 cases out of 22 there is at least a MILP formulation that is better than the MIQP formulation, namely only for instance 1 the MIQP solver has a lower CPU time (see Figures 5 and 6).

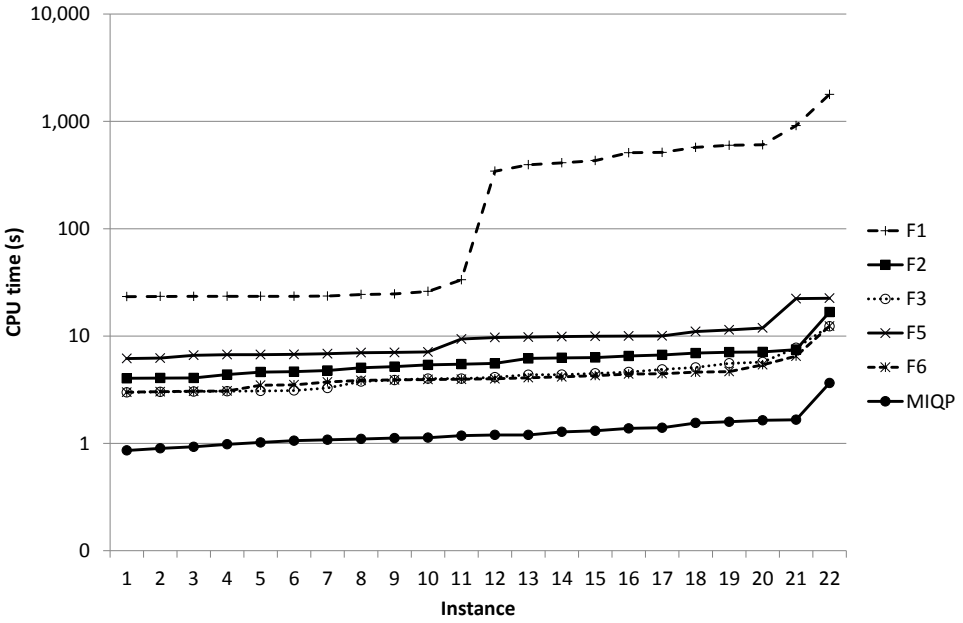

Figure 3: Results obtained with CPLEX using MILP and MIQP formulations, for $n = 50$ and $M = n/5$ for 22 instances with different matrices Q.

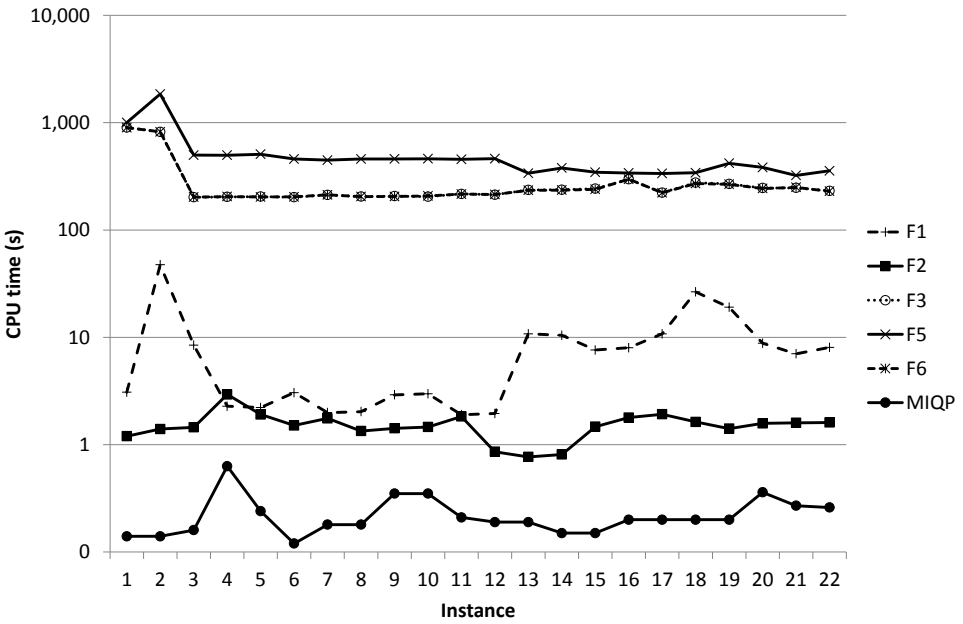

Figure 4: Results obtained with CPLEX using MILP and MIQP formulations, for $n = 50$ and $M = n/1.25$ for 22 instances with different matrices Q.

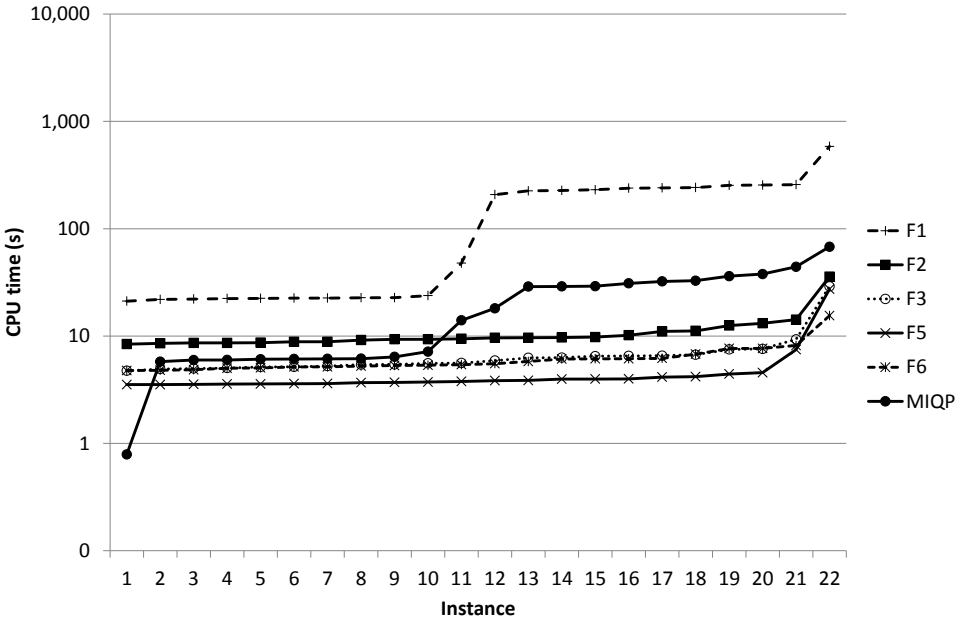

Figure 5: Results obtained with GUROBI using MILP and MIQP formulations, for $n = 50$ and $M = n/5$ for 22 instances with different matrices Q.

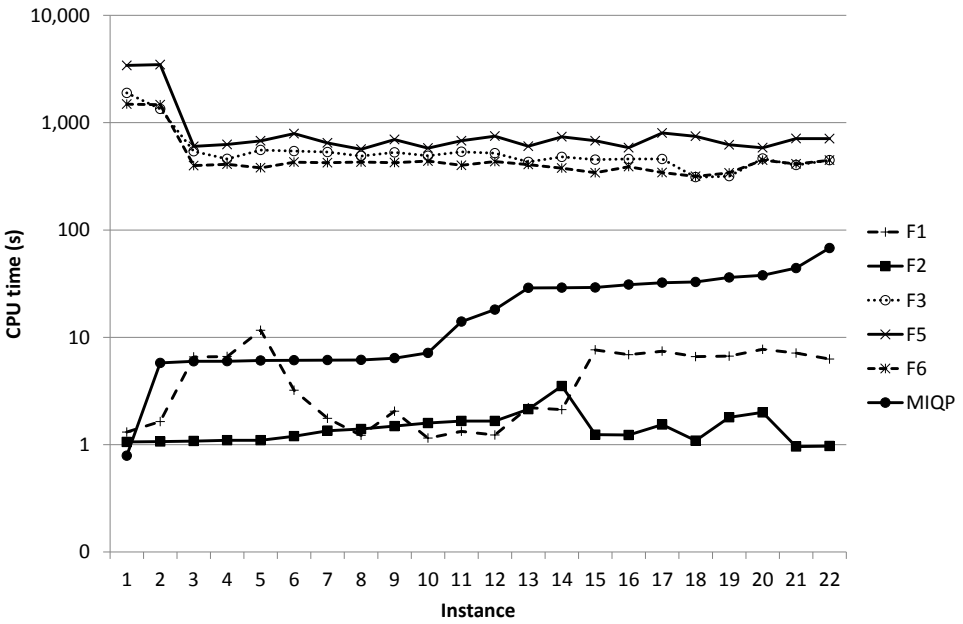

Figure 6: Results obtained with GUROBI using MILP and MIQP formulations, for $n = 50$ and $M = n/1.25$ for 22 instances with different matrices Q.

2.4. Contributions of algorithms and hardware developments

In this section, the objectives are to assess the contribution of software developments within one MILP solver, and evaluate the impact of using more than one CPU thread for the MILP and MIQP approaches. Here, the only solver used is CPLEX and two different versions are considered 7.1 and 12.4. The first version does not have an MIQP solver available, neither has the option to use more than one thread. In terms of hardware, the focus is on the number of threads used within the branch and bound search. The current CPLEX version is used with one and eight threads, and the results are then compared with the previous versions.

The formulations (F0) and (F3) with U(5,100) were solved using CPLEX 12.4 with one thread and 8 threads, and with CPLEX 7.1. The results show that using the best formulation, (F3), with CPLEX 7.1 remains a good option when compared with using the last version with a weak formulation (see Tables 14 and 15). This demonstrates that strong formulations are worth and still may overcome algorithmic advances implemented in CPLEX 12.4. In addition, for $n = 50$, U(5,100) and $M = 10$ and with the formulation (F0), CPLEX 7.1 has a better performance than CPLEX 12.4 with only one thread, which is surprising. The performance of both the MILP and MIQP solvers suggest that for this problem, the number of threads used has a significant impact over the CPU time required to close the gap.

Table 14: Comparison between CPLEX 7.1 (2001) and CPLEX 12.4 (2012) when applied to two MILP formulations with $n = 50, M = 10$ and U(5,100).

	CPLEX 7.1		CPLEX 12.4 - 1 threads		CPLEX 12.4 - 8 threads	
Formulation	Gap(%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)
F0	0	1437	0	2249	0	944
F3	0	88	0	56	0	12
MIQP	-	-	0	19	0	4

U(5,100) - Uniform distribution between 5 and 100 used to generate the matrix Q.

Table 15: Comparison between CPLEX 7.1 (2001) and CPLEX 12.4 (2012) when applied to two MILP formultions with $n = 50, M = 40$ and U(5,100).

	CPLEX 7.1		CPLEX 12.4 - 1 thread		CPLEX 12.4 - 8 threads	
Formulation	Gap(%)	CPU (s)	Gap (%)	CPU (s)	Gap (%)	CPU (s)
F2	0	8	0	3	0	3
F5	1.38	3600	1.09	3600	0	1,002
MIQP	-	-	0	1	0	1

U(5,100) - Uniform distribution between 5 and 100 used to generate the matrix Q.

3. Conclusions

In this paper we have presented the computational results obtained for the solution of a nonconvex boolean quadratic programming problem. The identification of a computationally efficient MILP formulation and the comparison with the other commercial approaches to solve this problem has been the main subject of this work. The computational results show that the MILP formulations are not efficient when compared with the MIQP solver available from CPLEX, but they are clearly more efficient than the MIQP algorithms implemented in the other solvers that were tested. In the specific case of CPLEX, a pre-processing stage has been implemented that is able to convexify the objective function and generate a good lower bound that is better than most of the lower bounds of the MILP formulations studied. The quality of the lower bound, plus the fact that no additional inequalities are added to the formulation, makes the CPLEX MIQP approach very fast when compared with the other MIQP solvers and MILP formulations. The superiority of the MILP formulations over the other solvers is explained by the weak lower bound determined initially by these solvers with the MIQP solver. The remaining solvers used, DICOPT, BARON, GloMIQO and SCIP, do not seem to be appropriate tools to handle these type of problems. The main reason seems to be the lack of specific convexification procedures for this type of nonlinearities. The matrix Q has some impact over the computational results of the approaches tested, but it does not change the relative performance of the MILP formulations or the performance of the MIQP or MINLP solvers. The comparison of the results obtained with the last version of CPLEX with 1 and 8 threads, and with version 7.1 shows the importance of a good formulation and the positive impact that the number of threads used may have on the required CPU time.

Acknowledgments

The authors would like to thank the Center for Advanced Process Decision-making at Carnegie Mellon University for their financial support. 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 Foundation for Science and Technology of Portugal.

References

Achterberg, T. 2009. SCIP solving constraint integer programs. *Math. Program.* **1** 1–41.

Barahona, F., M. Junger, G. Reinelt. 1989. Experiments in quadratic 0-1 programming. *Math. Program.* **44** 127 – 137.

Billionnet, A. 2005. Different formulations for solving the heaviest k-subgraph problem. *INFOR* **43** 171 – 186.

Billionnet, A., M. C. Costa, A. Sutter. 1992. An efficient algorithm for a task allocation problem. *J. ACM* **39** 502–518.

Billionnet, A., S. Elloumi. 2007. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Math. Program.* **109** 55 – 68.

Billionnet, S., A.and Elloumi, M.C. Plateau. 2009. Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: The QCR method. *Discrete Appl. Math.* **157** 1185–1197.

Bixby, R., E. Rothberg. 2007. Progress in computational mixed integer programming- a look back from the other side of the tipping point. *Ann. Oper. Res.* **49** 37–41.

Boros, E., P. L. Hammer. 1993. Cut-polytopes, boolean quadric polytopes and nonnegative quadratic pseudo-boolean functions. *Math. Oper. Res.* **18** 245 – 253.

Bruglieri, M., M. Ehrgott, H.W. Hamacher, F. Maffioli. 2006. An annotated bibliography of combinatorial optimization problems with fixed cardinality constraints. *Discrete Appl. Math.* **154** 1344 – 1357.

Burer, S., A.N. Letchford. 2009. On nonconvex quadratic programming with box constraints. *SIAM J. Optimiz.* **20** 1073 – 1089.

Faye, A., Q. Trinh. 2003. Polyhedral results for a constrained quadratic 0-1 problem. Tech. Rep. CEDRIC-03-511, CEDRIC laboratory, CNAM-Paris, France.

Faye, A., Q. A. Trinh. 2005. A polyhedral approach for a constrained quadratic 0-1 problem. *Discrete Appl. Math.* **149** 87 – 100.

Glover, F., E. Woolsey. 1974. Converting 0-1 polynomial programming problem to a 0-1 linear program. *Oper. Res.* **2** 180–182.

Gueye, S., P. Michelon. 2005. "Miniaturized" linearizations for quadratic 0/1 problems. *Ann. Oper. Res.* **140** 235 – 261.

Gueye, S., P. Michelon. 2009. A linearization framework for unconstrained quadratic (0-1) problems. *Discrete Appl. Math.* **157** 1255 – 1266.

Hansen, P., C. Meyer. 2009. Improved compact linearizations for the unconstrained quadratic 0-1 minimization problem. *Discrete Applied Mathematics* **157** 1267–1290.

Johnson, E. L., A. Mehrotra, G. L. Nemhauser. 1993. Min-cut clustering. *Math. Program.* **62** 133 – 151.

Liberti, L. 2007. Compact linearization for binary quadratic problems. *4OR* **5** 231–245.

Lima, R. M., I. E. Grossmann. 2011. *Chemical Engineering Greetings to Prof. Sauro Pierucci*, chap. Computational advances in solving Mixed Integer Linear Programming problems. AIDIC, 151–160.

Loiola, E. M., N. M. M. Abreu, P.O. Boaventura-Netto, P. Hahn, T. Querido. 2007. A survey for the quadratic assignment problem. *Eur. J. Oper. Res.* **176** 657 – 690.

Mehrotra, A. 1997. Cardinality constrained Boolean quadratic polytope. *Discrete Appl. Math.* **79** 137 – 154.

Misener, R., A. C. Floudas. 2012. GloMIQO: Global mixed-integer quadratic optimizer. *J. Global Optim* DOI 10.1007/s10898-012-9874-7.

Padberg, M. 1989. The boolean quadric polytope - some characteristics, facets and relatives. *Math. Program.* **45** 139 – 172.

Pardalos, P. M., G. P. Rodgers. 1990. Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing* **45** 131 – 144.

Phillips, A.T., J.B. Rosen. 1994. A quadratic assignment formulation of the molecular conformation problem. *J. Global Optim* **4** 229–241.

Raman, R., I. E. Grossmann. 1991. Relation between MILP modeling and logical inference for chemical process synthesis. *Comput. Chem. Eng.* **15** 73 – 84.

Rothberg, E. 2007. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS J. Comput.* **19** 534–541.

Sherali, H. D., Y. H. Lee, W. P. Adams. 1995. A simultaneous lifting strategy for identifying new classes of facets for the boolean quadric polytope. *Oper. Res. Lett.* **17** 19 – 26.

Tawarmalani, M., N.V. Sahinidis. 2005. A polyhedral branch-and-cut approach to global optimization. *Math. Program.* **103** 225–249.