Dipartimento di Informatica
e Sistemistica Antonio Ruberti

Sapienza
Università di Roma

An Exact Algorithm for Quadratic Integer Minimization using Nonconvex Relaxations

Christoph Buchheim
Marianna De Santis
Laura Palagi
Mauro Piacentini

Technical Report n. 5, 2012

An Exact Algorithm for Quadratic Integer Minimization using Nonconvex Relaxations [*]

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

May 23, 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. 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.

[*]This work was partially supported by the Vigoni Project 2010 "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`

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 be solved (in theory and in practice) by the presence of two types of nonconvexity: first, the objective function or constraints can be nonconvex, and second, the presence of integer variables makes the domains of the variables nonconvex.

We address quadratic integer optimization problems with box constraints,

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

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) has been proposed by Buchheim et al. [7]. 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 bound is given as the minimum of the objective function over the boundary of this ellipsoid, 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,

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

is an *NP*-hard problem in the case $Q \not\succeq 0$; it is equivalent to the so-called BoxQP problem [24].

In this paper, we present a novel approach for computing lower bounds in the nonconvex case, which tries to exploit the same interesting features used by Buchheim et al. [7] in the convex case. The main ingredient in the algorithm is the definition and solution of an appropriate continuous relaxation of Problem (1). We embedded this relaxation into a branch-and-bound framework in order to obtain an exact algorithm for general integer quadratic optimization problems. Experiments with various types of ternary instances show that this approach is competitive with other methods for integer quadratic optimization, or even yields significantly faster running times for larger instances.

Different from [7], we choose an ellipsoid E that contains all feasible solutions of Problem (1),

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

where $H \succ 0$ 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 idea goes back to Kamath and Karmarkar [15, 16]. 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 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 positive definite 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), strong Lagrangian duality holds; see e.g. [31, 26]. Hence we can also use the dual formulation to obtain a 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, while we use the dual formulation at each node of the branching tree. By the integrality constraints, instances with 50–100 variables can already be considered very challenging, different from a pure continuous optimization context. For solving the primal problem, we can thus use the approach of Lucidi and Palagi [19], while for the dual formulation we can use the algorithm of Moré and Sorensen [22].

As in [7], 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 variables fixed only depends on the depth of the node in the 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 matrices 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. In Section 3, we propose different strategies for choosing these ellipsoids. The main components of the overall branch-and-bound algorithm are discussed in Section 4. Finally, Section 5 contains the results of an experimental evaluation of this algorithm.

1.1 Related Work

Most software developed for integer nonlinear optimization can guarantee global optimality of the computed solutions only if the problem is convex [3, 14, 7]. 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 [6] propose a branch-and-bound approach based on 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 outperforms other software such as Couenne [1]. Another approach for mixed-integer quadratic optimization was recently devised by Saxena et al. [28]. 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, (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 [23] 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 [6] 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 [21, 29]. Fast algorithms for BoxQP based on semidefinite programming have been devised by Vandenbussche and Nemhauser [32] and by Burer [8].

All approaches for quadratic optimization discussed above share two features: the original problem (1) is first linearized by adding one new variable 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 to linearize 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 [16], 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 [18] exploit this idea within a branch-and-bound algorithm using rectangular partitions for solving BoxQP problems. After a suitable transformation, 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 nonconvex continuous optimization problems, while most approaches in the literature are based on convex relaxations. This is possible since our nonconvex relaxation can still be solved to global optimality efficiently.

Use of dual problems for bound computation. We solve the dual formulation of our nonconvex continuous optimization problems instead of the primal one. This allows us to use approximate solutions, which still yield valid bounds. 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 the pruning of the node.

Acceleration by preprocessing. For solving the continuous problems more efficiently, we compute a spectral decomposition of the respective matrices in the 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.

Reordering of input matrix. We try to reorder the variables of Problem (1) such that a convex problem is reached as soon as possible, assuming that variables are fixed in the resulting order. In the convex case, bounds tend to be tighter and nodes can be pruned more efficiently, as done in [7].

2 Lower Bounds from Axis-parallel Ellipsoids

The main ingredient of our algorithm are lower bounds obtained by minimizing the objective function $q(x)$ of Problem (1) over appropriate ellipsoids. In the following, we describe the general idea of the bound computation. The choice of the ellipsoid is discussed in Section 3, the details of the computation are reported in Section 4.3. We first consider the nonconvex case; the convex case has to be handled differently, see Section 2.2 below.

2.1 Nonconvex Case

We are interested in finding a lower bound for Problem (1) that is efficiently computable when the matrix Q is not positive semidefinite. To this end, we first relax the integrality constraint to get

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

Since integrality is not required here, we can scale the problem and hence assume that $l_j = -1$ and $u_j = 1$ for all $j = 1, \dots, n$. The resulting relaxation is still an *NP*-hard problem in the nonconvex case, so that we resort to the continuous relaxation (2) where we restrict ourselves to ellipsoids that are centered in the origin.

Hence we consider ellipsoids $E(H)$ with

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

such that H is a positive definite matrix and, by scaling again, the radius r

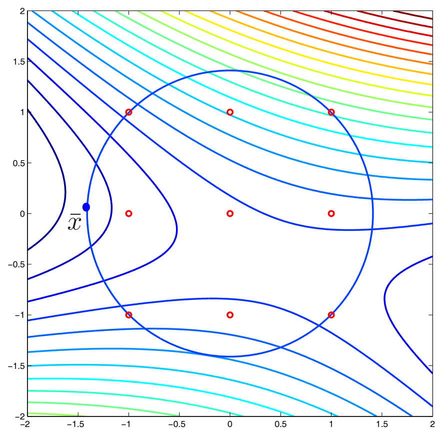

Figure 1: Computation of lower bounds in the nonconvex case.

can always be fixed to one, so that the relaxed problem becomes

$$\begin{array}{ll} \min & q(x) = x^\top Q x + L^\top x \\ \text{s.t.} & x^\top H x \le 1\ . \end{array} \tag{4}$$

Furthermore, since the objective function $q(x)$ is assumed to be nonconvex, the optimal solution of the problem above lies on the boundary of the ellipsoid $E(H)$. Indeed, the second order necessary optimality condition in the interior of the convex set $E(H)$, $\nabla^2 q(x^*) = 2Q \succeq 0$, cannot be satisfied. Hence we can restrict ourselves to studying the nonconvex problem

$$\begin{array}{ll} z^* = \min & q(x) = x^\top Q x + L^\top x \\ \text{s.t.} & x^\top H x = 1\ . \end{array} \tag{5}$$

See Figure 1 for an illustration of the case of indefinite $q(x)$ with a spherical constraint.

Despite its nonconvexity, it is well known that an optimal solution $\bar{x}$ of problem (5) can be fully characterized by the following conditions [12, 30]:

$$\begin{array}{rcl} 2(Q - \bar{\mu} H)\bar{x} & = & -L \\ Q - \bar{\mu} H & \succeq & 0 \\ \bar{x}^\top H \bar{x} & = & 1\ . \end{array} \tag{6}$$

The optimal multiplier $\bar{\mu}$ is uniquely determined in closed form as [19]

$$\mu(\bar{x}) = -\frac{1}{2}(2\bar{x}^\top Q\bar{x} + L^\top \bar{x}). \tag{7}$$

Furthermore, if $Q - \bar{\mu}H$ is positive definite, then Problem (5) has a unique global solution that is obtained as $\bar{x} = -\frac{1}{2}(Q - \bar{\mu}H)^{-1}L$. In general, an optimal solution is obtained as $\bar{x} = -\frac{1}{2}(Q - \bar{\mu}H)^\dagger L$, where $(\cdot)^\dagger$ denotes the Moore-Penrose generalized-inverse. It has also been proved that an approximation to the global solution z^* can be computed in polynomial time; see e.g. [2, 33, 34]. Hence Problem (5) can be considered an "easy" problem from a theoretical point of view. Within our branch-and-bound context, it is crucial to solve this problem very quickly using appropriate algorithms. This is explained in detail in Section 4.3.

Another important question is how to choose the matrix H in order to obtain tight bounds. We address this question in Section 3.

2.2 Convex Case

In the following, we consider the case where the objective function $q(x)$ of Problem (1) is convex. In this case, it is easy to see that the bound given by Problem (4) is still a valid lower bound. However, we have to distinguish two very different cases at this point.

If the global continuous minimizer x^* of $q(x)$ does not belong to $E(H)$, then it is easy to see that even the bound given by Problem (5) is a valid lower bound. This bound is tighter than the global continuous minimum $q(x^*)$. We can thus use the same approach as in the nonconvex case in order to compute an improved lower bound in the convex case; see Figure 2 (left).

On the other hand, if $x^* \in E(H)$, then the bound given by Problem (4) is just the continuous minimum itself an hence useless in a branch-and-bound-approach. At the same time, Problem (5) does not necessarily yield a valid lower bound in this case. We thus use a slightly different approach, inspired by the method proposed in [7] but similar to our method used in the nonconvex case: we choose the point $z \in (\frac{1}{2}, \dots, \frac{1}{2})^\top + \mathbb{Z}^n$ closest to x^* and determine an ellipsoid

$$E_z(H) = \{x \in \mathbb{R}^n \mid (x - z)^\top H(x - z) \le \alpha\} ,$$

where α is chosen maximally such that $E_z(H)$ does not contain any integer feasible point in its interior; see Figure 2 (right). Clearly, $E_z(H)$ contains also x^*. From this, it follows that a lower bound for Problem (1) is obtained

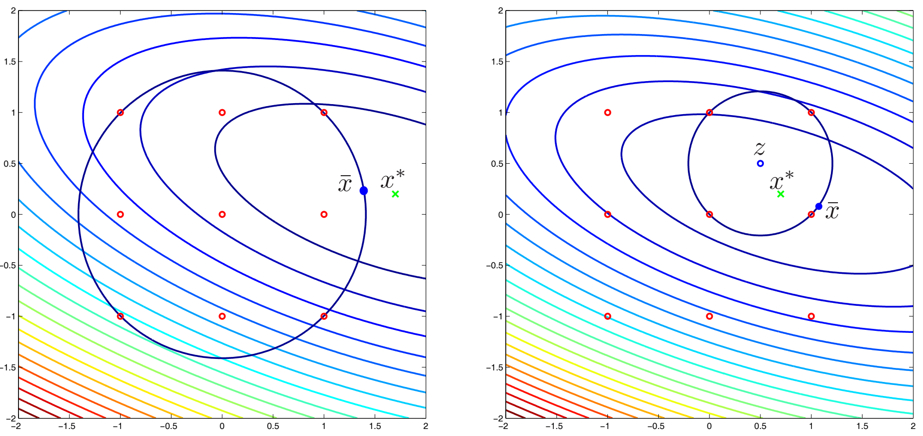

Figure 2: Computation of lower bounds in the convex case.

by solving

$$\begin{array}{ll} \min & q(x) = x^\top Q x + L^\top x \\ \text{s.t.} & (x-z)^\top H (x-z) = \alpha \,. \end{array}$$

By an appropriate transformation, which does not affect the matrix Q, this problem can be reduced to Problem (5), and solved by the same techniques as discussed in Section 4.3.

3 Choice of the Ellipsoid

We next discuss the problem of finding a "good" ellipsoid $E(H)$ of the form (3). The main objective is to find a matrix H such that the lower bound given by (5) is as tight as possible. At the same time, the matrix H should be easily computable.

In the following, we restrict ourselves to axis-parallel ellipsoids, i.e., to matrices $H = \text{Diag}(h)$ with $h \in \mathbb{R}^n_+$. More formally, we consider the set

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

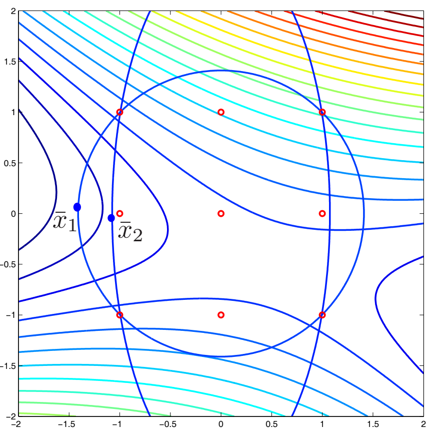

Figure 3: Different choices of the ellipsoid $E(H)$ give rise to different bounds.

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

$$\max_{H\in\mathcal{H}_{diag}} \min_{x\in\mathbb{R}^n} \{q(x) \mid x^\top Hx = 1\} \ . \tag{8}$$

Figure 3 illustrates how different choices of H may lead to different values of the resulting lower bound.

A straightforward feasible choice of the ellipsoid corresponds to $H = \frac{1}{n}I$. Indeed, this choice does not require any computation, 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 (8) or obtain H as a heuristic solution of (8).

The homogeneous version of problem (8) consists in ignoring the linear term in the objective function, so that it reduces to

$$\max_{H\in\mathcal{H}_{diag}} \lambda_{\min}(Q,H) = \max_{H\in\mathcal{H}_{diag}} \min_{x\in\mathbb{R}^n} \frac{x^\top Qx}{x^\top Hx} \ , \tag{9}$$

where $\lambda_{\min}(Q,H)$ is the generalized smallest eigenvalue for the pencil (Q,H). This approach was originally proposed by Kamath and Karmarkar [16] for the special case of optimizing a quadratic form over $\{-1,1\}^n$. They show the equivalence of Problem (9) with a semidefinite optimization problem that, in the nonconvex case, can be easily written as

$$\begin{aligned} \max_{H,t} \quad & t \\ & tQ + H \succeq 0, \\ & H \in \mathcal{H}_{diag} . \end{aligned} \tag{10}$$

This problem satisfies the Slater condition (i.e., strict feasibility holds for the primal and dual problem) and we can apply an interior point method to solve (10), yielding an optimal solution of Problem (9).

The third possibility consists in applying a subgradient-type (first order) method to the nonsmooth Problem (8); see e.g. [5]. The objective function of (8), namely

$$\min_{x \in \mathbb{R}^n} \{q(x) \mid x^\top Hx = 1\} ,$$

is not continuously differentiable in general and a generalized gradient is easily obtained as $-\mathrm{Diag}(\bar{\mu} \cdot \bar{x}^2)$, where $\bar{x}^2 \in \mathbb{R}^n$ is the vector with components $\bar{x}_i^2$; for more details see [25]. We cannot prove that this function is concave over $H \in \mathcal{H}_{diag}$, so that this approach is not guaranteed to yield optimal solutions of Problem (8). However, our results presented in Section 5 show that it yields very tight bounds in general.

4 Branch-and-Bound Algorithm

Our aim is to solve Problem (1) to proven optimality. To this end, we embed the lower bounds obtained by the strategies illustrated in Section 2.1 and 2.2 into a branch-and-bound framework. In the following, we describe the main components of this algorithm: the enumeration strategy (Section 4.1), the order in which variables are fixed (Section 4.2), the computation of lower bounds (Section 4.3), and the preprocessing phase (Section 4.4).

4.1 Enumeration Strategy

We branch by fixing a variable x_j to any integer value within the domain $[l_j, u_j]$. By this, the number of subproblems is at most $u_j - l_j + 1$. It follows that after each branching step, the resulting subproblem is an integer quadratic programming problem again, with a dimension decreased by one. We use a depth-first strategy.

In order to enumerate subproblems as quickly as possible, our aim is to perform the most time consuming computations in a preprocessing phase. Following [7], we decide to always fix the first unfixed variable according to an order $\{i_1, \ldots, i_n\}$ which is determined in the preprocessing phase, i.e., before starting the enumeration; see Section 4.2. 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 $i = 1, \ldots, d$, so that the vector x can be rewritten as

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

Fixing the first d variables results in subproblems depending only on the variables $(x_{d+1}, \ldots, x_n)$ 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

$$L_{j-d}^{(s,d)} = L_j + 2\sum_{i=1}^{d} q_{ij} r_i^{(s,d)}, \quad j = d+1, \ldots, n, \tag{11}$$

and

$$c^{(s,d)} = \sum_{i=1}^{d} L_i r_i^{(s,d)} + \sum_{i=1}^{d}\sum_{j=1}^{d} q_{ij} r_i^{(s,d)} r_j^{(s,d)}. \tag{12}$$

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. Namely, it is obtained from Q by deleting the first d rows and columns. If follows that all nodes at a given level d share the same quadratic term. As described in Section 4.3 below, the efficient solution of the relaxation needs some expensive operations on $Q^{(d)}$. Thus using a fixed order of variables implies that we have to perform such expensive operations only on n different matrices $Q^{(d)}$, which can be done in a preprocessing phase. The same is true for other operations concerning $Q^{(d)}$; see Section 4.4. 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 we compute a lower bound by using the strategies described in Section 2.1, distinguishing 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 lower bound is always used. In Section 4.2, we explain how the fixing order can be defined in order to exploit this fact.

Once the lower bound is obtained, 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 goes on until the list is empty, so that the current best solution is proved to be optimal.

For the computation of upper bounds, the algorithm produces feasible solutions by rounding the components of a vector $\bar{x}$ obtained in the lower bound computation. In the nonconvex case, this vector can be computed as an approximate solution of Problem (5). Since we solve the dual problem of (5), we use

$$\bar{x} = -\frac{1}{2}(Q - \bar{\mu}H)^{\dagger}L$$

to compute a corresponding primal vector $\bar{x}$. In the convex case, we simply choose $\bar{x} = x^*$, the global optimizer of $q(x)$.

In the following, we give an outline of the 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 $x_1, \dots, x_n$.

Pre-processing. Compute the matrix $Q^{(d)}$ for $d = 0, \dots, n-1$ 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 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 (11), (12).
    2. Compute lower bound $z_{lb}^{(s,d)}$ and corresponding $\bar{x}^{(s,d)} \in \mathbb{R}^{n-d}$.
    3. Update upper bound: let $x_I = (r^{(s,d)}, \lceil \bar{x}_1^{(s,d)} \rfloor, \dots, \lceil \bar{x}_{n-d}^{(s,d)} \rfloor)$;
       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. $x^*$, $z_{ub} = q(x^*)$.
```

4.2 Reordering of the Variables

As shown by the experimental results in Section 5, the performance of our algorithm depends strongly on the first level of convexity $\bar{d}$. 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 more sophisticated pruning rules can be applied [7].

We would thus like to find a permutation of columns and rows of Q in order to get $\bar{d}$ as small as possible. To this end, we propose a heuristic choice of the fixing order based on diagonal dominance properties. Let m_i be defined as

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

It is well known [13] that if all m_i are nonnegative with $q_{ii} \geq 0$, the matrix Q is positive semidefinite. Roughly speaking, the rule we propose to use consists in selecting the order of the variables by increasing values of m_i. The underlying idea is that the smaller the value of m_ℓ, the more significant is the role of x_ℓ for the nonconvexity of Q. Hence we proceed selecting variable x_ℓ and reducing the size of the matrix recursively to obtain the final order as reported in the following scheme.

```
Diagonal Dominance Rule for Reordering

Data. $Q \in \mathbb{S}^n$. Set $Q^{(n)} = Q$.

Do $t = n, \dots, 1$
        1. calculate $m_i$ for $i = 1, \dots, t$.
        2. choose the index $\ell = \arg\min_{i=1,\dots t} \left( q_{ii}^{(t)} - \sum_{\substack{j=i \\ j\neq i}}^{t} |q_{ij}^{(t)}| \right)$.
        3. set $j_{n-t+1} = \ell$.
        4. obtain $Q^{(t-1)}$ by removing the row and column $\ell$ from $Q^{(t)}$
End Do

Return. $(x_{j_1}, \dots, x_{j_n})$.
```

4.3 Computation of the Lower Bounds

As discussed above, the nonconvex problem (5) can be solved efficiently. Most of the algorithms proposed in literature for finding a global solution of Problem (5) (and thus a lower bound in our context) have been proposed in the context of trust region methods for unconstrained nonlinear minimization [22, 30, 12, 10]. Indeed, Problem (5) has been deeply studied in the nonlinear continuous optimization community, 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 [31, 26, 20, 17, 9, 11].

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 cannot be solved in reasonable time due to integrality constraints. 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 in each node.

The main idea is to perform heavy computational operations on the matrices Q and H in the preprocessing, in order to get a simplified form of Problem (5). Indeed, we can apply the linear transformation $y = H^{\frac{1}{2}}x$ to the variable space – which is particulary cheap as $H = \text{Diag}(h)$ is diagonal, in which case $H^{\frac{1}{2}} = \text{Diag}(\sqrt{h_1}, \dots, \sqrt{h_n})$ – thus obtaining

$$\begin{array}{ll} \min & y^\top H^{-\frac{1}{2}} Q H^{-\frac{1}{2}} y + (H^{-\frac{1}{2}} L)^\top y \\ \text{s.t.} & ||y||^2 = 1 \ . \end{array}$$

At this point we assume that we can calculate the spectral decomposition of the matrix $H^{-\frac{1}{2}} Q H^{-\frac{1}{2}}$ as

$$H^{-\frac{1}{2}} Q H^{-\frac{1}{2}} = P \, \Lambda \, P^\top,$$

where $P = [v_1 \, v_2 \dots v_n]$ is the orthogonal matrix of orthonormal eigenvectors of $H^{-\frac{1}{2}} Q H^{-\frac{1}{2}}$, and Λ is the diagonal matrix with elements $\lambda_1 \leq \cdots \leq \lambda_n$ being the eigenvalues of $H^{-\frac{1}{2}} Q H^{-\frac{1}{2}}$ in ascending order. In Section 4.4, we will explain how to perform such operations in the preprocessing phase. For sake of simplicity, we rename $y = P^\top y$, so that our problem takes the form

$$\begin{array}{rl} z^* = \min & y^\top \Lambda y + \widetilde{L}^\top y \\ \text{s.t.} & ||y||^2 = 1 \end{array} \tag{13}$$

where $\widetilde{L} = P^\top H^{-\frac{1}{2}} L$. Note that the size of this problem is linear in the dimension n. Moreover, in this simplified form the optimality conditions (6) reduce to

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

Most algorithms proposed in the literature look for an accurate primal solution $\bar{y}$, 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 [19], which is based on an unconstrained reformulation of Problem (13) and on some properties of its first order stationary points. We will use this approach for obtaining a global solution $\bar{y}$ in the preprocessing phase of our branch-and-bound scheme. In [19], the complete equivalence of Problem (13) with the minimization of an exact penalty function

$$\min_{y\in\mathbb{R}^n} \; y^\top \Lambda y + \widetilde{L}^\top y + \frac{1}{\epsilon}(||y||^2 - 1)^2 + \mu(y)(1 - ||y||^2)$$

has been proven, where $\epsilon > 0$ is a computable penalty parameter and $\mu(y)$ is obtained via (7) as

$$\mu(y) = -\frac{1}{2}(2y^\top \Lambda y + \widetilde{L}^\top y). \tag{14}$$

However, although not explicitly stated in the first papers [30, 12], most of the algorithms are based on duality results. Indeed in [31, 26] different dual programs have been proposed which are equivalent to the original primal one and which exhibit strong duality. A dual problem of (13) without duality gap is

$$\begin{aligned} \phi^* = \max_{\mu} \quad & \phi(\mu) = \mu - \tfrac{1}{4}\widetilde{L}^\top(\Lambda - \mu I)^\dagger \widetilde{L} \\ & \Lambda - \mu I \succeq 0, \end{aligned} \tag{15}$$

where $\phi(\mu)$ is a concave function (so that it follows implicitly that Problem (13) is convex).

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 of $\phi(\mu)$ for any feasible dual solution represents a safe bound. Hence we do not need to solve problem (15) to a high degree of accuracy to get $\bar{\mu}$. On contrary, if we considered the primal problem, we would need to find a very good approximation of the global solution $\bar{y}$.

So let us recall a scheme for the solution of the dual problem (15). First we note that the pseudoinverse of a diagonal matrix is obtained by taking the reciprocal of each non-zero element on the diagonal, leaving the zeros in place. This fact underlines that the pseudoinverse is not a continuous operation: slight changes on the zero diagonal entries result in significant changes on the pseudoinverse.

Indeed, solution methods for the pair of problems (13)–(15) distinguish two main cases: the *easy case* and the *hard case*, which can in turn be subdivided into two further sub-cases. For sake of completeness, we summarize these possibilities in the following proposition. Here, ϕ' denotes the derivative of the scalar function $\phi(\mu)$.

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

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

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

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

Proof. Because $\bar{\mu}$ is optimal for (15), there exists $\bar{y}$ such that the optimality conditions are satisfied. Consider case (i) with $J \neq \emptyset$. 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 $\phi'(\bar{\mu}) = 0$. As opposed, consider situations where $J = \emptyset$. 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. On the contrary, 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 have a diagonal matrix Λ, we can easily obtain the index set J. If 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 $\mu \in (-\infty, \lambda_1)$. We observe that

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

In the literature [22], on small-scale instances, this problem has been addressed by using a Newton method for finding the zero of the so-called secular equation

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

The Newton iteration is

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

and 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 [22]. We just mention that the Moré-Sorensen algorithm can be significantly simplified thanks to the knowledge of the spectral decomposition $P\Lambda P^T$. In particular both function and gradient evaluation are performed in $O(n)$ and also we can get rid of the estimate of λ_1 which is known in advance. For technical and practical details of the implementation we refer to Piacentini [25].

The choice of the initial value for μ^k is crucial for obtaining a good estimate of the solution in a few Newton steps. In a branch-and-bound framework, we will use a warm start strategy described in Section 4.4 which is based on the knowledge of the optimal primal solution of certain subproblems of type (13) and on the estimate (14).

4.4 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 $H^{(d)}$. Depending on the given matrix $Q^{(d)}$, we compute an appropriate matrix $H^{(d)}$ following the lines of Section 3, using one of the methods proposed. In our implementation, we actually compute only $H^{(0)}$

in the way described, while deriving all other $H^{(d)}$ by deleting appropriate rows and columns in $H^{(0)}$. This descreases preprocessing times without yielding significantly weaker bounds.

Compute weighted $Q^{(d)}$. Once the matrices $H^{(d)}$ are determined, we can compute the transformed matrices

$$\tilde{Q}^{(d)} = H^{(d)^{-\frac{1}{2}}} Q^{(d)} H^{(d)^{-\frac{1}{2}}}$$

for $d = 0, \ldots, n-1$. These are needed in the bound computation.

Spectral decomposition of $\tilde{Q}^{(d)}$. One of the most important tasks in the preprocessing is the computation of $(\Lambda^{(d)}, P^{(d)})$, the eigenvalues and eigenvectors of the weighted matrix $\tilde{Q}^{(d)}$, again needed in the bound computation.

Initial solutions for warm starts. For each subproblem, we need a starting value μ^0 for the dual algorithm described in Section 4.3. In the preprocessing, we compute an optimizer $y^{(d)}$ of

$$\begin{array}{ll} \min & y^\top \Lambda^{(d)} y + (L^{(d)} H^{(d)^{-\frac{1}{2}}})^\top P^{(d)} y \\ \text{s.t.} & ||y||^2 = 1 \; , \end{array} \tag{16}$$

where $L^{(d)} \in \mathbb{R}^{n-d}$ is obtained from L by dropping the first d components. By (7), we obtain an optimal solution of the corresponding dual problem as

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

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 [7]. 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.

5 Experimental Results

In this section we report our computational experience on an implementation named `GQIP` of the branch-and-bound scheme presented in Section 4.

Algorithm GQIP 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 $i = 1, \dots, n$. For our experiments, we consider objective functions $q : \mathbb{R}^n \to \mathbb{R}$ generated randomly as in [6]: given a $p \in [0, 1]$, 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. Finally, we determine L by choosing all entries uniformly at random from $[-1, 1]$. As in [6], we created ten random instances from different random seeds for each $n \in \{10, 20, \dots, 50\}$ and each $p \in \{0, 0.1, \dots, 1\}$. We thus consider 110 instances in total for each size n. In all experiments, we set the time limit to one hour; instances not solved within the time limit are considered a failure.

We first examine the impact of the choice of the matrix H, discussed in Section 3. We implemented three different choices:

- $\texttt{GQIP}_1$; $H = \frac{1}{n} \cdot I$
- $\texttt{GQIP}_2$; H obtained by solving problem (9), using CSDP [4]
- $\texttt{GQIP}_3$; H chosen by heuristically solving problem (8) by a simple implementation of a subgradient method.

All the three choices above have been run with the same reordering rule for the variables, namely the one described in Section 4.2. In Table 1, for each dimension n, we report the number of instances solved to proven optimality (solved), the maximum time spent to solve a problem (max time), and 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 # node). All running times are given in CPU-seconds.

As reported in Table 1, the three choices of H can be considered equivalent in terms of robustness and computational time for dimensions up to 30 or 40. However the average number of nodes of the branching tree is significantly smaller with $\texttt{GQIP}_3$. This is more evident for dimension 50 where the other two versions $\texttt{GQIP}_1$ and $\texttt{GQIP}_2$ need a much longer running time than $\texttt{GQIP}_3$. It can also be noticed that, for higher dimension, the preprocessing time is negligible with respect to the overall time.

n	alg	solved	max time	avg prep	avg time	avg # node
10	GQIP$_1$	110	0,0	0,0	0,0	33,5
	GQIP$_2$	110	0,0	0,0	0,0	60,7
	GQIP$_3$	110	0,1	0,0	0,0	28,5
20	GQIP$_1$	110	0,0	0,0	0,0	1068,5
	GQIP$_2$	110	0,0	0,0	0,0	1068,3
	GQIP$_3$	110	0,6	0,1	0,1	646,1
30	GQIP$_1$	110	1,1	0,0	0,2	24467,6
	GQIP$_2$	110	3,6	0,0	0,4	62028,3
	GQIP$_3$	110	1,2	0,5	0,4	5494,5
40	GQIP$_1$	110	55,4	0,1	5,8	904886,6
	GQIP$_2$	110	56,1	0,1	6,7	979664,4
	GQIP$_3$	110	10,4	1,1	1,9	102523,3
50	GQIP$_1$	109	1431,3	0,2	181,8	23425439,3
	GQIP$_2$	109	1597,3	0,2	208,4	23602304,1
	GQIP$_3$	110	309,6	2,2	31,8	2871660,7

Table 1: Results on ternary instances with different choice of H.

In the following evaluations, we thus concentrate on the most successful variant GQIP$_3$. To highlight the role of the reordering of variables, we report in Tables 2 and 3 the average results of the comparison between GQIP$_3$ using the diagonal dominance ordering described in section 4.2 (GQIP$_3$) and without using any reordering of the variables (GQIP$_3$NO). Table 2 shows that using diagonal dominance ordering decreases running times significantly. This is due to a much smaller number of nodes to be enumerated on average. In other words, the bounds are generally tighter after reordering.

We next examine the level of convexity of the problems when using the two different rules: as before, we denote by $\bar{d}_{\text{alg}}$ the smallest d such that $Q^{(d)}$ is a positive semidefinite submatrix of Q, depending on the reordering algorithm being applied. In Table 3, we report the values $\bar{d}_{\text{alg}}/n$. In other words, if Q is positive semidefinite, we obtain 0 %, while the result is 100 % if even the last nontrivial level is nonconvex. The diagonal dominance ordering yields much better results in terms of reaching a convex level quickly. However, the value of $\bar{d}_{\text{alg}}/n$ is far away from p in general, for both methods. This is also obvious from Figure 4, where we plot the results for $n = 50$.

We finally compare the performance of GQIP$_3$ with the following alternative solvers:

- COUENNE [1]: an algorithm for solving nonconvex mixed-integer non-

n	alg	solved	max time	avg prep	avg time	avg # node
10	GQIP$_3$NO	110	0,1	0,0	0,0	29,2
	GQIP$_3$	110	0,1	0,0	0,0	28,5
20	GQIP$_3$NO	110	0,3	0,1	0,1	700,5
	GQIP$_3$	110	0,6	0,1	0,1	646,1
30	GQIP$_3$NO	110	1,1	0,5	0,4	11639,9
	GQIP$_3$	110	1,2	0,5	0,4	5494,5
40	GQIP$_3$NO	110	13,3	2,4	1,0	174798,3
	GQIP$_3$	110	10,4	1,1	1,9	102523,3
50	GQIP$_3$NO	110	1542,7	2,2	112,7	11921425,2
	GQIP$_3$	110	309,6	2,2	31,8	2871660,7

Table 2: Results on ternary instances when using diagonal dominance ordering (GQIP$_3$) and without reordering (GQIP$_3$NO).

n	alg	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
10	GQIP$_3$NO	0%	56%	69%	79%	87%	93%	96%	98%	100%	100%	100%
	GQIP$_3$	0%	32%	49%	64%	73%	88%	91%	97%	100%	100%	100%
20	GQIP$_3$NO	0%	71%	86%	91%	98%	99%	99%	99%	100%	100%	100%
	GQIP$_3$	0%	45%	64%	76%	82%	93%	95%	100%	100%	100%	100%
30	GQIP$_3$NO	0%	72%	79%	88%	96%	98%	99%	99%	100%	100%	100%
	GQIP$_3$	0%	54%	65%	75%	87%	94%	97%	99%	100%	100%	100%
40	GQIP$_3$NO	0%	75%	87%	93%	95%	98%	99%	99%	100%	100%	100%
	GQIP$_3$	0%	59%	73%	83%	92%	96%	99%	100%	100%	100%	100%
50	GQIP$_3$NO	0%	76%	88%	94%	96%	99%	99%	100%	100%	100%	100%
	GQIP$_3$	0%	59%	74%	88%	92%	99%	99%	100%	100%	100%	100%

Table 3: Depth of the first convex level detected with either diagonal dominance ordering (GQIP$_3$) or no reordering (GQIP$_3$NO).

linear programs, based on convex underestimators;

- BARON [27]: a general purpose solver for optimization problems with nonlinear constraints;
- Q-MIST [6]: a branch-and-bound algorithm based on SDP relaxations for mixed integer quadratic programming.

We use as test bed the same random instances as before. In Table 4, we report the results grouped by dimension n. 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.

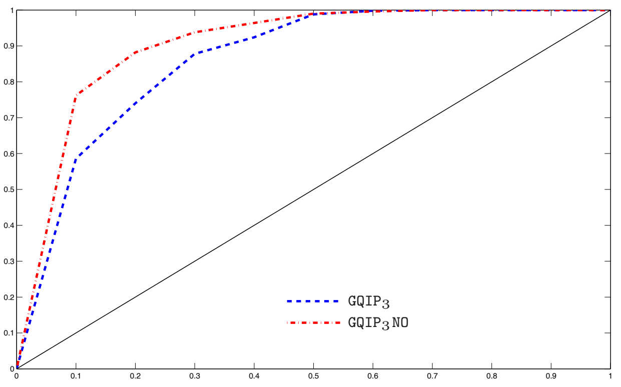

Figure 4: Graphical comparison between `GQIP`$_3$ and `GQIP`$_3$`NO`.

In order to analyze the performance of `GQIP` against `Q-MIST` in more detail, we investigate its dependance on the percentage of negative eigenvalues p. In Table 5, we report, for each dimension $n \in \{20, 30, 40, 50\}$ and each $p \in \{0, 0.1, \ldots, 1\}$, the average of the computational time over the 10 random instances solved. In Figure 5, we plot the corresponding running times on a logarithmic scale; Figure 6 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.

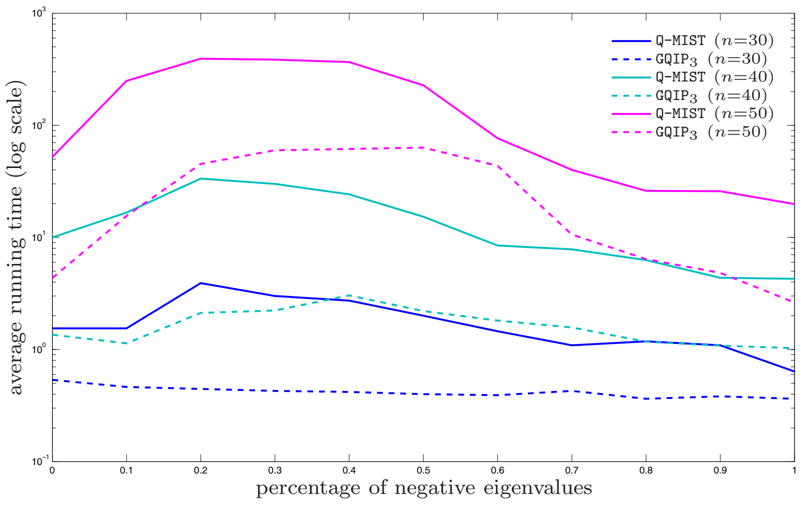

Figure 5: GQIP$_3$ versus Q-MIST: time comparison for $n = 30, 40, 50$

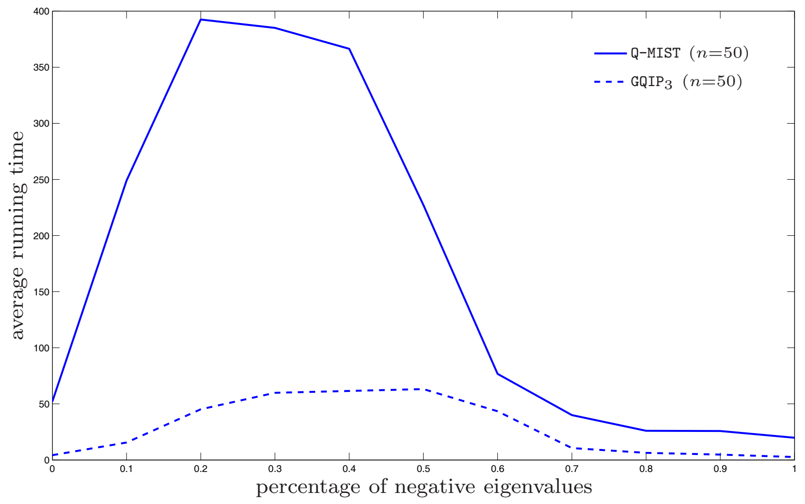

Figure 6: GQIP$_3$ versus Q-MIST: time comparison for $n = 50$

n	alg	solved	max time	avg time	avg # node
10	`COUENNE`	110	0,4	0,1	18,0
	`BARON`	110	0,4	0,1	8,9
	`Q-MIST`	110	1,0	0,0	9,1
	`GQIP`$_3$	110	0,1	0,0	28,5
20	`COUENNE`	110	51,4	0,0	3822,0
	`BARON`	110	477,1	24,1	1548,8
	`Q-MIST`	110	1,0	0,2	53,5
	`GQIP`$_3$	110	0,6	0,1	646,1
30	`COUENNE`	78	3567,3	1476,7	181127,6
	`BARON`	82	3552,1	1173,9	36218,1
	`Q-MIST`	110	10,0	2,0	199,7
	`GQIP`$_3$	110	1,2	0,4	5494,5
40	`COUENNE`	2	3148,3	2012,3	99500,0
	`BARON`	4	2233,8	13494,4	19411,3
	`Q-MIST`	110	106,0	16,1	831,6
	`GQIP`$_3$	110	10,4	1,9	102523,3
50	`COUENNE`	0	***	***	***
	`BARON`	0	***	***	***
	`Q-MIST`	110	1593,0	186,0	5463,7
	`GQIP`$_3$	110	309,6	31,8	2871660,7

Table 4: Comparitive results for ternary instances for different solvers.

n	alg	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
20	GQIP$_3$	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$_3$	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$_3$	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$_3$	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 5: Detailed comparison between `GQIP`$_3$ and `Q-MIST`: average times at different percentages of negative eigenvalues.

References

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

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

[3] 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.

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

[5] S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing Markov chain on a graph. *SIAM Review*, 46(4):667–689, 2004.

[6] C. Buchheim and A. Wiegele. Semidefinite relaxations for non-convex quadratic mixed-integer programming. *Mathematical Programming (Series A)*, 2012. doi: 10.1007/s10107-012-0534-y. To appear.

[7] C. Buchheim, A. Caprara, and A. Lodi. An effective branch-and-bound algorithm for convex quadratic integer programming. *Mathematical Programming (Series A)*, 2011. doi: 10.1007/s10107-011-0475-x. Online first.

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

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

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

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

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

[13] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins Press, Baltimore, 1989.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

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

[23] 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.

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

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

[26] F. Rendl and H. Wolkowicz. A semidefinite framework to trust region subproblems with applications to large scale minimization. *Mathematical Programming*, 77(2):273–299, 1997.

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

[28] 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.

[29] 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.

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

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

[32] 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.

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

[34] 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.