Math. Program., Ser. A (2013) 141:435–452
DOI 10.1007/s10107-012-0534-y

FULL LENGTH PAPER

Semidefinite relaxations for non-convex quadratic mixed-integer programming

Christoph Buchheim · Angelika Wiegele

Received: 14 September 2010 / Accepted: 6 March 2012 / Published online: 30 March 2012

Abstract We present semidefinite relaxations for unconstrained non-convex quadratic mixed-integer optimization problems. These relaxations yield tight bounds and are computationally easy to solve for medium-sized instances, even if some of the variables are integer and unbounded. In this case, the problem contains an infinite number of linear constraints; these constraints are separated dynamically. We use this approach as a bounding routine in an SDP-based branch-and-bound framework. In case of a convex objective function, the new SDP bound improves the bound given by the continuous relaxation of the problem. Numerical experiments show that our algorithm performs well on various types of non-convex instances.

Mathematics Subject Classification 90C10 · 90C11 · 90C20 · 90C22 · 90C26

1 Introduction

Nonlinear mixed-integer optimization has recently moved to the focus of mathematical programming. While linear mixed-integer programming problems have been investigated for many decades, leading to a tremendous progress in their practical solution, fast algorithms for solving nonlinear problems are still rare and often restricted to special classes of objective functions or variable domains.

Even in the case of a quadratic objective function, state-of-the-art optimization techniques still perform weakly compared to those designed for linear problems. In contrast to the linear case, quadratic optimization is hard even in the absence of constraints:

C. Buchheim
Fakultät für Mathematik, Technische Universität Dortmund, Dortmund, Germany
e-mail: christoph.buchheim@tu-dortmund.de

A. Wiegele (✉)
Institut für Mathematik, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria
e-mail: angelika.wiegele@aau.at

while the problem is tractable in the convex, purely continuous case, it becomes hard in general both by non-convexity of the objective function and by the presence of integrality constraints. More precisely, the problem is known to be NP-hard for box-constrained continuous variables in the non-convex case and for unbounded integer variables in the convex case, the latter being equivalent to the closest vector problem.

Most algorithms and software tools for quadratic mixed-integer optimization are restricted to convex objective functions, or they can handle non-convex objective functions but do not guarantee optimality in this case [3,6,9]. This is due to the fact that these approaches mainly rely on dual bounds given by the global continuous minima of the respective objective functions. In case of a non-convex objective function, even the computation of these bounds is NP-hard. If bounds are given by local minima instead of global ones, the resulting solutions are suboptimal in general. Most methods for exactly solving non-convex mixed-integer optimization problems are based on the idea of convex estimators, combined with branching and range reduction techniques [2,15].

A well-studied NP-hard special case of quadratic mixed-integer optimization is unconstrained quadratic binary optimization. In this case, convexity of the objective function can be assumed without loss of generality. The problem is known to be equivalent to the maximum cut problem, which can easily be modeled as an unconstrained quadratic minimization problem over variables with domain $\{-1, 1\}$. A well-known and very successful method for solving such problems relies on a strong dual bound given by an SDP relaxation of the problem. This relaxation results from an exact problem formulation by dropping a (non-convex) rank-one constraint [8]; combining this approach with additional cutting planes yields a very fast algorithm for solving the maximum cut problem [13].

In the following, our aim is to generalize this approach to unconstrained non-convex quadratic minimization problems where variable domains are arbitrary closed subsets of $\mathbb{R}$. By this, we get a common framework for SDP relaxations of various NP-hard problems: for the maximum cut problem (domains $\{-1, 1\}$), the closest vector problem (domains $\mathbb{Z}$), and quadratic minimization with box constraints (domains $[0, 1]$). The basic observation motivating our approach is that this general problem can still be modeled as an SDP plus a rank-one constraint. This SDP can have an infinite number of linear constraints, but under very mild conditions separation can be done quickly. Moreover, it turns out that very few separation steps are necessary in practice for solving the SDP relaxation.

We embedded this SDP relaxation into a branch-and-bound framework in order to obtain an exact algorithm for general quadratic optimization problems. Experiments with various types of instances show that this approach, though very general, is competitive with other methods for mixed-integer quadratic optimization, or even yields significantly faster running times.

This paper is organized as follows. In the remainder of Sect. 1, we review the recent literature on non-convex quadratic programming and discuss our contribution. In Sect. 2, we describe our new approach to mixed-integer quadratic optimization. The basic SDP relaxation is introduced in Sect. 2.1, while in Sect. 2.2 we explain how to embed this SDP relaxation into a branch-and-bound scheme. In Sect. 2.3, we discuss the case of unbounded relaxations. Experimental results are presented in Sect. 3. Section 4 concludes.

1.1 Related work

Most of the literature on non-convex quadratic optimization does not consider any integrality constraints on the variables. In this case, one may assume that all variable domains are [0, 1]. The resulting problem is often called *BoxQP*; this problem is NP-hard. In order to address this problem, one usually tries to construct hopefully tight convex relaxations. The most successful types of relaxations are based on semidefinite programming or on RLT inequalities.

The standard SDP approach consists in linearizing the objective function by introducing a matrix Y and by relaxing the desired equation $Y = xx^\top$ to the convex constraint $Y \succeq xx^\top$. Equivalently, one can replace the matrix

$$\begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^\top$$

by a matrix X satisfying $X \succeq 0$, $X_{00} = 1$ and $\operatorname{rank}(X) = 1$, and then remove the non-convex rank constraint [18].

In the same linearized model of the original quadratic problem, the connection between X_{ij} and x_i, x_j can be described by linear inequalities called RLT inequalities [10,17]. These take the bounds on both variables into account; in the BoxQP case they can be written as $X_{ij} \geq 0$, $X_{ij} \geq x_i + x_j - 1$, $X_{ij} \leq x_i, x_j$. Anstreicher [1] showed that SDP and RLT relaxations do not contain each other and that a combination of both can yield significantly better bounds than both relaxations alone.

From a practical point of view, the combination of SDP and RLT relaxations is difficult, since solving semidefinite programs with a large number of constraints is time-consuming. For this reason, Qualizza et al. [12] propose to approximate the constraint $X \succeq 0$ by linear inequalities, so-called *psd inequalities*. The main drawback of this approach is the fact that a large number of psd inequalities is needed to obtain a relaxation that is close to the SDP relaxation.

For the case of discrete variables, the literature on non-convex quadratic optimization is more limited. Most work is concentrated on the special case of binary variables. In this case, the problem is equivalent to the well-studied max-cut problem, and the above linearized model coincides with the standard max-cut model with edge variables. Both integer programming and SDP approaches for max-cut and binary quadratic programming have been investigated intensively; see e.g. [13] and the references therein.

In the case of general integers, Saxena et al. [16] present a cutting-plane approach for non-convex mixed-integer quadratically constrained programming based on the idea of disjunctive cuts. These cuts are not only derived from integrality constraints, but also from the non-convex constraint $Y \preceq xx^\top$. In other words, this approach does not explicitly use the constraint $X \succeq 0$ in an SDP-framework, but rather uses cutting planes derived from this constraint inside an integer programming approach. By the idea of disjunctive programming, however, these cutting planes implicitly take into account also problem information beyond $X \succeq 0$.

The special case of convex integer quadratic programming has also been studied recently. If variables are unbounded, it is equivalent to the well-known closest vector

problem, but applications arise also in the case of bounded variables. See Buchheim et al. [6] for a recent branch-and-bound approach.

1.2 Our contribution

The main focus of our approach is on discrete optimization problems. We present SDP relaxations that do not only model the non-convexity of the objective function, but also non-convexity of the variable domains. The strength of our approach lies in the fact that we address both types of non-convexity simultaneously: if we deal with the non-convexity of the objective function in an appropriate way, we can model the variable domains by convex constraints. We show that both types of non-convexity can be accommodated by a single non-convex rank constraint.

Experimentally, we show that this approach clearly outperforms Couenne [2], a software for non-convex optimization, on integer or mixed-integer instances. In our evaluation, we are also interested in the dependance of performance on the degree of non-convexity of the problem, i.e., on the percentage of positive eigenvalues of the quadratic matrix in the objective function. Our algorithm turns out to be particularly strong for instances that are convex or nearly convex. For these instances, modeling the non-convexity of variable domains is crucial, so we are led to the conclusion that our method of addressing discrete variables is appropriate and leads to very good results in practice.

Another conclusion of our experiments is the observation that adding more linear inequalities to our relaxation does not pay off in terms of running time. In our model, we need very few inequalities per variable to obtain a rather tight relaxation. Even adding only RLT inequalities leads to a much slower solution of the SDP relaxations, which cannot be compensated by the improvement of lower bounds due to tighter models. Our method therefore differs significantly from approaches based on cutting-planes such as [16].

2 The new approach

In the following, let $Q \in S_n$ denote a symmetric, but not necessarily positive semi-definite matrix of dimension $n \in \mathbb{N}$. Moreover, let $l \in \mathbb{R}^n$ and $c \in \mathbb{R}$. We consider the problem

$$\begin{array}{ll} \min & f(x) = x^\top Q x + l^\top x + c \\ \text{s.t.} & x \in D_1 \times \cdots \times D_n \,, \end{array} \tag{1}$$

where each D_i is a closed subset of $\mathbb{R}$. For example, we can choose $D_i = \{-1, 1\}$ to obtain the maximum cut problem, or $D_i = \mathbb{Z}$ and a convex objective function to obtain the closest vector problem. However, Problem (1) is more general, e.g., one can define $D_i = \{0\} \cup [1, \infty)$. This type of domain appears in portfolio optimization problems with buy-in thresholds [5]. Note that it is allowed to choose all D_i differently, so that Problem (1) also contains the mixed-integer case.

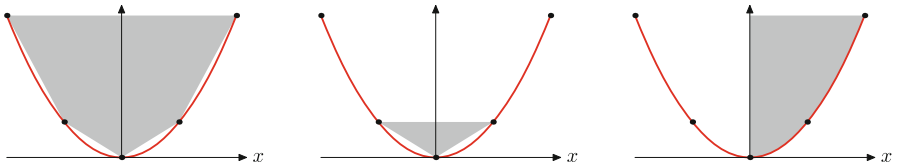

Fig. 1 The set $P(D)$ for $D = \mathbb{Z}$, $D = \{-1, 0, 1\}$ and $D = [0, \infty)$

2.1 SDP relaxation

We aim at deriving an SDP relaxation of Problem (1). The first steps of the construction are folklore; see e.g. [1,18]. We first linearize the problem using the function $\ell\colon \mathbb{R}^n \to S_{n+1}$ defined as

$$\ell(x) = \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^\top,$$

and the matrix

$$\tilde{Q} = \begin{pmatrix} c & \frac{1}{2} l^\top \\ \frac{1}{2} l & Q \end{pmatrix},$$

which is indexed over $\{0, \ldots, n\}$ in the following. We can rewrite Problem (1) as

$$\begin{array}{ll} \min & \langle \tilde{Q}, \ell(x) \rangle \\ \text{s.t.} & x \in D_1 \times \cdots \times D_n \end{array} \tag{2}$$

where $\langle A, B \rangle = \sum_{i,j} a_{ij} b_{ij}$ is the inner product of A and B.

Since the objective function in Problem (2) is linear in $\ell(x)$, solving (2) reduces to a linear optimization problem over the convex set $\operatorname{conv} \ell(D_1 \times \cdots \times D_n)$. To investigate this set, we first define $P(D_i)$ as the closure of $\operatorname{conv} \{(x, x^2) \mid x \in D_i\}$ in $\mathbb{R}^2$. See Fig. 1 for some examples. Notice that taking the closure is necessary in case D is bounded to exactly one side, e.g. for $D = [0, \infty)$.

Theorem 1 *Let $X \in S_{n+1}$. Then $X \in \ell(D_1 \times \cdots \times D_n)$ if and only if*

(a) $(x_{0i}, x_{ii}) \in P(D_i)$ *for all* $i = 1, \ldots, n$,
(b) $x_{00} = 1$,
(c) $\operatorname{rank}(X) = 1$, *and*
(d) $X \succeq 0$.

Proof Let X satisfy (a)–(d). From $X \succeq 0$, $\operatorname{rank}(X) = 1$, and $x_{00} = 1$, we derive $X = \ell(x)$ for some $x \in \mathbb{R}^n$. Hence, $x_{ii} = x_{0i}^2$. With $(x_{0i}, x_{ii}) \in P(D_i)$, this implies $x_{0i} \in D_i$, using the strict convexity of the function $x_{0i} \mapsto x_{0i}^2$. Thus $(x_{01}, \ldots, x_{0n})$ is a feasible solution for Problem (1). On the other hand, if x is a feasible solution for Problem (1), then $\ell(x)$ satisfies (a)–(d). □

Note that the only non-convex constraint in Theorem 1 is the rank-constraint. In particular, not only the non-convexity of the objective function but also the potential non-convexity of the domains D_i is accommodated in the rank-constraint. We derive

Corollary 1 *Let $X \in \operatorname{conv} \ell(D_1 \times \cdots \times D_n)$. Then $X \succeq 0$, $x_{00} = 1$, and $(x_{0i}, x_{ii}) \in P(D_i)$ for all $i = 1, \ldots, n$.*

Corollary 1 suggests to consider the following SDP relaxation of Problem (1):

$$\begin{array}{rl} \min & \langle \tilde{Q}, X \rangle \\ \text{s.t.} & (x_{0i}, x_{ii}) \in P(D_i) \ \forall i = 1, \ldots, n \\ & x_{00} = 1 \\ & X \succeq 0\,. \end{array} \tag{3}$$

It is easy to see that $(x_{0i}, x_{ii}) \in P(\mathbb{R})$ already follows from the positive semidefiniteness of X:

Fact 1 *Let $X \in S^+_{n+1}$ with $x_{00} = 1$. Then $x_{ii} \geq x_{0i}^2$ for all $i = 1, \ldots, n$. Moreover, if $x_{ii} = x_{0i}^2$ for all $i = 1, \ldots, n$, then $X = \ell(x)$ for $x = (x_{01}, \ldots, x_{0n})^\top$.*

Proof Let $i, j \in \{1, \ldots, n\}$. Substituting $x_{00} = 1$, the determinant of the submatrix of X given by the rows and columns 0 and i is $x_{ii} - x_{0i}^2$, so $X \succeq 0$ implies $x_{ii} \geq x_{0i}^2$. Moreover, substituting $x_{00} = 1, x_{ii} = x_{0i}^2$, and $x_{jj} = x_{0j}^2$, the determinant of the submatrix of X given by the rows and columns 0, i, and j turns out to be $-(x_{ij} - x_{0i}x_{0j})^2$. From $X \succeq 0$ we derive $x_{ij} = x_{0i}x_{0j}$. □

If D_i is a proper subset of $\mathbb{R}$, we need further linear constraints to model $(x_{0i}, x_{ii}) \in P(D_i)$. In general this requires an infinite number of inequalities. However, we can easily devise an exact separation algorithm for $P(D_i)$ provided that the following tasks can be accomplished efficiently:

1. given $x \in \mathbb{R} \cup \{-\infty\}$, find the smallest element of $D_i \cap [x, \infty)$
2. given $x \in \mathbb{R} \cup \{\infty\}$, find the largest element of $D_i \cap (-\infty, x]$

The sets $D_i \cap [x, \infty)$ and $D_i \cap (-\infty, x]$ are closed by assumption, so that the corresponding minima and maxima exist (though they can be $\pm\infty$). Algorithm SepP describes the separation procedure for $P(D_i)$.

The point to be separated is a pair (x_{0i}, x_{ii}) from a positive semidefinite matrix X. By Fact 1, it follows that $x_{ii} \geq x_{0i}^2$, meaning that it belongs to the epigraph of the function $x \mapsto x^2$, see Fig. 2a. If x_{0i} belongs to D_i it follows that (x_{0i}, x_{ii}) is in $P(D_i)$ and we are done. Otherwise, if x_{0i} is not in D_i, we determine the nearest points $u, l \in D_i$ to the left and right of x_{0i}, respectively (see Fig. 2b). If u and l are finite we only have to check the secant through the points (u, u^2) and (l, l^2) as a potential cutting plane for (x_{0i}, x_{ii}), see Fig. 2c. If one of the two is infinite, we generate either the cutting plane $x \geq l$ or $x \leq u$, see Fig. 3a. If D_i is bounded, we finally have to check whether (x_{0i}, x_{ii}) lies above $P(D_i)$, as in Fig. 3b.

This rises the question of how many iterations of the separation procedure are necessary until all constraints are satisfied. As we will see later, the number of iterations

Algorithm SepP: separation algorithm for $P(D)$

Input: a closed non-empty set $D \subseteq \mathbb{R}$ and a pair $(x, y) \in \mathbb{R}^2$ with $y \geq x^2$
Output: a valid inequality $a^\top x \leq b$ for $P(D)$ violated by (x, y), if one exists
compute $l = \min\big(D \cap [x, \infty)\big)$;
compute $u = \max\big(D \cap (-\infty, x]\big)$;
if $l \in \mathbb{R}$ *and* $u \in \mathbb{R}$ **then**
  if $y < (u + l)x - ul$ **then**
    return $y \geq (u + l)x - ul$;
  end
end
if $u = -\infty$ **then**
  return $x \geq l$;
end
if $l = \infty$ **then**
  return $x \leq u$;
end
compute $\tilde{l} = \max D = \max\big(D \cap (-\infty, \infty]\big)$;
compute $\tilde{u} = \min D = \min\big(D \cap [-\infty, \infty)\big)$;
if $\tilde{l} \in \mathbb{R}$ *and* $\tilde{u} \in \mathbb{R}$ **then**
  if $y > (\tilde{u} + \tilde{l})x - \tilde{u}\tilde{l}$ **then**
    return $y \leq (\tilde{u} + \tilde{l})x - \tilde{u}\tilde{l}$;
  end
end
return "no violated inequality exists";

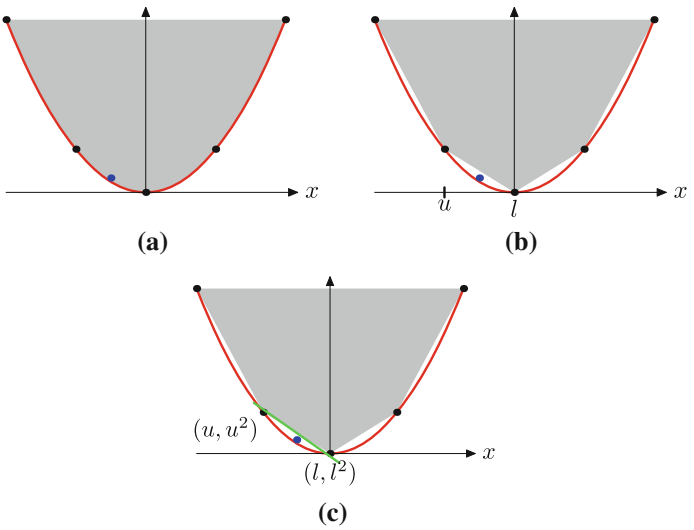

Fig. 2 Separating inequalities

remains very small in practice, even in the pure integer case. It is thus a feasible approach to solve the SDP relaxation (3) by calling an interior-point algorithm after each iteration of Algorithm SepP.

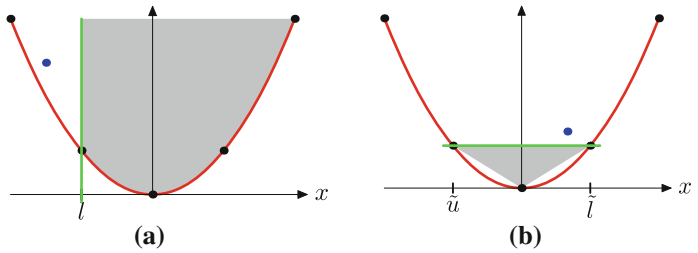

Fig. 3 Separating inequalities, part 2

In case of a convex objective function, we can compare the optimum of (3) with the global continuous minimum of f. Clearly, the latter is obtained when relaxing each D_i to $\mathbb{R}$.

Fact 2 *If f is a convex function with a continuous minimizer $\bar{x}$, then the value of (3) is at least $f(\bar{x})$.*

Proof The convexity of f implies $x_{ii} = x_{0i}^2$ for all optimal solutions X of (3), for all $i = 1, \ldots, n$. In particular, for $D_i = \mathbb{R}$, condition (a) of Theorem 1 holds. Moreover, Fact 1 yields $X = \ell(x)$, so that conditions (b)–(d) are also satisfied. Hence the continuous minimum $f(\bar{x})$ can be computed as

$$\begin{aligned} \min \quad & \langle \tilde{Q}, X \rangle \\ \text{s.t.} \quad & x_{00} = 1 \\ & X \succeq 0 \,. \end{aligned} \tag{4}$$

The latter is a relaxation of (3), hence the result. □

Finally, note that (3) is a generalization of the well-known SDP relaxation for the maximum cut problem: choosing $D_i = \{-1, 1\}$ for all $i = 1, \ldots, n$, the set $P(D_i)$ is essentially given by the equation $x_{ii} = 1$.

2.2 Branch-and-bound algorithm

Our aim is to solve Problem (1) to proven optimality. To this end we use the SDP relaxation (3) in a branch-and-bound framework. For branching, we choose a variable x_i, having value $\bar{x}_{0i}$ in the solution of (3), and produce two subproblems by splitting up the domain D_i into $D_i \cap (-\infty, \bar{x}_{0i}]$ and $D_i \cap [\bar{x}_{0i}, \infty)$. See Fig. 4 for two examples.

The choice of the branching variable is motivated by Fact 1: we can choose the variable i that maximizes $x_{ii} - x_{0i}^2$. Note that Fact 1 guarantees feasibility as soon as $x_{ii} = x_{0i}^2$ for all i. However, under the presence of continuous variables, we will never reach equality in practical computations after a finite number of branching steps. Therefore we use the following variant of Fact 1.

Lemma 1 *Let $D_i \subseteq [0, 1]$ for all $i = 1, \ldots, n$ and let x^* be an optimal solution of (1). Let $\varepsilon > 0$. Then there exists $\delta > 0$ such that for every optimal solution X of (3) with $x_{ii} - x_{0i}^2 \leq \delta$ it follows that $f(x^*) - \langle \tilde{Q}, X \rangle \leq \varepsilon$.*

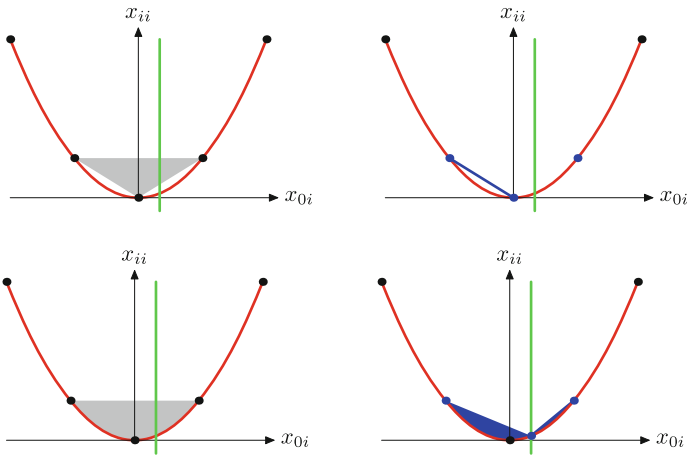

Fig. 4 Branching on integer and continuous variables. We branch at $\bar{x} = \frac{1}{3}$. *Top* splitting up $D = \{-1, 0, 1\}$ yields the new domains $\{-1, 0\}$ and $\{1\}$. *Bottom* splitting up $D = [-1, 1]$ yields the new domains $[-1, \frac{1}{3}]$ and $[\frac{1}{3}, 1]$

Proof It suffices to prove that $|x_{ii} - x_{0i}^2| \leq \delta$ for all i implies

$$|x_{ij} - x_{0i}x_{0j}| \leq \sqrt{\delta^2 + 2\delta}$$

for all i, j. This follows from considering the determinant of the submatrix given by the rows and columns indexed by 0, i and j. □

The complete branch-and-bound algorithm Q-MIST (Quadratic Mixed-Integer Semidefinite programming Technique) is now described below.

Theorem 2 *Let all D_i be bounded. Then for any $\varepsilon > 0$, Algorithm Q-MIST terminates in finite time with a feasible solution x such that $f(x) \leq f^* + \varepsilon$, where f^* is the optimum of Problem (1).*

Proof We may assume that $D_i \subseteq [0, 1]$. By Lemma 1, there exists $\delta > 0$ such that Algorithm Q-MIST terminates as soon as $x_{ii} - x_{0i}^2 \leq \delta$ for all $i = 1, \ldots, n$. The latter condition is always satisfied for intervals of length at most $2\sqrt{\delta}$. In other words, the algorithm will never choose a branching position within an interval of length less than $2\sqrt{\delta}$. □

Note that in case of unbounded sets D_i, the algorithm might not terminate in finite time, as discussed in Sect. 2.3.

A further improvement of Algorithm Q-MIST is obtained by applying the range reduction technique [14]. For this, assume again that all domains D_i are bounded, with $l_i = \min D_i$ and $u_i = \max D_i$. Let λ_i denote the dual multiplier corresponding to the constraint $x_{ii} \leq (u_i + l_i)x_{i0} - u_i l_i$. Let U denote the value of the currently best known solution and $L = \langle \tilde{Q}, X \rangle$ the optimal solution value of the current SDP

Algorithm Q-MIST: branch-and-bound algorithm for Problem (1)

Input: $Q \in S_n, l \in \mathbb{R}^n, c \in \mathbb{R}$, closed subsets $D_i \subseteq \mathbb{R}$ for $i = 1, \dots, n, \varepsilon > 0$
Output: $x^* \in \mathbb{R}^n$ s.t. $f(x^*)$ differs at most ε from $\min f(x), x \in D_1 \times \cdots \times D_n$
let $U = \infty$;
let $\mathcal{P}$ be the SDP (3);
let $\mathcal{S} = \{\mathcal{P}\}$;
while $\mathcal{S} \neq \emptyset$ **do**
  choose some $\mathcal{P}$ from $\mathcal{S}$;
  let $\mathcal{S} = \mathcal{S} \backslash \{\mathcal{P}\}$;
  repeat
    solve $\mathcal{P}$ (e.g., using an interior point algorithm) to obtain X;
    for $i = 1$ **to** n **do**
      apply Algorithm SepP to separate (x_{0i}, x_{ii}) from $P(D_i)$;
      add generated cutting planes to $\mathcal{P}$
    end
  until *no cutting plane has been generated*;
  choose $\hat{x} \in D_1 \times \cdots \times D_n$ to obtain a feasible solution $\ell(\hat{x})$ of $\mathcal{P}$;
  if $f(\hat{x}) < U$ **then**
    let $U = f(\hat{x})$;
    let $x^* = \hat{x}$;
  end
  if $\langle \tilde{Q}, X \rangle < U - \varepsilon$ **then**
    find i maximizing $x_{ii} - x_{0i}^2$;
    obtain $\mathcal{P}_1$ from $\mathcal{P}$ by replacing D_i by $D_i \cap (-\infty, x_{0i}]$;
    obtain $\mathcal{P}_2$ from $\mathcal{P}$ by replacing D_i by $D_i \cap [x_{0i}, \infty)$;
    let $\mathcal{S} = \mathcal{S} \cup \{\mathcal{P}_1, \mathcal{P}_2\}$;
  end
end

relaxation; see Algorithm Q-MIST. If $\lambda_i > 0$, we can derive that all optimal solutions of our problem satisfy

$$x_{i0} \leq \frac{u_i + l_i}{2} - \sqrt{\Delta} \quad \text{or} \quad x_{i0} \geq \frac{u_i + l_i}{2} + \sqrt{\Delta}$$

if $\Delta = \frac{1}{4}(u_i + l_i)^2 - u_i l_i - \frac{U-L}{\lambda_i} \geq 0$. This information can be used for the branching variable x_i in order to decrease the size of the domains D_i in the two resulting subproblems.

2.3 Unbounded relaxations

If all domains D_i in (1) are bounded, clearly all $P(D_i)$ are bounded and thus relaxation (3) is bounded as well. In case of unbounded domain but convex objective function Problem (3) is bounded by the continuous minimum since the latter is the optimum of Relaxation (4).

However, in case of an unbounded domain and a non-convex objective function, the initial relaxation (3) may be unbounded, even if the original Problem (1) is bounded. As an example, consider the problem of testing copositivity of a matrix Q. This problem can be modeled as

$$\begin{array}{l} \min\ x^\top Q x \\ \text{s.t.}\ \ x \geq 0\ . \end{array}$$

If Q is copositive, this problem is bounded by zero. On the other hand, it is easy to see that the SDP relaxation (3) is unbounded as soon as Q is not positive semidefinite. Hence we have an infinite gap between (1) and (3) for each matrix being copositive but not positive semidefinite, e.g., for the Horn matrix

$$H = \begin{pmatrix} 1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & 1 & 1 & -1 & 1 \end{pmatrix}\ .$$

Notice however that the copositivity test can be modeled alternatively by

$$\begin{array}{l} \min\ x^\top Q x \\ \text{s.t.}\ \ x \in [0,1]^n, \end{array}$$

which also fits into our framework.

3 Experiments

We implemented Algorithm Q-MIST in C++ using CSDP 6.0.1 [4] for solving the SDP relaxations. We carried out experiments on numerous classes of instances, containing both convex and non-convex objective functions and both integer and continuous variables. All tests were performed on an Intel Core Duo Processor running at 3.00 GHz. We stop our algorithm as soon as the absolute gap falls below 10^{-6}. Moreover, we do not branch on variables already restricted to intervals of length at most 10^{-6}.

For most of our experiments, we created random objective functions f as follows: given a percentage $p \in [0, 100]$, we choose n numbers μ_i, where the first $\lfloor pn/100 \rfloor$ are chosen uniformly at random from $[-1, 0]$ and the remaining ones are chosen uniformly at random from $[0, 1]$. Next, we generate n vectors of dimension n each, now choosing all entries uniformly at random from $[-1, 1]$, and orthonormalize them to obtain vectors v_i. The coefficient matrix Q is then calculated as $Q = \sum_{i=1}^{n} \mu_i v_i v_i^\top$. In particular, the parameter p allows to control whether the matrix Q is positive semidefinite ($p = 0$), negative semidefinite ($p = 100$) or indefinite. Finally, we determine l by choosing all entries uniformly at random from $[-1, 1]$, and set $c = 0$.

Note that for $p = 0$ the resulting problems have a convex objective function. For these problems specialized algorithms exist, e.g. [6]. Since in this paper we focus on the non-convex case, which is considerably harder, we do not propose to apply our algorithm when solving purely convex instances. We do not expect our algorithm to be competitive in this case.

Table 1 Results for integer instances with domains $\{-10, \ldots, 10\}$

n	Q-MIST			COUENNE		
	Solved	Time	Nodes	Solved	Time	Nodes
10	110	0.1	17.7	110	29.6	5,720.8
20	110	2.3	206.9	*96*	*643.9*	*75,464.5*
30	110	43.1	1,435.3	*0*	–	–
40	*104*	*334.3*	*6,058.2*	*0*	–	–
50	*70*	*593.4*	*10,347.7*	*0*	–	–

3.1 Non-convex integer instances

We first generated random instances with $D_i = \{-10, \ldots, 10\}$ for all $i = 1, \ldots, n$. The number of variables n was chosen between 10 and 50. The objective function was generated randomly as described above, for each p out of $\{0, 10, \ldots, 100\}$ and each n we created 10 instances. Thus we consider 110 instances for each size n.

We compare Algorithm Q-MIST to the state of the art non-convex mixed-integer programming software COUENNE [2] (running on the same machine). Results are summarized in Table 1. The first column contains the number n of variables. The following columns contain, for both approaches, the number of solved instances, the average running time in seconds, and the average number of branch-and-bound nodes, respectively. For all our experiments we set a time limit of 1 h. Instances not solved to proven optimality within this time limit are not considered in the averages, averages not taken over all 110 instances are printed in italics. As can be seen from Table 1, we can solve all but six instances up to $n = 40$, whereas COUENNE is incapable of solving any instance beyond $n = 20$. In particular, the small number of nodes enumerated on average by Q-MIST compared with COUENNE is remarkable.

As explained above, we can control the number of negative eigenvalues of the coefficient matrix Q by the parameter p. Clearly, this parameter has a strong influence on running times. In order to investigate this influence, we plotted the average running times of Algorithm Q-MIST depending on p; see Fig. 5. We use a logarithmic scale. Different from the tables, we count the running times of unsolved instances as 3,600 s; the time limit is depicted as a dashed line. The different colors correspond to the different numbers of variables.

One can observe that instances having a convex or concave objective function are solved more easily by Algorithm Q-MIST than instances with indefinite Q. Note however that in both the convex and the concave case the problem remains NP-hard due to integrality constraints. On the other hand, COUENNE does not seem to profit at all from (near-)convexity.

To conclude this section we take a closer look at the separation algorithm. For the instances considered above, we have $D_i = \{-10, \ldots, 10\}$. It follows that the number of linear inequalities needed to define a single set $P(D_i)$ is 21. Indeed, for each $i \in \{-10, \ldots, 9\}$, we have a facet $x_{ii} \geq (2i + 1)x_{i0} - i(i + 1)$ of $P(D_i)$; the last facet is $x_{ii} \leq 100$. However, it turns out that only a small part of these facets is needed

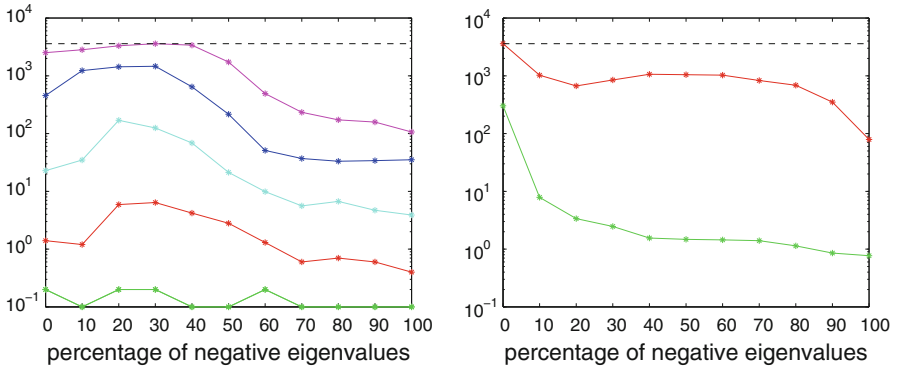

Fig. 5 The average running time for integer instances depending on the percentage of negative eigenvalues; Q-MIST (*left*) and COUENNE (*right*)

Table 2 Results for integer instances without separation

n	Q-MIST					
	w/ separation			w/o separation		
	Solved	Time	Nodes	Solved	Time	Nodes
10	110	0.1	17.7	110	0.1	27.2
20	110	2.3	206.9	110	4.8	345.3
30	110	43.1	1,435.3	110	92.6	3,580.7
40	*104*	*334.3*	*6,058.2*	*94*	*317.9*	*6,295.9*
50	*70*	*593.4*	*10,347.7*	*64*	*492.5*	*10,426.9*

when solving the SDP relaxation (3): for the random instances considered above, for $n = 50$, we need to generate only 3.24 such facets on average within one node of the branch-and-bound tree. In other words, very few calls of Algorithm SepP and hence very few reoptimizations of the SDP suffice to solve the relaxation (3). In general, more inequalities are needed for instances with less negative eigenvalues. To be more precise, the number of inequalities separated on average at each node is as follows.

p	0	10	20	30	40	50	60	70	80	90	100
avg. inequalities	31.20	2.38	0.95	0.25	0.03	0.02	0.00	0.00	0.00	0.00	0.00

The small number of iterations in the separation routine raises the question whether the constraints $(x_{0i}, x_{ii}) \in P(D_i)$ for $i = 1, \ldots, n$ have a crucial effect on the bound given by the SDP relaxation (3). Replacing these constraints by the single inequalities $x_{ii} \leq 100$ and switching off the separation algorithm, we still obtain an exact algorithm for solving Problem (1). Table 2 presents a comparison of both approaches. It is obvious from the average numbers of branch-and-bound nodes that our cutting planes improve the bounds significantly. Moreover, even if the number of SDPs to be solved per node is increased by separation, the overall running times decrease considerably for larger n, so that more instances can be solved to optimality within the time limit.

Table 3 Instances of Phuong et al. [11]

Example	Q-MIST		Phuong et al.
	Time	Nodes	Time
1	27	1,127	54
2	37	1,683	1
3	28	1,179	441
4	530	24,973	474

The size of these instances is $n = 40$

Table 4 Results for ternary instances

n	Q-MIST			COUENNE		
	Solved	Time	Nodes	Solved	Time	Nodes
10	110	0.1	9.1	110	0.4	18.0
20	110	0.4	53.5	110	61.5	3,822.0
30	110	3.4	199.7	*55*	*1,531.5*	*101,487.2*
40	110	22.1	831.6	*1*	*1,965.3*	*51,968.0*
50	110	231.9	5,463.7	*0*	–	–

3.2 Non-convex ternary instances

A problem arising in engineering applications is the following [11]:

$$\begin{aligned} \max \quad & x^\top C x \\ \text{s.t.} \quad & x \in \{-1, 0, 1\}, \quad \sum_{i=1}^{n} x_i = 0 \,. \end{aligned}$$

Instances of this type can be addressed by Algorithm Q-MIST by explicitly adding the constraint $\sum_{i=1}^{n} x_i = 0$ to the SDP relaxation (3).

A special case of this problem is considered in [11]. Here, the matrix C is circulant. In Table 3, we compare our algorithm with the algorithm presented in [11]. The latter is based on a problem-specific enumeration scheme. The timings for the algorithm of Phuong et al. are taken from [11], as the code is not available. They used a Pentium IV 2.53 GHz, a slightly slower machine than we used for our experiments. Taking this into account, we can still see that our algorithm performs comparably well, even if their algorithm is tailored for ternary quadratic problems with circulant cost matrices, while our algorithm does not exploit any special problem structure.

In order to run more tests on ternary instances, we generated (non-circulant) random matrices Q. Again, we chose dimensions n from 10 to 50, and random coefficient matrices with p % negative eigenvalues, with $p \in \{0, 10, \ldots, 100\}$. For each n and p, we again generated 10 matrices. Results are presented in Table 4, along with the running times of COUENNE. Note that in these experiments we omitted the constraint $\sum_{i=1}^{n} x_i = 0$. The effect of the number of negative eigenvalues on the average running time of Q-MIST is shown in Fig. 6.

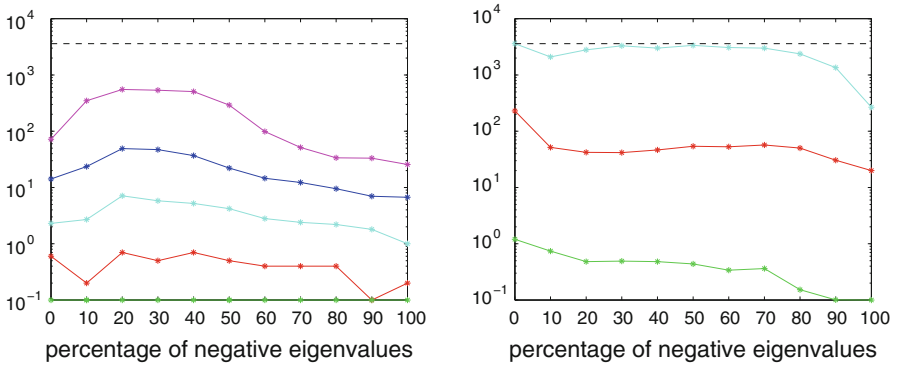

Fig. 6 The average running time for ternary instances depending on the percentage of negative eigenvalues; Q-MIST (*left*) and COUENNE (*right*)

Table 5 Results for mixed-binary instances

n	Q-MIST			COUENNE		
	Solved	Time	Nodes	Solved	Time	Nodes
10	110	0.1	12.2	110	0.1	5.5
20	110	0.3	35.5	110	8.7	105.6
30	110	6.7	255.5	110	81.9	559.4
40	*104*	*83.0*	*1,618.8*	*109*	*385.4*	*6,156.0*
50	*99*	*133.7*	*1,785.9*	*70*	*1,806.9*	*36,455.8*

3.3 Non-convex mixed-binary instances

We next generated random mixed-integer instances with box constraints, with half of the variables being integer. In other words, we chose $D_i = [0, 1]$ for half of the variables and $D_i = \{0, 1\}$ for the other half. The objective function was again produced as described above. Results are summarized in Table 5.

In Fig. 7, we plot the results by the percentage of negative eigenvalues. Again, instances having a convex or concave objective function are solved more easily than instances with indefinite Q. Note that in the concave case the problem is equivalent to a pure binary quadratic optimization problem. It turns out that the hardest instances for Algorithm Q-MIST arise when the percentage of negative eigenvalues lies between 10 and 30.

3.4 BoxQP instances

Our algorithm is mainly designed for problems containing integer variables. However, it can also be applied to problem instances without any integrality conditions. In this section, we compare our approach to the branch-and-bound algorithm of Burer [7].

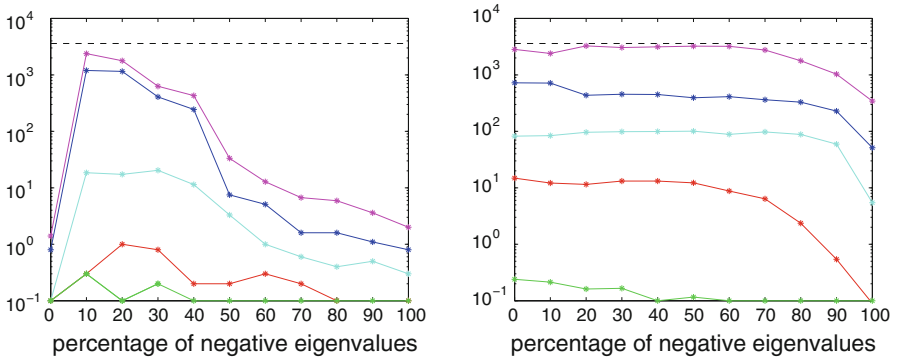

Fig. 7 The average running time for mixed-binary instances depending on the percentage of negative eigenvalues; Q-MIST (*left*) and COUENNE (*right*)

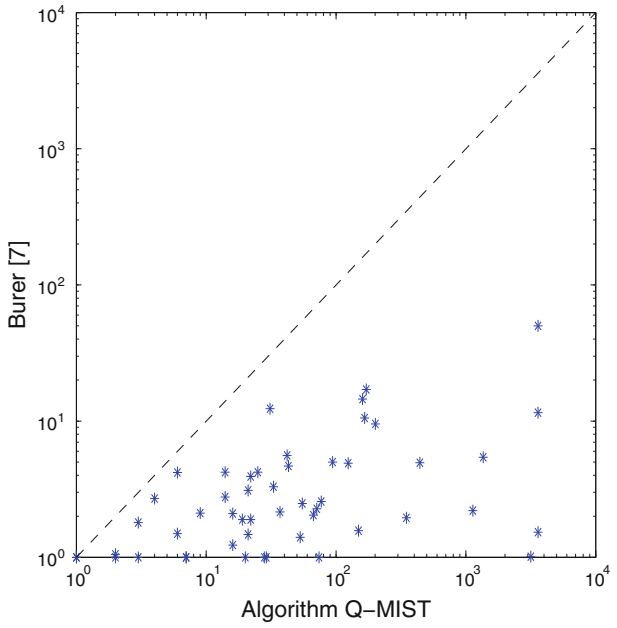

Fig. 8 Comparison of Algorithm Q-MIST and the branch-and-bound algorithm of Burer [7] for globally solving the 90 instances of BoxQP. Time in seconds, plotted on a logarithmic scale. The line $x = y$ is plotted for reference only

The set of instances we evaluated has dimension 20–60 and is taken from [19]. All variables have range [0, 1]. Such problems are called *BoxQP* problems.

We performed all experiments on the same computing system. As can be seen in Fig. 8, our algorithm is performing worse than the algorithm of [7], which is specialized for solving BoxQP problems. Nevertheless, our approach is still capable of solving the instances in reasonable running time.

4 Conclusion

We presented an exact solution method for solving a large class of quadratic optimization problems. This class includes integer and continuous variables, as well as non-convex objective functions. The new algorithm is based on a semidefinite programming relaxation embedded into a branch-and-bound framework. To the best of our knowledge, this is the first algorithm using pure semidefinite programming techniques for solving general non-convex quadratic mixed-integer problems.

Extensive experiments show that this algorithm can solve medium-sized instances within reasonable running time. A comparison with COUENNE, which is one of the few software packages that is capable of solving non-convex problems to optimality, demonstrates the outstanding performance of our algorithm. Even when compared to specialized solvers for smaller problem classes, namely ternary quadratic problems and BoxQP, our algorithm performs reasonably well.

Acknowledgments We thank Brian Borchers for his help in using CSDP and Sam Burer for making his codes available. Furthermore, we thank two anonymous referees for their constructive comments that helped to improve this paper.

References

1. Anstreicher, K.: Semidefinite programming versus the reformulation-linearization technique for non-convex quadratically constrained quadratic programming. J. Glob. Optim. **43**(2–3), 471–484 (2009)
2. Belotti, P.: Couenne: a user's manual. Technical report, Lehigh University (2009)
3. Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wächter, A.: An algorithmic framework for convex mixed integer nonlinear programs. Discret. Optim. **5**, 186–2004 (2008)
4. Borchers, B.: CSDP, a C library for semidefinite programming. Optim. Methods Softw. **11/12** (1–4), 613–623 (1999)
5. Branke, J., Scheckenbach, B., Stein, M., Deb, K., Schmeck, H.: Portfolio optimization with an envelope-based multi-objective evolutionary algorithm. Eur. J. Oper. Res. **199**(3), 684–693 (2009)
6. Buchheim, C., Caprara, A., Lodi, A.: An effective branch-and-bound algorithm for convex quadratic integer programming. Math. Program. (2012, to appear)
7. Burer, S.: Optimizing a polyhedral-semidefinite relaxation of completely positive programs. Math. Program. Comput. **2**(1), 1–19 (2010). ISSN 1867-2949
8. Goemans, M., Williamson, D.: Improved approximation algorithms for maximum cut and satisfiability problems. J. ACM **42**, 1115–1145 (1995)
9. ILOG, Inc. ILOG CPLEX 12.1 (2009) http://www.ilog.com/products/cplex
10. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs. I. Convex underestimating problems. Math. Program. **10**(2), 147–175 (1976). ISSN 0025-5610
11. Phuong, N.T.H., Tuy, H., Al-Khayyal, F.: Optimization of a quadratic function with a circulant matrix. Comput. Optim. Appl. **35**(2), 135–159 (2006)
12. Qualizza, A., Belotti, P., Margot, F.: Linear programming relaxations of quadratically constrained quadratic programs. In: IMA Volume Series, pp. 407–426. Springer (2012)
13. Rendl, F., Rinaldi, G., Wiegele, A.: Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. Math. Program. **121**(2), 307–335 (2010)
14. Ryoo, H.S., Sahinidis, N.V.: A branch-and-reduce approach to global optimization. J. Glob. Optim. **8**(2), 107–138 (1996)
15. Sahinidis, N.V., Tawarmalani, M.: BARON 9.0.4: global optimization of mixed-integer nonlinear programs. User's manual (2010)
16. Saxena, A., Bonami, P., Lee, J.: Convex relaxations of non-convex mixed integer quadratically constrained programs: extended formulations. Math. Program. **124**(1–2), 383–411 (2010)

17. Sherali, H.D., Adams, W.P.: A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems, vol. 31 of Nonconvex Optimization and Its Applications. Kluwer, Dordrecht (1999)
18. Vandenberghe, L., Boyd, S.: Semidefinite programming. SIAM Rev. **38**, 49–95 (1996)
19. Vandenbussche, D., Nemhauser, G.L.: A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. Math. Program. **102**(3), 559–575 (2005)