Quadratic 0–1 optimization using separable underestimators

Christoph Buchheim · Emiliano Traversi

Received: date / Accepted: date

Abstract Binary programs with a quadratic objective function are NP-hard in general, even if the linear optimization problem over the same feasible set is tractable. In this paper, we address such problems by computing quadratic global underestimators of the objective function that are separable but not necessarily convex. Exploiting the binarity constraint on the variables, a minimizer of the separable underestimator over the feasible set can be computed by solving an appropriate linear minimization problem over the same feasible set. Embedding the resulting lower bounds into a branch-and-bound framework, we obtain an exact algorithm for the original quadratic binary program. The main practical challenge is the fast computation of an appropriate underestimator, which in our approach reduces to solving a series of semidefinite programs. We exploit the special structure of the resulting problems and propose a tailored coordinate-descent method for their solution. Our extensive experimental results on various quadratic combinatorial optimization problems show that our approach outperforms both Cplex and the related QCR method. For the quadratic shortest path problem, we thus provide the fastest exact approach currently available.

Keywords Binary Quadratic Optimization · Separable Underestimators · Quadratic Shortest Path Problem

1 Introduction

We consider binary quadratic optimization problems of the form

$$\min \ f(x) := x^T Q x + L^T x$$

s.t. \( x \in X \),

where \( Q \in \mathbb{R}^{n \times n} \) is a symmetric matrix, \( L \in \mathbb{R}^n \) is a vector and \( X \subseteq \{0,1\}^n \) is the set of feasible binary vectors. Many combinatorial optimization problems can
be naturally formulated in this fashion, e.g., network design problems with reload costs [2,14], the angular metric TSP [1], or crossing minimization problems for bipartite graphs [9].

In this paper, we focus on combinatorial optimization problems where the linear counterpart of Problem (1),

\[
\begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad x \in X,
\end{align*}
\]

(2)
can be solved efficiently for any vector \( c \in \mathbb{R}^n \). However, we do not make any assumptions on how Problem (2) is solved. In particular, any combinatorial algorithm can be used, neither a compact linear description nor a polynomial-time separation algorithm for \( \text{conv} \ (X) \) are required. From a practical point of view, our approach is also beneficial if Problem (2) is NP-hard but significantly easier to solve than Problem (1), e.g., for the quadratic knapsack problem. Even if Problem (2) is tractable, the quadratic problem (1) is usually NP-hard. This is true, e.g., for the unconstrained case \( X = \{0,1\}^n \), where Problem (1) is equivalent to unconstrained quadratic binary optimization and hence to the max-cut problem. For another example, the quadratic spanning tree problem is NP-hard [3], while the linear counterpart can be solved very quickly, e.g., by Kruskal’s algorithm [16].

The standard approach for solving problems of type (1) is based on linearization. In a first step, a new variable \( y_{ij} \) representing the product \( x_i x_j \) is introduced for each pair \( i,j \). Then the convex hull of feasible solutions in the extended space is usually approximated either by a polyhedral relaxation or by semidefinite programming (SDP) models, or by a combination of both. The main focus lies on enforcing the connection between \( x \)- and \( y \)-variables. For the unconstrained case, we point the reader to [17] and the reference therein. In the constrained case, most approaches presented in the literature are highly problem-specific; only few general techniques have been devised, see e.g. [11].

In this paper, we address Problem (1) by computing underestimators \( g \) of the quadratic objective function \( f \). A lower bound on Problem (1) can then be computed by minimizing \( g(x) \) over \( x \in X \). Unlike most other approaches based on underestimators, we however do not use convex functions in general, but separable non-convex functions. The main idea of our approach is to determine a good separable underestimator \( g \) of \( f \) in the first step; in the second step we can reduce the separable quadratic function to a linear function exploiting the binarity of all variables. The minimization of \( g(x) \) over \( x \in X \) can thus be performed by solving Problem (2); convexity of the underestimator is not required. The resulting lower bounds are embedded into a branch-and-bound scheme for solving Problem (1) to optimality. Our approach can also be used when the linear counterpart (2) is not efficiently solvable, but when a lower bound for (2) can be computed quickly. In this case, the same approach yields a lower bound for the Problem (1). Compared with linearization, the advantage of our approach lies in the fact that we do not need to add any additional variables. Moreover, we do not require any polyhedral knowledge about \( \text{conv} \ (X) \) and do not use any LP solver at all. At the same time, any algorithmic knowledge about the linear problem (2) is exploited directly.

A related approach to binary optimization is the QCR technique [4]. Instead of linearizing the problem, it is reformulated as an equivalent binary optimization problem with a convex quadratic objective function. This allows to apply more
powerful software tailored for convex problems. In particular, it is now possible
to solve the continuous relaxation of the problem efficiently. The QCR approach
is designed such that this relaxation yields as tight lower bounds as possible. The
resulting bounds are not directly comparable to the bounds obtained by our ap-
proach: while the former may be stronger because separability is not required, our
approach does not require convexity of the underestimator and, more importantly,
we optimize the resulting underestimator over the binary feasible set, while the
QCR bounds are derived from a continuous relaxation of the convex quadratic
problem. This implies that our new bounds are particularly strong whenever the
matrix \(Q\) is diagonal-dominant. Experimentally, we show that our root bounds for
general matrices \(Q\) are considerably weaker than the QCR bounds, but the overall
running times of our approach are shorter.

An important question in our approach is how to compute the separable un-
derestimator \(g\). We first fix a point \(z \in \mathbb{R}^n\) where \(f(z) = g(z)\), i.e., where the
underestimator touches the original objective function. It turns out that a reason-
able choice is the box center \(\frac{1}{2}1\). Under this restriction, we compute a separable
quadratic function \(g\) that is a global underestimator for \(f\) and that maximizes
the minimum of \(g(x)\) over \(X\), i.e., that yields a best possible lower bound. We
show that this task can be accomplished efficiently either by solving an SDP or
by applying a subgradient method, depending on \(z\) and \(X\).

In order to obtain a fast branch-and-bound scheme, we however avoid to apply
a subgradient method or to solve an SDP in each node of the enumeration tree. In-
stead, we compute underestimators that only depend on the matrix \(Q\), but not on
the linear part \(L\) of the objective function, leading to slightly weaker underestima-
tors. Using an appropriate branching scheme as already proposed in [6,7], we can
make sure that only \(n\) different matrices \(Q\) arise in the entire branch-and-bound
process. For each of the corresponding matrices, an underestimator is computed
in the preprocessing phase by solving an SDP. As it is still too time-consuming
to solve each SDP by a standard interior point method, we developed a tailored
coordinate-descent algorithm exploiting the structure of these problems.

This paper is organized as follows. In the next section, we formalize the main
ideas of our approach. In Section 3, we present strategies to determine best possi-
ble separable underestimators; we also discuss the connection to other SDP-based
bounds and a possible improvement of lower bounds by taking into account valid
linear equations or box constraints. Details of our branch-and-bound algorithm are
given in Section 4. The coordinate-descent algorithm for computing underestima-
tors is sketched in Section 5. In Section 6, we evaluate our approach computa-
tionally, applying it to instances of the quadratic assignment problem, the quadratic
knapsack problem and the quadratic shortest path problem. We compare our ap-
proach both with Cplex and with the QCR method. A preliminary version of this
paper has been published in the Proceedings of SEA 2013 [8].

2 Notation and basic idea

Our aim is to derive a lower bound for Problem (1) by solving the linear coun-
terpart (2) for an appropriate vector \(c \in \mathbb{R}^n\). To this end, for an arbitrary point
\(z \in \mathbb{R}^n\), we can rewrite \(f(x)\) as
\[
f(x) = (x - z)^\top Q(x - z) + (L + 2Qz)^\top x - z^\top Qz.
\]
For a given vector \( t \in \mathbb{R}^n \), define
\[
g_z(t)(x) := (x - z)^\top \text{Diag}(t)(x - z) + (L + 2Qz)^\top x - z^\top Qz
\]
\[
= \sum_{i=1}^n t_i x_i^2 + \sum_{i=1}^n (-2z_i t_i + l_i + 2q_i^\top z) x_i + \sum_{i=1}^n z_i^2 t_i - z^\top Qz,
\]
where \( q_i \) denotes the \( i \)-th row of \( Q \). Then \( g_z(t)(z) = f(z) \), i.e., the function \( g_z(t) \) touches \( f \) in \( z \). By (3), it is easy to see that the function \( g_z(t) \) is a global underestimator of \( f \) if and only if \( Q \succeq \text{Diag}(t) \). In this case, the desired lower bound can be obtained as
\[
\min g_z(t)(x) \quad \text{s.t. } x \in X.
\]
(4)

As \( X \subseteq \{0,1\}^n \), we can replace Problem (4) by the equivalent problem
\[
\min l_z(t)(x) \quad \text{s.t. } x \in X
\]
(5)

where the function
\[
l_z(t)(x) := \sum_{i=1}^n t_i x_i + \sum_{i=1}^n (-2z_i t_i + l_i + 2q_i^\top z) x_i + \sum_{i=1}^n z_i^2 t_i - z^\top Qz
\]
is bilinear in \( x, t \in \mathbb{R}^n \). Here we use \( \cdot \) to denote entrywise multiplication and define \( z^\cdot := z \cdot z \). Note that Problem (5) is of type (2).

3 Optimal separable underestimators

The choice of \( t \) is crucial for the strength of the lower bound resulting from (5). As discussed above, this lower bound is valid for each \( t \in \mathbb{R}^n \) with \( Q \succeq \text{Diag}(t) \). Our objective is to maximize the lower bound induced by \( t \). In other words, our aim is to solve the problem
\[
\max \min_{x \in X} l_z(t)(x) \quad \text{s.t. } Q \succeq \text{Diag}(t).
\]
(6)

We will show that Problem (6) can be solved efficiently with a subgradient method for an arbitrary choice of the touching point \( z \), while it reduces to an SDP for the particular choice \( z = \frac{1}{2} \mathbf{1} \). Before, we note that the bound given by (6) is invariant under any shifting of weights between \( \text{diag}(Q) \) and \( L \), provided that the touching point \( z \) remains unchanged.

3.1 Invariance under reformulation of the objective function

As \( X \subseteq \{0,1\}^n \), minimizing \( f(x) \) over \( x \in X \) is equivalent to minimizing
\[
f_\alpha(x) = x^\top (Q + \text{Diag}(\alpha)) x + (L - \alpha)^\top x
\]
over \( x \in X \), for any \( \alpha \in \mathbb{R}^n \). One may ask whether the choice of \( \alpha \) has an effect on the lower bounds computed above. If \( z \) is not fixed but depends on \( \alpha \), the answer
Quadratic combinatorial optimization is yes. However, when choosing a fixed touching point $z$ not depending on $\alpha$, we can show that the resulting bound does not depend on $\alpha$ as well. Let $g_{z,\alpha}^{(t)}$ denote the family of underestimators of $f_\alpha$ as constructed in Section 2.

**Theorem 1** Let $z \in \mathbb{R}^n$. Then the optimal value of

$$
\max_{x \in X} \min_{\alpha} g_{z,\alpha}^{(t)}(x)
\quad \text{s.t.} \quad Q + \text{Diag}(\alpha) \succeq \text{Diag}(t)
$$

(7)

does not depend on $\alpha$.

**Proof** Choose any $\alpha \in \mathbb{R}^n$. For each $x \in X$, we have

$$
g_{z,\alpha}^{(t)}(x) = (x - z)^\top \text{Diag}(t)(x - z)
+ (L - \alpha + 2(Q + \text{Diag}(\alpha))z)^\top x - z^\top (Q + \text{Diag}(\alpha))z
= (x - z)^\top \text{Diag}(t - \alpha)(x - z) + (x - z)^\top \text{Diag}(\alpha)(x - z)
+ (L - \alpha + 2(Q + \text{Diag}(\alpha))z)^\top x - z^\top (Q + \text{Diag}(\alpha))z
= (x - z)^\top \text{Diag}(t - \alpha)(x - z) + (L + 2Qz)^\top x - z^\top Qz
= g_{z}^{(t-\alpha)}(x).
$$

Here we use $x^\top \text{Diag}(\alpha)x = \alpha^\top x$, which follows from $x \in \{0,1\}^n$. Consequently, Problem (7) is equivalent to

$$
\max_{x \in X} \min_{\alpha} g_{z}^{(t-\alpha)}(x)
\quad \text{s.t.} \quad Q \succeq \text{Diag}(t - \alpha)
$$

and hence to (6) by translation. \hfill \Box

### 3.2 Computation of the lower bound

For general $X \subseteq \{0,1\}^n$, Problem (6) can be solved by a subgradient method. For this, we can model the constraint $Q \succeq \text{Diag}(t)$ by an exact penalty approach, using the following result. Here $\lambda_{\min}(A)$ denotes the smallest eigenvalue of a symmetric matrix $A$.

**Theorem 2** Let $\mu \in \mathbb{R}$ such that $\mu \geq ||x - z||^2$ for all $x \in X$. Then Problem (6) is equivalent to

$$
\max_{t \in \mathbb{R}^n} \min_{x \in X} l_z^{(t)}(x) + \mu \min\{0, \lambda_{\min}(Q - \text{Diag}(t))\}
\quad \text{s.t.} \quad t \in \mathbb{R}^n.
$$

(8)

**Proof** It is clear that the optimal value of (8) is greater or equal to the optimal value of (6), it thus remains to show the converse. For this, let $t^*$ be an optimal solution of (8). If $Q - \text{Diag}(t^*) \succeq 0$, then $t^*$ is also feasible for (6) with the same objective function value, so we may assume $\lambda_{\min}(Q - \text{Diag}(t^*)) < 0$. Consider

$$
\tilde{t} := t^* + \lambda_{\min}(Q - \text{Diag}(t^*))1.
$$
By construction, $\bar{t}$ is a feasible solution for (6). For each $x \in X$, we have

$$l_\bar{t}(x) = (x - z)\top \text{Diag}(\bar{t})(x - z) + (L + 2Qz)\top x - z\top Qz$$

$$= (x - z)\top \text{Diag}(t^\star)(x - z) + (L + 2Qz)\top x - z\top Qz$$

$$+ \lambda_{\min}(Q - \text{Diag}(t^\star))||x - z||^2$$

$$\geq l_{t^\star}(x) + \mu \lambda_{\min}(Q - \text{Diag}(t^\star)) .$$

Hence the objective value of $\bar{t}$ in (6) is greater or equal to the objective value of $t^\star$ in (8).

\[ \square \]

In particular, we can always choose

$$\mu := \max_{x \in \{0,1\}^n} ||x - z||^2 = \sum_{i=1}^n \max\{z_i^2, (1 - z_i)^2\}$$

in Theorem 2. For the touching point $z = \frac{1}{2}1$, we can thus use $\mu := \frac{1}{4}n$ to obtain an exact penalty approach, while for each $z \in [0,1]^n$ we may use $\mu := n$.

The objective function of (8) is concave, so that a subgradient approach can be used to solve the problem efficiently. The supergradient of

$$\min_{x \in X} l_{t^k}(x)$$

at a given point $t^k$ can be computed by using the black box (2), as $l_{t^k}(x)$ is a linear function in $x$. Given the optimal solution $\hat{x}^k$, the desired supergradient is the gradient of $l_{t^k}(\hat{x}^k)$, which is easily computed since $l_{t^k}(\hat{x}^k)$ is a linear function also in $t$. If $\lambda_{\min}(Q - \text{Diag}(t^k)) < 0$, the supergradient of the penalty term can be obtained as $-\mu v$, where $v$ is a normalized eigenvector corresponding to the eigenvalue $\lambda_{\min}(Q - \text{Diag}(t^k))$. Note that the subgradient algorithm can be stopped at any time and the best $t^k$ obtained so far can be used in order to obtain a feasible solution

$$t := t^k + \min\{0, \lambda_{\min}(Q - \text{Diag}(t^k))\}1$$

as already observed in the proof of Theorem 2.

In the special case of $z = \frac{1}{2}1$, Problem (6) can be solved more easily: in this case, the function $l_{t}(x)$ can be simplified as

$$l_{t}(x) = \frac{1}{4}1\top t + (L + Q1)\top x - \frac{1}{4}1\top Q1 .$$

Note that for this particular choice of $z$, the function $l_{t}(x)$ does not contain any product of $z$ and $t$. Problem (6) thus becomes

$$\max \frac{1}{2}1\top t \quad \text{subject to} \quad Q \succeq \text{Diag}(t)$$

$$\min (L + Q1)\top x \quad \text{subject to} \quad x \in X$$

and hence decomposes. The first problem is an SDP, while the second problem can be solved by calling the oracle (2) once. In particular, the optimal underestimator only depends on $Q$ in this case, but not on $L$. This fact can be exploited in our branch-and-bound algorithm, as explained in Section 4. A fast algorithm for solving the SDP on the left hand side, exploiting its specific structure, is discussed in Section 5.
3.3 Comparison with other SDP-based relaxations

For any touching point \( z \in \mathbb{R}^n \), Problem (6) could be reformulated as an SDP with a potentially exponential number of constraints as follows:

\[
\begin{align*}
\max & \quad \beta \\
\text{s.t.} & \quad \beta \leq l^{(t)}_z(x) \quad \forall x \in X \\
& \quad Q \succeq \text{Diag}(t).
\end{align*}
\] (9)

All constraints in (9) except for the last one are linear. The corresponding separation problem is of type (2) again, as it amounts to checking whether

\[
l^{(t^*)}_z(x) < \beta^*
\]

for some \( x \in X \), for fixed \( t^* \) and \( \beta^* \). The dual of (9) turns out to be

\[
\begin{align*}
\min & \quad \langle Q,Y \rangle + \sum_{x \in X} \lambda_x ((L + 2Qz)^\top x) - z^\top Qz \\
\text{s.t.} & \quad \text{diag}(Y) = \sum_{x \in X} \lambda_x (x - z)^2 \\
& \quad \sum_{x \in X} \lambda_x = 1, \quad \lambda_x \geq 0 \quad \forall x \in X \\
& \quad Y \succeq 0,
\end{align*}
\] (10)

which is a relaxation of the basic problem (1). Indeed, adding integrality of \( \lambda \) and the non-convex constraint \( \text{rk}(Y) = 1 \), we obtain

\[
\begin{align*}
\min & \quad y^\top Qy + (L + 2Qz)^\top(y + z) - z^\top Qz = f(y + z) \\
\text{s.t.} & \quad y^2 \in \{(x - z)^2 | x \in X\} \quad \Leftrightarrow \quad y + z \in X.
\end{align*}
\]

In order to compare our relaxation (6) with other relaxations discussed in the literature, we consider two special cases for the touching point \( z \) again. First assume \( z = 0 \). Then the dual problem (10) simplifies to

\[
\begin{align*}
\min & \quad \langle Q,Y \rangle + \sum_{x \in X} \lambda_x (L^\top x) \\
\text{s.t.} & \quad \text{diag}(Y) = \sum_{x \in X} \lambda_x x \\
& \quad \sum_{x \in X} \lambda_x = 1, \quad \lambda_x \geq 0 \quad \forall x \in X \\
& \quad Y \succeq 0,
\end{align*}
\]

which is equivalent to

\[
\begin{align*}
\min & \quad \langle Q,Y \rangle + L^\top x \\
\text{s.t.} & \quad \text{diag}(Y) = x \\
& \quad x \in \text{conv} X \\
& \quad Y \succeq 0.
\end{align*}
\] (11)
This formulation is very similar to the standard SDP relaxation for constrained binary quadratic optimization, which can be written as

\[
\begin{align*}
\min & \quad \langle Q, Y \rangle + L^\top x \\
\text{s.t.} & \quad \text{diag}(Y) = x \\
& \quad x \in \text{conv} \ X \\
& \quad \begin{pmatrix} 1 & x^\top \\ x & Y \end{pmatrix} \succeq 0,
\end{align*}
\]

assuming that a complete polyhedral description of conv \( X \) is known. This shows that our relaxation is dominated by the standard SDP relaxation in the case \( z = 0 \), as the SDP constraint in (12) implies the SDP constraint in (11). Unfortunately, the stronger SDP constraint of (12) cannot easily be generalized to other choices of \( z \). Note that each other point \( z \in \{0,1\}^n \) leads to the same conclusions by symmetry.

However, when using \( z = \frac{1}{2}1 \) as touching point, the situation changes. The problem then decomposes again and (10) simplifies to

\[
\begin{align*}
\min & \quad \langle Q, Y \rangle + \sum_{x \in X} \lambda_x ((L + Q1)^\top x) - \frac{1}{4}1^\top Q1 \\
\text{s.t.} & \quad \text{diag}(Y) = \sum_{x \in X} \lambda_x \frac{1}{2}1 \\
& \quad \sum_{x \in X} \lambda_x = 1, \; \lambda_x \geq 0 \; \forall x \in X \\
& \quad Y \succeq 0,
\end{align*}
\]

which is equivalent to

\[
\begin{align*}
\min & \quad \frac{1}{4}(Q, Y) + \min_{x \in X} (L + Q1)^\top x - \frac{1}{4}1^\top Q1 \\
\text{s.t.} & \quad \text{diag}(Y) = 1 \\
& \quad Y \succeq 0.
\end{align*}
\]

In this case, there is no dominance between the standard SDP relaxation and our relaxation (6). The disadvantage of (13) lies again in the fact that the SDP constraint only takes the quadratic part of the objective function into account. However, in the special case \( Q = 0 \), our relaxation yields the optimal value of (1), and by Theorem 1 this remains true if \( Q \) is any diagonal matrix. None of the SDP based relaxations discussed in the literature will achieve this in general. This reflects the main idea of our approach, namely to address the underlying linear problem by exact methods.

It follows that our approach is particularly beneficial in cases where the diagonal and linear parts of the objective function dominate the quadratic part. However, our experimental results in Section 6 show that our approach usually even outperforms the QCR method on instances where all entries of \( Q \) and \( L \) have the same order of magnitude. This is due to the fact that our relaxations can be computed more quickly when using a smart branching scheme and preprocessing, as discussed in Section 4.
3.4 Feasible sets of low dimension

As already discussed in [8], the underestimators given by (6) can be improved if the set $X$ satisfies a set of linear equations $Ax = b$. For the convenience of the reader, we shortly summarize this idea in the following.

Let the columns of $V \in \mathbb{R}^{n \times k}$ form an orthonormal basis of the kernel of $A$ and choose $w \in \mathbb{R}^n$ with $Aw = b$. Then $g^{(t)}_z$ underestimates $f$ on the affine subspace given by $Ax = b$ if and only if

$$V^T QV \succeq V^T \text{Diag}(t)V.$$  

The latter constraint can replace the stronger constraint $Q \succeq \text{Diag}(t)$ both in the subgradient approach and in the SDP based approach. In the former, the penalty term may be replaced by $\min\{0, \lambda_{\min}(V^T (Q - \text{Diag}(t))V)\}$; the corresponding supergradient is $-(Vv)^2$, where $v$ is a normalized eigenvector of $V^T (Q - \text{Diag}(t))V$ corresponding to its smallest eigenvalue.

In our experimental results presented in [8], it turns out that this replacement can improve both the bounds and the total running times significantly, even if only one equation is considered, as in the case of the quadratic spanning tree problem. Note that other approaches such as the QCR method usually do not exploit the presence of equations but take all binary vectors into account.

3.5 Taking box constraints into account

We next discuss how the underestimator can be improved when taking into account that it only needs to underestimate $f$ on the set $\{0, 1\}^n$. We have

$$g^{(t)}_z(x) \leq f(x) \forall x \in \{0, 1\}^n$$

$$\iff (x - z)^T (Q - \text{Diag}(t))(x - z) \geq 0 \forall x \in \{0, 1\}^n.$$ 

A sufficient condition is

$$y^T (Q - \text{Diag}(t))y \geq 0 \forall y \in \mathbb{R}^n: y_i \geq 0 \text{ if } z_i \leq 0, y_i \leq 0 \text{ if } z_i \geq 1.$$  

(14)

This condition is also necessary if $z \in [0, 1]^n$. In particular, when choosing the touching point as $z = \frac{1}{2}1$, this shows that we need to enforce $Q - \text{Diag}(t) \succeq 0$ for obtaining a valid underestimator. In this sense, the methods presented above yield best possible underestimators.

On the contrary, if $z = 0$, the condition above is equivalent to $Q - \text{Diag}(t)$ being copositive, which is a weaker condition than positive semidefiniteness. For computational matters, as copositive optimization is NP-hard in general, the cone of copositive matrices is often replaced by the sum of the positive semidefinite cone and the cone of nonnegative matrices; the latter sum is a proper subcone of the former.

In the same spirit, we can replace (14) by the slightly stronger condition that $Q - \text{Diag}(t)$ be a sum of a positive semidefinite matrix and a symmetric matrix $N$ such that

$$N_{ij} \geq 0 \text{ if } (z_i \leq 0 \text{ and } z_j \leq 0) \text{ or } (z_i \geq 1 \text{ and } z_j \geq 1)$$

$$N_{ij} \leq 0 \text{ if } (z_i \leq 0 \text{ and } z_j \geq 1) \text{ or } (z_i \geq 1 \text{ and } z_j \leq 0)$$

$$N_{ij} = 0 \text{ otherwise.}$$  

(15)
In other words, we require \( Q - \text{Diag}(t) - N \succeq 0 \) for some matrix \( N \) satisfying the conditions (15). In general, this requirement is less strict than \( Q - \text{Diag}(t) \succeq 0 \), so it can lead to tighter lower bounds while still being tractable: the optimal \( t \) can still be computed by solving a semidefinite program.

This improvement can also be combined with the techniques presented in Section 3.4 for taking valid equations into account. In this case, the resulting relaxed condition on \( t \) is

\[
V^\top (Q - \text{Diag}(t) - N)V \succeq 0
\]

for some \( N \) that satisfies (15). Again, this condition can be modeled within a semidefinite program.

4 Branch-and-bound algorithm

In order to solve Problem (1) exactly, we embed the lower bounds derived in Section 3 into a branch-and-bound framework. Our main objective is to exploit the fact that Problem (6) only changes slightly from one node in the enumeration tree to a neighboring node. In fact, we enforce this similarity in two ways, as already described in [8] and shortly summarized in the following.

4.1 Enumeration scheme and preprocessing

Firstly, we determine an order of variables at the beginning and fix variables always in this order. More precisely, if \( x_1, \ldots, x_n \) is the chosen order, the next variable to be fixed is the free variable with smallest index. The same idea has been used in [6] and [7]. In this way, the reduced matrices \( Q \) in the nodes of the enumeration tree only depend on the depth of the node but not on the specific subproblem. Consequently, only \( n \) such matrices can appear in the enumeration tree, instead of \( 2^n \) when applying other branching strategies.

Secondly, we do not call the subgradient method or SDP to compute an optimal \( t \) in every node, but try to find one fixed \( t \) for each level of the enumeration tree that yields strong lower bounds on average. This reduces the number of oracle calls to one per node. By our first restriction, the matrix \( Q \) is fixed on each level, so that we can compute the vector \( t \) depending on \( Q \).

In summary, all time-consuming computations concerning the matrix \( Q \) can now be performed in a preprocessing phase, including the computation of an underestimator for each level of the enumeration tree, and the same remains true when considering valid equations. All problem data can be updated quickly in an incremental way. In summary, if no subgradient approach is used and touching points are consistent on all depths, all operations except for the solution of the black box problem can be performed in a total running time of \( O((n - d)^2) \) for a node on depth \( d \). For technical details, we refer to [8].

Our decision to use the same order of variable fixings throughout the entire enumeration still allows to choose this order once in the beginning. In order to compute tight underestimators, we aim at matrices that are as close to a diagonal matrix as possible, as motivated by the discussion in Section 3.3. To this end, we first fix the variable \( x_i \) that maximizes \( \sum_{j \neq i} |q_{ij}| \) and then apply the same
rule recursively. Each variable is first fixed to the value it had in the last solution computed by the linear optimization oracle.

### 4.2 Computation of an underestimator

In Section 3 we showed that for the special touching point $\frac{1}{2} \mathbf{1}$ the optimal lower bound can be computed as

$$
\max \frac{1}{2} \mathbf{1}^\top t \quad \text{s.t.} \quad Q \succeq \text{Diag}(t) \quad \text{s.t.} \quad x \in X
+ \min (L + Q \mathbf{1})^\top x - \frac{1}{2} \mathbf{1}^\top Q \mathbf{1}.
$$

In order to speed up our algorithm, we apply the same approach for any touching point $z$. More precisely, we determine an underestimator $t$ solving the SDP on the left hand side, which only depends on the matrix $Q$. By our ideas sketched above, the matrix $Q$ in turn only depends on the level in the enumeration tree, so that only $n$ such underestimators need to be computed in the preprocessing phase. The second problem is a linear optimization problem depending on the current node in the enumeration tree, it can be solved by calling the oracle for Problem (2) once. This approach may lead to weaker bounds if the touching point is different from $\frac{1}{2} \mathbf{1}$, but the advantage of solving $n$ instead of exponentially many semidefinite programs is crucial here. Note that in the presence of equations we may be forced to choose a touching point different from $\frac{1}{2} \mathbf{1}$, as it needs to satisfy all considered equations.

As discussed above, the strength of the underestimator can be improved by exploiting valid equations. For this, the above SDP can be replaced by

$$
\max \mathbf{1}^\top t \quad \text{s.t.} \quad V^\top Q V \succeq V^\top \text{Diag}(t) V.
$$

However, this problem can be unbounded, as the objective function is not restricted to the feasible subspace. To avoid this, we replace the objective function by

$$
\langle I, V^\top \text{Diag}(t) V \rangle = \langle V V^\top, \text{Diag}(t) \rangle = \langle V V^\top \rangle^\top t,
$$

thus obtaining

$$
\max \text{diag}(V V^\top)^\top t \quad \text{s.t.} \quad V^\top Q V \succeq V^\top \text{Diag}(t) V. \quad (16)
$$

The dual of (16) is

$$
\min \langle V^\top Q V, Y \rangle \quad \text{s.t.} \quad \text{diag}(V Y V^\top) = \text{diag}(V V^\top) \quad Y \succeq 0
$$

and thus strictly feasible. In particular, the primal problem (16) is bounded. A strictly feasible solution for (16) is given by $t := (\lambda_{\text{min}}(Q) - 1) \cdot \mathbf{1}$.

Note that it is possible to speed up the solution of the semidefinite program by increasing the tolerances. In order to guarantee a feasible $t$, it is possible to adapt any given $t_0$ by setting

$$
t := t_0 + \lambda_{\text{min}}(Q - \text{Diag}(t_0)) \cdot \mathbf{1}
$$
as discussed above, or, when taking equations into account, by
\[
  t := t_0 + \lambda_{\text{min}}(V^T(Q - \text{Diag}(t_0))V) \mathbf{1}
\]
where we use \( V^TV = I \). In Section 5, we will describe a tailored algorithm for solving problems of type (16).

Finally, note that this approach can be generalized if box constraints are taken into account as described in Section 3.5. The dual problem then becomes

\[
  \min \quad (V^TQV, Y) \\
  \text{s.t.} \quad \text{diag}(VV^T) = \text{diag}(VV^T) \\
  \quad (VYV^T)_{ij} \geq 0 \text{ if } (z_i \leq 0 \text{ and } z_j \leq 0) \text{ or } (z_i \geq 1 \text{ and } z_j \geq 1) \\
  \quad (VYV^T)_{ij} \leq 0 \text{ if } (z_i \leq 0 \text{ and } z_j \geq 0) \text{ or } (z_i \geq 1 \text{ and } z_j \leq 1) \\
  \quad Y \succeq 0.
\]

5 Fast computation of the underestimator

The special structure of the semidefinite program (16) can be exploited in order to solve it more quickly than with standard interior point methods. We adapt an algorithm devised by Dong [13] for a very similar type of SDP. It is based on a barrier approach using a coordinate descent method with exact line search.

To keep the analogy with [13], switching the sign of \( t \), we first rewrite (16) as a minimization problem:

\[
  \min \quad \text{diag}(VV^T)^Tt \\
  \text{s.t.} \quad V^T(Q + \text{Diag}(t))V \succeq 0.
\]

We introduce a penalty term \( \sigma > 0 \) and obtain

\[
  \min \quad f(t; \sigma) := \text{diag}(VV^T)^Tt - \sigma \log \det(V^T(Q + \text{Diag}(t))V) \\
  \text{s.t.} \quad V^T(Q + \text{Diag}(t))V \succ 0. \tag{17}
\]

The gradient of the objective function of (17) is

\[
  \nabla_t f(t; \sigma) = \text{diag}(VV^T)^T - \sigma \text{diag}(V[V^T(Q + \text{Diag}(t))V]^{-1}V^T).
\]

An important ingredient in our algorithm is the quick update of the matrix

\[
  W := [V^T(Q + \text{Diag}(t))V]^{-1}
\]

using the Sherman-Morrison formula and of the vector

\[
  x := \text{diag}(VWV^T).
\]

We initialize our iterative algorithm by calculating

\[
  t^{(0)} := -(1 + \varepsilon)\lambda_{\text{min}}(V^TQV) \\
  W^{(0)} := [V^T(Q + \text{Diag}(t^{(0)}))V]^{-1} \\
  x^{(0)} := \text{diag}(W^{(0)}V^T)
\]
directly, for some $\varepsilon > 0$. By construction, $t_0$ is a feasible solution for (17). Now in each iteration we improve the objective function coordinatewise, choosing a coordinate $i^{(k)}$ maximizing

$$|\nabla_i f(t; \sigma)| = |v_i^T v_i - \sigma x_i^{(k)}|.$$

We perform an exact line search along the chosen coordinate direction, i.e., we minimize $f(t^{(k)} + s e_i^{(k)})$ over the feasible region. This is equivalent to finding some $s$ satisfying

$$0 = \nabla_s f(t^{(k)} + s e_i^{(k)}) = v_i^{(k)} - \sigma i^{(k)} [V^T (Q + \text{Diag}(t^{(k)} + s e_i^{(k)}))V]^{-1} v_i^{(k)}.$$

Now

$$[V^T (Q + \text{Diag}(t^{(k)} + s e_i^{(k)}))V]^{-1} = [W^{(k)} + s v_i^{(k)} v_i^{(k)}]^{-1} = W^{(k)} - \frac{s W^{(k)} v_i^{(k)} v_i^{(k)} W^{(k)}}{1 + s v_i^{(k)} W^{(k)} v_i^{(k)}}$$

by the Sherman-Morrison formula. We obtain the unique solution

$$s^{(k)} = \frac{\sigma}{v_i^{(k)} v_i^{(k)}} = \frac{1}{x_i^{(k)}}$$

as our optimal step length. The resulting updates

$$t^{(k+1)} := t^{(k)} + s^{(k)} e_i^{(k)}$$

$$W^{(k+1)} := W^{(k)} - \frac{s^{(k)} W^{(k)} v_i^{(k)} v_i^{(k)} W^{(k)}}{1 + s^{(k)} x_i^{(k)}}$$

$$x^{(k+1)} := x^{(k)} - \frac{s^{(k)} V W^{(k)} v_i^{(k)} v_i^{(k)} W^{(k)} V^T}{1 + s^{(k)} x_i^{(k)}}$$

can now be performed in $O(n^2)$ time, using Sherman-Morrison again as well as the fact that $W^{(k)} v_i^{(k)} v_i^{(k)} W^{(k)}$ is of rank one.

In the course of this algorithm, the penalty factor $\sigma$ needs to converge to zero. In our implementation, we multiply $\sigma$ by 0.99 in each iteration. We stop the process as soon as the relative improvement of the objective value of $t^{(k)}$ according to (16) falls short of $10^{-5}$ in three consecutive iterations. Note that the iterates $t^{(k)}$ computed by this algorithm are all feasible for Problem (17) and hence for Problem (16), independently of the (current) value of $\sigma$. In particular, each iterate yields a valid underestimator. We can thus stop our algorithm at any point, which gives us the possibility to balance the running time to be spend for computing underestimators with the quality of the latter.
6 Experiments

A preliminary version of our algorithm has been evaluated in [8]. In particular, we were able to show that taking valid equations into account can improve both lower bounds and total running times significantly, even if only one equation is considered. A further observation was that the touching point $\frac{1}{2}1$ yields by far the best results out of the considered alternatives, so that we fix this choice in the following, up to the necessary projections when considering equations.

In this section, we provide a much more extensive experimental evaluation of our approach based on a much larger class of test instances from different problem types. We compare our algorithm computationally to both Cplex and the QCR method; an implementation of the latter has been provided by the authors of [4]. The main improvement of our algorithm with respect to the version tested in [8] is the new algorithm presented in Section 5. Note that taking box constraints into account, as discussed in Section 3.5, did not lead to practical improvements in our current implementation, as it requires to choose a touching point that does not belong to the interior of the box $[0, 1]^n$.

We tested our approach on various quadratic combinatorial optimization problems, as listed in the following. For each problem we provide the following information:

- a brief description of the problem,
- a mathematical model in the form of (1),
- a set of valid equations used to improve the underestimator,
- the testbed used in the experiments,
- the algorithm for solving the linear counterpart (2) of the problem.

Note that the integer programming formulations given here are just for illustration. As argued above, our approach does not rely on any integer programming model for the underlying problem.

**Quadratic Shortest Path Problem (QSPP):** We solve a generalization of the minimum (single pair) shortest path problem on a directed graph. In addition to the weights associated to each arc, we also have weights associated to the simultaneous use of pairs of arcs. The mathematical model for solving the problem on a directed graph $G = (N, A)$ is the following:

\[
\begin{align*}
\min & \quad \sum_{a,b \in A} Q_{ab} x_a x_b + \sum_{a \in A} L_a x_a \\
\text{s.t.} & \quad \sum_{a \in \delta^+(i)} x_a - \sum_{a \in \delta^-(i)} x_a = 0 & \forall i \in N \setminus \{s, t\} \\
& \quad \sum_{a \in \delta^+(s)} x_a = 1 \\
& \quad \sum_{a \in \delta^-(t)} x_a = 1 \\
& \quad x_a \in \{0, 1\} & \forall a \in A,
\end{align*}
\]

(18)

with $s$ and $t$ being the origin and destination of the shortest path and $\delta^+(i)$ (resp. $\delta^-(i)$) being the set of outgoing (resp. ingoing) arcs of a node $i$. In this case we can exploit all equations given in (18), out of which $|N| - 1$ are linearly independent (if $G$ is connected). To the best of our knowledge, no instances are available in the literature for the QSPP. We thus considered grid graphs with $k \times k$ nodes, for $k = 10, \ldots, 15$, and generated quadratic costs uniformly at random.
from \{1, \ldots, 10\}. For each size, five instances were generated with different seeds for a total of 30 instances. As black box we used the network simplex algorithm of Cplex 12.4 [12] with standard settings.

**Quadratic Assignment Problem (QAP):** The QAP is a well know generalization of the assignment problem. As for the quadratic version of the shortest path and spanning tree problem, also in this case we have an additional weight corresponding to the use of two edges in the solution. The mathematical model for solving the problem on a bipartite graph \( G = (N, E) \) is the following:

\[
\begin{align*}
\min & \sum_{e,f \in E} Q_{ef}x_ex_f + \sum_{e \in E} L_ex_e \\
\text{s.t.} & \sum_{e \in \delta(i)} x_e = 1 \quad \forall i \in N \\
& x_e \in \{0, 1\} \quad \forall e \in E.
\end{align*}
\] (19)

Also in this case we can exploit all equations given in (19), out of which \(|N| - 1\) are linearly independent. We used all instances from QAPLIB [10] on 10 or 12 nodes, so that \(|N| = 20, 24\). As black box, we reformulate the assignment problem as a min-cost-flow problem and solve it using the network simplex algorithm of Cplex 12.4, with standard settings.

**Quadratic Knapsack Problem (QKP):** The QKP is a well know generalization of the knapsack problem where in addition to profits associated to each object, we also have profits associated to the presence of pairs of objects in the solution. The mathematical model for solving the problem on a set of objects \( N \) is the following:

\[
\begin{align*}
\min & \sum_{i,j \in N} Q_{ij}x_ix_j + \sum_{i \in N} L_ix_i \\
\text{s.t.} & \sum_{i \in N} c_ix_i \leq k \\
& x_i \in \{0, 1\} \quad \forall i \in N
\end{align*}
\] (20)

with \(c_i\) being the weight of object \(i \in N\) and \(k\) being the capacity. No equations can be exploited in this case. We use the library of QKP instances with 100 items proposed in [5], for a total of 40 instances. As black box, we implemented the well-known dynamic programming procedure for the linear knapsack problem; see, e.g., [15]. Note that we model QKP as a minimization problem here, in order to stay analogous to the other problem classes.

In [8], we also considered unconstrained binary quadratic programs (UBQP) and instances of the quadratic spanning tree problem (QSTP). Note that any LP based approach to spanning tree problems suffers from the fact that the classical integer programming formulation of the latter requires an exponential number of linear inequalities and hence a separation algorithm. For this reason, applying Cplex or the QCR method is not a promising approach. On contrary, our new method does not rely on any LP formulation and the linear problem (2) can be solved by any combinatorial algorithm such as Kruskal’s algorithm [16]. For this reason, we do not include a comparison here.

For all tests reported in the following, we used Intel Xeon E5-2670 processors, running at 2.60 GHz with 64 GB of RAM. All running times are stated in cpu seconds. The running time for each instance is limited to 3 cpu hours.
6.1 Lower bound comparison

In this section, we compare the value of the lower bound in the root node of the branch-and-bound tree obtained by our algorithm with the root bounds obtained by Cplex and the QCR algorithm. The results concerning QSPP, QAP and QKP are provided in Tables 1, 2 and 3. All tables are vertically divided into four blocks: in the first block, we state the name of the instance and its dimension. In the next blocks, we report lower bounds given by Cplex, by the QCR method, and by our algorithm. As all problems we consider are purely binary with integer coefficients, we round up the values obtained to the closest integer value. Finally, we state the optimal value of the instance. If none of the three algorithms tested was able to solve to optimality an instance, we report the best integer solution found and we mark the value with an $\ast$.

When compared to Cplex, our algorithm provides much better bounds for the QSPP and QAP and comparable bounds for the QKP. On the other hand, for all the instances considered, our bounds are considerably weaker than the ones provided by the QCR method. Nevertheless, we will see in the following section that the picture changes when considering total running times. Comparing lower bounds with optimal solutions, it turns out that all approaches behave similarly: root bounds are weak for QSPP and desastrous for QAP instances, whereas all approaches compute rather strong bounds for the given QKP instances.

<table>
<thead>
<tr>
<th>Inst</th>
<th>n</th>
<th>Cplex LB</th>
<th>QCR LB</th>
<th>Our LB</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10_1</td>
<td>180</td>
<td>206</td>
<td>485</td>
<td>357</td>
<td>621</td>
</tr>
<tr>
<td>10_2</td>
<td>180</td>
<td>211</td>
<td>501</td>
<td>323</td>
<td>635</td>
</tr>
<tr>
<td>10_3</td>
<td>180</td>
<td>217</td>
<td>498</td>
<td>367</td>
<td>636</td>
</tr>
<tr>
<td>10_4</td>
<td>180</td>
<td>209</td>
<td>491</td>
<td>359</td>
<td>661</td>
</tr>
<tr>
<td>10_5</td>
<td>180</td>
<td>233</td>
<td>504</td>
<td>367</td>
<td>665</td>
</tr>
<tr>
<td>11_1</td>
<td>220</td>
<td>251</td>
<td>609</td>
<td>420</td>
<td>815</td>
</tr>
<tr>
<td>11_2</td>
<td>220</td>
<td>251</td>
<td>533</td>
<td>417</td>
<td>788</td>
</tr>
<tr>
<td>11_3</td>
<td>220</td>
<td>225</td>
<td>592</td>
<td>384</td>
<td>795</td>
</tr>
<tr>
<td>11_4</td>
<td>220</td>
<td>236</td>
<td>619</td>
<td>402</td>
<td>782</td>
</tr>
<tr>
<td>11_5</td>
<td>220</td>
<td>228</td>
<td>582</td>
<td>404</td>
<td>767</td>
</tr>
<tr>
<td>12_1</td>
<td>264</td>
<td>271</td>
<td>711</td>
<td>475</td>
<td>959</td>
</tr>
<tr>
<td>12_2</td>
<td>264</td>
<td>282</td>
<td>707</td>
<td>524</td>
<td>963</td>
</tr>
<tr>
<td>12_3</td>
<td>264</td>
<td>259</td>
<td>647</td>
<td>491</td>
<td>900</td>
</tr>
<tr>
<td>12_4</td>
<td>264</td>
<td>236</td>
<td>698</td>
<td>481</td>
<td>960</td>
</tr>
<tr>
<td>12_5</td>
<td>264</td>
<td>289</td>
<td>701</td>
<td>479</td>
<td>976</td>
</tr>
<tr>
<td>13_1</td>
<td>312</td>
<td>335</td>
<td>805</td>
<td>586</td>
<td>1159</td>
</tr>
<tr>
<td>13_2</td>
<td>312</td>
<td>333</td>
<td>821</td>
<td>590</td>
<td>1176</td>
</tr>
<tr>
<td>13_3</td>
<td>312</td>
<td>325</td>
<td>822</td>
<td>558</td>
<td>1164</td>
</tr>
<tr>
<td>13_4</td>
<td>312</td>
<td>301</td>
<td>845</td>
<td>568</td>
<td>1110</td>
</tr>
<tr>
<td>13_5</td>
<td>312</td>
<td>322</td>
<td>842</td>
<td>567</td>
<td>1115</td>
</tr>
<tr>
<td>14_1</td>
<td>364</td>
<td>364</td>
<td>959</td>
<td>680</td>
<td>1563</td>
</tr>
<tr>
<td>14_2</td>
<td>364</td>
<td>357</td>
<td>963</td>
<td>669</td>
<td>1367</td>
</tr>
<tr>
<td>14_3</td>
<td>364</td>
<td>334</td>
<td>934</td>
<td>651</td>
<td>1320</td>
</tr>
<tr>
<td>14_4</td>
<td>364</td>
<td>348</td>
<td>982</td>
<td>661</td>
<td>1347</td>
</tr>
<tr>
<td>14_5</td>
<td>364</td>
<td>354</td>
<td>949</td>
<td>704</td>
<td>1344</td>
</tr>
<tr>
<td>15_1</td>
<td>420</td>
<td>412</td>
<td>1099</td>
<td>806</td>
<td>1588</td>
</tr>
<tr>
<td>15_2</td>
<td>420</td>
<td>419</td>
<td>1067</td>
<td>762</td>
<td>1561</td>
</tr>
<tr>
<td>15_3</td>
<td>420</td>
<td>386</td>
<td>1061</td>
<td>744</td>
<td>1569</td>
</tr>
<tr>
<td>15_4</td>
<td>420</td>
<td>380</td>
<td>1084</td>
<td>791</td>
<td>1582</td>
</tr>
</tbody>
</table>

Table 1 Lower bound comparison for QSPP

6.2 Running time comparison

We next compare our algorithm with Cplex and with the QCR method in terms of total running times for solving the instances to optimality. The comparison for QSPP, QAP and QKP is presented in Tables 4, 5, and 6, respectively. In all
Table 2 Lower bound comparison for QAP

<table>
<thead>
<tr>
<th>inst</th>
<th>n</th>
<th>Cplex LB</th>
<th>QCR LB</th>
<th>Our LB</th>
<th>OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>tai10a</td>
<td>100</td>
<td>-50985</td>
<td>32225</td>
<td>-13475</td>
<td>135028</td>
</tr>
<tr>
<td>tai10b</td>
<td>100</td>
<td>-153757</td>
<td>225336</td>
<td>-5972174</td>
<td>1183760</td>
</tr>
<tr>
<td>chr12a</td>
<td>144</td>
<td>-416796</td>
<td>-103848</td>
<td>-217851</td>
<td>9552</td>
</tr>
<tr>
<td>chr12b</td>
<td>144</td>
<td>-419180</td>
<td>-86164</td>
<td>-177090</td>
<td>11156</td>
</tr>
<tr>
<td>chr12c</td>
<td>144</td>
<td>-415954</td>
<td>-99747</td>
<td>-179110</td>
<td>11156</td>
</tr>
<tr>
<td>had12</td>
<td>144</td>
<td>-7041</td>
<td>1198</td>
<td>672</td>
<td>1652</td>
</tr>
<tr>
<td>nug12</td>
<td>144</td>
<td>-3484</td>
<td>-216</td>
<td>779</td>
<td>578</td>
</tr>
<tr>
<td>rou12</td>
<td>144</td>
<td>-88198</td>
<td>76922</td>
<td>-313</td>
<td>235852</td>
</tr>
<tr>
<td>scr12</td>
<td>144</td>
<td>-423727</td>
<td>-111147</td>
<td>-227213</td>
<td>31410</td>
</tr>
<tr>
<td>tai12a</td>
<td>144</td>
<td>-1029578</td>
<td>77893</td>
<td>-27238</td>
<td>224416</td>
</tr>
<tr>
<td>tai12b</td>
<td>144</td>
<td>-621081</td>
<td>-134673000</td>
<td>-402161178</td>
<td>39464925</td>
</tr>
</tbody>
</table>

Table 3 Lower bound comparison for QKP

<table>
<thead>
<tr>
<th>inst</th>
<th>n</th>
<th>Cplex LB</th>
<th>QCR LB</th>
<th>Our LB</th>
<th>BEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>100_1</td>
<td>100</td>
<td>-21395</td>
<td>-16875</td>
<td>-24756</td>
<td>-18558</td>
</tr>
<tr>
<td>100_2</td>
<td>100</td>
<td>-58465</td>
<td>-57451</td>
<td>-60485</td>
<td>-56525</td>
</tr>
<tr>
<td>100_3</td>
<td>100</td>
<td>-7794</td>
<td>-6194</td>
<td>-14986</td>
<td>-3752</td>
</tr>
<tr>
<td>100_4</td>
<td>100</td>
<td>-54204</td>
<td>-53143</td>
<td>-7266</td>
<td>-12928</td>
</tr>
<tr>
<td>100_5</td>
<td>100</td>
<td>-62410</td>
<td>-61746</td>
<td>-63137</td>
<td>-61494</td>
</tr>
<tr>
<td>100_6</td>
<td>100</td>
<td>-40748</td>
<td>-39868</td>
<td>-47826</td>
<td>-36360</td>
</tr>
<tr>
<td>100_7</td>
<td>100</td>
<td>-19300</td>
<td>-18321</td>
<td>-29519</td>
<td>-14657</td>
</tr>
<tr>
<td>100_8</td>
<td>100</td>
<td>-24870</td>
<td>-24135</td>
<td>-36474</td>
<td>-204552</td>
</tr>
<tr>
<td>100_9</td>
<td>100</td>
<td>-39664</td>
<td>-38922</td>
<td>-46032</td>
<td>-35438</td>
</tr>
<tr>
<td>100_10</td>
<td>100</td>
<td>-30081</td>
<td>-29596</td>
<td>-38858</td>
<td>-24930</td>
</tr>
<tr>
<td>100_1</td>
<td>50</td>
<td>-92176</td>
<td>-95449</td>
<td>-104435</td>
<td>-88611</td>
</tr>
<tr>
<td>100_2</td>
<td>50</td>
<td>-112191</td>
<td>-109655</td>
<td>-117406</td>
<td>-105996</td>
</tr>
<tr>
<td>100_3</td>
<td>50</td>
<td>-44140</td>
<td>-42695</td>
<td>-66364</td>
<td>-34006</td>
</tr>
<tr>
<td>100_4</td>
<td>50</td>
<td>-111322</td>
<td>-109162</td>
<td>-110670</td>
<td>-108056</td>
</tr>
<tr>
<td>100_5</td>
<td>50</td>
<td>-66171</td>
<td>-65104</td>
<td>-84867</td>
<td>-56464</td>
</tr>
<tr>
<td>100_6</td>
<td>50</td>
<td>-26588</td>
<td>-23914</td>
<td>-42152</td>
<td>-16893</td>
</tr>
<tr>
<td>100_7</td>
<td>50</td>
<td>-63788</td>
<td>-62824</td>
<td>-82084</td>
<td>-54246</td>
</tr>
<tr>
<td>100_8</td>
<td>50</td>
<td>-79085</td>
<td>-77998</td>
<td>-93732</td>
<td>-69974</td>
</tr>
<tr>
<td>100_9</td>
<td>50</td>
<td>-36117</td>
<td>-32902</td>
<td>-59888</td>
<td>-19570</td>
</tr>
<tr>
<td>100_10</td>
<td>50</td>
<td>-120973</td>
<td>-119196</td>
<td>-141193</td>
<td>-104341</td>
</tr>
</tbody>
</table>

Our new algorithm is able to solve to optimality all instances of QSPP and QAP and 12 out of 40 instances of QKP, whereas Cplex is able to solve only 2 instances of QAP and 9 instances of QKP, while it cannot solve the biggest instances of QSPP. When both approaches are able to solve an instance to optimality within the time limit, our algorithm is about 2 to 5 times faster. Also in comparison with the QCR algorithm, our algorithm is faster in solving QAP and QSPP instances and comparable in terms of time in solving the QKP instances. Moreover, it is able to solve two more instance of QKP than the QCR method. Interestingly, running times can differ drastically in both directions, particularly for the QKP instances, we report for each algorithm the number of branch-and-bound nodes and the overall computing time (TL if the time limit is reached). In the last columns, we report the corresponding ratios between the values obtained by Cplex or QCR and ours, where a value lower than one means an improvement in terms of number of nodes or computing time. Here, the time limit is used as running time for non-solved instances.
instances, showing that both algorithms have very different strengths. The best relative results of our algorithm are obtained for the QAP instances, where for the instances with 144 variables on average we only spend 4% of the running time of QCR, while Cplex always runs into the time limit.

7 Conclusion

We proposed a generic framework for solving binary quadratic programming problems that exploits the underlying combinatorial structure. The bounding procedure uses an integer linear relaxation of the original problem based on separable underestimators and it proved to have a good trade-off between quality of the bound provided and computing time. Our algorithm is effective in solving several classes of quadratic 0–1 problems and is the first exact approach available in the literature for the quadratic shortest path problem. The use of the proposed algorithm is recommended in any situation where an efficient algorithm is known for solving the linear counterpart of the problem.

Table 4 Computational results for QSP

<table>
<thead>
<tr>
<th>Inst</th>
<th>Cplex</th>
<th>QCR</th>
<th>Our Alg</th>
<th>Ratios wrt Cplex</th>
<th>Ratios wrt QCR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nodes</td>
<td>time</td>
<td>nodes</td>
<td>time</td>
<td>nodes</td>
</tr>
<tr>
<td>10_1</td>
<td>180</td>
<td>1.00e+4</td>
<td>5.3e+6</td>
<td>4.8</td>
<td>5.3e+6</td>
</tr>
<tr>
<td>10_2</td>
<td>180</td>
<td>1.02e+4</td>
<td>13.3</td>
<td>3.3e+4</td>
<td>4.2</td>
</tr>
<tr>
<td>10_3</td>
<td>180</td>
<td>1.01e+4</td>
<td>13.0</td>
<td>3.9e+4</td>
<td>4.2</td>
</tr>
<tr>
<td>10_4</td>
<td>180</td>
<td>1.67e+4</td>
<td>16.2</td>
<td>9.6e+4</td>
<td>9.8</td>
</tr>
<tr>
<td>10_5</td>
<td>180</td>
<td>1.46e+4</td>
<td>18.0</td>
<td>6.7e+4</td>
<td>5.3</td>
</tr>
<tr>
<td>11_1</td>
<td>220</td>
<td>4.10e+4</td>
<td>69.4</td>
<td>9.5e+3</td>
<td>32.3</td>
</tr>
<tr>
<td>11_2</td>
<td>220</td>
<td>3.80e+4</td>
<td>66.1</td>
<td>9.0e+3</td>
<td>31.3</td>
</tr>
<tr>
<td>11_3</td>
<td>220</td>
<td>4.07e+4</td>
<td>59.0</td>
<td>9.2e+3</td>
<td>27.3</td>
</tr>
<tr>
<td>11_4</td>
<td>220</td>
<td>3.04e+4</td>
<td>59.2</td>
<td>3.1e+3</td>
<td>20.5</td>
</tr>
<tr>
<td>11_5</td>
<td>220</td>
<td>2.58e+4</td>
<td>47.2</td>
<td>7.8e+3</td>
<td>25.4</td>
</tr>
</tbody>
</table>

Table 5 Computational results for QAP

<table>
<thead>
<tr>
<th>Inst</th>
<th>Cplex</th>
<th>QCR</th>
<th>Our Alg</th>
<th>Ratios wrt Cplex</th>
<th>Ratios wrt QCR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nodes</td>
<td>time</td>
<td>nodes</td>
<td>time</td>
<td>nodes</td>
</tr>
<tr>
<td>10_1</td>
<td>180</td>
<td>1.00e+4</td>
<td>15.6</td>
<td>6.2e+2</td>
<td>9.4</td>
</tr>
<tr>
<td>10_2</td>
<td>180</td>
<td>1.02e+4</td>
<td>13.3</td>
<td>1.5e+3</td>
<td>8.7</td>
</tr>
<tr>
<td>10_3</td>
<td>180</td>
<td>1.01e+4</td>
<td>13.0</td>
<td>2.0e+4</td>
<td>8.5</td>
</tr>
<tr>
<td>10_4</td>
<td>180</td>
<td>1.67e+4</td>
<td>16.2</td>
<td>6.3e+4</td>
<td>9.8</td>
</tr>
<tr>
<td>10_5</td>
<td>180</td>
<td>1.46e+4</td>
<td>18.0</td>
<td>3.2e+4</td>
<td>9.7</td>
</tr>
<tr>
<td>11_1</td>
<td>220</td>
<td>4.10e+4</td>
<td>69.4</td>
<td>9.5e+3</td>
<td>32.3</td>
</tr>
<tr>
<td>11_2</td>
<td>220</td>
<td>3.80e+4</td>
<td>66.1</td>
<td>9.0e+3</td>
<td>31.3</td>
</tr>
<tr>
<td>11_3</td>
<td>220</td>
<td>4.07e+4</td>
<td>59.0</td>
<td>9.2e+3</td>
<td>27.3</td>
</tr>
<tr>
<td>11_4</td>
<td>220</td>
<td>3.04e+4</td>
<td>59.2</td>
<td>3.1e+3</td>
<td>20.5</td>
</tr>
<tr>
<td>11_5</td>
<td>220</td>
<td>2.58e+4</td>
<td>47.2</td>
<td>7.8e+3</td>
<td>25.4</td>
</tr>
</tbody>
</table>

Acknowledgments

This work has been supported by the German Research Foundation (DFG) under grant BU 2313/4. The authors would like to thank Antonio Frangioni for fruitful discussions and suggestions that improved the present paper significantly. Moreover, they are grateful to Sourour Elloumi for providing her implementation of the QCR method.

References


Table 6: Computational results for QKP