An Exact Algorithm for Nonconvex Quadratic Integer Minimization using Ellipsoidal Relaxations *

Christoph Buchheim† Marianna De Santis‡ Laura Palagi§ Mauro Piacentini§

October 29, 2012

Abstract

We propose a branch-and-bound algorithm for minimizing a not necessarily convex quadratic function over integer variables. The algorithm is based on lower bounds computed as continuous minima of the objective function over appropriate ellipsoids. In the nonconvex case, we use ellipsoids enclosing the feasible region of the problem. In spite of the nonconvexity, these minima can be computed quickly; the corresponding optimization problems are equivalent to trust-region subproblems. We present several ideas that allow to accelerate the solution of the continuous relaxation within a branch-and-bound scheme and examine the performance of the overall algorithm by computational experiments.

1 Introduction

Motivated by the progress made in recent decades both in nonlinear optimization and in integer programming, the focus of research has recently moved to the study of mixed-integer nonlinear optimization problems. These problems are usually hard to solve (in theory and in practice) due to the

*This work was partially supported by the Vigoni 2010 Project "Exact Methods for Integer Non Linear Programs"

†Fakultät für Mathematik, TU Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany – email: christoph.buchheim@tu-dortmund.de

‡Istituto di Analisi dei Sistemi e Informatica Antonio Ruberti – IASI CNR, Viale Manzoni 30, Roma, Italy – email: mdesantis@dis.uniroma1.it

§Department of Computer, Control, and Management Engineering Antonio Ruberti – Sapienza Università di Roma, Via Ariosto 25, Roma, Italy – email: {palagi,piacentini}@dis.uniroma1.it

presence of two types of nonconvexity: first, the objective function or constraints can be nonconvex, and second, the presence of integrality constraints leads to nonconvex variable domains.

We address quadratic integer optimization problems with box constraints,

$$\begin{aligned} \min \quad & q(x) = x^\top Q x + L^\top x \\ \text{s.t.} \quad & l_j \le x_j \le u_j \ (j = 1, \ldots, n) \\ & x \in \mathbb{Z}^n \ , \end{aligned} \tag{1}$$

where we may assume $l < u$ and $l, u \in \mathbb{Z}^n$. Recently, a fast branch-and-bound algorithm for the convex special case of Problem (1), i.e., the case when $Q \succeq 0$, has been proposed by Buchheim et al. [9]. Its main features are a fast incremental computation of lower bounds given by unconstrained continuous minimizers and an improvement of these bounds by considering lattice-free ellipsoids centered in the continuous minimizers. More precisely, the improved lower bounds are given as the minima of the objective function over the boundaries of these ellipsoids, which can be computed efficiently.

For a nonconvex objective function, this approach is not feasible any more; the unconstrained continuous minimizer does not even exist in this case. Moreover, even the continuous relaxation of this problem,

$$\begin{aligned} \min \quad & q(x) = x^\top Q x + L^\top x \\ \text{s.t.} \quad & l_j \le x_j \le u_j \ (j = 1, \ldots, n) \\ & x \in \mathbb{R}^n \end{aligned}$$

is an *NP*-hard problem in the case $Q \not\succeq 0$; it is equivalent to the so-called BoxQP problem [30].

In this paper, we present a novel approach for computing lower bounds for Problem (1) in the nonconvex case, which tries to apply ideas similar to those being used by Bienstock [5] and by Buchheim et al. [9] in the convex case. The main ingredient of our algorithm is the definition and solution of an appropriate continuous relaxation of Problem (1). We embed this relaxation into a branch-and-bound framework in order to obtain an exact algorithm for general integer quadratic optimization problems. Experiments with ternary instances, where $x \in \{-1, 0, 1\}^n$ is required, show that this approach is competitive or even faster than other methods for integer quadratic optimization.

Different from [9], we choose an ellipsoid E that contains all feasible solutions of Problem (1),

$$[l, u] \subseteq E = \{x \in \mathbb{R}^n \mid (x - x^0)^\top H (x - x^0) \le 1\}$$

where $H \succeq 0$ and x^0 denotes the center of the ellipsoid. We then define a relaxation of Problem (1) as

$$\begin{aligned} \min \quad & q(x) = x^\top Q x + L^\top x \\ \text{s.t.} \quad & x \in E \ . \end{aligned} \tag{2}$$

This relaxation is equivalent to a trust-region subproblem. This idea goes back to Kamath and Karmarkar [20, 21]. They consider the problem of minimizing a quadratic indefinite form (i.e. with $L = 0$) over $\{-1, 1\}^n$ and obtain a lower bound by solving the generalized eigenvalue problem $\min \frac{x^\top Q x}{x^\top H x}$ by means of suitable interior point methods.

A crucial aspect for obtaining a tight bound from Problem (2) is the choice of the ellipsoid E. E.g., if $l_j = -1$ and $u_j = 1$ for all $j = 1, \ldots, n$, a straighforward choice is the sphere

$$E = \{x \in \mathbb{R}^n \mid \|x\| \leq \sqrt{n}\} \ ,$$

which corresponds to $H = \frac{1}{n} I$ and $x^0 = 0$. However, different choices of the matrix H and of the center x^0 yield different bounds and can have a strong impact on the efficiency of the overall algorithm.

For Problem (2) with $H \succ 0$, strong Lagrangian duality holds; see e.g. [39, 32]. We can thus use the dual formulation to obtain an optimal solution. Switching to the dual problem has the advantage that even an approximate solution represents a safe lower bound to be used in a branch-and-bound framework. For this reason, we solve the primal formulation only in a preprocessing phase using the approach of Lucidi and Palagi [24], while using the dual formulation at each node of the branching tree. By the presence of integrality constraints, instances with 50–100 variables can already be considered very challenging, so that the continuous subproblems to be solved are not large scale in the usual sense. Hence for the dual formulation we can use a simplified version of the algorithm of Moré and Sorensen [28] designed for small scale problems.

As in [9], our enumeration strategy is depth-first and we always branch by fixing the value of one of the n variables. A crucial property of our algorithm is that we restrict the order in which variables are fixed. In other words, the set of fixed variables only depends on the depth of the node in the branch-and-bound tree. We thus loose the flexibility of choosing the best branching variable, but this strategy allows us to process a single node in the tree much faster. This is due to the fact that only n different submatrices of Q appear in the tree in this case, so that many time-consuming calculations can be done in a preprocessing phase.

This paper is organized as follows. In Section 2, we describe the basic idea of our approach for computing lower bounds using axis-parallel ellipsoids. Moreover, we propose different strategies for choosing these ellipsoids. The main components of the overall branch-and-bound algorithm are discussed in Section 3. Finally, Section 4 contains the results of an experimental evaluation of this algorithm and Section 5 concludes.

1.1 Related Work

Most software developed for integer nonlinear optimization can guarantee global optimality of the computed solutions only when the problem is convex [6, 19]. This is due to the fact that the underlying algorithms rely on solving continuous relaxations of the integer problems to proven optimality, which in a general situation requires convexity. For nonconvex objective functions, Buchheim and Wiegele [8] propose a branch-and-bound approach based on semidefinite programming (SDP) relaxations of Problem (1). In case of a convex objective function, this SDP bound improves over the bound given by the continuous relaxation of the problem. Numerical experiments are reported for various types of nonconvex instances, showing that this approach significantly outperforms other software such as Couenne [2]. Another approach for mixed-integer quadratic optimization was recently devised by Saxena et al. [35]. This approach uses cutting planes derived from the positive semidefiniteness of the matrix $xx^\top$.

An important special case of Problem (1) arises when all variables are binary. In this case, Problem (1) is equivalent to the well-studied Max-Cut problem. The latter problem has been addressed by both integer programming (IP) and SDP based methods; see [29] for recent advances and further references. These methods strongly rely on the binarity of variables, so that a generalization to larger domains is not straightforward. This is true in particular for IP based approaches. In the SDP context, the algorithm of [8] can be considered a generalization of the corresponding Max-Cut techniques.

Most literature on nonconvex quadratic minimization does not deal with integrality constraints. For the BoxQP problem, again both SDP based approaches and approaches using linear constraints have been examined. In particular, an effective set of valid linear inequalities is given by the so-called RLT-inequalities [26, 36]. Fast algorithms for BoxQP based on semidefinite programming have been devised by Vandenbussche and Nemhauser [40] and by Burer [10].

All approaches for quadratic optimization discussed above share two features: the original problem (1) is first linearized by adding one new vari-

able x_{ij} for each product $x_i x_j$ of original variables. Second, the relaxations used to compute lower bounds are all convex. In fact, the aim of obtaining convex relaxations is the basic motivation for linearizing the problem in the first step. However, this usually leads to a large number of new variables.

There are also approaches based on nonconvex quadratic relaxations. Among them we mention the seminal paper of Kamath and Karmarkar [21], proposing for the first time to approximate the feasible region by an ellipsoid. However, they do not examine the quality of the resulting bounds and do not embed them into a branch-and-bound algorithm. Le Thi Hoai and Pham Dinh [23] and later De Angelis et al. [12] exploit this idea within a branch-and-bound algorithm using rectangular partitions for solving BoxQP problems. After a suitable adaptation, this approach can also be applied to binary problems.

1.2 Our Contribution

The main contribution of this paper is a novel approach to nonconvex quadratic integer optimization, combining techniques from nonlinear and discrete optimization in an appropriate way. In particular, this approach has the following features:

Lower bounds from nonconvex relaxations. We obtain lower bounds by solving appropriate continuous optimization problems, without considering convex relaxations of the quadratic objective function. Our relaxation is a trust-region type problem which possesses some "hidden convexity" property so that it is possible to solve it to global optimality efficiently, as discussed in Section 2.

Use of dual problems for bound computation. Instead of solving the primal trust-region relaxation, we solve its dual formulation which is known to have zero duality gap. Approximate solutions of the dual relaxation still yield valid lower bounds, whereas in the primal formulation we need to obtain a very accurate solution. In practice, it does not pay off to solve the dual problems with a high accuracy, since a small increase in the bound is unlikely to allow pruning of the node. This is discussed in Section 2.2.

Acceleration by preprocessing. For solving the continuous problems more efficiently, we compute a spectral decomposition of the respective matrices during preprocessing. For a single solution of the problem, this does not pay off. However, within our branch-and-bound algorithm, problems

that share the same matrix are solved exponentially often, so that the preprocessing has a very positive effect on the overall running time. Moreover, we compute initial solutions for the relaxations in the preprocessing. See Section 3.2 for more details.

Reordering of input matrix. We reorder the variables of Problem (1) so that a convex subproblem is reached as soon as possible in the branching tree, assuming that variables are fixed in the resulting order; see Section 3.3. In the convex case, bounds tend to be tighter and nodes can be pruned more efficiently, as done in [9].

2 Lower Bounds

Our objective is to develop a fast branch-and-bound algorithm for solving Problem (1) to proven optimality. The main step towards this objective is the definition of a fast method for computing tight lower bounds for Problem (1), which will be discussed in the present section. Other features of our branch-and-bound approach will be presented in Section 3.

Starting from (1), we first relax the integrality constraints and obtain the continuous problem

$$\begin{aligned} \min \quad & q(x) = x^\top Q x + L^\top x \\ \text{s.t.} \quad & l_j \leq x_j \leq u_j \ (j = 1, \ldots, n) \ . \end{aligned}$$

Next, for sake of simplicity, we apply the transformation

$$x_j \leftarrow \frac{2x_j - (l_j + u_j)}{u_j - l_j}, \quad j = 1, \ldots, n$$

so that from now on we can assume $l_j = -1$ and $u_j = 1$ for all $j = 1, \ldots, n$. This relaxation is still an *NP*-hard problem in the nonconvex case, so that we further relax the problem to minimizing the objective function $q(x)$ over appropriate ellipsoids $E(H)$ such that

$$[-1, 1]^n \subseteq E(H) = \{x \in \mathbb{R}^n \mid (x - x^0)^\top H (x - x^0) \leq 1\} \ , \tag{3}$$

with $H \succeq 0$. The resulting relaxation is

$$\begin{aligned} \min \quad & q(x) = x^\top Q x + L^\top x \\ \text{s.t.} \quad & x \in E(H) \ . \end{aligned} \tag{4}$$

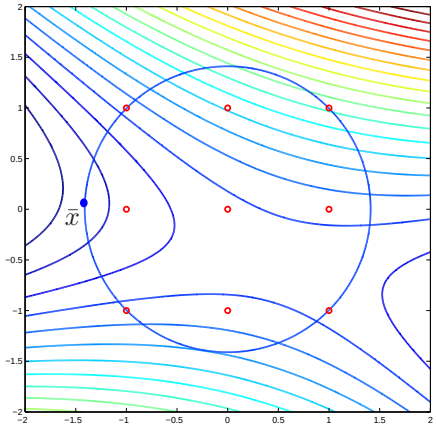

Figure 1: Computation of lower bounds in the nonconvex case. The minimizer of $q(x)$ over the given sphere is denoted by $\bar{x}$.

See Figure 1 for an illustration of the case of indefinite $q(x)$ with a spherical constraint ($H = I$) where $x^0 = 0$.

At this point, we have some flexibility in choosing the matrix H and the center point x^0; the radius can be assumed to be 1 after scaling. The choice of the ellipsoid $E(H)$ clearly has an influence on the strength of the bound resulting from (4). For different reasons, we restrict ourselves to axis-parallel ellipsoids in our approach, i.e., we assume that H is a diagonal matrix. In the following, this will be discussed in more detail.

Problems of type (4) are known as generalized trust-region subproblems [11, 27]. Such problems have been deeply studied in the field of unconstrained minimization. It is well known that necessary and sufficient optimality conditions exist even when $Q \not\succeq 0$, so that Problem (4) can be considered an "easy" problem from a theoretical point of view. Indeed, it has been shown that an approximation of the optimal value can be computed in polynomial time; see e.g. [3, 41, 42]. This is discussed in more detail in Section 2.2.

In our approach, we apply the relaxation (4) in two different ways, depending on whether we have $Q \succeq 0$ or not. In the nonconvex case $Q \not\succeq 0$, the optimal solution of (4) cannot be in the interior of the convex set $E(H)$, because the second order necessary condition would require $Q \succeq 0$.

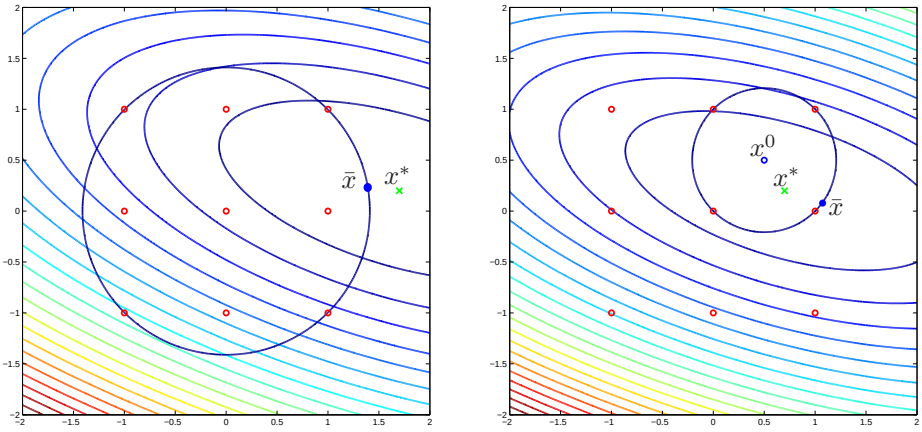

Figure 2: Computation of lower bounds in the convex case. The minimizer of $q(x)$ over the boundary of the sphere is denoted by $\bar{x}$.

In the case $Q \succeq 0$, we can still use the same approach if the global continuous minimizer x^* of $q(x)$ does not belong to $E(H)$. Then it is easy to see that even the value obtained by minimizing $q(x)$ over the boundary of $E(H)$ is a valid lower bound for Problem (1), which is tighter than the global minimum $q(x^*)$; see Figure 2 (left).

Still assuming $Q \succeq 0$, we can additionally improve the lower bound in the following way, inspired by the approach proposed in [5] and [9]: we can choose an ellipsoid containing x^* but not containing any integer feasible solution of Problem (1) in its interior. Then again the minimizer of $q(x)$ over its boundary will yield a valid lower bound improving the bound $q(x^*)$; see Figure 2 (right).

More precisely, we choose as center x^0 a point in $(\frac{1}{2}, \ldots, \frac{1}{2})^\top + \mathbb{Z}^n$ closest to x^*, which can be computed easily by rounding. Then we determine the maximal scaling of the ellipsoid $E(H)$,

$$\lambda E(H) := \{x \in \mathbb{R}^n \mid (x - x^0)^\top \big(\tfrac{1}{\lambda^2} H\big)(x - x^0) \leq 1\} \ ,$$

that does not contain any integer feasible point in its interior. Since we assume H to be a diagonal matrix, this is an easy task: the first integer

feasible point touched by the scaled ellipsoid will be $x^1 \in \mathbb{Z}^n$ with

$$x_i^1 = \begin{cases} l_i & \text{if } x_i^0 \leq l_i \\ u_i & \text{if } x_i^0 \geq u_i \\ \lfloor x_i^0 \rceil & \text{otherwise.} \end{cases}$$

Now λ can be chosen as $\sqrt{(x^1 - x^0)^\top H (x^1 - x^0)}$. Clearly, $x^* \in \lambda E(H)$, so that the minimum of $q(x)$ over the boundary of $\lambda E(H)$ again improves over $q(x^*)$.

In summary, in each of the three cases we need to optimize over the boundary of a given ellipsoid. In other words, we need to solve the equality constrained version of Problem (4),

$$\begin{aligned} \min \quad & q(x) = x^\top Q x + L^\top x \ , \\ \text{s.t.} \quad & x^\top H x = 1 \end{aligned} \tag{5}$$

for an appropriate matrix H. Here, after a simple transformation which does not affect the matrices Q or H, we have assumed that $x^0 = 0$.

At this point, the main questions are how to choose the matrix H in order to obtain tight bounds and how to solve Problem (5) efficiently. These questions are discussed in the following subsections.

2.1 Choice of the Ellipsoid

A key ingredient for the definition of an efficient branch-and-bound algorithm based on the ideas discussed above is the choice of a matrix H that leads to tight lower bounds but at the same time allows to simplify the calculation of this bound in order to obtain fast running times. This is one of our motivations for restricting ourselves to axis-parallel ellipsoids $E(H)$, as mentioned above.

We thus aim at obtaining a tight bound from Problem (5) by choosing an appropriate diagonal positive semidefinite matrix H. More formally, we consider the set

$$\mathcal{H}_{diag} := \left\{ H \succeq 0 \mid H = \mathrm{Diag}(h), \ \sum_{i=1}^{n} h_i = 1 \right\} ,$$

which defines a closed simplex in $\mathbb{R}^n$, and look for a solution of the problem

$$\max_{H \in \mathcal{H}_{diag}} \ q^*(H) \tag{6}$$

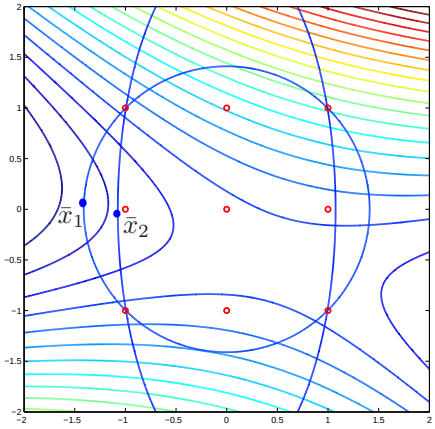

Figure 3: Different choices of the ellipsoid $E(H)$ give rise to different bounds.

where

$$q^*(H) := \min_{x \in \mathbb{R}^n} \{q(x) \mid x^\top H x = 1\} \ .$$

Note that $q^*(H) < \infty$ for all $H \in \mathcal{H}_{diag}$, but $q^*(H) = -\infty$ is possible if $H \not\succ 0$. However, as $\frac{1}{n} I \in \mathcal{H}_{diag}$, it follows that (6) is finite. Figure 3 illustrates how different choices of H may lead to different values of the resulting lower bound $q^*(H)$.

A straightforward feasible choice of the ellipsoid corresponds to $H = \frac{1}{n} I$. Indeed, this choice does not require any computation for solving Problem (6), but it may result in a weak bound. We analyse two other approaches: either choose H as the solution of a homogeneous version of Problem (6) or obtain H as an approximate solution of (6).

The homogeneous version of Problem (6) consists in ignoring the linear term of $q(x)$ in the inner problem, so that we consider the generalized pencil problem

$$\max_{H \in \mathcal{H}_{diag}} \lambda_{\min}(Q, H) = \max_{H \in \mathcal{H}_{diag}} \min_{x \in \mathbb{R}^n} \frac{x^\top Q x}{x^\top H x} \ , \tag{7}$$

where $\lambda_{\min}(Q, H)$ is the generalized smallest eigenvalue for the pencil (Q, H). This approach was originally proposed by Kamath and Karmarkar [21] for the special case of optimizing a quadratic form over $\{-1, 1\}^n$. They show the equivalence of Problem (7) with a semidefinite optimization problem

that, in the nonconvex case $Q \not\succeq 0$, can be easily written [31] as

$$\begin{array}{ll} \max\limits_{H,t} & t \\ & tQ + H \succeq 0, \\ & H \in \mathcal{H}_{diag}\ . \end{array} \tag{8}$$

Problem (8) satisfies the Slater condition and we can apply any interior point method to solve it, thus yielding an optimal solution of Problem (7).

The third possibility is to apply a subgradient-type (first order) method to the nonsmooth Problem (6); see e.g. [37, 1]. The objective function $q^*(H)$ of (6) is not continuously differentiable, but a generalized gradient is easily obtained; see [31]. The numerical experiments presented in Section 4 show that this choice yields significantly better bounds than the other ones.

2.2 Computation of the Lower Bounds

In this section, we assume that the ellipsoid $E(H)$ has been chosen by one of the methods presented in Section 2.1, so that $H \in \mathcal{H}_{diag}$ is fixed. We address the question of how to efficiently compute a solution of Problem (5).

Most of the algorithms proposed in the literature for finding a global solution of Problem (5) (and thus a lower bound in our approach) have been proposed in the context of trust-region methods for unconstrained nonlinear minimization [28, 38, 17, 13]. Indeed, in the nonlinear continuous optimization community, Problem (5) has been deeply studied with the main aim of defining efficient algorithms being able to treat large scale instances and exploit sparsity, thus avoiding expensive operations on the matrices [39, 32, 25, 22, 11, 16].

However, sparsity is not given in our context and instances are comparably small from a continuous optimization point of view. Indeed, in our case the dimension of the problems is usually below one hundred variables, as larger instances of Problem (1) cannot be solved in reasonable time due to integrality. On the other hand, within a branch-and-bound scheme, many problems sharing the same data must be solved. This different situation must be taken into account and exploited in order to solve the continuous problems quickly at each node.

We begin with a brief review of results about existence, characterization, and uniqueness of the optimal solution of Problem (5). Since the choice of H has been driven by the maximization of the bound, as explained in Section 2.1, we conclude that Problem (5), namely the inner problem in (6), admits a global solution.

We first note that the existence of a solution of Problem (5) implies that

$$w^T H w = 0,\ w \neq 0 \implies w^T Q w \geq 0\ .$$

The converse is not true in general, so that in the trust-region literature the stronger assumption

$$w^T H w = 0,\ w \neq 0 \implies w^T Q w > 0 \tag{9}$$

is made; see [27, 3] and comments therein. Indeed, under condition (9), a basic result on simultaneous diagonalization of Q and H can be applied [18]: there exists a non-singular matrix P such that both $P^\top Q P$ and $P^\top H P$ are diagonal. Hence we can apply the transformation $x \leftarrow Py$ and restate Problem (5) in diagonal form as

$$\begin{aligned} \min \quad & y^\top \Lambda y + \widetilde{L}^\top y\ , \\ \text{s.t.} \quad & y^\top \widetilde{H} y = 1 \end{aligned} \tag{10}$$

where $P^\top Q P = \Lambda$ and $P^\top H P = \widetilde{H}$ are diagonal matrices.

Next, let $J = \{i :\ \tilde{h}_i > 0\}$ and $\bar{J} = \{i :\ \tilde{h}_i = 0\}$. Since $\widetilde{H} \succeq 0$, we have $J \cup \bar{J} = \{1, \ldots, n\}$. We note that all variables y_i with $i \in \bar{J}$ are unconstrained and, thanks to diagonalization, the objective function is separable. Hence an optimal solution of Problem (10) exists if and only if $\lambda_i \geq 0$ and $2\lambda_i y_i + \widetilde{L}_i = 0$ for all $i \in \bar{J}$; see e.g. [4]. In this case, it can be obtained as the sum

$$z^* - \sum_{i \in \bar{J}} \frac{\widetilde{L}_i^2}{4\lambda_i}$$

where z^* is the optimal solution of the reduced problem

$$\begin{aligned} z^* := \min \quad & y_J^\top \Lambda_J y_J + \widetilde{L}_J^\top y_J \\ \text{s.t.} \quad & y_J^\top \widetilde{H}_J y_J = 1\ . \end{aligned}$$

Note that $\widetilde{H}_J \succ 0$ by construction. Hence this problem is of the form (5) with $\widetilde{H}$ positive definite. In this case, a further transformation can be applied, allowing to rewrite the problem as

$$\begin{aligned} z^* = \min \quad & y^\top \Lambda y + \widetilde{L}^\top y \\ \text{s.t.} \quad & ||y||^2 = 1\ . \end{aligned} \tag{11}$$

Note that the size of this problem is linear in the dimension n. An optimal solution $\bar{y}$ of problem (11) can be fully characterized [17, 38, 27]. Indeed,

a point $\bar{y} \in \mathbb{R}^n$ is a global solution if and only if a multiplier $\bar{\mu} \in \mathbb{R}$ exists such that the following conditions hold:

$$\begin{aligned} 2(\Lambda - \bar{\mu} I)\bar{y} &= -\widetilde{L} \\ \Lambda - \bar{\mu} I &\succeq 0 \\ \|\bar{y}\|^2 &= 1 \end{aligned}$$

It is worth noting that given any feasible solution y satisfying the stationarity condition $2(\Lambda - \mu I)y = -\widetilde{L}$, the corresponding multiplier μ is uniquely determined in closed form as [24]

$$\mu(y) = -\frac{1}{2}(2y^\top \Lambda y + \widetilde{L}^\top y). \tag{12}$$

On the other hand, given the optimal value $\bar{\mu}$, if $\Lambda - \bar{\mu} I$ is positive definite, then Problem (11) has a unique global solution that is obtained as

$$\bar{y} = -\frac{1}{2}(\Lambda - \bar{\mu} I)^{-1}\widetilde{L} \ .$$

In general, an optimal solution can be derived using

$$-\frac{1}{2}(\Lambda - \bar{\mu} I)^{\dagger} L \ ,$$

where $(\cdot)^{\dagger}$ denotes the Moore-Penrose generalized-inverse, which in the diagonal case is obtained by taking the reciprocal of each non-zero element on the diagonal, leaving the zeroes in place.

Most algorithms proposed in the literature look for an accurate primal solution $\bar{y}$ of Problem (11), as they have been developed in the context of trust-region methods for unconstrained nonlinear minimization. In particular, research has been devoted to the development of methods for large scale problems (up to thousands of variables) in which spectral decomposition is too heavy to be performed. We cite here the approach proposed by Lucidi and Palagi [24], which is based on an unconstrained reformulation of Problem (11) and on some properties of its first order stationary points. We will use this approach for obtaining an accurate global primal-dual solution $(\bar{y}, \bar{\mu})$ in the root node of our branch-and-bound scheme. The solution $\bar{y}$ is used to find a first integer solution heuristically while $\bar{\mu}$ is used for a warm start strategy as devised in Section 3.2.

However, although not explicitly stated in the first papers [38, 17], most of the algorithms are based on duality results. Indeed in [39, 32] different

dual programs have been proposed which are equivalent to the original primal one and which exhibit strong duality. A dual problem of (11) without duality gap is [39]

$$\begin{aligned} \phi^* = \max_{\mu} \quad & \phi(\mu) = \mu - \tfrac{1}{4}\widetilde{L}^\top(\Lambda - \mu I)^\dagger \widetilde{L} \\ & \Lambda - \mu I \succeq 0, \end{aligned} \tag{13}$$

where $\phi(\mu)$ is a concave function, so that Problem (11) can be considered an implicit convex program, as fully analysed in [3].

Considering the dual problem has an important side effect in our context. Indeed, we want to use the optimal value ϕ^* in a branch-and-bound framework as a lower bound to decide whether to prune a certain subtree or to explore it. Actually, the value $\phi(\mu)$ for any feasible dual solution μ represents a safe bound, so that we do not need to solve Problem (13) to a high degree of accuracy. The opposite is true when solving the primal problem: we would need to find a very good approximation of the global solution $\bar{y}$ to be able to use this as a lower bound.

Motivated by this, we resort to the solution of the dual problem (13) to obtain a computationally cheap lower bound. We specialized the Moré and Sorensen [28] algorithm to Problem (11). The diagonal form of the matrices Q and H allows to simplify the iterations of the algorithm significantly. In particular, both function and gradient evaluations can now be performed in $O(n)$ time. For technical details of the implementation we refer to Piacentini [31].

First we recall that methods for the solution of Problem (11) distinguish between two main cases: the *easy case* and the *hard case*, which can in turn be subdivided into two further sub-cases. For the sake of completeness, we summarize these possibilities in the following proposition. The original proof can be found in, e.g., [39, 32]. Here, ϕ' denotes the derivative of the scalar function $\phi(\mu)$. We assume that the elements of the diagonal matrix Λ are sorted in ascending order $\lambda_1 \leq \cdots \leq \lambda_n$.

Proposition 1 *Let* $\mathcal{J} = \left\{ i : \ \widetilde{L}_i \neq 0, \ \lambda_i = \lambda_1 \right\}$.

(i) (easy case) *If* $\mathcal{J} \neq \emptyset$, *then* $\bar{\mu} < \lambda_1$ *and* $\phi'(\bar{\mu}) = 0$.

(ii) (nearly hard case) *If* $\mathcal{J} = \emptyset$ *and* $\phi'(\lambda_1) < 0$, *then* $\bar{\mu} < \lambda_1$ *and* $\phi'(\bar{\mu}) = 0$.

(iii) (hard case) *If* $\mathcal{J} = \emptyset$ *and* $\phi'(\lambda_1) \geq 0$, *then* $\bar{\mu} = \lambda_1$.

Proof. Because $\bar{\mu}$ is optimal for (13), there exists $\bar{y}$ such that the optimality conditions

$$\begin{aligned} 2(\lambda_i - \bar{\mu})\bar{y}_i &= -\widetilde{L}_i \\ \bar{\mu} &\leq \lambda_1 \\ \|\bar{y}\|^2 &= 1 \end{aligned}$$

are satisfied. Consider case (i) with $\mathcal{J} \neq \emptyset$. Without loss of generality, we can assume $\widetilde{L}_1 \neq 0$. It is not hard to see that global optimality conditions cannot be satisfied with $\bar{\mu} = \lambda_1$, so that $\bar{\mu} < \lambda_1$. Moreover, since ϕ is strictly concave, $\bar{\mu}$ satisfies $\phi'(\bar{\mu}) = 0$. Now assume, on contrary, that $\mathcal{J} = \emptyset$. Since ϕ is strictly concave, if $\phi'(\lambda_1) < 0$, then $\phi(\mu) > \phi(\lambda_1)$ for any $\mu < \lambda_1$. It follows that $\bar{\mu} < \lambda_1$ and $\phi'(\bar{\mu}) = 0$, so that (ii) is proved. Otherwise, if $\phi'(\lambda_1) \geq 0$ then $\phi(\mu) < \phi(\lambda_1)$ for any $\mu < \lambda_1$. Hence $\bar{\mu} = \lambda_1$ and (iii) follows. □

Since we assume to deal with a diagonal matrix Λ, we can easily obtain the index set $\mathcal{J}$. If $\mathcal{J}$ is empty, we need to evaluate the first derivative of ϕ in the smallest eigenvalue λ_1 to decide whether we are in case (ii) or (iii). Hence the *hard case* does not represent at all a numerical difficulty, whereas in either case (ii) and (i) the main effort is finding the zero of the nonlinear function ϕ' over the open interval $(-\infty, \lambda_1)$. We observe that

$$0 = \phi'(\mu) \quad \Longleftrightarrow \quad 0 = \|y(\mu)\|^2 - 1 = \sum_{i=1}^{n} \frac{\widetilde{L}_i^2}{4(\lambda_i - \mu)^2} - 1 \; .$$

The Moré and Sorensen [28] method consists in finding the zero of the so-called secular equation

$$\varphi(\mu) = 1 - \frac{1}{\|y(\mu)\|} = 0$$

by means of a Newton method whose generic iteration is written as

$$\mu_N^{k+1} = \mu^k - \frac{\varphi(\mu^k)}{\varphi'(\mu^k)} \; .$$

A *safeguard* step is performed to check whether μ_N^{k+1} lies outside the region of interest $(-\infty, \lambda_1]$. If so, the value μ^{k+1} is forced to belong to a prefixed interval which is known to contain the optimal solution. For theoretical details we refer to Moré and Sorensen [28].

The choice of the initial value μ^0 is crucial for obtaining a good estimate of the solution in a few Newton steps. In our branch-and-bound framework, we will use a warm start strategy as described in Section 3.2.

3 Branch-and-Bound Algorithm

In order to solve Problem (1) to proven optimality, we embed the lower bounds obtained by the strategies illustrated in Section 2 into a branch-and-bound algorithm. In the following, we describe the main components of this algorithm: the enumeration strategy (Section 3.1), the preprocessing phase (Section 3.2), and the order in which variables are fixed (Section 3.3).

3.1 Enumeration Strategy

We always branch by fixing a variable x_j to one of the integer values within the domain $[l_j, u_j]$. By this, the number of subproblems is equal to $u_j - l_j + 1$. It follows that after each branching step, the resulting subproblem is an integer quadratic programming problem of type (1) again, with a dimension decreased by one.

Compared to standard branching given by a binary subdivision of the domain $\{l_j, \ldots, u_j\}$, we have to enumerate a much larger number of nodes when $u - l$ is large. In other words, our branching strategy is particularly well suited for the case of small variable domains. For our experimental results presented in Section 4 we only use ternary instances, i.e., instances with $l_i = -1$ and $u_i = 1$ for all $i = 1, \ldots, n$.

In fact, we also experimented with binary branching rules. The resulting subproblems are still of the form (1). However, for this type of branching it often happens that the lower bound in a subproblem is lower than the lower bound in the parent problem, no matter by which strategy we choose H, so that this approach is not well-suited for a branch-and-bound approach. For this reason, we stick to the branching by fixing as described above.

We use a depth-first strategy in order to find good feasible solutions quickly. For running time reasons, we do not use any primal heuristics in our implementation (except in the root node), instead we wait until all variables are fixed, which works very well in practice. Because of our branching by fixing strategy, deeper nodes of the tree correspond to problems of small dimension, so that the algorithm can dive down to the last level very quickly.

In order to enumerate subproblems as quickly as possible, our aim is to perform the most time consuming computations in a preprocessing phase. To this aim, following [9], we decide to always fix the first unfixed variable according to an order $\{i_1, \ldots, i_n\}$ which is determined before starting the enumeration; see Section 3.3. Without loss of generality, we assume here that the fixing order is $\{1, 2, \ldots, n\}$.

At a certain node s at level d of the branch-and-bound tree, the first d

components of the vector x have been already fixed to integer values $r_i^{(s,d)}$ with $i = 1, \dots, d$, so that the vector x can be rewritten as

$$(r_1^{(s,d)}, \dots, r_d^{(s,d)},\ x_{d+1}, \dots, x_n)\ .$$

Fixing the first d variables results in subproblems depending only on the variables $(x_{d+1}, \dots, x_n)$ and produces changes in the linear and constant terms of the quadratic objective function $q(x)$. More precisely, the reduced objective function $q^{(s,d)} : \mathbb{R}^{n-d} \to \mathbb{R}$ is of the form

$$q^{(s,d)}(x) = x^\top Q^{(d)} x + {L^{(s,d)}}^\top x + c^{(s,d)},$$

where $Q^{(d)}$ is obtained from Q by removing the first d rows and columns,

$$L_{j-d}^{(s,d)} = L_j + 2\sum_{i=1}^{d} q_{ij} r_i^{(s,d)}, \quad j = d+1, \dots, n, \tag{14}$$

and

$$c^{(s,d)} = \sum_{i=1}^{d} L_i r_i^{(s,d)} + \sum_{i=1}^{d}\sum_{j=1}^{d} q_{ij} r_i^{(s,d)} r_j^{(s,d)}. \tag{15}$$

We observe that the matrix $Q^{(d)}$ does not depend on the values at which the first d variables are fixed, i.e., it does not depend on the node s. If follows that all nodes at a given level d share the same quadratic term. As described in Section 2.2, the efficient solution of the relaxation needs some expensive operations on $Q^{(d)}$. Using a fixed order of variables implies that we have to perform these operations only on n different matrices $Q^{(0)}, \dots, Q^{(n-1)}$. See Section 3.2 for a detailed description of these operations. If the variables to be fixed were chosen freely, the number of such matrices would become exponential.

At each iteration of the branch-and-bound algorithm, we pick a subproblem given by $(Q^{(d)}, L^{(s,d)}, c^{(s,d)})$ and compute a lower bound using the strategies described in Section 2. We distinguish between nonconvex ($Q^{(d)} \not\succeq 0$) and convex ($Q^{(d)} \succeq 0$) subproblems. We note that, once an order of fixing is established, we know in advance at which level $\bar{d}$ the submatrix $Q^{(\bar{d})}$ becomes positive semidefinite. This implies that $Q^{(d)}$ is positive semidefinite for all $d \geq \bar{d}$, so that, starting at that level, the convex strategy for the calculation of the lower bounds is always used. We call $\bar{d}$ the *first level of convexity*. In Section 3.3, we explain how the fixing order can be defined in order to exploit this fact.

Once the lower bound is computed, either the node s may be pruned (if the lower bound exceeds the value of the best known feasible solution) or a bunch of subproblems are added to the list. The algorithm continues until the list of open subproblems is empty, so that the current best solution is proven to be optimal.

For the computation of upper bounds, the algorithm produces feasible solutions for Problem (1) by rounding the components of a vector $\hat{x}$, obtained in the lower bound computation, to the nearest integer in $[l_i, u_i]$. In a convex subproblem, we can simply choose $\hat{x} := x^*$, the continuous minimizer of $q(x)$. In a nonconvex subproblem, we determine $\hat{x}$ as an optimizer of the lower bounding problem (5): in the root node, this optimizer is calculated exactly as explained above, while in each other nonconvex subproblem we use our approximate solution $\bar{\mu}$ of the dual problem (13) and we compute $\hat{x}$ as a solution of the system

$$2(Q - \bar{\mu}H)\hat{x} = -L \; .$$

In the following, we give an outline of the overall algorithm.

Branch-and-Bound Scheme `GQIP` for Problem (1)

Data. $Q \in \mathbb{S}^n$, $L \in \mathbb{R}^n$, $l, u \in \mathbb{Z}^n$.

Fixing Order. Determine a variable order $x_1, \ldots, x_n$.

Preprocessing. Compute the matrix $Q^{(d)}$ for $d = 0, \ldots, n-1$ and perform all expensive operations on $Q^{(d)}$.

Initialization. $z_{ub} = \infty$, $x^* = ()$, $L^{(1,0)} = L$, $c^{(1,0)} = 0$, $r^{(1,0)} = ()$,

$$\mathcal{L} = \left\{ \left(Q^{(0)}, L^{(1,0)}, c^{(1,0)}, r^{(1,0)} \right) \right\}.$$

While $\mathcal{L} \neq \emptyset$

1. Pick some problem $\left(Q^{(d)}, L^{(s,d)}, c^{(s,d)}, r^{(s,d)}\right)$ with largest d in $\mathcal{L}$ where $L^{(s,d)}, c^{(s,d)}$ are computed by (14), (15).
2. Compute lower bound $z_{lb}^{(s,d)}$ and corresponding $\hat{x}^{(s,d)} \in \mathbb{R}^{n-d}$.
3. Update upper bound: let $x_I = (r^{(s,d)}, \lceil \hat{x}_1^{(s,d)} \rfloor, \ldots, \lceil \hat{x}_{n-d}^{(s,d)} \rfloor)$; if $q(x_I) < z_{ub}$, then $x^* = x_I$ and $z_{ub} = q(x^*)$.
4. If $z_{lb}^{(s,d)} < z_{ub}$ and $d < n$, then branch on variable x_{d+1} and add to $\mathcal{L}$ the corresponding $u_{d+1} - l_{d+1} + 1$ subproblems.

End While

Return. x^*, $z_{ub} = q(x^*)$.

3.2 Preprocessing

As discussed above, an important feature of our algorithm is the preparation of the lower bound computation in a preprocessing phase. Recall that the matrix $Q^{(d)}$ only depends on the depth d and hence can only take n different values, as we fix the order of branching variables in advance. In this section, we summarize the main components of the preprocessing.

Choice of $H^{(d)}$. In principle, the optimal ellipsoids or their approximations can be calculated in each node of the branching tree as described in

Section 2.1, thus taking the local shape of the feasible region into account. However, a fast computation of lower bounds at each node is crucial for the efficiency of the overall algorithm. We thus propose to compute an optimal (or near-optimal) ellipsoid $E(H^{(0)})$ only once at the root node, using one of the methods proposed in Section 2.1. In order to avoid numerical problems, in the implementation we restrict ourselves to positive definite matrices $H^{(0)}$ in the closed simplex

$$\mathcal{H}_{diag}(\epsilon) = \left\{ H \in \mathcal{H}_{diag} \mid \ h_i \geq \epsilon \text{ for all } i = 1, \ldots, n \right\}$$

for small $\epsilon > 0$. At each other node, we derive $H^{(d)} \succ 0$ by deleting appropriate rows and columns in $H^{(0)}$. This decreases both preprocessing times and running times per node without yielding significantly weaker bounds.

Computation of $\Lambda^{(d)}$ and $P^{(d)}$ Once the matrices $H^{(d)} \succ 0$ are fixed, the most important tasks in the preprocessing is the computation of $(\Lambda^{(d)}, P^{(d)})$ as needed in the transformed problem (11). Since we assume that $H^{(d)}$ is positive definite, the matrix $P^{(d)}$ is obtained by first applying the simple linear transformation $y = H^{(d)\frac{1}{2}} x$ to the variable space and then calculating the spectral decomposition of the matrix $H^{(d)^{-\frac{1}{2}}} Q^{(d)} H^{(d)^{-\frac{1}{2}}}$ as

$$H^{(d)^{-\frac{1}{2}}} Q^{(d)} H^{(d)^{-\frac{1}{2}}} = U^{(d)} \, \Lambda^{(d)} \, U^{(d)^\top} \ ,$$

thus obtaining $\Lambda^{(d)}$ and $P^{(d)} = H^{(d)^{-\frac{1}{2}}} U^{(d)}$ for $d = 0, \ldots, n-1$.

Initial solutions for warm starts. For each subproblem, we need a starting value μ^0 for the dual algorithm described in Section 2.2. In the preprocessing, we compute for each $d = 1, \ldots, n-1$ an optimizer $y^{(d)}$ of

$$\begin{array}{ll} \min & y^\top \Lambda^{(d)} y + L^{(d)^\top} P^{(d)} y \\ \text{s.t.} & ||y||^2 = 1 \ , \end{array}$$

where $L^{(d)} \in \mathbb{R}^{n-d}$ is obtained from L by dropping the first d components. By (12), we obtain a feasible solution of the corresponding dual problem in a given node s as

$$\mu(y^{(d)}) = -\frac{1}{2} \left(2 y^{(d)^\top} \Lambda^{(d)} y^{(d)} + L^{(s,d)^\top} P^{(d)} y^{(d)} \right) \ .$$

Computation of stationary points. In each convex subproblem, we need the stationary point of $q(x)$ and the corresponding function value in order to compute a lower bound. For this, we use the incremental approach proposed in [9]. This approach remains feasible in the nonconvex case without changes. Using an appropriate preprocessing, the update of the stationary point takes only linear time in n per node.

3.3 Reordering of the Variables

As shown by the experimental results in Section 4, the performance of our algorithm depends strongly on permutations of the matrix Q. Let Π denote a permutation matrix representing an ordering of the n dimensions of the problem and let $\Pi^\top Q \Pi$ be the resulting reordered matrix. Our aim is to find a matrix Π such that the resulting first level of convexity $\bar{d}$ is as small as possible, or, equivalently, to find a small number of indices such that removing the corresponding rows and columns in Q results in a positive semidefinite matrix. This allows to switch from the nonconvex to the convex case in the branch-and-bound scheme as soon as possible.

Indeed, an early appearance of convex nodes makes the overall scheme much more efficient, as lower bounds tend to be tighter in this case and we can apply more sophisticated pruning rules as presented in [9]. In particular, we fix variables to values in increasing distance from the global minimizer. As the lower bound obtained after fixing a given variable is a convex quadratic function in the value to which the variable is fixed, we can prune all siblings immediately if the global minimizer reaches the current upper bound.

We propose several heuristics for choosing the fixing order based on different indicators of convexity. The rules we propose consist in defining a vector $m \in \mathbb{R}^n$, which is roughly speaking an indicator of positive semidefiniteness, and in selecting the order of the variables by increasing values of the components m_i. In particular, we examine the following rules:

- ordering by the diagonal elements of the matrix, i.e.,

$$m_i = q_{ii}, \quad i = 1, \ldots, n.$$

- use of a partial Cholesky factorization based rule as in [15], so that

$$m_i = q_{ii} - \max_{j \neq i} |q_{ij}|, \quad i = 1, \ldots, n.$$

- ordering based on diagonal dominance, i.e.,

$$m_i = q_{ii} - \sum_{j \neq i} |q_{ij}|, \quad i = 1, \ldots, n.$$

These rules are compared from a computational point of view in the following section. Of course, these choices do not cover all possibile reordering rules; more sophisticated indicators might exists.

4 Experimental Results

In this section, we report computational results obtained with the branch-and-bound scheme `GQIP` presented in Section 3. Our Algorithm is implemented in C++ and Fortran 90. All numerical experiments have been performed on an Intel Core2 processor running at 3.16 GHz with 4GB of RAM.

We restrict ourselves to ternary quadratic instances, i.e., instances where variables x_i are constrained to assume values in $\{-1, 0, 1\}$ for all $i = 1, \ldots, n$. For our experiments, we consider objective functions $q : \mathbb{R}^n \to \mathbb{R}$ generated randomly as in [8]: given a percentage $p \in [0, 1]$, we choose n eigenvalues, where $\lfloor p \cdot n \rfloor$ are chosen uniformly at random from $[-1, 0]$ and the remaining ones are chosen uniformly at random from $[0, 1]$. In particular, the parameter p allows to control whether the matrix Q is positive semidefinite ($p = 0$), negative semidefinite ($p = 1$) or indefinite. The matrix Q is obtained by combining these eigenvalues with a random orthonormal basis of $\mathbb{R}^n$. Finally, we determine L by choosing all entries uniformly at random from $[-1, 1]$. As in [8], we created ten random instances from different random seeds for each $n \in \{10, 20, \ldots, 50\}$ and each $p \in \{0, 0.1, \ldots, 1\}$. We thus consider 110 instances in total for each size n. Instances with $n = 10$ were all solved within a second by all considered approaches and are thus not reported in the following. In all experiments, we set the time limit to one hour; instances not solved to proven optimality (with accuracy less than 10^{-6}) within the time limit are considered a failure.

The behaviour of the algorithm strongly depends on two main decisions: the rule applied for reordering variables and the choice of the ellipsoid; these two features also interact with each other. We considered three different choices for H, following the lines of Section 2.1,

- $H_1 = \frac{1}{n} \cdot I$
- H_2 obtained by solving Problem (7), using CSDP [7]

- H_3 chosen by approximately solving Problem (6) using a subgradient method which exploits the diagonal form of H [31]

and four different choices for the ordering, following the lines of Section 3.3,

- O_1 no reordering
- O_2 diagonal elements
- O_3 Partial Cholesky
- O_4 diagonal dominance rule.

We compare the different versions of our code corresponding to these choices, named $\texttt{GQIP}_{H_iO_j}$ for $i = 1, 2, 3$ and $j = 1, 2, 3, 4$, on the basis of total running time. Rather than reporting detailed tables, we use a *performance profile*, as proposed in [14]: given a set of solvers S and a set of problems P, we compare the performance of a solver $s \in S$ on problem $p \in P$ against the best performance obtained by any solver in S on the same problem. To this end we define the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s'} : s' \in S\}} ,$$

where $t_{p,s}$ is the computational time, and we consider a cumulative distribution function

$$\rho_s(\tau) = \frac{1}{|P|} \left|\{p \in P : r_{p,s} \leq \tau\}\right| .$$

Then the performance profile for $s \in S$ is a plot of the function ρ_s.

From Figure 4 it is obvious that the best version corresponds to the choice H_3O_4. In order to examine the impact of the choice of H_i once the ordering O_4 is fixed, or the impact of the ordering O_j once the method H_3 is chosen, we next report separate results. In all following tables, we state average results for each dimension n. In the columns of the tables we list the number of instances solved to proven optimality within the time limit of one hour, out of all 110 instances (SOLVED); the maximum time spent to solve a problem (MAX TIME); the average over the successfully solved instances of: the time spent in the preprocessing phase (AVG PREP), the overall time spent to solve the instance (AVG TIME), and the number of nodes (AVG # NODES). All running times are given in CPU-seconds.

Table 1 compares $\texttt{GQIP}_{H_iO_4}$ for the different choices of H_i, $i = 1, 2, 3$. It turns out that the three choices of H_i can be considered equivalent in terms of robustness and computational time for dimensions up to 30 or 40. However,

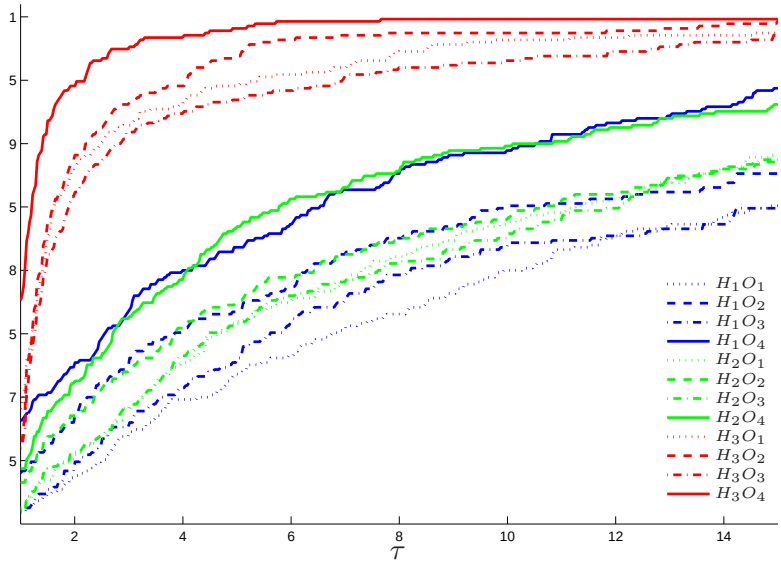

Figure 4: Performance profiles for CPU time of $\texttt{GQIP}_{H_iO_j}$.

the average number of nodes of the branching tree is significantly smaller with H_3. This is more evident for dimension 50 where the other two versions $\texttt{GQIP}_{H_1O_4}$ and $\texttt{GQIP}_{H_2O_4}$ need a much longer running time than $\texttt{GQIP}_{H_3O_4}$; one of the instances could only be solved by $\texttt{GQIP}_{H_3O_4}$ within the time limit. It can also be noticed that, for higher dimension, the preprocessing time is negligible with respect to the overall running time.

We next turn to highlight the role of the reordering of variables. We first examine the detected level of convexity of the problems when using the four different ordering rules. We recall that problems where randomly generated with $p \in \{0,\ 0.1,\ \ldots,\ 0.9,\ 1\}$, so that $p{\cdot}n$ eigenvalues are negative. We denote by $\bar{d}_{O_j}$ the smallest d such that $Q^{(d)}$ is a positive semidefinite submatrix of Q, depending on the reordering rule O_j being applied.

For each dimension n and for each ordering O_j, we calculate the value of $\widetilde{p} = 1 - \bar{d}_{O_j}/n$ on each of the 110 instances. The idea is to evaluate the difference $\widetilde{p} - p$. For each $p \in \{0,\ 0.1,\ \ldots,\ 0.9,\ 1\}$, we consider the average value of $\widetilde{p}$ over the corresponding 10 instances. To have a quick view, we analyse only the case $n = 50$; the behaviour for $n = 20, 30, 40$ is similar. In Figure 5 we plot the values of $\widetilde{p}$ for all orderings O_i and, as a reference value, also the curve $\widetilde{p} = p$. For none of the rules, the value of $\widetilde{p}$ agrees with

n	H	SOLVED	MAX TIME	AVG PREP	AVG TIME	AVG # NODES
20	H_1	110	0.1	0.0	0.0	3 177.9
	H_2	110	0.1	0.0	0.0	4 788.1
	H_3	110	0.6	0.1	0.1	1 914.5
30	H_1	110	1.1	0.0	0.1	73 329.5
	H_2	110	3.5	0.0	0.3	185 961.8
	H_3	110	1.1	0.4	0.5	16 435.7
40	H_1	110	54.6	0.0	5.8	2 714 442.8
	H_2	110	54.2	0.1	6.4	2 938 746.0
	H_3	110	10.0	1.0	1.8	307 492.7
50	H_1	109	1 470.4	0.1	185.1	70 275 080.0
	H_2	109	1 591.8	0.2	207.5	70 805 601.5
	H_3	110	306.7	2.1	31.6	8 614 806.8

Table 1: Results on random ternary instances with different choice of H_i but fixed ordering rule O_4.

or even comes close to the value of p. The plots show that, in terms of $\widetilde{p}$, none of the rules dominates all other rules, but all yield better results when compared to the original random ordering. However, the detailed results reported in Table 2 show that using diagonal dominance ordering O_4 yields much better results in terms of overall running time. This is due to a much smaller number of nodes to be enumerated on average. All instances could be solved to optimality within the time limit by all ordering rules.

In summary, we obtain the best results with the version H_3O_4 of our code, which we simply refer to as `GQIP` in the remainder of this section. We next compare `GQIP` with the following other solvers:

- `COUENNE` [2]: an algorithm for solving nonconvex mixed-integer nonlinear programs, based on convex underestimators;
- `BARON` [34]: a general purpose solver for optimization problems with nonlinear constraints;
- `Q-MIST` [8]: a branch-and-bound algorithm based on SDP relaxations for mixed integer quadratic programming.

We use `COUENNE` and `BARON` under `GAMS` [33] for Windows, while running `Q-MIST` and `GQIP` under Linux on the same machine. For the two latter codes, we use an absolute optimality tolerance of 10^{-6}. We use as test bed the same random instances as before. In Table 3, we report the results grouped by dimension n.

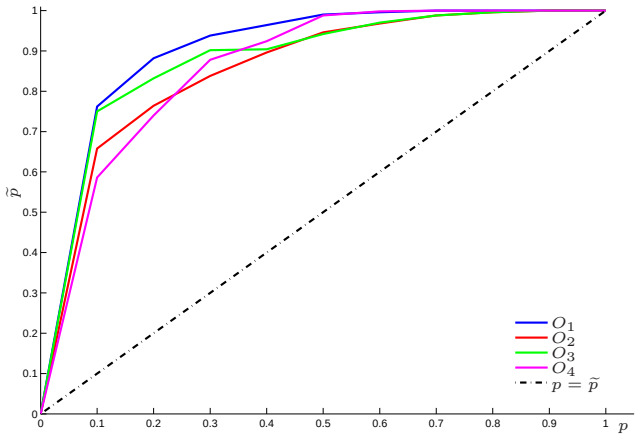

Figure 5: Comparison of average $\widetilde{p}$ by p for $\texttt{GQIP}_{O_j}$, $j = 1, 2, 3, 4$ ($n = 50$).

We see that BARON and COUENNE are able to solve, within the time limit, only instances of dimensions up to 40, while even some instances with dimension 30 could not be solved to optimality. This is due to the fact that they are general purpose solvers that cannot exploit the particular structure of Problem (1) such as Q-MIST and GQIP. For this reason, neither BARON nor COUENNE is competitive in terms of both robustness and computational time for this class of problems.

In order to analyse the performance of GQIP against Q-MIST in more detail, we investigate its dependance on the percentage of negative eigenvalues p. In Table 4, for each dimension $n \in \{20, 30, 40, 50\}$, we report the average of the computational time over the 10 random instances solved for each $p \in \{0, 0.1, \ldots, 1\}$. In Figure 6, we plot the corresponding running times on a logarithmic scale; Figure 7 shows the results for $n = 50$ on a linear scale. It turns out that GQIP outperforms Q-MIST for every percentage p, however, the main improvement (in absolute terms) is obtained when p lies between 20% and 40%. On the other hand, Q-MIST enumerates a significantly smaller number of nodes, but at the cost of a much longer running time for computing lower bounds. Finally, Figure 8 shows a performance profile comparing GQIP and Q-MIST.

n	O	MAX TIME	AVG TIME	AVG # NODES
20	O_1	0.3	0.1	2 078.7
	O_2	0.3	0.1	1 979.7
	O_3	0.5	0.1	1 809.6
	O_4	0.6	0.1	1 914.5
30	O_1	1.1	0.5	34 876.7
	O_2	1.4	0.5	22 844.4
	O_3	1.1	0.5	31 168.3
	O_4	1.1	0.5	16 435.7
40	O_1	12.9	2.3	524 312.4
	O_2	10.2	2.1	423 184.1
	O_3	22.3	2.6	658 593.4
	O_4	10.0	1.8	307 492.7
50	O_1	1 531.1	111.9	35 764 080.6
	O_2	1 011.5	67.2	21 158 615.6
	O_3	923.1	103.0	32 761 223.9
	O_4	306.7	31.6	8 614 806.8

Table 2: Results on ternary instances when using different ordering rules with fixed choice of ellipsoid H_3.

5 Conclusion

We devise a branch-and-bound algorithm for quadratic integer optimization problems with box constraints. Building on well-known techniques developed in the context of trust-region algorithms for computing lower bounds, we modify and adapt these methods to the integer optimization setting. In particular, the branch-and-bound framework requires to solve a huge number of subproblems sharing a significant portion of the problem data, whereas the size of a single problem is small from a continuous optimization point of view. In fact, the largest instances we can solve by our approach in reasonable time have 50 variables; the number of subproblems enumerated for these instances is in the order of 10^7. It is thus crucial to reengineer the given methods for lower bound computation, the main ideas being an approximate solution of the dual problem instead of the primal one and the movement of as many calculations as possible into a preprocessing phase. We are convinced that similar ideas can be applied successfully to many other classes of nonlinear integer optimization problems.

n	ALG	SOLVED	MAX TIME	AVG TIME	AVG # NODES
20	COUENNE	110	51.4	13.0	3 822.0
	BARON	110	477.1	24.1	1 548.8
	Q-MIST	110	1.0	0.2	53.5
	GQIP	110	0.6	0.1	1 914.5
30	COUENNE	78	3 567.3	1 476.7	181 127.6
	BARON	82	3 552.1	1 173.9	36 218.1
	Q-MIST	110	11.0	2.0	199.7
	GQIP	110	1.1	0.5	16 435.7
40	COUENNE	2	3 148.3	2 012.3	99 500.0
	BARON	4	2 233.8	1 394.4	19 411.3
	Q-MIST	110	105.0	15.9	831.6
	GQIP	110	10.0	1.8	307 492.7
50	COUENNE	0	***	***	***
	BARON	0	***	***	***
	Q-MIST	110	1 573.0	184.0	5 463.7
	GQIP	110	306.7	31.6	8 614 806.8

Table 3: Comparitive results for ternary instances for different solvers.

n	ALG	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
20	GQIP	0.20	0.17	0.10	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09
	Q-MIST	0.30	0.20	0.30	0.30	0.40	0.30	0.20	0.20	0.20	0.10	0.10
30	GQIP	0.53	0.46	0.44	0.42	0.41	0.40	0.39	0.43	0.36	0.38	0.36
	Q-MIST	1.70	1.70	4.30	3.30	3.00	2.20	1.60	1.20	1.30	1.20	0.70
40	GQIP	1.35	1.13	2.12	2.22	3.04	2.20	1.81	1.57	1.17	1.08	1.03
	Q-MIST	11.00	18.30	36.80	33.00	26.60	16.80	9.30	8.60	6.90	4.80	4.70
50	GQIP	4.35	15.56	45.19	59.89	61.22	63.21	43.46	10.62	6.40	4.82	2.63
	Q-MIST	57.30	273.80	431.70	423.50	403.10	249.90	84.40	44.00	28.60	28.40	21.80

Table 4: Detailed comparison between GQIP and Q-MIST: average running times at different percentages of negative eigenvalues.

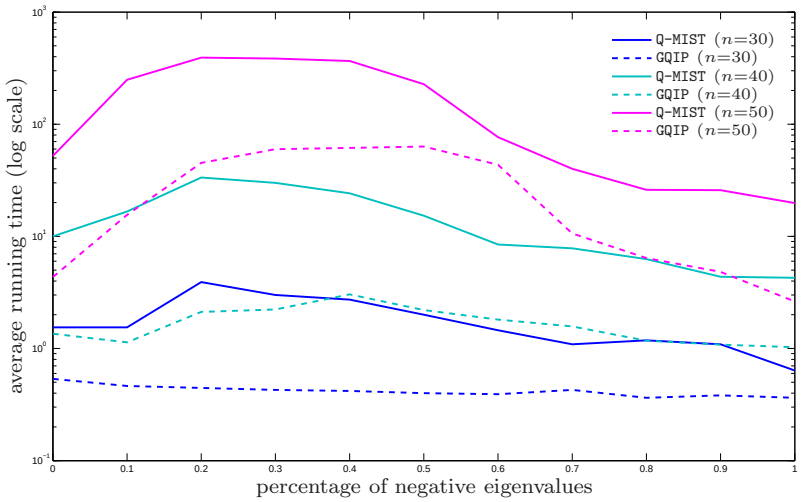

Figure 6: GQIP versus Q-MIST: time comparison for $n = 30, 40, 50$.

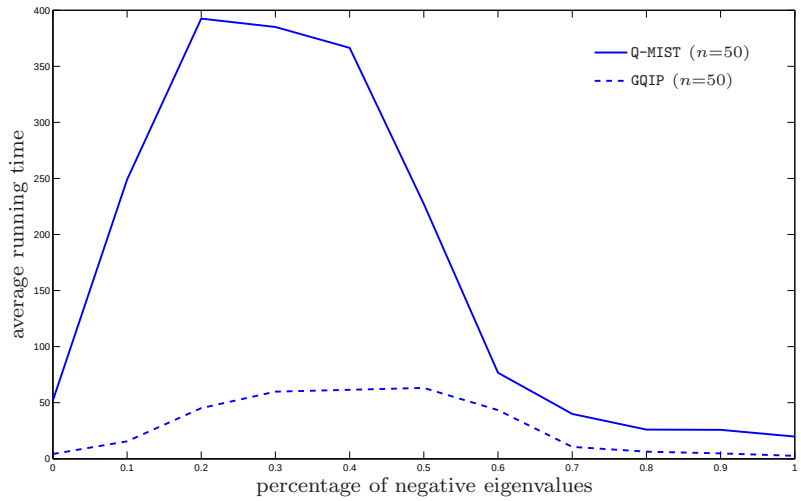

Figure 7: GQIP versus Q-MIST: time comparison for $n = 50$.

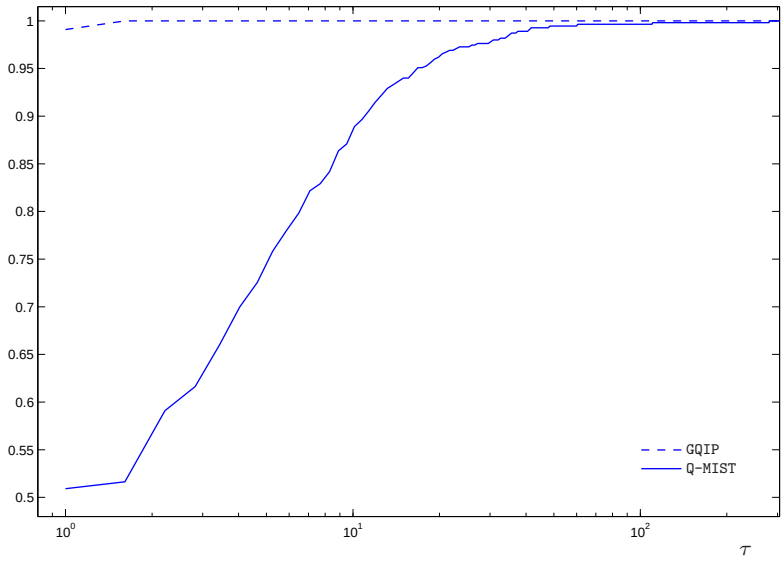

Figure 8: `GQIP` versus `Q-MIST`: CPU time performance profile for all n.

References

[1] Ya. I. Alber, A. N. Iusem, and M. V. Solodov. On the projected subgradient method for nonsmooth convex optimization in a Hilbert space. *Mathematical Programming*, 81(1):23–35, 1998.

[2] P. Belotti. *Couenne: a users manual*. Lehigh University, 2009. Technical report.

[3] A. Ben-tal and M. Teboulle. Hidden convexity in some nonconvex quadratically constrained quadratic programming. *Mathematical Programming*, 72:51–63, 1996.

[4] D. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1982.

[5] D. Bienstock. Eigenvalue techniques for convex objective, nonconvex optimization problems. In F. Eisenbrand and F. B. Shepherd, editors, *IPCO 2010*, pages 29–42, 2010.

[6] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5:186–2004, 2008.

[7] B. Borchers. CSDP, a C library for semidefinite programming. *Optimization Methods and Software*, 11(1):613–623, 1999.

[8] C. Buchheim and A. Wiegele. Semidefinite relaxations for non-convex quadratic mixed-integer programming. *Mathematical Programming*, 2012. To appear.

[9] C. Buchheim, A. Caprara, and A. Lodi. An effective branch-and-bound algorithm for convex quadratic integer programming. *Mathematical Programming*, 135(1–2):369–395, 2012.

[10] S. Burer. Optimizing a polyhedral-semidefinite relaxation of completely positive programs. *Mathematical Programming Computation*, 2(1):1–19, 2010.

[11] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-region methods*. SIAM/MPS Series on Optimization, SIAM, 2000.

[12] P. L. De Angelis, I. M. Bomze, and G. Toraldo. Ellipsoidal approach to box-constrained quadratic problems. *Journal of Global Optimization*, 28:1–15, 2004.

[13] R. S. Dembo and T. Steihaug. Truncated-Newton methods algorithms for large-scale unconstrained optimization. *Mathematical Programming*, 26:190–212, 1983.

[14] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[15] A. Forsgren, P. Gill, and W. Murray. Computing modified Newton directions using a partial Cholesky factorization. *SIAM Journal on Scientific Computing*, 16(1):139–150, 1995.

[16] C. Fortin and H. Wolkowicz. The trust region subproblem and semidefinite programming. *Optimization Methods and Software*, 19(1):41–67, 2004.

[17] D. M. Gay. Computing optimal locally constrained steps. *SIAM Journal on Scientific and Statistical Computing*, 2(2):186–197, 1981.

[18] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[19] ILOG, Inc. ILOG CPLEX 12.1, 2009. `www.ilog.com/products/cplex`.

[20] A. Kamath and N. Karmarkar. A continuous approach to compute upper bounds in quadratic maximization problems with integer constraints. In C. A. Floudas and P. M. Pardalos, editors, *Recent Advances in Global Optimization*, pages 125–140, Princeton University, 1991. Princeton University Press.

[21] A. Kamath and N. Karmarkar. A continuous method for computing bounds in integer quadratic optimization problems. *Journal of Global Optimization*, 2(3):229–241, 1992.

[22] An Le Thi Hoai and Tao Pham Dinh. A d.c. optimization algorithm for solving the trust-region subproblem. *SIAM Journal on Optimization*, 8, 1998.

[23] An Le Thi Hoai and Tao Pham Dinh. A branch and bound method via d.c. optimization algorithms and ellipsoidal technique for box constrained nonconvex quadratic problems. *Journal of Global Optimization*, 13:171–206, 1998.

[24] S. Lucidi and L. Palagi. *Topics in Semidefinite and Interior-Point methods*, volume 18 of *Field Institute Communications AMS*, chapter Solution of the trust region problem via a smooth unconstrained reformulation, pages 237–250. American Mathematical Society, 1998.

[25] S. Lucidi, L. Palagi, and M. Roma. On some properties of quadratic programs with a convex quadratic constraint. *SIAM Journal on Optimization*, 8(1):105–122, 1998.

[26] G. P. McCormick. Computability of global solutions to factorable nonconvex programs. I. Convex underestimating problems. *Mathematical Programming*, 10(2):147–175, 1976.

[27] J. Moré. Generalization of the trust region problem. *Optimization Methods and Software*, 2:189–209, 1993.

[28] J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983.

[29] L. Palagi, V. Piccialli, F. Rendl, G. Rinaldi, and A. Wiegele. *Handbook on Semidefinite, Conic and Polynomial Optimization*, chapter Computational approaches to Max-Cut, pages 821–849. Springer, 2012.

[30] P. M. Pardalos and S. A. Vavasis. Quadratic programming with one negative eigenvalue is NP-hard. *Journal of Global Optimization*, 1:15–22, 1991.

[31] M. Piacentini. *Nonlinear formulation of Semidefinite Programming and Eigenvalue Optimization – Application to Integer Quadratic Problems.* PhD thesis, Sapienza – Università di Roma, June 2012.

[32] F. Rendl and H. Wolkowicz. A semidefinite framework to trust region subproblems with applications to large scale minimization. *Mathematical Programming*, 77(1):273–299, 1997.

[33] R. E. Rosenthal. *GAMS – A User's Guide.* GAMS Development Corporation, Washington, DC, USA, 2012. A tutorial.

[34] N. V. Sahinidis and M. Tawarmalani. *BARON 9.0.4: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2010.

[35] A. Saxena, P. Bonami, and J. Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: extended formulations. *Mathematical Programming*, 124(1–2):383–411, 2010.

[36] H. D. Sherali and W. P. Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, volume 31 of *Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, Dordrecht, 1999.

[37] N. Z. Shor, K. C. Kiwiel, and A. Ruszczyński. *Minimization Methods for Non-differentiable Functions*. Springer-Verlag, 1985.

[38] D. C. Sorensen. Newton's method with a model trust region modification. *SIAM Journal on Numerical Analysis*, 19(2):409–427, 1982.

[39] R. J. Stern and H. Wolkowicz. Indefinite trust region subproblems and nonsymmetric eigenvalue perturbations. *SIAM Journal on Optimization*, 5(2):286–313, 1995.

[40] D. Vandenbussche and G. L. Nemhauser. A branch-and-cut algorithm for nonconvex quadratic programs with box constraints. *Mathematical Programming*, 102(3):559–575, 2005.

[41] S. A. Vavasis. *Nonlinear optimization*. Oxford University Press, 1991.

[42] Y. Ye. A new complexity result on minimization of a quadratic function with a sphere constraint. In C. A. Floudas and P. M. Pardalos, editors, *Recent Advances in Global Optimization*, pages 19 – 31, Princeton University, 1991. Princeton University Press.