Submitted to *INFORMS Journal on Computing*
manuscript (Please, provide the manuscript number!)

Quadratic combinatorial optimization using separable underestimators

Christoph Buchheim
Fakultät für Mathematik, Technische Universität Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany,
christoph.buchheim@tu-dortmund.de

Emiliano Traversi
LIPN, Université Paris 13, 99 Avenue Jean-Baptiste Clément, 93430 Villetaneuse, France,
emiliano.traversi@lipn.univ-paris13.fr

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 binary 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 adapt an algorithm of Dong (2014) in order to obtain 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 as well as the SDP-based software BiqCrunch on instances of the quadratic shortest path problem and the quadratic assignment problem.

Key words: Binary Quadratic Optimization, Separable Underestimators, Quadratic Shortest Path Problem

1. Introduction

We consider binary quadratic optimization problems of the form

$$\begin{aligned} \min \quad & f(x) := x^\top Q x + L^\top x \\ \text{s.t.} \quad & x \in X\,, \end{aligned} \tag{1}$$

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 (Amaldi et al. 2011, Gamvros et al. 2012), the angular metric TSP (Aggarwal et al. 1999), or crossing

minimization problems for bipartite graphs (Buchheim et al. 2010). In this paper, we focus on combinatorial optimization problems where the linear counterpart of Problem (1),

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & x \in X\,, \end{aligned} \tag{2}$$

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 $\operatorname{conv}(X)$ are required. From a practical point of view, our approach can also be useful if Problem (2) is NP-hard but significantly easier to solve than Problem (1), e.g., for the quadratic knapsack problem. Note that the quadratic problem (1) is usually NP-hard even if the underlying linear problem (2) is tractable. 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 (Assad and Xu 1992), while the linear counterpart can be solved very quickly, e.g., by Kruskal's algorithm (Kruskal 1956).

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 (Palagi et al. 2012) 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. (Caprara 2008). A different approach to binary optimization is the QCR technique (Billionnet et al. 2009). 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. Finally, outer approximation techniques have been devised, but mostly for more general (mixed-)integer versions of (1) and often assuming convexity; see, e.g., (Bonami et al. 2008, Buchheim and Trieu 2013).

In this paper, we address Problem (1) by computing underestimators g of the quadratic objective function f. A lower bound for (1) can then be computed by minimizing $g(x)$ over $x \in X$. Unlike the QCR approach, 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 replace the separable quadratic function by 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. Compared with linearization-based methods, 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 $\mathrm{conv}\,(X)$ and do not use any LP solver at all. At the same time, any algorithmic knowledge about the linear problem (2) is exploited directly.

In order to obtain a fast branch-and-bound scheme, we compute underestimators that only depend on the matrix Q, but not on the linear part L of the objective function. Using an appropriate branching scheme as already proposed in (Buchheim et al. 2012, 2013), 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. For this, we developed a tailored coordinate-descent algorithm exploiting the particular structure of the latter, which is an adaptation of an algorithm proposed by Dong (2014).

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 possible 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 underestimators is sketched in Section 5. In Section 6, we evaluate our approach computationally, applying it to instances of the quadratic assignment problem, the quadratic knapsack problem and the quadratic shortest path problem. We compare our approach with Cplex, with the QCR method, and with BiqCrunch.

2. Notation and basic idea

Our aim is to derive a lower bound for Problem (1) by solving the linear problem (2) for an appropriate vector $c \in \mathbb{R}^n$. To this end, for an arbitrary $z \in \mathbb{R}^n$, we rewrite $f(x)$ as

$$f(x) = (x-z)^\top Q(x-z) + (L+2Qz)^\top x - z^\top Qz\,. \tag{3}$$

For a given vector $t \in \mathbb{R}^n$, define

$$\begin{aligned} g_z^{(t)}(x) &:= (x-z)^\top \mathrm{Diag}(t)(x-z) + (L+2Qz)^\top x - z^\top Qz \\ &= \sum_{i=1}^n t_i x_i^2 + \sum_{i=1}^n (-2z_i t_i + l_i + 2q_i^\top z)x_i + \sum_{i=1}^n z_i^2 t_i - z^\top Qz\,, \end{aligned}$$

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 $Q \succeq \mathrm{Diag}(t)$. In this case, the desired lower bound can be obtained as

$$\min\ g_z^{(t)}(x) \quad \text{s.t. } x \in X\,. \tag{4}$$

Since $g_z^{(t)}$ is separable by construction and $X \subseteq \{0,1\}^n$, we can replace Problem (4) by the equivalent problem

$$\min\ l_z^{(t)}(x) \quad \text{s.t. } x \in X \tag{5}$$

where the function

$$\begin{aligned} l_z^{(t)}(x) &:= \sum_{i=1}^n t_i x_i + \sum_{i=1}^n (-2z_i t_i + l_i + 2q_i^\top z)x_i + \sum_{i=1}^n z_i^2 t_i - z^\top Qz \\ &= ((\mathbf{1}-2z)\cdot t + L + 2Qz)^\top x + (z^2)^\top t - z^\top Qz \end{aligned}$$

is bilinear in $x, t \in \mathbb{R}^n$. Here we use $\cdot$ to denote entrywise multiplication and define $z^2 := z \cdot z$. 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 $Q \succeq \mathrm{Diag}(t)$. Our objective is to maximize the lower bound induced by t. In other words, our aim is to solve the problem

$$\begin{aligned} \max\ & \min\nolimits_{x\in X} l_z^{(t)}(x) \\ \text{s.t. }\ & Q \succeq \mathrm{Diag}(t)\,. \end{aligned} \tag{6}$$

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 $\mathrm{diag}(Q)$ and L, provided that the touching point z remains unchanged.

3.1. Invariance under reformulation of the objective function

As we have $X \subseteq \{0,1\}^n$, minimizing $f(x)$ over $x \in X$ is equivalent to minimizing

$$f_\alpha(x) = x^\top (Q + \mathrm{Diag}(\alpha))x + (L - \alpha)^\top x$$

over $x \in X$, for any $\alpha \in \mathbb{R}^n$. One may ask whether the choice of α has an effect on the lower bounds computed above. The answer is no as long as z does not depend on α. To show this, let $g^{(t)}_{z,\alpha}$ denote the family of underestimators of f_α as constructed in Section 2.

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

$$\begin{aligned} \max \quad & \min_{x \in X} g^{(t)}_{z,\alpha}(x) \\ \text{s.t.} \quad & Q + \mathrm{Diag}(\alpha) \succeq \mathrm{Diag}(t) \end{aligned} \tag{7}$$

does not depend on α.

Proof. Choose any $\alpha \in \mathbb{R}^n$. For each $x \in X$, we have $x^\top \mathrm{Diag}(\alpha) x = \alpha^\top x$, which implies

$$g^{(t)}_{z,\alpha}(x) = g^{(t-\alpha)}_z(x)\,.$$

Consequently, Problem (7) is equivalent to

$$\begin{aligned} \max \quad & \min_{x \in X} g^{(t-\alpha)}_z(x) \\ \text{s.t.} \quad & Q \succeq \mathrm{Diag}(t - \alpha) \end{aligned}$$

and hence to (6) by translation. □

However, if z depends on α, this result does not hold true in general. To see this, consider the trivial case where X contains only the zero vector, so that

$$\min_{x \in X} g^{(t)}_{z,\alpha}(x) = g^{(t)}_{z,\alpha}(0) = z^\top (\mathrm{Diag}(t - \alpha) - Q) z\,.$$

Then (7) reduces to

$$\begin{aligned} \max \quad & z^\top (\mathrm{Diag}(t - \alpha) - Q) z \\ \text{s.t.} \quad & Q - \mathrm{Diag}(t - \alpha) \succeq 0\,. \end{aligned} \tag{8}$$

Now, for instance, consider the bivariate function $f_\alpha(x)$ with

$$Q = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad L = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

In this case, it can be verified that the optimal value of (8) and hence of (7) is $-4z_1z_2$, it thus depends on the choice of z. In particular, if z is chosen as the stationary point of f, the result of Theorem 1 does not hold any more.

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 $Q \succeq \mathrm{Diag}(t)$ 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*

$$\begin{aligned} \max \quad & \min_{x\in X} l_z^{(t)}(x) + \mu \min\{0, \lambda_{\min}(Q - \mathrm{Diag}(t))\} \\ \text{s.t.} \quad & t \in \mathbb{R}^n . \end{aligned} \tag{9}$$

Proof. It is clear that the optimal value of (9) 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 (9). If $Q - \mathrm{Diag}(t^*) \succeq 0$, then t^* is also feasible for (6) with the same objective function value, so we may assume $\lambda_{\min}(Q - \mathrm{Diag}(t^*)) < 0$. Consider

$$\bar{t} := t^* + \lambda_{\min}(Q - \mathrm{Diag}(t^*))\mathbf{1} .$$

By construction, $\bar{t}$ is a feasible solution for (6). For each $x \in X$, we have

$$\begin{aligned} l_z^{(\bar{t})}(x) &= (x-z)^\top \mathrm{Diag}(\bar{t})(x-z) + (L+2Qz)^\top x - z^\top Q z \\ &= (x-z)^\top \mathrm{Diag}(t^*)(x-z) + (L+2Qz)^\top x - z^\top Q z + \lambda_{\min}(Q - \mathrm{Diag}(t^*))||x-z||^2 \\ &\geq l_z^{(t^*)}(x) + \mu\lambda_{\min}(Q - \mathrm{Diag}(t^*)) \end{aligned}$$

and hence

$$\min_{x\in X} l_z^{(\bar{t})}(x) \geq \min_{x\in X} l_z^{(t^*)}(x) + \mu\lambda_{\min}(Q - \mathrm{Diag}(t^*)) ,$$

which shows that the value of $\bar{t}$ in the objective function of (6) is greater or equal to the value of t^* in the objective function of (9). As $\bar{t}$ is feasible for (6) and t^* is optimal for (9), we obtain that the optimal value of (6) is greater or equal to the optimal value of (9). □

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 $z = \frac{1}{2}\mathbf{1}$, 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 (9) is concave, so that a subgradient approach can be used to solve the problem efficiently. The supergradient of

$$\min_{x\in X} \; l_z^{(t)}(x)$$

at a given point t^k can be computed by using the black box (2), as $l_z^{(t^k)}(x)$ is a linear function in x. Given the optimal solution $\hat{x}^k$, the desired supergradient is the gradient of $l_z^{(t)}(\hat{x}^k)$, which is easily computed since $l_z^{(t)}(\hat{x}^k)$ is a linear function also in t, thus

$$\nabla_t \, l_z^{(t)}(\hat{x}^k) = (\mathbf{1} - 2z) \cdot \hat{x}^k + (z^2) \, .$$

A supergradient of the penalty term $\mu \min\{0, \lambda_{\min}(Q - \mathrm{Diag}(t))\}$ in t^k can be calculated as follows: if $\lambda_{\min}(Q - \mathrm{Diag}(t^k)) \geq 0$, the zero vector is a feasible choice. Otherwise, we need to find a supergradient of $\lambda_{\min}(Q - \mathrm{Diag}(t))$ in t^k, for this we can use $-\mu v^2$, where v is a normalized eigenvector corresponding to the eigenvalue $\lambda_{\min}(Q - \mathrm{Diag}(t^k))$. The subgradient algorithm is summarized in Algorithm 1; see also Buchheim and Traversi (2013).

Note that Algorithm 1 can be stopped at any time and the best t^k obtained so far can be used in order to obtain a feasible solution

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

as already observed in the proof of Theorem 2.

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

$$l_z^{(t)}(x) = \frac{1}{4}\mathbf{1}^\top t + (L + Q\mathbf{1})^\top x - \frac{1}{4}\mathbf{1}^\top Q\mathbf{1} \, .$$

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

$$\begin{array}{lcl} \max & \frac{1}{4}\mathbf{1}^\top t & + \quad \min \quad (L + Q\mathbf{1})^\top x \quad - \quad \frac{1}{4}\mathbf{1}^\top Q\mathbf{1} \\ \text{s.t.} & Q \succeq \mathrm{Diag}(t) & \quad \text{s.t.} \quad x \in X \end{array}$$

Algorithm 1: Subgradient algorithm for computing an optimal underestimator

input : function f, set X, touching point z, penalty parameter μ,
procedure for solving Problem (2)

output: a (near-)optimal solution to Problem (6)

$t^0 \leftarrow \lambda_{\min}(Q)\mathbf{1}$;
$k \leftarrow 0$, STOP$\leftarrow$ false;
while STOP = false **do**
  solve $\min_{x\in X} l_z^{(t^k)}(x)$ using the black box, let $\hat{x}^k$ be the optimal solution;
  $\Delta t^k \leftarrow \big(\nabla_t\, l_z^{(t)}(\hat{x}^k)\big)(t^k)$;
  $\lambda \leftarrow \lambda_{\min}(Q - \mathrm{Diag}(t^k))$;
  if $\lambda < 0$ **then**
    choose normalized eigenvector v of $Q - \mathrm{Diag}(t^k)$ to eigenvalue λ;
    $\Delta t^k \leftarrow \Delta t^k - \mu v^2$;
  end
  if $\Delta t^k \approx 0$ **then**
    STOP$\leftarrow$true;
  else
    $t^{k+1} \leftarrow t^k + \Delta t^k$; $k \leftarrow k+1$;
  end
end

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:

$$\begin{aligned} &\max\ \beta \\ &\text{s.t. } \beta \le l_z^{(t)}(x) \quad \forall x \in X \\ &\qquad Q \succeq \mathrm{Diag}(t)\,. \end{aligned} \tag{10}$$

All constraints in (10) except for the last one are linear. The corresponding separation problem is of type (2) again, as it amounts to checking whether

$$l_z^{(t^*)}(x) < \beta^*$$

for some $x \in X$, for fixed t^* and β^*. The dual of (10) turns out to be

$$\begin{aligned} \min \quad & \langle Q, Y \rangle + \textstyle\sum_{x \in X} \lambda_x ((L + 2Qz)^\top x) - z^\top Q z \\ \text{s.t.} \quad & \operatorname{diag}(Y) = \textstyle\sum_{x \in X} \lambda_x (x - z)^2 \\ & \textstyle\sum_{x \in X} \lambda_x = 1, \ \lambda_x \geq 0 \ \forall x \in X \\ & Y \succeq 0 \,. \end{aligned} \tag{11}$$

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 (11) simplifies to

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

which is equivalent to

$$\begin{aligned} \min \quad & \langle Q, Y \rangle + L^\top x \\ \text{s.t.} \quad & \operatorname{diag}(Y) = x \\ & x \in \operatorname{conv} X \\ & Y \succeq 0 \,. \end{aligned} \tag{12}$$

This formulation is very similar to the standard SDP relaxation for constrained binary quadratic optimization, which can be written as

$$\begin{aligned} \min \quad & \langle Q, Y \rangle + L^\top x \\ \text{s.t.} \quad & \operatorname{diag}(Y) = x \\ & x \in \operatorname{conv} X \\ & \begin{pmatrix} 1 & x^\top \\ x & Y \end{pmatrix} \succeq 0 \,, \end{aligned} \tag{13}$$

assuming that a complete polyhedral description of $\operatorname{conv} X$ is known. This shows that our relaxation is dominated by the standard SDP relaxation in the case $z = 0$, as the SDP constraint in (13) implies the SDP constraint in (12), and the same holds for any other binary point $z \in \{0,1\}^n$ by symmetry.

Unfortunately, the stronger SDP constraint of (13) cannot easily be generalized to other choices of z. However, when using $z = \frac{1}{2}\mathbf{1}$ as touching point, the situation changes. The problem then decomposes again and (11) simplifies to

$$\begin{aligned} \min \quad & \langle Q, Y \rangle + \textstyle\sum_{x \in X} \lambda_x ((L + Q\mathbf{1})^\top x) - \frac{1}{4}\mathbf{1}^\top Q \mathbf{1} \\ \text{s.t.} \quad & \operatorname{diag}(Y) = \textstyle\sum_{x \in X} \lambda_x \frac{1}{4}\mathbf{1} \\ & \textstyle\sum_{x \in X} \lambda_x = 1, \ \lambda_x \geq 0 \ \forall x \in X \\ & Y \succeq 0 \,, \end{aligned}$$

which is equivalent to

$$\begin{aligned} \min \quad & \tfrac{1}{4}\langle Q, Y \rangle + \min_{x \in X} (L + Q\mathbf{1})^\top x - \tfrac{1}{4}\mathbf{1}^\top Q \mathbf{1} \\ \text{s.t.} \quad & \operatorname{diag}(Y) = \mathbf{1} \\ & Y \succeq 0 \,. \end{aligned} \tag{14}$$

In this case, there is no dominance between the standard SDP relaxation (13) and our relaxation (6) being equivalent to (14): the disadvantage of (14) lies in the fact that the SDP constraint only takes the quadratic part of the objective function into account, which generally leads to weaker relaxations. On the other hand, in the special case $Q = 0$, our relaxation trivially yields the optimal value of (1), and by Theorem 1 this remains true

if Q is any diagonal matrix. This just reflects the main idea of our approach, namely to address the underlying linear problem by exact methods. None of the SDP based relaxations discussed in the literature will achieve this in general.

3.4. Feasible sets of low dimension

As already discussed in (Buchheim and Traversi 2013), 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 any $w \in \mathbb{R}^n$ with $Aw = b$. Then $g_z^{(t)}$ underestimates f on the affine subspace given by $Ax = b$ if and only if

$$V^\top Q V \succeq V^\top \mathrm{Diag}(t) V \,.$$

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

In our experimental results presented in (Buchheim and Traversi 2013), it turns out that this replacement can improve both the bounds and the total running times significantly, even if only one equation is taken into account, as in the case of the quadratic spanning tree problem.

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

$$\begin{aligned} & g_z^{(t)}(x) \le f(x) \ \forall x \in \{0,1\}^n \\ \Leftrightarrow\ & (x-z)^\top (Q - \mathrm{Diag}(t))(x-z) \ge 0 \ \forall x \in \{0,1\}^n \,. \end{aligned}$$

A sufficient condition is

$$y^\top (Q - \mathrm{Diag}(t)) y \ge 0 \ \forall y \in \mathbb{R}^n \colon y_i \ge 0 \text{ if } z_i \le 0,\ y_i \le 0 \text{ if } z_i \ge 1 \,. \tag{15}$$

This condition is also necessary if $z \in [0,1]^n$. In particular, when choosing the touching point as $z = \frac{1}{2}\mathbf{1}$, this shows that we need to enforce $Q - \mathrm{Diag}(t) \succeq 0$ 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 $Q - \mathrm{Diag}(t)$ 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 (15) by the stronger condition that $Q - \mathrm{Diag}(t)$ be a sum of a positive semidefinite matrix and a symmetric matrix N such that

$$\begin{aligned} &N_{ij} \geq 0 \text{ if } (z_i \leq 0 \text{ and } z_j \leq 0) \text{ or } (z_i \geq 1 \text{ and } z_j \geq 1) \\ &N_{ij} \leq 0 \text{ if } (z_i \leq 0 \text{ and } z_j \geq 1) \text{ or } (z_i \geq 1 \text{ and } z_j \leq 0) \\ &N_{ij} = 0 \text{ otherwise.} \end{aligned} \tag{16}$$

In other words, we require $Q - \mathrm{Diag}(t) - N \succeq 0$ for some matrix N satisfying the conditions (16). In general, this requirement is less strict than $Q - \mathrm{Diag}(t) \succeq 0$, 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

$$V^\top (Q - \mathrm{Diag}(t) - N)V \succeq 0$$

for some N that satisfies (16). 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 (Buchheim and Traversi 2013) 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 $x_1, \dots, x_n$ is the chosen order, the next variable to be fixed is the free variable with smallest index. The same idea has been used in (Buchheim et al. 2012) and (Buchheim et al. 2013). 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 (Buchheim and Traversi 2013).

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

$$\begin{array}{lll} \max & \frac{1}{4}\mathbf{1}^\top t & + \quad \min \quad (L+Q\mathbf{1})^\top x \quad - \quad \frac{1}{4}\mathbf{1}^\top Q\mathbf{1}\,. \\ \text{s.t.} & Q \succeq \operatorname{Diag}(t) & \text{s.t.} \quad x \in X \end{array}$$

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

$$\begin{aligned} \max \quad & \mathbf{1}^\top t \\ \text{s.t.} \quad & V^\top Q V \succeq V^\top \mathrm{Diag}(t) V \,. \end{aligned}$$

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

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

thus obtaining

$$\begin{aligned} \max \quad & \mathrm{diag}(V V^\top)^\top t \\ \text{s.t.} \quad & V^\top Q V \succeq V^\top \mathrm{Diag}(t) V \,. \end{aligned} \tag{17}$$

The dual of (17) is

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

and thus strictly feasible. In particular, the primal problem (17) is bounded. A strictly feasible solution for (17) is given by $t := (\lambda_{\min}(Q) - 1) \cdot \mathbf{1}$. In Section 5, we will describe a tailored algorithm for solving problems of type (17).

Note that it is possible to obtain a feasible solution t from any given t_0 by setting

$$t := t_0 + \lambda_{\min}(Q - \mathrm{Diag}(t_0))\, \mathbf{1}$$

as discussed above, or, when taking equations into account, by

$$t := t_0 + \lambda_{\min}(V^\top(Q - \mathrm{Diag}(t_0))V)\,\mathbf{1}$$

where we use $V^\top V = I$.

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

$$\begin{aligned}
\min\ & \langle V^\top Q V, Y\rangle \\
\text{s.t.}\ & \mathrm{diag}(VYV^\top) = \mathrm{diag}(VV^\top) \\
& (VYV^\top)_{ij} \geq 0 \text{ if } (z_i \leq 0 \text{ and } z_j \leq 0) \text{ or } (z_i \geq 1 \text{ and } z_j \geq 1) \\
& (VYV^\top)_{ij} \leq 0 \text{ if } (z_i \leq 0 \text{ and } z_j \geq 0) \text{ or } (z_i \geq 1 \text{ and } z_j \leq 1) \\
& Y \succeq 0.
\end{aligned}$$

5. Fast computation of the underestimator

The special structure of the semidefinite program (17) can be exploited in order to solve it more quickly than with standard interior point methods. We adapt an algorithm devised by Dong (2014) 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 (Dong 2014), switching the sign of t, we first rewrite (17) as a minimization problem:

$$\begin{aligned}
\min\ & \mathrm{diag}(VV^\top)^\top t \\
\text{s.t.}\ & V^\top(Q + \mathrm{Diag}(t))V \succeq 0\,.
\end{aligned}$$

We introduce a penalty term $\sigma > 0$ and obtain

$$\begin{aligned}
\min\ & f(t;\sigma) := \mathrm{diag}(VV^\top)^\top t - \sigma \log\det(V^\top(Q + \mathrm{Diag}(t))V) \\
\text{s.t.}\ & V^\top(Q + \mathrm{Diag}(t))V \succ 0\,.
\end{aligned} \tag{18}$$

The gradient of the objective function of (18) is

$$\nabla_t f(t;\sigma) = \mathrm{diag}(VV^\top) - \sigma\mathrm{diag}(V[V^\top(Q + \mathrm{Diag}(t))V]^{-1}V^\top)\,.$$

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

$$W := [V^\top(Q + \mathrm{Diag}(t))V]^{-1}$$

using the Sherman-Morrison formula and of the vector

$$x := \mathrm{diag}(VWV^\top)\,.$$

We initialize our iterative algorithm by calculating

$$\begin{aligned} t^{(0)} &:= -(1+\varepsilon)\lambda_{\min}(V^\top Q V) \\ W^{(0)} &:= [V^\top (Q + \mathrm{Diag}(t^{(0)}))V]^{-1} \\ x^{(0)} &:= \mathrm{diag}(VW^{(0)}V^\top) \end{aligned}$$

directly, for some $\varepsilon > 0$. By construction, $t^{(0)}$ is a feasible solution for (18). Now in each iteration we improve the objective function coordinatewise, choosing a coordinate $i^{(k)}$ maximizing

$$|\nabla_t f(t;\sigma)_i| = |v_i^\top v_i - \sigma x_i^{(k)}| \,.$$

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

$$\begin{aligned} 0 &= \nabla_s f(t^{(k)} + se_{i^{(k)}}) \\ &= v_{i^{(k)}}^\top v_{i^{(k)}} - \sigma v_{i^{(k)}}^\top [V^\top (Q + \mathrm{Diag}(t^{(k)} + se_{i^{(k)}}))V]^{-1} v_{i^{(k)}} \,. \end{aligned}$$

Now

$$\begin{aligned} [V^\top (Q + \mathrm{Diag}(t^{(k)} + se_{i^{(k)}}))V]^{-1} &= [W^{(k)} + sv_{i^{(k)}} v_{i^{(k)}}^\top]^{-1} \\ &= W^{(k)} - \frac{sW^{(k)} v_{i^{(k)}} v_{i^{(k)}}^\top W^{(k)}}{1 + sv_{i^{(k)}}^\top W^{(k)} v_{i^{(k)}}} \end{aligned}$$

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

$$s^{(k)} = \frac{\sigma}{v_{i^{(k)}}^\top v_{i^{(k)}}} - \frac{1}{x_{i^{(k)}}^{(k)}}$$

as our optimal step length. The resulting updates

$$\begin{aligned} t^{(k+1)} &:= t^{(k)} + s^{(k)} e_{i^{(k)}} \\ W^{(k+1)} &:= W^{(k)} - \frac{s^{(k)} W^{(k)} v_{i^{(k)}} v_{i^{(k)}}^\top W^{(k)}}{1 + s^{(k)} x_{i^{(k)}}^{(k)}} \\ x^{(k+1)} &:= x^{(k)} - \frac{s^{(k)} V W^{(k)} v_{i^{(k)}} v_{i^{(k)}}^\top W^{(k)} V^\top}{1 + s^{(k)} x_{i^{(k)}}^{(k)}} \end{aligned}$$

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

In the course of this algorithm, the penalty factor σ needs to converge to zero. In our implementation, we multiply σ 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 (17) falls short of 10^{-5} in three consecutive iterations. Note that the iterates $t^{(k)}$ computed by this algorithm are all feasible for Problem (18) and hence for Problem (17), independently of the (current) value of σ. 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 (Buchheim and Traversi 2013). 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 $\frac{1}{2}\mathbf{1}$ 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 Cplex 12.6 (Cplex 2015), to the SDP-based solver BiqCrunch 2.0 (Malick and Roupin 2013), and to the QCR method; an implementation of the latter has been provided by the authors of (Billionnet et al. 2009). All these methods can address general constrained binary quadratic optimization problems and are thus applicable to all types of instances considered here.

The main improvement of our algorithm with respect to the version tested in (Buchheim and Traversi 2013) 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 algorithm for solving the linear counterpart (2) of the problem,
- an overview over state-of-the-art problem-specific solution methods, and
- the testbed used in the experiments.

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:

$$\begin{aligned} \min \quad & \textstyle\sum_{a,b\in A} Q_{ab} x_a x_b + \sum_{a\in A} L_a x_a \\ \text{s.t.} \quad & \textstyle\sum_{a\in\delta^+(i)} x_a - \sum_{a\in\delta^-(i)} x_a = 0 \qquad \forall i \in N \setminus \{s,t\} \\ & \textstyle\sum_{a\in\delta^+(s)} x_a = 1 \\ & \textstyle\sum_{a\in\delta^-(t)} x_a = 1 \\ & x_a \in \{0,1\} \; \forall a \in A\,, \end{aligned} \tag{19}$$

with s and t being the origin and destination of the 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 (19), out of which $|N| - 1$ are linearly independent (if G is connected).

Comparatively few publications have addressed the QSPP so far. They mostly deal with special cases, e.g., only products between consecutive edges are taken into account (Gourvès et al. 2010, Amaldi et al. 2011, Hu and Sotirov 2016). For the general case, the problem has been shown to be NP-hard, and problem-specific methods to compute lower bounds have been proposed in the meantime (Rostami et al. 2015). The latter have been evaluated on the same instances we consider in the following: we use grid graphs with $k \times k$ nodes, for $k = 10, \ldots, 15$, and generate quadratic costs uniformly at random from $\{1, ..., 10\}$. 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 with standard settings.

Quadratic Assignment Problem (QAP): The QAP is a well known generalization of the assignment problem. As for the quadratic version of the shortest path 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 $G = (N_1 \dot\cup N_2, E)$ with $|N_1| = |N_2|$ is the following:

$$\begin{aligned} \min \quad & \textstyle\sum_{e,f\in E} Q_{ef} x_e x_f + \sum_{e\in E} L_e x_e \\ \text{s.t.} \quad & \textstyle\sum_{e\in\delta(i)} x_e = 1 \qquad \forall i \in N_1 \\ & \textstyle\sum_{e\in\delta(i)} x_e = 1 \qquad \forall i \in N_2 \\ & x_e \in \{0,1\} \; \forall e \in E\,. \end{aligned} \tag{20}$$

Also in this case we can exploit all equations given in (20), out of which $|N_1| + |N_2| - 1$ are linearly independent. 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.

In contrast to the QSPP, the QAP has been investigated intensively in the literature. It is probably the most classical example of a combinatorial optimization problem with a quadratic objective function, introduced by Koopmans and Beckmann in 1957. Early approaches contain the well-known Gilmore-Lawler bound (Gilmore 1962, Lawler 1963), which results from a solution of $|N_1|+1$ linear assignment problems, as well as linearization approaches, combined with quadratic reformulations of the assignment constraints (Adams and Johnson 1994). Since then, methods computing much stronger lower bounds have been devised, in particular based on semidefinite relaxations (Zhao et al. 1998, de Klerk and Sotirov 2012, de Klerk et al. 2015). Compared to LP-based bounds, the resulting bounds turn out to be very strong, but their computation is more time-consuming (both is confirmed by our results for BiqCrunch reported below). When aiming at exact solutions, instances with $|N_1| \geq 20$ are still difficult to solve in general. Using convex QP-relaxations and a huge computational grid, Anstreicher et al. (2002) were able to solve instances with $|N_1| = 30$. Such dimensions are clearly out of reach for our generic approach; in our experiments we use instances from QAPLIB (Burkard et al. 1997) with $|N_1| = 10, 12$.

Quadratic Knapsack Problem (QKP): The QKP is a well known 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:

$$\begin{aligned} \min \quad & \textstyle\sum_{i,j\in N} Q_{ij}x_ix_j + \sum_{i\in N} L_ix_i \\ \text{s.t.} \quad & \textstyle\sum_{i\in N} c_ix_i \leq k \\ & x_i \in \{0,1\} \ \forall i \in N\,, \end{aligned} \tag{21}$$

with c_i being the weight of object $i \in N$ and k being the capacity. No equations can be exploited in this case. As black box, we implemented the well-known dynamic programming procedure for the linear knapsack problem; see, e.g., (Kellerer et al. 2004). Note that we model QKP as a minimization problem here, in order to stay analogous to the other problem classes.

The QKP is a well-studied problem. Again, dual bounds are often obtained from either linear (Billionnet and Soutif 2004b) or semidefinite relaxations (Helmberg et al. 2000). For the case of non-negative quadratic coefficients, exact algorithms based on Lagrangian relaxation have been devised by Caprara et al. (1999) and Billionnet and Soutif (2004a). They are able to solve instances on up to 150 items to optimality (up to 400 items if the objective function is sparse). For a more recent survey of dual bounds and exact algorithms, see Pisinger (2007). In our experiments, we use the library of QKP instances (with 100 items each) proposed in (Billionnet and Soutif 2004b), for a total of 40 instances.

We also considered *Unconstrained Binary Quadratic Programs (UBQP)* as well as the *Quadratic Spanning Tree Problem (QSTP)* in (Buchheim and Traversi 2013). Note that any LP based approach to spanning tree problems suffers from the fact that the classical integer programming formulation requires an exponential number of linear inequalities and hence a separation algorithm. For this reason, applying Cplex, BiqCrunch, 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 (Kruskal 1956). We thus do not include a comparison here.

We would like to emphasize that our algorithm is designed to solve a large class of quadratic binary optimization problems. As such, we do not expect that it can compete with problem-specific methods for, e.g., the QAP. For this reason, we do not include a comparison with state-of-the-art solvers for the problems listed above.

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 lower bounds in the root node of the branch-and-bound tree obtained by our algorithm with the root bounds obtained by Cplex, the QCR algorithm, and BiqCrunch. For Cplex, we examine both the variant where the quadratic objective function is linearized (Cplex-MIP) and the one where QP-relaxations are used to obtain dual bounds (Cplex-QP); this distinction is made via the parameter `CPX_PARAM_QTOLININD`. In the SDP formulations passed to BiqCrunch, we always consider quadratic reformulations of constraints, as suggested to us by one of the developers of BiqCrunch. More precisely, for the problems QSPP and QAP containing linear equations, we multiply each of these by all variables and add the resulting quadratic constraints to the linear formulation. For QKP, we add the quadratic inequality $(a^\top x)(a^\top x - b) \leq 0$ to the original contraint $a^\top x \leq b$. To the resulting problem formulations, we apply both the version of BiqCrunch without triangle inequalities (SDP) and with triangle inequalities (SDP+Δ). Otherwise, we use standard parameters for all software considered.

The results concerning QSPP, QAP and QKP are provided in Tables 1–3. All tables are vertically divided into eight blocks: in the first block, we state the name of the instance and its dimension. In the next six blocks, we report lower bounds given by the different approaches. 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.

As obvious from the tables, SDP+Δ yields by far the strongest bounds, being close to the optima for all three instance types. Without triangle inequalities, the bounds are considerably weaker: roughly, the simple SDP bounds are similary strong as the bounds computed by QCR. Also Cplex-QP leads to similar bounds in the QKP case, but is considerably weaker in the QSPP case and extremely weak in the QAP case. The bounds calculated by Cplex-MIP are very weak for QSPP, trivial for QAP, and comparable to QCR and SDP for QKP. Compared to the other approaches, our new algorithm in general yields bounds of medium strength: they are always weaker than those obtained by SDP+Δ, SDP, and QCR, but stronger than those given by Cplex-QP except for the QKP case. Note that

most algorithms considered here produce negative bounds for many instances in the QAP case, in spite of the fact that zero is a trivial lower bound.

inst	n	Cplex-MIP LB	Cplex-QP LB	QCR LB	SDP LB	SDP+Δ LB	Our LB	OPT
10_1	180	79	200	489	487	620	357	621
10_2	180	79	211	501	489	634	323	635
10_3	180	91	217	498	495	635	367	636
10_4	180	81	209	491	489	660	359	661
10_5	180	91	233	504	500	664	367	665
11_1	220	83	253	609	607	812	420	813
11_2	220	86	251	593	590	787	417	788
11_3	220	87	225	592	588	794	384	795
11_4	220	98	236	619	618	781	402	782
11_5	220	93	228	582	581	766	404	767
12_1	264	105	271	714	709	958	479	959
12_2	264	85	282	707	705	962	524	963
12_3	264	84	259	687	683	899	491	900
12_4	264	87	236	698	693	959	481	960
12_5	264	94	289	701	699	975	479	976
13_1	312	105	335	805	803	1154	586	1159
13_2	312	108	333	821	820	1176	590	1178
13_3	312	104	325	822	814	1159	558	1164
13_4	312	103	301	805	801	1109	568	1110
13_5	312	95	322	842	837	1114	567	1115
14_1	364	102	364	959	954	1350	680	1363
14_2	364	114	357	963	954	1357	669	1367
14_3	364	117	334	934	932	1319	651	1320
14_4	364	119	348	982	977	1345	661	1347
14_5	364	108	354	949	941	1341	704	1344
15_1	420	131	367	1094	1089	1545	729	1551
15_2	420	120	412	1099	1096	1554	806	1588
15_3	420	102	419	1067	1064	1522	762	1561
15_4	420	122	386	1061	1057	1531	744	1569
15_5	420	122	389	1084	1080	1558	791	1582

Table 1 Lower bound comparison for QSPP

inst	n	Cplex-MIP LB	Cplex-QP LB	QCR LB	SDP LB	SDP+Δ LB	Our LB	OPT
tai10a	100	0	-509824	32225	32171	135025	-13475	135028
tai10b	100	0	-15375723	-2253366	-2253753	1183625	-5972174	1183760
chr12a	144	0	-416795	-103848	-103887	9551	-217851	9552
chr12b	144	0	-419179	-86164	-86312	9741	-177099	9742
chr12c	144	0	-415953	-99747	-99819	11155	-179110	11156
had12	144	0	-7040	1198	1197	1651	672	1652
nug12	144	0	-3483	-216	-220	558	-759	578
rou12	144	0	-881997	76922	76862	231778	-313	235852
scr12	144	0	-423726	-111147	-111189	31005	-227213	31410
tai12a	144	0	-1029577	77893	77839	224412	-27238	224416
tai12b	144	0	-621081036	-134673000	-134818410	-134818410	-402161178	39464925

Table 2 Lower bound comparison for QAP

6.2. Running time comparison

We next compare our algorithm with Cplex-MIP, Cplex-QP, the QCR method, and BiqCrunch in terms of total running times for solving the instances to optimality. The latter is applied to the quadratic model as described above, also making use of triangle inequalities. The comparison for QSPP, QAP and QKP is presented in Tables 4–6, respectively. In all tables, we report for each of the five algorithms the number of branch-and-bound nodes and the overall computing time (TL if the time limit is reached). In each row, the running time of the fastest solver is highlighted.

Our new algorithm is able to solve to optimality all 30 instances of QSPP and all 11 instances of QAP, as well as 12 out of the 39 instances of QKP. Cplex-MIP can only solve 7

inst	n	Cplex-MIP LB	Cplex-QP LB	QCR LB	SDP LB	SDP+Δ LB	Our LB	OPT
100_25_1	100	-19125	-23846	-23141	-18904	-18615	-34756	-18558
100_25_2	100	-56576	-58465	-57451	-56600	-56575	-60485	-56525
100_25_3	100	-4064	-7794	-6164	-3986	-3800	-14986	-3752
100_25_4	100	-51065	-54204	-53143	-50527	-50404	-57694	-50382
100_25_5	100	-61622	-62410	-61746	-61649	-61644	-63137	-61494
100_25_6	100	-36655	-40748	-39868	-36541	-36419	-47826	-36360
100_25_7	100	-14855	-19300	-18321	-14852	-14719	-29519	-14657
100_25_8	100	-20529	-24870	-24135	-20656	-20502	-36474	-20452
100_25_9	100	-35488	-39664	-38922	-35601	-35482	-46032	-35438
100_25_10	100	-25497	-30081	-29596	-25217	-25042	-38858	-24930
100_50_1	100	-83921	-92176	-95449	-83978	-83911	-104435	-83742
100_50_2	100	-105322	-112191	-109655	-105089	-105023	-117406	-104856
100_50_3	100	-34324	-44140	-42695	-34270	-34209	-66364	-34006
100_50_4	100	-106319	-111322	-109162	-106099	-106079	-115734	-105996
100_50_5	100	-56952	-66171	-65104	-56719	-56692	-84867	-56464
100_50_6	100	-17357	-26388	-23914	-16271	-16144	-43215	-16083
100_50_7	100	-53062	-62775	-62148	-52980	-52942	-81743	-52819
100_50_8	100	-54534	-63788	-62824	-54510	-54479	-82084	-54246
100_50_9	100	-70794	-79085	-77998	-69039	-68974	-93372	-68974
100_50_10	100	-89966	-97413	-90142	-89041	-88956	-106246	-88634
100_75_1	100	-189137	-190150	-151179	-189139	-189138	-190664	-189137
100_75_2	100	-98716	-113262	-111376	-95516	-95450	-135560	-95074
100_75_3	100	-63870	-78403	-77256	-62287	-62280	-108892	-62098
100_75_4	100	-75164	-89190	-88035	-72779	-72719	-116203	-72245
100_75_5	100	-29235	-43606	-40327	-27906	-27844	-72765	-27616
100_75_6	100	-145418	-155748	-151780	-145452	-145416	-166175	-145273
100_75_7	100	-113111	-127900	-125222	-111361	-111316	-147115	-110979
100_75_8	100	-25408	-36117	-32902	-19713	-19673	-59888	-19570
100_75_9	100	-108298	-120973	-119196	-104905	-104834	-141193	-104341
100_75_10	100	-144220	-154822	-189571	-143969	-143957	-164457	-143740
100_100_1	100	-82642	-99418	-97043	-82628	-82624	-142746	-81978
100_100_2	100	-193522	-208411	-203642	-190903	-190886	-220342	-190424
100_100_3	100	-226138	-234897	-231054	-225565	-225538	-239245	-225434
100_100_5	100	-230441	-239462	-235360	-230286	-230268	-243602	-230076
100_100_6	100	-76250	-95725	-93766	-74719	-74714	-135979	-74358
100_100_7	100	-10620	-25261	-17864	-10748	-10441	-49966	-10330
100_100_8	100	-63167	-82854	-80870	-62895	-62890	-124939	-62582
100_100_9	100	-234558	-241061	-237797	-234288	-234280	-243448	-232754
100_100_10	100	-195087	-208945	-204046	-195080	-195077	-222563	-193262

Table 3 Lower bound comparison for QKP

		Cplex-MIP		Cplex-QP		QCR		BiqCrunch		Our Algo	
inst	n	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
10_1	180	1.54e+3	66.2	8.77e+3	11.2	6.21e+2	9.4	1.00e+0	22.4	5.13e+4	4.8
10_2	180	2.00e+3	73.5	9.42e+3	11.8	1.53e+3	8.7	1.00e+0	27.1	3.32e+4	4.2
10_3	180	1.72e+3	67.8	8.24e+3	9.9	2.04e+3	8.5	1.00e+0	20.3	2.96e+4	4.2
10_4	180	2.52e+3	89.8	1.33e+4	16.2	4.24e+3	9.8	1.00e+0	26.0	6.67e+4	5.1
10_5	180	2.43e+3	90.7	1.25e+4	15.1	3.23e+3	9.7	1.00e+0	34.7	6.76e+4	5.2
11_1	220	8.53e+3	677.0	3.26e+4	50.7	9.51e+3	32.3	1.00e+0	55.1	2.72e+5	15.8
11_2	220	7.89e+3	661.9	2.83e+4	44.8	8.04e+3	31.1	1.00e+0	52.7	1.34e+5	11.2
11_3	220	8.91e+3	598.2	3.16e+4	49.9	9.28e+3	27.8	1.00e+0	81.7	1.64e+5	12.0
11_4	220	7.79e+3	481.4	2.34e+4	42.9	3.10e+3	20.5	1.00e+0	55.0	1.09e+5	10.1
11_5	220	6.68e+3	566.4	2.03e+4	36.9	4.78e+3	25.4	1.00e+0	50.8	1.30e+5	11.1
12_1	264	1.30e+4	1506.9	7.49e+4	146.2	1.82e+4	92.8	1.00e+0	126.7	7.41e+5	42.8
12_2	264	1.52e+4	1762.3	9.70e+4	201.8	2.39e+4	99.7	1.00e+0	151.3	7.16e+5	41.1
12_3	264	9.68e+3	1028.3	5.63e+4	115.2	9.40e+3	80.8	1.00e+0	113.0	3.01e+5	24.0
12_4	264	1.61e+4	1860.5	1.08e+5	197.0	2.77e+4	124.0	1.00e+0	162.9	9.84e+5	53.8
12_5	264	1.46e+4	1680.2	8.76e+4	172.6	2.46e+4	132.0	3.00e+0	457.8	7.33e+5	43.5
13_1	312	4.47e+4	6399.4	3.94e+5	1183.4	1.07e+5	625.3	5.00e+0	1131.2	3.19e+6	176.8
13_2	312	4.82e+4	6213.2	3.36e+5	1043.5	8.79e+4	502.0	3.00e+0	908.5	4.36e+6	240.8
13_3	312	4.60e+4	6826.3	3.48e+5	1060.0	7.80e+4	353.5	3.00e+0	1146.6	4.10e+6	218.7
13_4	312	3.09e+4	5082.1	2.25e+5	726.3	4.48e+4	271.4	1.00e+0	297.2	1.98e+6	121.1
13_5	312	2.24e+4	3847.8	1.72e+5	590.6	2.51e+4	206.4	1.00e+0	272.2	2.29e+6	138.5
14_1	364	3.98e+4	TL	1.07e+6	3204.9	2.44e+5	1416.5	7.00e+0	2949.1	9.75e+6	618.1
14_2	364	3.89e+4	TL	1.14e+6	4062.2	2.41e+5	1396.9	5.00e+0	2244.8	1.12e+7	702.1
14_3	364	3.35e+4	TL	7.73e+5	3079.3	1.25e+5	795.2	3.00e+0	1487.3	1.01e+7	618.6
14_4	364	2.98e+4	TL	9.25e+5	3475.1	1.50e+5	935.6	3.00e+0	1585.9	1.88e+7	1140.8
14_5	364	4.09e+4	TL	8.85e+5	3363.6	2.06e+5	1230.6	3.00e+0	1470.8	1.08e+7	678.9
15_1	420	1.48e+4	TL	2.30e+6	TL	4.27e+5	2951.3	5.00e+0	3667.4	3.86e+7	2697.7
15_2	420	2.21e+4	TL	2.42e+6	TL	8.36e+5	5117.1	1.50e+1	9797.8	6.90e+7	4903.6
15_3	420	2.21e+4	TL	2.39e+6	TL	6.10e+5	3910.2	1.30e+1	7918.2	5.10e+7	3682.6
15_4	420	1.66e+4	TL	2.77e+6	TL	9.09e+5	5556.5	1.70e+1	9684.1	6.15e+7	4312.2
15_5	420	1.92e+4	TL	2.76e+6	TL	6.97e+5	4317.0	9.00e+0	5655.8	6.52e+7	4712.4

Table 4 Computational results for QSPP

		Cplex-MIP		Cplex-QP		QCR		BiqCrunch		Our Algo	
inst	n	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
tai10a	100	1.03e+5	672.8	4.30e+6	916.4	2.72e+5	72.9	1.95e+2	441.7	8.33e+5	14.6
tai10b	100	7.72e+3	46.1	3.62e+6	703.3	2.34e+5	56.9	5.12e+3	TL	9.56e+4	2.1
chr12a	144	4.06e+3	6.6	5.53e+7	TL	4.42e+6	1110.5	2.87e+2	1975.6	1.53e+5	8.6
chr12b	144	1.44e+3	1.0	4.88e+7	9057.7	3.82e+6	950.7	2.77e+2	977.5	6.44e+3	4.7
chr12c	144	1.63e+4	23.8	4.39e+7	TL	2.10e+7	5110.9	3.51e+2	3201.8	1.32e+6	32.8
had12	144	8.76e+5	TL	3.32e+7	TL	1.62e+6	783.4	3.00e+0	48.9	7.00e+6	168.5
nug12	144	1.27e+6	TL	3.59e+7	TL	2.75e+7	10585.1	3.70e+1	1353.3	7.56e+6	146.4
rou12	144	6.72e+5	TL	3.32e+7	TL	1.94e+7	9005.8	3.47e+2	7132.0	2.15e+7	416.7
scr12	144	2.75e+4	126.6	4.13e+7	TL	1.88e+7	5790.5	2.39e+2	TL	1.01e+6	24.6
tai12a	144	6.91e+5	TL	3.27e+7	TL	2.91e+6	1403.0	2.93e+2	1421.4	4.31e+6	92.4
tai12b	144	1.86e+5	2471.5	3.30e+7	TL	9.07e+6	3687.5	4.20e+3	TL	2.78e+6	51.2

Table 5 Computational results for QAP

		Cplex-MIP		Cplex-QP		QCR		BiqCrunch		Our Algo	
inst	n	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
100_25_1	100	6.29e+2	1.1	5.00e+7	TL	8.45e+7	TL	2.06e+3	713.5	3.09e+8	TL
100_25_2	100	0.00e+0	0.1	5.49e+4	8.5	9.39e+3	2.1	5.07e+2	99.2	8.06e+4	18.9
100_25_3	100	4.60e+1	0.5	6.33e+7	TL	9.38e+7	TL	6.71e+2	302.2	8.89e+7	732.4
100_25_4	100	4.10e+1	0.5	6.26e+7	TL	3.91e+7	8280.0	1.75e+2	58.1	3.26e+7	5865.3
100_25_5	100	0.00e+0	0.1	1.50e+3	0.3	2.26e+2	0.1	3.87e+2	80.7	1.07e+3	1.3
100_25_6	100	2.79e+2	0.9	5.54e+7	TL	5.99e+7	TL	3.19e+2	145.0	1.77e+8	TL
100_25_7	100	2.37e+2	0.7	5.22e+7	TL	7.73e+7	TL	1.68e+3	485.3	3.79e+8	TL
100_25_8	100	1.90e+1	0.2	5.08e+7	TL	6.76e+7	TL	7.59e+2	251.7	2.98e+8	TL
100_25_9	100	1.70e+1	0.3	5.29e+7	TL	6.97e+7	TL	2.15e+2	93.8	1.10e+8	TL
100_25_10	100	5.90e+2	1.9	5.02e+7	TL	7.68e+7	TL	2.87e+3	1302.9	2.48e+8	TL
100_50_1	100	7.00e+1	2.1	5.16e+7	TL	6.83e+7	TL	1.95e+3	637.8	1.40e+8	TL
100_50_2	100	4.69e+2	8.8	1.86e+7	3629.4	1.01e+7	2540.0	2.60e+3	932.0	1.14e+7	1952.4
100_50_3	100	3.21e+2	3.1	4.73e+7	TL	7.93e+7	TL	3.16e+3	896.8	4.65e+8	TL
100_50_4	100	3.00e+1	1.1	1.18e+6	221.8	3.39e+5	85.6	2.31e+2	87.8	1.29e+6	253.1
100_50_5	100	3.98e+2	5.1	4.66e+7	TL	6.72e+7	TL	3.14e+3	569.9	1.91e+8	TL
100_50_6	100	5.53e+2	3.6	4.99e+7	TL	9.03e+7	TL	4.97e+2	149.3	7.80e+8	TL
100_50_7	100	1.77e+2	4.2	4.67e+7	TL	7.41e+7	TL	4.95e+2	100.8	2.13e+8	TL
100_50_8	100	1.60e+2	2.1	4.52e+7	TL	7.37e+7	TL	2.20e+3	736.9	2.53e+8	TL
100_50_9	100	1.83e+2	4.9	4.48e+7	TL	6.95e+7	TL	1.00e+0	1.4	1.37e+8	TL
100_50_10	100	3.06e+3	33.1	4.84e+7	TL	5.69e+7	TL	1.86e+4	8017.5	9.35e+7	TL
100_75_1	100	0.00e+0	0.6	3.50e+1	0.0	2.21e+7	6680.0	8.30e+1	19.3	1.31e+2	1.0
100_75_2	100	1.02e+4	93.0	4.20e+7	TL	6.05e+7	TL	2.56e+4	10070.3	1.87e+8	TL
100_75_3	100	3.57e+2	6.0	4.05e+7	TL	6.95e+7	TL	8.35e+2	140.8	3.40e+8	TL
100_75_4	100	4.43e+3	59.3	4.24e+7	TL	6.30e+7	TL	3.03e+4	TL	2.70e+8	TL
100_75_5	100	4.76e+2	7.2	4.44e+7	TL	8.38e+7	TL	3.38e+3	749.3	7.09e+8	TL
100_75_6	100	6.00e+0	0.8	1.92e+7	4642.2	5.55e+6	1750.0	5.63e+2	181.1	2.01e+7	2994.6
100_75_7	100	1.25e+3	20.6	4.40e+7	TL	6.25e+7	TL	9.39e+3	3186.6	1.47e+8	TL
100_75_8	100	2.62e+3	82.3	4.46e+7	TL	7.88e+7	TL	2.75e+2	78.7	9.00e+8	TL
100_75_9	100	1.19e+4	171.5	4.02e+7	TL	6.72e+7	TL	2.57e+4	TL	1.50e+8	TL
100_75_10	100	3.46e+2	8.0	4.42e+7	TL	1.50e+2	0.5	1.96e+3	425.6	7.79e+7	TL
100_100_1	100	1.26e+2	5.0	4.04e+7	TL	5.52e+7	TL	7.87e+3	1259.6	3.17e+8	TL
100_100_2	100	1.49e+3	46.6	3.76e+7	TL	5.10e+7	TL	6.26e+3	1888.1	7.84e+7	TL
100_100_3	100	9.40e+1	5.7	3.43e+5	94.0	3.49e+5	118.4	5.23e+2	89.7	1.43e+5	32.0
100_100_5	100	1.28e+2	9.4	2.73e+5	68.8	2.42e+5	81.7	6.67e+2	156.2	9.83e+4	21.4
100_100_6	100	6.93e+2	23.4	4.09e+7	TL	6.47e+7	TL	3.36e+3	723.7	3.58e+8	TL
100_100_7	100	1.56e+2	7.2	4.27e+7	TL	8.10e+7	TL	2.01e+2	87.9	4.96e+7	251.6
100_100_8	100	2.91e+2	5.5	3.81e+7	TL	7.10e+7	TL	1.93e+3	351.2	5.01e+8	TL
100_100_9	100	4.58e+4	1610.5	2.74e+5	68.5	3.12e+5	93.2	3.29e+4	9707.7	9.08e+4	14.5
100_100_10	100	7.22e+4	1185.0	3.65e+7	TL	4.20e+7	TL	1.69e+4	TL	1.51e+8	TL

Table 6 Computational results for QKP

instances of QAP and is the weakest solver for QSPP with 10 instances unsolved; for QKP, however, this approach clearly outperforms all other methods, being the fastest method in all but 5 cases. Altogether, Cplex-QP is weaker than Cplex-MIP, solving only 3 instances

for QAP, 25 for QSPP and 9 for QKP. Our approach clearly outperforms both versions of Cplex on all instances of QSPP and most instances of QAP. Moreover, it performs slightly better than Cplex-QP on the QKP instances.

Also in comparison with the QCR algorithm, our algorithm is faster in solving QSPP in general and much faster in solving QAP instances, while having comparable running times on average for solving the QKP instances. Interestingly, running times can differ drastically in both directions, particularly for the QKP instances, showing that both algorithms have very different strengths.

Finally, also BiqCrunch shows different performance on the three classes of instances. On the one hand, it is clearly outperformed by our approach for QSPP and QAP instances, except for one instance. On the other hand, it is very strong for the QKP instances, where it is beaten only by Cplex-MIP.

In summary, we can conclude that our algorithm outperforms the other approaches on almost all QSPP and most of the QAP instances considered, while the picture for QKP is different, with Cplex-MIP and BiqCrunch being the fastest algorithms on average.

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.

Acknowledgments

This work has been supported by the German Research Foundation under grant BU 2313/4. A preliminary version of this paper has been published in the Proceedings of SEA 2013 (Buchheim and Traversi 2013). 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, and to Frédéric Roupin for information concerning the best models and parameter settings for BiqCrunch.

References

Adams, W. P., T. A. Johnson. 1994. Improved linear programming-based lower bounds for the quadratic assignment problem. P.M. Pardalos, H. Wolkowicz, eds., *Quadratic Assignment and Related Problem*, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 16. AMS, 43–75.

Aggarwal, A., D. Coppersmith, S. Khanna, R. Motwani, B. Schieber. 1999. The angular-metric traveling salesman problem. *SIAM Journal on Computing* **29** 697–711.

Amaldi, E., G. Galbiati, F. Maffioli. 2011. On minimum reload cost paths, tours, and flows. *Networks* **57** 254–260.

Anstreicher, K. M., N. W. Brixius, J. P. Goux, J. Linderoth. 2002. Solving large quadratic assignment problems on computational grids. *Mathematical Programming* **91** 563–588.

Assad, A., W. Xu. 1992. The quadratic minimum spanning tree problem. *Naval Research Logistics* **39** 399–417.

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

Billionnet, A., E. Soutif. 2004a. An exact method based on lagrangian decomposition for the 0–1 quadratic knapsack problem. *European Journal of Operational Research* **157** 565–575.

Billionnet, A., E. Soutif. 2004b. Using a mixed integer programming tool for solving the 0–1 quadratic knapsack problem. *INFORMS Journal on Computing* **16** 188–197.

Bonami, P., L.T. Biegler, A.R. Conn, G. Cornuéjols, I.E. Grossmann, C.D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, A. Wächter. 2008. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization* **5** 186–204.

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

Buchheim, C., M. De Santis, L. Palagi, M. Piacentini. 2013. An exact algorithm for nonconvex quadratic integer minimization using ellipsoidal relaxations. *SIAM Journal on Optimization* **23** 1867–1889.

Buchheim, C., E. Traversi. 2013. Separable non-convex underestimators for binary quadratic programming. *12th International Symposium on Experimental Algorithms – SEA 2013*, *LNCS*, vol. 7933. 236–247.

Buchheim, C., L. Trieu. 2013. Quadratic outer approximation for convex integer programming. *12th International Symposium on Experimental Algorithms – SEA 2013*, *LNCS*, vol. 7933. 224–235.

Buchheim, C., A. Wiegele, L. Zheng. 2010. Exact algorithms for the quadratic linear ordering problem. *INFORMS Journal on Computing* **22** 168–177.

Burkard, R.E., S.E. Karisch, F. Rendl. 1997. QAPLIB – a quadratic assignment problem library. *Journal of Global Optimization* **10** 391–403.

Caprara, A. 2008. Constrained 0–1 quadratic programming: Basic approaches and extensions. *European Journal of Operational Research* **187** 1494–1503.

Caprara, A., D. Pisinger, P. Toth. 1999. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing* **11** 125–137.

Cplex. 2015. IBM ILOG CPLEX Optimizer 12.6. `www.ibm.com/software/integration/optimization/cplex-optimizer`.

de Klerk, E., R. Sotirov. 2012. Improved semidefinite programming bounds for quadratic assignment problems with suitable symmetry. *Mathematical Programming* **133** 75–91.

de Klerk, E., R. Sotirov, U. Truetsch. 2015. A new semidefinite programming relaxation for the quadratic assignment problem and its computational perspectives. *INFORMS Journal on Computing* **27** 378–391.

Dong, H. 2014. Relaxing nonconvex quadratic functions by multiple adaptive diagonal perturbations. Tech. rep., Optimization Online.

Gamvros, I., L. Gouveia, S. Raghavan. 2012. Reload cost trees and network design. *Networks* **59** 365–379.

Gilmore, P. C. 1962. Optimal and suboptimal algorithms for the quadratic assignment problem. *Journal of the Society for Industrial & Applied Mathematics* **10** 305–313.

Gourvès, L., A. Lyra, C. Martinhon, J. Monnot. 2010. The minimum reload s–t path, trail and walk problems. *Discrete Applied Mathematics* **158** 1404–1417.

Helmberg, C., F. Rendl, R. Weismantel. 2000. A semidefinite programming approach to the quadratic knapsack problem. *Journal of Combinatorial Optimization* **4** 197–215.

Hu, H., R. Sotirov. 2016. Special cases of the quadratic shortest path problem. Tech. rep., arXiv.org. `https://arxiv.org/abs/1611.07682`.

Kellerer, H., U. Pferschy, D. Pisinger. 2004. *Knapsack problems*. Springer Verlag.

Kruskal, Jr., J. B. 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society* **7** pp. 48–50.

Lawler, E. L. 1963. The quadratic assignment problem. *Management science* **9** 586–599.

Malick, J., F. Roupin. 2013. On the bridge between combinatorial optimization and nonlinear optimization: a family of semidefinite bounds for 0-1 quadratic problems leading to quasi-Newton methods. *Mathematical Programming B* **140** 99–124.

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

Pisinger, D. 2007. The quadratic knapsack problem – a survey. *Discrete Applied Mathematics* **155** 623–648.

Rostami, B., F. Malucelli, D. Frey, C. Buchheim. 2015. On the quadratic shortest path problem. *14th International Symposium on Experimental Algorithms – SEA 2015*, *LNCS*, vol. 9125. Springer-Verlag, 379–390.

Zhao, Q., S. E. Karisch, F. Rendl, H. Wolkowicz. 1998. Semidefinite programming relaxations for the quadratic assignment problem. *Journal of Combinatorial Optimization* **2** 71–109.