SIAM J. OPTIM.
Vol. 29, No. 2, pp. 1076–1105


# SOLVING QUADRATIC PROGRAMMING BY CUTTING PLANES*

PIERRE BONAMI†, ANDREA LODI‡, JONAS SCHWEIGER§, AND ANDREA TRAMONTANI†

**Abstract.** We propose new cutting planes for strengthening the linear relaxations that appear in the solution of nonconvex quadratic problems with linear constraints. By a famous result of Motzkin and Straus, these problems are connected to the clique number of a graph. Our cuts are derived in the context of a spatial branch-and-bound algorithm where linearization variables are introduced to represent products. Their validity is based on the result of Motzkin and Straus, in that it depends on the clique number of certain graphs. For convenience, we derive our cuts for the special case of the so-called standard quadratic programs, where the only (linear) constraint imposes that variables must belong to a simplex. We specifically consider cuts that correspond to carefully constructed complete bipartite graphs. We study the relation between these cuts and the classical ones obtained at the first level of the reformulation-linearization technique. By studying this relation, we derive a new type of valid inequalities that generalizes both types of cuts, and thus leading to a large family of cutting planes to strengthen the linear relaxation. We present extensive computational results using the different cutting planes we propose within the spatial branch and bound implemented by the commercial solver CPLEX. We show that our cuts allow us to obtain a significantly better bound than reformulation-linearization cuts and reduce computing times for global optimality. Finally, we formally establish the scaling necessary for using the proposed cuts to solve nonconvex quadratic problems with linear constraints, e.g., quadratic knapsack problems.

**Key words.** nonconvex programming, standard quadratic programming, global optimization, cutting planes, reformulation-linearization technique

**AMS subject classifications.** 90C20, 90C26

**DOI.** 10.1137/16M107428X

## 1. Introduction.

In this paper, we focus on strong convex relaxations for quadratic programs with linear constraints in a setting where auxiliary variables are used to represent products of two variables. This structure appears in the solution of general quadratic programs with linear constraints through spatial branch-and-bound (see, e.g., [3]).

We focus on constraint sets with only one linear constraint. Assuming a linear constraint $ax = b$ with $a \in \mathbb{Q}^d_+$ and $b \in \mathbb{Q}_+$, we seek to generate cutting planes for the set

$$\Gamma_{a,b} = \{(x, Y) \in \mathbb{R}^d \times \mathbb{R}^{d\times d} \mid Y = xx^T,\ x \in \Delta_{a,b}\}, \tag{1}$$

where $\Delta_{a,b} = \{x \in \mathbb{R}^d \mid a^T x = b,\ x \geq 0\}$. We will show that the proposed inequalities are also valid if the linear constraint is an inequality and under slightly weaker restrictions on $a$.

*Received by the editors May 9, 2016; accepted for publication (in revised form) December 21, 2018; published electronically April 16, 2019.

http://www.siam.org/journals/siopt/29-2/M107428.html

**Funding:** The work of the third author was partially conducted when he was a member of CPLEX Optimization, IBM Italy, and has been supported by the EU Initial Training Network MINO (grant agreement 316647), which is strongly acknowledged. The work of the second author was partially conducted when he was a faculty member at the University of Bologna and supported by MIUR, Italy, under the PRIN 2012 grant.

†CPLEX Optimization, IBM Spain, Madrid, Spain (pierre.bonami@es.ibm.com, andrea.tramontani@it.ibm.com).

‡Polytechnique de Montréal – Canada Excellence Research Chair, C.P. 6079, Succ. Centre-ville, Montréal, Québec, Canada H3C 3A7 (andrea.lodi@polymtl.ca).

§Zuse Institute Berlin, Germany (schweiger@zib.de).

Instead of studying $\Gamma_{a,b}$ directly, we may assume, without loss of generality, that $a$ is the vector $e = (1, \ldots, 1)$ and $b = 1$ by rescaling the variables in $\Gamma_{a,b}$. Indeed, by applying the linear mapping $\tilde{x}_i = \frac{a_i x_i}{b}$ and $\tilde{Y}_{ij} = \frac{a_i a_j Y_{ij}}{b^2}$, we have $(x, Y) \in \Gamma_{a,b}$ if and only if $(\tilde{x}, \tilde{Y}) \in \Gamma_{e,1}$. This equivalence leads us to study the solution of the important special case of quadratic programs

$$\text{(StQP)} \qquad \min \left\{ x^T Q x \;\middle|\; x \in \Delta_{e,1} \right\},$$

where $Q \in \mathbb{R}^d \times \mathbb{R}^d$ is a symmetric matrix and $\Delta_{e,1}$ is the *standard simplex*. Problem (StQP) is generally referred to as the *standard quadratic program* [5] and, besides being the subject of several ad hoc contributions [5, 28, 6], it has fundamental relations with copositive programming [16]. In particular, (StQP) has an exact reformulation as a copositive programming problem [4], and the solution of (StQP) can be used to test if a matrix is copositive [8].

The classical linear relaxation of the nonconvex set $\Gamma_{e,1}$ is obtained by replacing constraints $Y = xx^T$ with the linear McCormick estimators [24]. In addition, it is well known that to obtain even stronger linear relaxations of $\Gamma_{e,1}$ one can use the so-called *reformulation-linearization technique* (RLT [30]), which will be discussed with the aim of establishing a formal relationship with the novel cutting planes (cuts, for short) we propose.

Although $\Gamma_{e,1}$ is a purely continuous set, it has strong connections with combinatorial optimization and, in particular, with the maximum clique problem by a remarkable result of Motzkin and Straus [26]. Before stating the result, we recall that a clique in a simple, undirected graph $G = (V, E)$ is a subset of nodes where every node is connected to all other nodes. The size of the largest clique in $G$ is called the *clique number of* $G$ and denoted by $\omega(G)$. The problem of computing the clique number is one of Karp's 21 NP-hard problems [22]. The Motzkin–Straus theorem connects the clique number of a graph with (StQP).

THEOREM 1 (Motzkin–Straus [26]). *Let $A$ be the adjacency matrix of a simple, undirected graph $G = (V, E)$, and $\omega(G)$ be its clique number. Then, the following relation holds:*

$$\max \left\{ x^T A x \;\middle|\; x \in \Delta_{e,1} \right\} = 1 - \frac{1}{\omega(G)}.$$

Note the identification of each variable $x_i$ with node $i \in G$. This can most conveniently be seen by rewriting the objective function as a summation over the edges in $G$, as follows:

$$x^T A x = \sum_{(i,j) \in E} 2 x_i x_j.$$

The factor of 2 is due to the symmetry of the adjacency matrix. For notational convenience, in the remainder of the paper we maintain the identification of the index set of $x$ with the set $V = \{1, \ldots, d\}$ of nodes, and all considered graphs $G$ are meant to have this node set. As a by-product, it follows directly from Theorem 1 that (StQP) is an NP-hard problem.

Several authors have studied (StQP) and proposed solution methods that exploit the relationship with the max-clique problem. Our goal in this paper is to exploit Theorem 1 to obtain strong convex relaxations of $\Gamma_{e,1}$ (and thus, as discussed above, for (1)). We place ourselves in the context of a solution algorithm for (StQP) by spatial

branch-and-bound (see, e.g., [3]). We employ a classical convex relaxation of the problem using McCormick estimators [24] and strengthen it by using cutting planes that are based on solving clique problems for certain graphs. We call these inequalities *Motzkin–Straus clique (MSC) inequalities*. A generalization of these inequalities for the special case of bipartite graphs is then proposed, which we call *generalized MSC (GMSC) bipartite inequalities*.

The paper is organized as follows. In section 2, we review the basics of spatial branch-and-bound and the RLT. In section 3, we present MSC inequalities based on the theorem of Motzkin and Straus, and show connections to RLT inequalities. Extending these results, in section 4 we propose GMSC bipartite inequalities that generalize both the RLT methodology and some of our inequalities. In section 5 we propose separation algorithms to find violated inequalities. Sections 6 and 7 provide computational results that show the effectiveness of the described inequalities. Finally, section 8 states our conclusions.

**2. Relaxations.** The first step toward solving (StQP) by spatial branch-and-bound approaches is to construct a convex relaxation of the problem, which is then iteratively refined by branching. In this section, we recall the main relaxation techniques that can be applied to (StQP).

**2.1. *Q*-space relaxation.** Here, we place ourselves in the context of a reformulation of the problem where all the nonconvex terms of the objective function are replaced with linearization variables. We say that a term $Q_{ij}x_ix_j$ is convex if and only if $i = j$ and $Q_{ij} \geq 0$ or $i \neq j$ and $Q_{ij} = 0$. Accordingly, the objective matrix $Q$ is decomposed as $Q = S + P$, where $S$ contains all positive diagonal entries of $Q$ and $P = Q - S$. Linearization variables $Y_{ij}$ are introduced for all nonzero entries of $P$, and $Y_{ij} = x_ix_j$ has to hold in every feasible solution. By abuse of notation, we interpret the linearization variables as a matrix

$$Y = xx^T,$$

with the understanding that, for those components for which $P_{ij} = 0$, $Y_{ij} = x_ix_j$ does not influence the optimal value and is omitted. Then, the reformulated (StQP) is

$$\min \left\{ x^T S x + \langle P, Yn \rangle \;\middle|\; Y = xx^T,\ x \in \Delta_{e,1} \right\},$$

where the function $\langle P, Yn \rangle = \sum_{i=1}^{d} \sum_{j=1}^{d} P_{ij}Y_{ij}$ is the trace of the matrix product (or matrix scalar product). This formulation has a convex quadratic objective function and all the nonconvexities have been moved into the constraint $Y = xx^T$.

Once this reformulation has been performed, a convex relaxation can be formed by taking any convex relaxation of the feasible set $\Gamma_{e,1}$:

$$\left\{ (x, Y) \in \mathbb{R}^d \times \mathbb{R}^{d \times d} \;\middle|\; Y = xx^T,\ x \in \Delta_{e,1} \right\}.$$

Based on such a relaxation, a spatial branch-and-bound can then be performed by branching on the variables $x$ and tightening the convex relaxation of $\Gamma_{e,1}$ with the resulting local bounds at each node; see, e.g., [3, 25, 31].

*McCormick estimators.* A common way of forming a convex relaxation of $\Gamma_{e,1}$ is to relax each nonconvex equality $Y_{ij} = x_ix_j$ separately using its convex hull given by

the McCormick inequalities [24]

$$\overline{x_j}x_i + \overline{x_i}x_j - \overline{x_i}\,\overline{x_j} \le Y_{ij}, \tag{2}$$

$$\underline{x_j}x_i + \underline{x_i}x_j - \underline{x_i}\,\underline{x_j} \le Y_{ij}, \tag{3}$$

$$\overline{x_j}x_i + \underline{x_i}x_j - \underline{x_i}\,\overline{x_j} \ge Y_{ij}, \tag{4}$$

$$\underline{x_j}x_i + \overline{x_i}x_j - \overline{x_i}\,\underline{x_j} \ge Y_{ij}, \tag{5}$$

where $\overline{x_i}$ and $\underline{x_i}$ are valid upper and lower bounds on $x_i$, respectively.

We use the lower bounds $\underline{x_i} = 0$ and upper bounds $\overline{x_i} = 1$ in our initial relaxation, and obtain the convex quadratic programming (QP) relaxation of (StQP):

$$\text{(MC-StQP)} \quad \min \left\{ x^T S x + \langle P, Yn \rangle \;\middle|\; \begin{array}{l} \max\{0, x_i + x_j - 1\} \le Y_{ij} \le \min\{x_i, x_j\}, \\ Y_{ij} = Y_{ji},\ x \in \Delta_{e,1} \end{array} \right\}.$$

We refer to this relaxation as the *Q-space relaxation*.

While $Y = xx^T$ has to hold for each feasible solution, the McCormick inequalities only give a coarse approximation of the set $\Gamma_{e,1}$. We therefore strive to find valid inequalities that tighten this set and the $Q$-space relaxation.

**2.2. Reformulation-linearization technique [29, 30].** The RLT consists of two steps. In the first step, valid constraints are multiplied by other constraints or by variables yielding an equation or inequality with higher-order terms. In the second step, these terms are reformulated using linearization variables to obtain a linear constraint. The result is a valid constraint on the linearization variables that is often a very strong cutting plane;[1] see, e.g., [25]. By repeatedly applying this procedure to all constraints in a model, a hierarchy of valid constraints using higher-order terms can be established.

In the case of the (StQP), we restrict this exposition to the first order, which involves only bilinear terms. In the first step, we multiply the (only) linear constraint $\sum_{i=1}^{d} x_i = 1$ by one of the variables $x_j$, which yields the equation

$$\sum_{i=1}^{d} x_i x_j = x_j. \tag{6}$$

In the second step, the quadratic and bilinear terms in (6) are replaced with the linearization variables, leading to

$$\sum_{i=1}^{d} Y_{ij} = x_j. \tag{7}$$

A second stronger relaxation, denoted by RLT-StQP, is obtained by adding (7) to (MC-StQP) for each $j = 1, \ldots, d$.

*Projected RLT inequalities.* The RLT constraints are known to be strong, but they might use linearization variables for zero entries in $P$, i.e., either zero entries or convex terms (those in $S$) of $Q$. Because we build our relaxation in the space of nonzero entries of $P$, we need to project out those variables for which no linearization

[1] Note that the McCormick inequalities (2)–(5) can be derived by applying this technique to the bound constraints of two variables.

variable exists. Precisely, let the set $V_j$ collect all indices $i$ for which a linearization variable $Y_{ij}$ exists, namely

$$V_j = \{\, i \in V \mid P_{ij} \neq 0 \}\,. \tag{8}$$

Terms $x_i x_j$, $i \notin V_j$, are replaced by linear over- and underestimators, i.e., linear functions $o_{ij}(x_i, x_j)$ and $u_{ij}(x_i, x_j)$, such that $o_{ij}(x_i, x_j) \geq x_i x_j$ and $u_{ij}(x_i, x_j) \leq x_i x_j$ hold, and the projected RLT inequalities for each $j$ then read

$$\sum_{i \in V_j} Y_{ij} + \sum_{i \notin V_j} o_{ij}(x_i, x_j) \geq x_j, \tag{9}$$

$$\sum_{i \in V_j} Y_{ij} + \sum_{i \notin V_j} u_{ij}(x_i, x_j) \leq x_j. \tag{10}$$

The details of our implementation for the (StQP) will be discussed in subsection 6.2.

**2.3. Semidefinite programming relaxations.** Another type of relaxation can be obtained using *semidefinite programming* (SDP). Namely, in $\Gamma_{e,1}$, the nonlinear constraint $Y = xx^T$ can be relaxed to $Y - xx^T \succeq 0$, where $\succeq 0$ denotes that $Y - xx^T$ is a symmetric positive semidefinite matrix.

SDP relaxations have been used for devising strong relaxations and branch-and-bound algorithms for nonconvex QPs [11, 14]. They have been shown to be particularly strong when combined with the $Q$-space relaxation and RLT inequalities [2]. In particular, Burer [9] devised a strong relaxation, based on copositive programming, that combines the strength of the SDP constraint and RLT inequalities (see [10, Proposition 1]). While SDP relaxations are not our main focus, we will compare both theoretically and computationally the strength of our inequalities with those obtained through the relaxation of Burer. More precisely, such a comparison is theoretically discussed in section 4 and computationally analyzed through the bounds obtained by our cuts versus those obtained with QuadProgBB [14].

**3. Motzkin–Straus clique inequalities.** We now come back to Theorem 1 and its use. On the one hand, Theorem 1 can be seen as a method to compute the clique number of a graph. On the other hand, as soon as the clique number of the graph is known, a valid inequality for $\Gamma_{e,1}$ can be derived, which is extended to more general sets, such as $\Gamma_{a,b}$, at the end of this section.

COROLLARY 2. *For any simple, undirected graph $G$ with adjacency matrix $A$ and clique number $\omega(G)$, the following inequality is valid for $(x, Y) \in \Gamma_{e,1}$:*

$$\langle A, Yn \rangle \leq 1 - \frac{1}{\omega(G)}.$$

*Proof.* The inequality $x^T A x \leq 1 - \frac{1}{\omega(G)}$ for $x \in \Delta_{e,1}$ follows immediately from the Motzkin–Straus theorem. A reformulation using the definition of $\Gamma_{e,1}$ yields the result. ∎

In the remainder of the paper, we call the inequalities derived from Corollary 2 *MSC inequalities*.

For any instance of $\Gamma_{e,1}$, one can derive a different MSC inequality from any graph $G$ on $d$ nodes. However, the following theorem establishes that the inequalities stemming from certain graphs are dominated.

THEOREM 3. *Let $G = (V, E)$ be a subgraph of $\tilde{G} = (V, \tilde{E})$ with the same clique number $\omega(G) = \omega(\tilde{G})$. Then, every point that violates the MSC inequality corresponding to $G$ also violates the MSC inequality corresponding to $\tilde{G}$.*

*Proof.* Let the point $(x, Y)$ be violated by the MSC inequality corresponding to $G$. Let $A$ and $\tilde{A}$ be the adjacency matrices of $G$ and $\tilde{G}$, respectively. Since $G$ is a subgraph of $\tilde{G}$, it holds[2] that $A \leq \tilde{A}$. Then, with $Y \geq 0$ we have

$$\langle \tilde{A}, Y \rangle \geq \langle A, Yn \rangle > 1 - \frac{1}{\omega(G)}.$$

Therefore, the MSC inequality corresponding to $\tilde{G}$ is also violated. ∎

We are therefore mostly interested in graphs that are "maximal" for a certain clique number, in the sense that adding any edge increases their clique number.

Note that MSC inequalities are not dominated by McCormick inequalities and RLT inequalities. Indeed, if the original quadratic objective $Q$ of (StQP) is the adjacency matrix of a graph, then the relaxation obtained by adding the corresponding MSC inequality to the $Q$-space relaxation of $\Gamma_{e,1}$ has the same objective function of (StQP). But neither relaxations MC-StQP nor RLT-StQP solve general clique problems directly.

This observation is formalized in the following theorem, whose proof shows that a feasible point for the RLT-StQP relaxation can violate an MSC inequality obtained from a graph $G$ with clique number 2. Before proceeding to the statement of the theorem, we remind the reader that a complete bipartite graph with partition $(M, \bar{M})$ (with $M \subset V$) is a graph where every node in $M$ is connected to all nodes in $\bar{M}$, but there are no edges between any pair of nodes in $M$ or in $\bar{M}$. An obvious property of bipartite graphs is that their clique number is 2 and the MSC inequality corresponding to a bipartite graph is given by $\sum_{j \in M} \sum_{i \in \bar{M}} Y_{ij} \leq \frac{1}{4}$.

THEOREM 4. *Let $G = (V, E)$ be a complete bipartite graph. The MSC inequality obtained from $G$ is not implied by RLT equations.*

*Proof.* Let $(M, \bar{M})$ be the partition of $V$ induced by $G$, let $m = |M|$ and define a point $(\tilde{x}, \tilde{Y})$ as

$$\tilde{x}_i = \begin{cases} \frac{1}{2m} & \text{if } i \in M, \\ \frac{1}{2(d-m)} & \text{otherwise,} \end{cases}$$

$$\tilde{Y}_{ij} = \begin{cases} \frac{1}{2m(d-m)} & \text{if } (i,j) \in (M \times \bar{M}) \cup (\bar{M} \times M), \\ 0 & \text{otherwise.} \end{cases}$$

The notation $(i, j) \in (M \times \bar{M}) \cup (\bar{M} \times M)$ means that exactly one of the two indices is in $M$ and the other in $\bar{M}$.

It is easy to verify that $\tilde{x} \in \Delta_{e,1}$ and $(\tilde{x}, \tilde{Y})$ fulfills the McCormick inequalities for bounds $x \in [0, 1]^d$. Furthermore, $(\tilde{x}, \tilde{Y})$ fulfills the RLT equations (7) for every $j \in V$. Indeed, for $j \in M$,

$$\sum_{i \in V} \tilde{Y}_{ij} = \sum_{i \in \bar{M}} \tilde{Y}_{ij} = \frac{d-m}{2m(d-m)} = \frac{1}{2m} = \tilde{x}_j,$$

[2] Unless otherwise stated, we understand comparisons between two matrices and between a matrix and a scalar to be componentwise.

and for $j \in \bar{M}$,

$$\sum_{i \in V} \tilde{Y}_{ij} = \sum_{i \in M} \tilde{Y}_{ij} = \frac{m}{2m(d-m)} = \frac{1}{2(d-m)} = \tilde{x}_j.$$

However, $\tilde{Y}$ violates the MSC inequality for the bipartite graph corresponding to the partition $(M, \bar{M})$:

$$\sum_{j \in M} \sum_{i \in \bar{M}} \tilde{Y}_{ij} = \sum_{j \in M} \frac{d-m}{2m(d-m)} = \frac{m(d-m)}{2m(d-m)} = \frac{1}{2} > \frac{1}{4}. \qquad \square$$

Even though the MSC inequalities are not dominated by the RLT equations, a close relation exists. In particular, if $G = (V, E)$ is a complete graph, we can derive the corresponding MSC inequality by summing the equations (7) for all $j \in V$ to obtain

$$\sum_{j \in V} \sum_{i \in V} Y_{ij} = \sum_{j \in V} x_j = 1.$$

The last equation holds because $x$ is in the standard simplex. Moving the quadratic terms to the right-hand side and observing that $\min_{x \in \Delta_{e,1}} \sum_{i \in V} x_i^2 = d^{-1}$, the MSC inequality for the complete graph with $d$ vertices is

$$\sum_{j \in V} \sum_{\substack{i \in V \\ i \neq j}} Y_{ij} = 1 - \sum_{j \in V} Y_{jj} = 1 - \sum_{j \in V} x_j^2 \leq 1 - \frac{1}{d}.$$

So far we have focused on inequalities for $\Gamma_{e,1}$. Now, we come back to the general case and consider the set $\Gamma_{a,b}$ as defined in (1).

PROPOSITION 5. *For any simple, undirected graph $G = (V, E)$ with clique number $\omega(G)$, the following inequality is valid for $\Gamma_{a,b}$:*

$$\sum_{(i,j) \in E} 2 \frac{a_i a_j}{b^2} Y_{ij} \leq 1 - \frac{1}{\omega(G)}.$$

*Proof.* By applying the linear mappings $\tilde{x}_i = \frac{a_i x_i}{b}$ and $\tilde{Y}_{ij} = \frac{a_i a_j Y_{ij}}{b^2}$, we have $(x, Y) \in \Gamma_{a,b}$ if and only if $(\tilde{x}, \tilde{Y}) \in \Gamma_{e,1}$. From Corollary 2, it follows that the MSC inequality

$$\sum_{(i,j) \in E} 2 \tilde{Y}_{ij} \leq 1 - \frac{1}{\omega(G)} \tag{11}$$

is valid for $\Gamma_{e,1}$. By applying the mapping above to (11) we obtain the result. $\square$

Note that the requirements $a \geq 0$, $b > 0$ from the definition of $\Gamma_{a,b}$ can be further relaxed, as the scaling argument works as long as $\tilde{x} \geq 0$. Consequently, depending on the bounds on $x$, the inequalities can also be valid for negative values of $a$ and $b$.

A different scaling is used to show that MSC inequalities are also valid if the linear constraint is an inequality instead of an equation. Consider the set

$$\Gamma_{\leq} = \left\{ (x, Y) \in \mathbb{R}^d \times \mathbb{R}^{d \times d} \;\middle|\; Y = xx^T,\ e^T x \leq 1,\ x \geq 0 \right\},$$

where the equation $e^T x = 1$ is relaxed to an inequality.

PROPOSITION 6. *MSC inequalities are valid for all points* $(x, Y) \in \Gamma_{\leq}$.

*Proof.* Consider a graph $G$ and a given $(x, Y) \in \Gamma_{\leq}$. Let

$$\beta = \frac{1}{e^T x}$$

be the scaling factor. Note that $\beta \geq 1$ is constant for fixed $x$. The vector $\bar{x} = \beta x$ is in the standard simplex such that

$$\langle A, Yn \rangle = x^T A x \leq \beta^2 x^T A x = \bar{x}^T A \bar{x} \leq 1 - \frac{1}{\omega(G)}. \qquad \square$$

We end this section by noting that the arguments from Propositions 5 and 6 can be combined in a way that further extends the applicability of the Motzkin–Straus theorem to a (surprisingly) large family of optimization problems characterized by an indefinite quadratic objective function subject to a linear inequality. Obviously, such an inequality could also be obtained by the aggregation of a large(r) system of inequalities. Thus, MSC inequalities are an extremely general tool for indefinite quadratic programming.

**4. Generalized Motzkin–Straus clique inequalities for bipartite graphs.** In the previous section, we introduced MSC inequalities. In this section, we show that by performing a specific aggregation of RLT inequalities we can obtain a nonlinear inequality that generalizes MSC inequalities for bipartite graphs.

PROPOSITION 7. *Consider any subset of variables* $M \subset V$. *Then*

$$\sum_{j \in M} \sum_{i \in \bar{M}} Y_{ij} = \sum_{j \in M} x_j - \Big( \sum_{j \in M} x_j \Big)^2 \tag{12}$$

*is valid for* $\Gamma_{e,1}$.

*Proof.* First, sum the equations (6), obtained by multiplying the standard simplex constraint with $x_j$ for each $j \in M$, to obtain

$$\sum_{j \in M} \sum_{i \in V} x_i x_j = \sum_{j \in M} x_j.$$

Next, regroup all the terms that have both indices in $M$ on the right-hand side, to obtain

$$\sum_{j \in M} \sum_{i \in \bar{M}} x_i x_j = \sum_{j \in M} x_j - \sum_{j \in M} \sum_{i \in M} x_i x_j \tag{13}$$

$$= \sum_{j \in M} x_j - \Big( \sum_{j \in M} x_j \Big)^2. \tag{14}$$

Finally, linearize the products on the left-hand-side by using $Y$ to obtain (12). $\square$

Note that to go from (13) to (14) we used basic algebra to factorize the right-hand side. This step would not be satisfied by using linearization variables $Y$ and only looking at the RLT equations (7).

THEOREM 8. *Let* $f_\alpha$ *be the tangent of* $g = z - z^2$ *at* $\alpha \in [0, 1]$. *Then, for any* $M \subset V$, *the following inequality is valid for* $(x, Y) \in \Gamma_{e,1}$:

$$\sum_{j \in M} \sum_{i \in \bar{M}} Y_{ij} \leq f_\alpha \Big( \sum_{j \in M} x_j \Big). \tag{15}$$

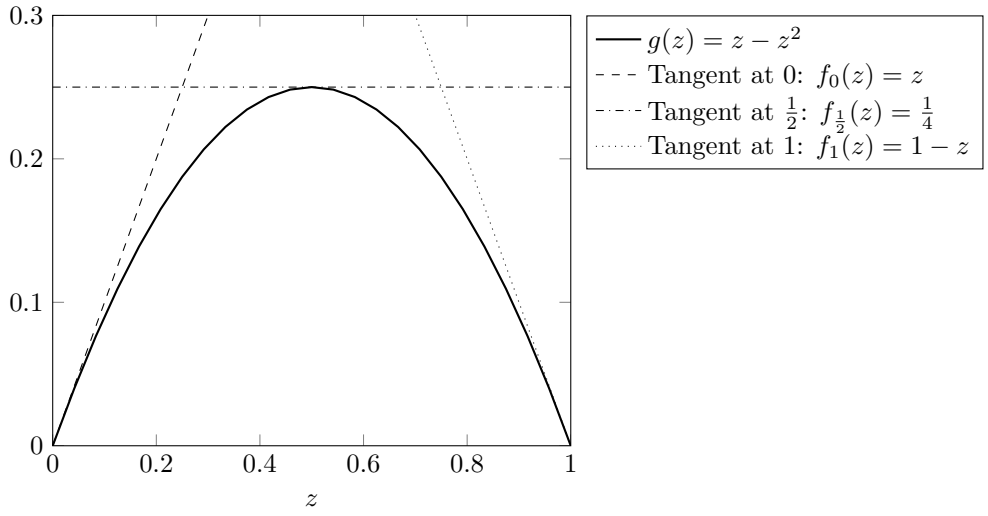


FIG. 1. *The function $g(z)$ with tangents at 0, $\frac{1}{2}$, and 1.*

*Proof.* Fix $M \subset V$ and $\alpha \in [0,1]$. Since $g$ is a concave function, the tangent $f_\alpha$ at $\alpha$ overestimates $g$. Using (12), we get

$$\sum_{j\in M}\sum_{i\in\bar{M}} Y_{ij} = g\Big(\sum_{j\in M} x_j\Big) \le f_\alpha\Big(\sum_{j\in M} x_j\Big). \qquad \square$$

Because $f_\alpha(z)$ is an affine function, the right-hand side of (15) is linear in $x$.

Of course, any missing linearization variable $Y_{ij}$ can also be projected out in a similar way to the RLT case. In this way, the inequality using a tangent $f_\alpha$ becomes

$$\sum_{j\in M}\sum_{i\in\bar{M}\cap V_j} Y_{ij} + \sum_{j\in M}\sum_{i\in\bar{M}\cap\overline{V_j}} u_{ij}(x_i,x_j) \le f_\alpha\Big(\sum_{j\in M} x_j\Big). \tag{16}$$

We denote the constraint (16) as a *generalized MSC bipartite inequality* that depends on the choice of the point $\alpha$ where the tangent is taken and of the subset $M$. The name is motivated by the fact that for $\alpha = \frac{1}{2}$ we obtain an MSC inequality from the previous section.

THEOREM 9. *For any $M \subset V$, the generalized MSC bipartite inequality at $\alpha = \frac{1}{2}$ equals the MSC inequality for the complete bipartite graph with partition $(M, \bar{M})$.*

*Proof.* Basic calculus tells us that $g(z)$ attains its maximum at $g(\frac{1}{2}) = \frac{1}{4}$ (see Figure 1 for a plot of $g(z)$ on the domain of interest, $[0,1]$). Therefore, we get that the right-hand side of (12) is less than or equal to $\frac{1}{4}$, which is exactly the MSC inequality for the complete bipartite graph with partition $(M, \bar{M})$. $\square$

It turns out that, regardless of the choice of the partition $(M, \overline{M})$, the generalized MSC bipartite inequalities obtained from $\alpha = 0$ and $\alpha = 1$ are always implied by the RLT inequalities (10).

THEOREM 10. *If a point $(x, Y) \ge 0$ satisfies the RLT inequality* (10) *for all $j \in V$, then it satisfies the generalized MSC bipartite inequalities* (16) *for $\alpha = 0$ and $\alpha = 1$ and for all $M \subset V$.*

*Proof.* Take any $M \subset V$. We assume without loss of generality that the underestimators $u_{ij}(x_i, x_j)$ are chosen to be nonnegative. Since $(x, Y) \geq 0$ and because of the validity of the RLT inequalities (10), the following chain of inequalities is valid for every $j \in V$:

$$\sum_{i \in \bar{M} \cap V_j} Y_{ij} + \sum_{i \in \bar{M} \cap \overline{V_j}} u_{ij}(x_i, x_j) \leq \sum_{i \in V_j} Y_{ij} + \sum_{i \in \overline{V_j}} u_{ij}(x_i, x_j) \leq x_j.$$

Summing over $j \in M$, we get

$$\sum_{j \in M} \sum_{i \in \bar{M} \cap V_j} Y_{ij} + \sum_{j \in M} \sum_{i \in \bar{M} \cap \overline{V_j}} u_{ij}(x_i, x_j) \leq \sum_{j \in M} x_j = f_0\Big( \sum_{j \in M} x_j \Big), \tag{17}$$

which is the generalized MSC bipartite inequality for $M$ at $f_0(z) = z$.

For the generalized MSC bipartite inequality at $f_1(z) = 1 - z$, it suffices to exchange $M$ and $\bar{M}$ in (17) and, due to $x \in \Delta_{e,1}$, it holds that

$$\sum_{j \in M} x_j = 1 - \sum_{j \in \bar{M}} x_j. \qquad \square$$

Finally, we show that GMSC bipartite inequalities are implied by the combination of the SDP inequality and RLT.

LEMMA 11. *Consider any subset of variables $M \subset V$. If $Y - xx^T \succeq 0$, then $(\sum_{i \in M} \sum_{j \in M} Y_{ij}) - (\sum_{j \in M} x_j)^2 \geq 0$.*

*Proof.* Note that $\sum_{j \in M} x_j = \mathbf{1}_M^T x$ (with $\mathbf{1}_M$ being the characteristic vector of $M$). Therefore, $(\sum_{j \in M} x_j)^2 = (\mathbf{1}_M^T x)(\mathbf{1}_M^T x) = \mathbf{1}_M^T x x^T \mathbf{1}_M$. Similarly, we have that $\sum_{i \in M} \sum_{j \in M} Y_{ij} = \mathbf{1}_M^T Y \mathbf{1}_M$. Thus, we get

$$\sum_{i \in M} \sum_{j \in M} Y_{ij} - \Big( \sum_{j \in M} x_j \Big)^2 = \mathbf{1}_M^T (Y - xx^T) \mathbf{1}_M.$$

By definition, if $Y - xx^T \succeq 0$, then $\mathbf{1}_M^T (Y - xx^T) \mathbf{1}_M \geq 0$. $\square$

THEOREM 12. *Consider any subset of variables $M \subset V$. The inequality*

$$\sum_{j \in M} \sum_{i \in \bar{M}} Y_{ij} \leq \sum_{j \in M} x_j - \Big( \sum_{j \in M} x_j \Big)^2$$

*is implied by RLT-StQP and $Y - xx^T \succeq 0$.*

*Proof.* Starting from (13) and substituting by linearization variables, we get

$$\sum_{j \in M} \sum_{i \in \bar{M}} Y_{ij} = \sum_{j \in M} x_j - \sum_{j \in M} \sum_{i \in M} Y_{ij}. \tag{18}$$

Applying Lemma 11 to (18), we obtain the result. $\square$

Figure 2 illustrates the generalized MSC bipartite inequalities that are separated in addition to RLT inequalities for one specific instance. More precisely, we separate generalized MSC bipartite inequalities as long as they are violated by using the separation algorithms that will be discussed in the next section. Each point in Figure 2

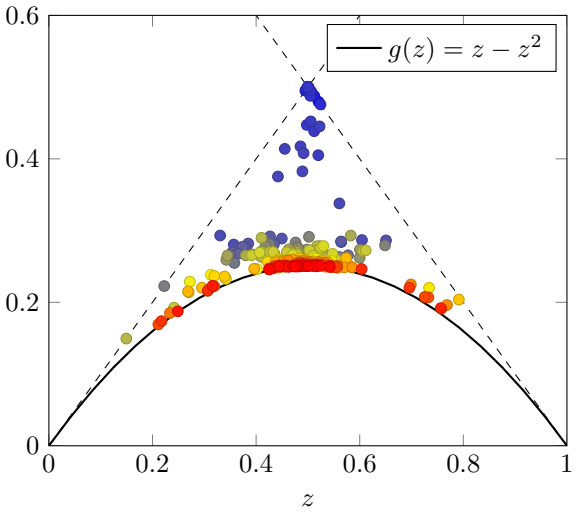


FIG. 2. *Violated points separated by generalized MSC bipartite inequalities. Notice that each point corresponds to a different set $M$.*

has the value $z = \sum_{i \in M} x_i^*$ on the horizontal axis and the value $\sum_{i \in M} \sum_{j \in \bar{M}} Y_{ij}^*$ on the vertical axis and corresponds to a different relaxation solution $(x^*, Y^*)$ and a different set $M$.

The color indicates the round in which the point was separated, and the warmer[3] the color the later the point was found in the cut loop. The plot clearly illustrates that the generalized MSC bipartite inequalities at 0 and 1 are implied for sets $M$ by the RLT inequalities, but many additional cutting planes can be separated.

By the same scaling arguments introduced in the previous section, Generalized MSC bipartite inequalities can be extended to $\Gamma_{\leq}$ and $\Gamma_{a,b}$. Precisely, inequalities (15) and (16) are also valid for $\Gamma_{\leq}$ and can be generalized to $\Gamma_{a,b}$. In particular, (15) can be generalized as

$$\sum_{i \in M} \sum_{j \in \bar{M}} \frac{a_i a_j}{b^2} Y_{ij} \leq f_\alpha \left( \sum_{i \in M} \frac{a_i}{b} x_i \right)$$

for the set $\Gamma_{a,b}$.

**5. Separation.** For the ease of notation, in this section we concentrate on the separation of MSC inequalities and generalized MSC bipartite inequalities valid for $\Gamma_{e,1}$. Using the scaling of variables, the separation models are easily adapted to $\Gamma_{a,b}$. For sets with an inequality, such as $\Gamma_{\leq}$, the models remain unchanged.

To separate a violated MSC inequality on $\Gamma_{e,1}$, a graph has to be determined and its clique number has to be computed. In addition to the fact that the latter computation is NP-hard, this boils down to a *bilevel separation problem* [23] with the determination of the graph in the first level and the computation of the clique number in the second one. This turns out to be computationally very hard because bilevel (integer-integer) optimization problems are, in general, extremely challenging both in theory and in practice (see, e.g., [12]).

[3]Generally, points closer to the curve in Figure 2 have been found later in the cut loop.

We focus on finding graphs with a fixed clique number, especially bipartite graphs, and devise exact separation algorithms for these two classes. This corresponds to removing the second level above and is by far the best way we have found to generate effective cutting planes. We have also tried a number of algorithms based on the idea of using the solution of the continuous relaxation as a starting point to heuristically find graphs that yield violated inequalities. This approach corresponds to eliminating the first level. Although some neat bound improvements can be obtained in this way (e.g., roughly 1% over some of the algorithms discussed in subsection 6.3), it turns out that computing the clique number of instances of relevant sizes is feasible but not computationally effective, leading to excessive separation times. Detailed results for these attempts are reported in [27].

In the following, we will study mathematical programming formulations to separate MSC inequalities on specific classes of graphs, striving for exact separation. This is computationally viable if we restrict ourselves to graphs with known clique number so as to avoid solving bilevel programming problems (to separate a single cut). First, we will concentrate on general graphs with fixed clique number. Iterating over all possible clique sizes yields a separation algorithm for general graphs. Second, we focus on complete bipartite graphs, which clearly have clique number 2.

*Graphs with fixed clique number.* Consider a point $(x^*, Y^*)$ and a fixed integer $k > 1$. The aim is to find a graph with clique size at most $k$ whose corresponding MSC inequality separates $(x^*, Y^*)$ from $\Gamma_{e,1}$. Since $x^* \in \Delta_{e,1}$, without loss of generality $(x^*, Y^*)$ can be assumed to be nonnegative. Then, the following mixed-integer linear programming (MILP) problem serves the purpose:

$$\max \quad \langle A, Y^* n \rangle - \left(1 - \frac{1}{k}\right) \tag{19}$$

$$\text{s.t.} \quad \sum_{\substack{i,j \in S \\ i<j}} A_{ij} \leq \frac{|S|\,(|S|-1)}{2} - 1 \qquad \text{for all } S \subseteq V,\ |S| = k+1, \tag{20}$$

$$A_{ij} = A_{ji} \qquad \text{for all } i, j \in V, \tag{21}$$

$$A_{ij} = 0 \qquad \text{for all } i \notin V_j, \tag{22}$$

$$A_{ij} \in \{0, 1\} \qquad \text{for all } i, j \in V. \tag{23}$$

The above mixed-integer linear program maximizes the violation of the cut, and $(x^*, Y^*)$ can be separated if and only if the objective value is greater than zero. Since the clique size is fixed, only the graph (in the form of its adjacency matrix $A$) has to be computed. Constraints (21), (22), and (23) ensure that $A$ is indeed the adjacency matrix of a simple undirected graph. The inequalities (20) ensure that $G$ contains no clique of size $k+1$. To this end, it requires that from every set of $S \subseteq V$ of cardinality $k+1$ at least one of the $\frac{|S|(|S|-1)}{2}$ possible edges is missing.

A posteriori, if the clique size of $G$ is smaller than $k$, then all edges connecting a maximum clique with the rest of the nodes have zero weight. Adding enough of these nodes will yield a graph with clique size $k$ and the same objective value.

The drawback of this formulation is its exponential size for fixed $k$, which makes it impractical for computational purposes.

*Complete bipartite graphs.* We now turn to the separation of an MSC inequality stemming from bipartite graphs with partition $(M, \bar{M})$. We will always assume that both sets in the partition are nonempty and restrict ourselves to complete bipartite graphs (bipartite graphs such that adding any edge forms a triangle), since these are

maximal w.r.t. the clique number and thus yield the strongest inequalities. For our purposes, bipartite graphs have two advantages: first, the clique number is 2 and therefore the MSC inequalities have the best right-hand-side value. Second, they have a very clean structure. For a fixed subset $M \subsetneq V$ of nodes, the MSC inequality corresponding to the complete bipartite graph with partition $(M, \bar{M})$ is

$$\sum_{i \in M} \sum_{j \in \bar{M}} 2Y_{ij} \leq 1 - \frac{1}{2}.$$

Separating a maximally violated MSC inequality corresponding to some complete bipartite graph means finding a bipartite graph with maximum weight, where the weight for each edge $(i, j)$ is given by $Y^*_{ij}$. This is equivalent to finding a maximum-weight cut and is thus NP-hard [22]. However, since both the number of nodes and the cardinality of the support (i.e., the nonzero values) of $Y^*$ are typically relatively small, it is computationally feasible to separate by solving a simple binary QP. To this end, we introduce a binary variable $z_i$ for each $i \in V$ and say that nodes whose variables take the same value are in the same set of the partition. The problem to be solved is

$$\begin{aligned} \max \quad & \sum_{i \in V} \sum_{j \in V} 2Y^*_{ij}\, z_i(1 - z_j) - \frac{1}{2} \\ \text{s.t.} \quad & z \in \{0,1\}^{|V|}. \end{aligned} \tag{24}$$

We assume without loss of generality that $Y^*$ is symmetric. The product $z_i(1 - z_j)$ ensures that $Y^*_{ij}$ is counted if and only if $z_i = 1$ and $z_j = 0$, i.e., $i$ and $j$ are in different sets. The objective function therefore maximizes the violation of the cut. Every solution with positive objective function value corresponds to a violated cut with partition $(M, \bar{M})$, where $M = \{\, i \in V \mid z_i = 1\}$. If the optimal objective value is nonpositive, no violated cut exists.

*Generalized MSC bipartite inequalities.* To separate a violated generalized MSC bipartite inequality a set $M$ has to be found such that

$$\sum_{i \in M} \sum_{j \in \bar{M}} Y^*_{ij} > g\Big( \sum_{i \in M} x^*_i \Big).$$

The generalized MSC bipartite inequality for $M$ and $\alpha = \sum_{i \in M} x^*_i$ will then separate this point. The separating binary QP for bipartite graphs can be generalized to separate violated generalized MSC bipartite inequalities. It maximizes the violation and adds the $\alpha$ as one of the decision variables. Namely,

$$\begin{aligned} \max \quad & \sum_{i \in V} \sum_{j \in V} Y^*_{ij}\, z_i(1 - z_j) - (\alpha - \alpha^2) \\ \text{s.t.} \quad & \alpha = \sum_{i \in V} x^*_i z_i, \\ & z \in \{0,1\}^{|V|},\ \alpha \geq 0. \end{aligned} \tag{25}$$

As for bipartite graphs, nodes are partitioned according to the value of their associated variables.

We end the section by proving the following simple proposition.

PROPOSITION 13. *Given* $(x^*, Y^*)$ *with* $x^* \in \Delta_{e,1}$, *if* $Y^*_{ij} \leq x^*_i x^*_j \ \forall (i,j)$, *no MSC inequality or GMSC bipartite inequality can be violated.*

*Proof.* We first prove the result for MSC inequalities. Given $x \in \Delta_{e,1}$, for every simple, undirected graph $G = (V, E)$ we have $\sum_{(i,j)\in E} 2x_i x_j \leq 1 - \frac{1}{\omega(G)}$ by Theorem 1. Therefore, if $Y^*_{ij} \leq x^*_i x^*_j \ \forall (i,j)$, MSC inequalities cannot be violated because $1 - \frac{1}{\omega(G)} \geq \sum_{(i,j)\in E} 2x^*_i x^*_j \geq \sum_{(i,j)\in E} 2Y^*_{ij}$.

Next, we prove the result for GMSC bipartite inequalities. Following Proposition 7 and Theorem 8, it is easy to observe that the inequality

$$\sum_{j\in M} \sum_{i\in \bar{M}} x_i x_j \leq f_\alpha\Big( \sum_{j\in M} x_j \Big)$$

holds for every $M \subset V$ and for every $x \in \Delta_{e,1}$. If $Y^*_{ij} \leq x^*_i x^*_j \ \forall (i,j)$, then GMSC bipartite inequalities cannot be violated because

$$f_\alpha\Big( \sum_{j\in M} x^*_j \Big) \geq \sum_{j\in M} \sum_{i\in \bar{M}} x^*_i x^*_j \geq \sum_{j\in M} \sum_{i\in \bar{M}} Y^*_{ij}. \qquad \square$$

**6. Computational experiments on (StQP) instances.** In this section, we present the results of a large set of computational experiments on (StQP) instances. They were carried out on a cluster of Intel Xeon 5160 quadcore CPUs running at 3.00 GHz with 8 GB RAM and using RHEL5 as the operating system. To avoid random noise by cache misses and similar, only one process was executed on each node at a time.

The implementation is based on a slightly modified version of the IBM CPLEX Optimizer 12.6.3 [21] (CPLEX for short) where the C-API has been extended to provide callbacks with access to the linearization variables $Y$. The cuts are separated from the user cut callback only at the root node, and are added with the *purgeable flag* set to `CPX_USECUT_PURGE`, in order to allow CPLEX to purge the cuts that are deemed to be ineffective according to its internal strategies.

Our computational investigation focuses on the application of the proposed cutting planes in the context of a general spatial branch-and-bound algorithm based on the $Q$-space relaxation (MC-StQP), and studies their impact on the root node and on the overall solution time. Therefore, we omit a direct comparison with specific formulations or solution approaches as presented, for example, in [28]. Nevertheless, in subsection 6.5 we report computational results on the small set of publicly available instances from [28], while in subsection 6.6 we compare our results with the ones obtained with QuadProgBB [14]. We begin by describing the large collection of randomly generated instances we based our extensive computation on.

**6.1. Instances.** We considered a large set of randomly generated instances and, in particular, two sizes, $d = 30$ and $d = 50$. Because the only linear constraint is a simplex one, the part of the problem to be randomly generated is the objective matrix. The instances are available at http://or.dei.unibo.it/library/msc.

The sign of the objective coefficients plays a major role in these instances. Assuming all terms are linearized (i.e., all diagonal entries are negative and all off-diagonal entries are not zero), the objective function only acts on the $Y$ variables. When optimizing the value of $Y$ over $\Gamma_{e,1}$, $Y_{ij}$ with $i \neq j$ is restricted by the McCormick inequalities, and whether it will take the upper or the lower bound is defined by the

sign of $Q_{ij}$, namely

$$Y_{ij} = \begin{cases} \max\{0, x_i + x_j - 1\} & \text{if } Q_{ij} > 0, \\ \min\{x_i, x_j\} & \text{if } Q_{ij} < 0. \end{cases}$$

We therefore strive to generate instances with different fractions of positive and negative entries in $Q$. Since the inequalities presented in this paper can only cut points where at least some entries $Y_{ij}$ exceed $x_i x_j$, the biggest impact is expected for instances with many negative entries in $Q$.

We used triangular distributions, which are characterized by three parameters, $a < c < b$. Namely, $a$ and $b$ are the minimum and the maximum of the value range. The mode $c$ describes the peak of the piecewise linear density function. Of course, the sign of the coefficients has a high impact in general; on the diagonal this even decides whether the respective terms are convex. We therefore use the triples $(-10, -5, 0)$ and $(0, 5, 10)$ to get instances with only negative and only positive coefficients, respectively. For instances with mixed signs, we used the triples $(-10, -3, 10)$, $(-10, 0, 10)$, and $(-10, 3, 10)$, where the second triple is a symmetric distribution and the other two are more likely to have positive and negative coefficients, respectively. The diagonal entries are divided by 2. In addition, 2 variants of each instance are generated by taking the positive and negative absolute values of the diagonal elements. For $(-10, -5, 0)$ only the positive and for $(0, 5, 10)$ only the negative variants are generated since the respective opposite would yield the same instance again. Furthermore, the instances with positive off-diagonal entries (distribution $(0, 5, 10)$) in the variant with negative diagonal entries are trivially solved by all approaches. Indeed, since the objective is to minimize, setting the variable $x_i$ with lowest diagonal entry $Q_{ii}$ in the objective to 1 gives the optimal solution. For this reason, these instances are excluded.

The 3 distributions in 3 variants, 1 distribution in 2 variants for the diagonal, and 1 distribution in 1 variant for the diagonal give 12 different instance types. For each instance type and for each size $d \in \{30, 50\}$ we generated 10 instances, yielding 120 instances with $d = 30$ and 120 instances with $d = 50$ in total. All instances are available at http://or.dei.unibo.it/library/msc. In all computational experiments we enforced a time limit of 2 hours for instances of size $d = 30$ and 6 hours for those of size $d = 50$.

Since, as shown in Proposition 13, we can only separate MSC inequalities and generalized MSC bipartite inequalities if the linearization variable $Y_{ij}$ exceeds the respective product, i.e., $Y_{ij} > x_i x_j$ for some $(i, j)$, one could assume that the instances $(0, 5, 10)$ will not be affected by MSC inequalities, given that the objective function drives the linearization variables toward zero. This is only true for instances with positive diagonal elements in $Q$. For these instances, the quadratic terms $Q_{ii} x_i^2$ are convex and thus not linearized, and the resulting projected RLT inequalities are redundant. For the variation with negative diagonal elements in $Q$, the quadratic terms are linearized and the RLT equations $\sum_i Y_{ij} = x_j$ for all $j \in V$ force some $Y_{ij}$ to be positive. In all these instances it is then possible to separate MSC inequalities and generalized MSC bipartite inequalities.

**6.2. Projected RLT.** As anticipated at the end of section 2, special care has to be taken to deal with writing RLT equations when there are missing bilinear terms. Specifically, RLT equations (7) cannot be enforced, and projected RLT inequalities (9) and (10) have to be separated instead, with a suitable choice of the over- and underestimators $o_{ij}(x_i, x_j)$ and $u_{ij}(x_i, x_j)$. In our implementation of (9) and (10),

we used $o_{ij}(x_i, x_j) = \overline{x_i}x_j$ and $u_{ij}(x_i, x_j) = \underline{x_i}x_j$. For bilinear terms, the McCormick over- and underestimators would also be natural choices. However, in our experiments, the objective matrix is dense, and thus only the diagonal entries of $P$ might be zero.

The same approach can be used to derive a valid RLT inequality from a general equation $ax = b$. There, the sign of $a_i$ has to be considered for the estimators of $a_i x_i x_j$ such that the RLT inequalities for each $j$ become

$$\sum_{i \in V_j} a_i Y_{ij} + \sum_{\substack{i \notin V_j \\ a_i \geq 0}} a_i o_{ij}(x_i, x_j) + \sum_{\substack{i \notin V_j \\ a_i < 0}} a_i u_{ij}(x_i, x_j) \geq b x_j,$$

$$\sum_{i \in V_j} a_i Y_{ij} + \sum_{\substack{i \notin V_j \\ a_i \geq 0}} a_i u_{ij}(x_i, x_j) + \sum_{\substack{i \notin V_j \\ a_i < 0}} a_i o_{ij}(x_i, x_j) \leq b x_j.$$

Similarly, an inequality $ax \leq b$ can be the starting point, but in this case the constraint needs to be multiplied by a nonnegative or nonpositive quantity. Usually, instead of multiplying the constraint by $x_j$ one multiplies it by $x_j - l_j$ or $u_j - x_j$.

**6.3. Optimizing over the bipartite closures.** As a first set of experiments, we want to evaluate the impact of MSC inequalities corresponding to bipartite graphs and generalized MSC bipartite inequalities at the root node of the branch-and-cut tree. Since RLT equations and inequalities can be easily separated by enumeration and are expected to be effective, we separate our inequalities only if no violated RLT inequalities can be found.

For MSC and GMSC bipartite inequalities we have exact separation algorithms and thus we can optimize over the associated closures. The closure of a class of inequalities is the relaxation obtained by adding all possible inequalities of this class. Although the closure itself may be intractable to compute, one can optimize a linear function over it by separation. A comparison of the values gives an indication of the strength of the class of inequalities.

MSC and GMSC bipartite inequalities are separated by solving the associated mathematical models (24) and (25) in a classical cutting-plane scheme, by using CPLEX as a black box. To limit the tailing-off effect that often arises in cutting plane algorithms, we try to separate up to five cuts per round, i.e., at every separation round we collect the first five incumbent solutions returned by CPLEX that correspond to violated cuts. Specifically, we consider the following four settings.

`CPLEX` CPLEX with empty cut callback.

`RLT` Violated RLT equations and inequalities are added.

`Bipartite` At each call of the cut callback, first violated RLT equations and inequalities and then violated MSC bipartite inequalities are added. If no inequality can be separated anymore, the final dual bound gives the value of the closure over these two types of cuts.

`GMSC` The same as `Bipartite`, but GMSC bipartite inequalities are separated instead of MSC bipartite inequalities.

Each of these configurations improves the closure of the previous ones, since `Bipartite` and `GMSC` also separate RLT equations and inequalities and since MSC inequalities for bipartite graphs are generalized MSC bipartite inequalities at $\alpha = 0.5$.

Tables 1 and 2 report aggregated results at the root node for these configurations on the instances of size $d = 30$ and $d = 50$, respectively. Both tables have the following structure. First, we report the average root gap to measure the strength of the separated cuts. For all considered approaches we give the percentage gap left

at the root, computed as $(UB - LB_{root})/|UB|$, where $LB_{root}$ is the dual bound at the root node and $UB$ is the optimal solution value as computed by our approaches or QuadProgBB (see subsections 6.4 and 6.6). Then we report the gap closed with respect to the CPLEX root (resp., CPLEX with RLT inequalities), computed as $(LB_{root} - LB_{base})/(UB - LB_{base})$, where $LB_{base}$ is the dual bound obtained by the CPLEX root (resp., CPLEX with RLT). Then, the number of instances solved to proven optimality is given, along with the number of time limits hit. Next, we give the average and maximum separation time, first considering all instances and then disregarding the instances where any of the compared approaches reach the time limit. In addition, we report the average and maximum number of separated cuts, along with the number of cuts applied to the root linear program (LP) at the end of root node, as reported by CPLEX. Finally, the average and maximum number of separation rounds are given, to specify how many times the callback was called (note that in the last round no cut was separated, otherwise the callback would have been called again). In both tables we do not report `CPLEX`, because no cuts are generated.

TABLE 1
*Comparing the closures on standard quadratic programming problems of size* 30.

| | `Bipartite` | `GMSC` | `GraphPool` | `Hybrid` | `bst200` |
|---|---|---|---|---|---|
| **Solved/timeout at the root** | | | | | |
| Solved | 0 | 3 | 0 | 3 | 3 |
| Timeout | 0 | 3 | 0 | 0 | 0 |
| **Separation time in seconds** | | | | | |
| Mean | 1.24 | 425.39 | 7.23 | 17.07 | 48.61 |
| Max | 14.50 | 7139.10 | 118.30 | 142.00 | 799.30 |
| **Separation time in seconds (exclude time limit)** | | | | | |
| Mean | 1.02 | 255.04 | 5.15 | 15.06 | 34.34 |
| Max | 14.60 | 5651.30 | 98.30 | 147.20 | 730.40 |
| **Number of cuts** | | | | | |
| Separated mean | 97.62 | 748.83 | 2456.58 | 433.13 | 5565.64 |
| Separated max | 382.00 | 4651.00 | 16491.00 | 782.00 | 32271.00 |
| Applied mean | 53.72 | 84.67 | 78.16 | 77.34 | 104.12 |
| Applied max | 137.00 | 228.00 | 200.00 | 168.00 | 256.00 |
| **Number of separation rounds** | | | | | |
| Mean | 47.29 | 225.28 | 96.39 | 216.73 | 161.24 |
| Max | 176.00 | 1032.00 | 515.00 | 376.00 | 710.00 |
| **Average root gap** [%] | | | | | |
| Gap left | 21.55 | 11.04 | 13.73 | 11.33 | 11.06 |
| Closed w.r.t. CPLEX root | 90.02 | 91.22 | 90.94 | 91.20 | 91.22 |
| Closed w.r.t. `RLT` | 68.81 | 86.19 | 81.93 | 85.69 | 86.17 |

The results reported in the tables clearly show that RLT inequalities are fundamental for (StQP). Indeed, by themselves, RLT inequalities already close about 85% of the root gap obtained by default CPLEX. On the other hand, MSC and GMSC bipartite inequalities are very effective at improving on the dual bound on top of RLT inequalities, and the GMSC bipartite closure appears definitely stronger than the MSC bipartite closure. `Bipartite` and `GMSC` greatly improve over `RLT`, reducing the arithmetic mean of the root gap, and `GMSC` halves the gap left by `Bipartite`. Concerning the separation time, `Bipartite` appears on average to be very fast, while `GMSC` is two orders of magnitude more time consuming. With `GMSC`, three instances of size $d = 30$ and ten of size $d = 50$ do not finish the root node within the time limit.

TABLE 2
*Comparing the closures on standard quadratic programming problems of size* 50.

| | RLT | Bipartite | GMSC | GraphPool | Hybrid |
|---|---|---|---|---|---|
| **Average root gap [%]** | | | | | |
| Gap left | 61.14 | 20.40 | 10.36 | 13.66 | 11.20 |
| Closed w.r.t. CPLEX root | 87.94 | 90.73 | 91.46 | 91.23 | 91.42 |
| Closed w.r.t. RLT | — | 68.46 | 86.80 | 81.10 | 85.24 |
| **Solved/timeout at the root** | | | | | |
| Solved | 0 | 0 | 0 | 0 | 0 |
| Timeout | 0 | 0 | 10 | 0 | 0 |
| **Separation time in seconds** | | | | | |
| Mean | 0.00 | 18.07 | 2183.95 | 203.22 | 92.98 |
| Max | 0.00 | 533.10 | 21488.30 | 4732.70 | 2097.00 |
| **Separation time in seconds (exclude time limit)** | | | | | |
| Mean | 0.00 | 2.23 | 439.81 | 20.02 | 28.17 |
| Max | 0.00 | 7.00 | 8266.00 | 230.20 | 121.40 |
| **Number of cuts** | | | | | |
| Separated mean | 45.63 | 240.19 | 1771.27 | 12281.52 | 676.42 |
| Separated max | 50.00 | 2068.00 | 9230.00 | 76722.00 | 2634.00 |
| Applied mean | 45.63 | 97.75 | 159.83 | 150.72 | 136.15 |
| Applied max | 50.00 | 326.00 | 354.00 | 447.00 | 349.00 |
| **Number of separation rounds** | | | | | |
| Mean | 1.96 | 102.91 | 477.81 | 285.40 | 285.59 |
| Max | 3.00 | 583.00 | 2175.00 | 1210.00 | 783.00 |

However, it is remarkable that GMSC is able to solve three instances of size $d = 30$ without resorting to branching.

To investigate the main differences between Bipartite and GMSC we analyzed closely the evolution of the root node for some specific instances. All the plots reported in the following are given for one instance of size $d = 30$ with positive diagonal entries and distribution $(-10, 0, -5)$ (i.e., instance *triangular_30_-10_0_-5__04_posDiag*). This instance has been selected as the one for which the separation time of both Bipartite and GMSC exceeds the respective arithmetic mean by the smallest amount, but the plots would look similar for other instances.

Figure 3 shows the evolution of the dual bound from round to round. Even if GMSC converges toward a stronger dual bound, Bipartite is superior in the first rounds and shows a very limited tailing-off effect. On the contrary, GMSC stalls after about 200 rounds, and after that each round of cuts increases the bound only by a very small amount.

Figure 4 plots the time for each round of separation for Bipartite and GMSC for the same instance. For Bipartite the separation times remain almost constant at a very low value. For GMSC, in contrast, separation times are modest for the first rounds, but soon start to increase, with outliers taking more than 20 seconds. Such a difference in the separation time between Bipartite and GMSC can be easily explained: the former separation problem is a binary QP that can be linearized and solved by MILP techniques, while the latter has a nonconvex quadratic continuous variable (namely $\alpha$) and requires spatial branch-and-bound to be solved.

Finally, we analyze the diversity of the graphs that are generated by Bipartite and GMSC. To this end, we compare each graph returned by the separation problems to all graphs that have been separated previously. The difference between two bipartite

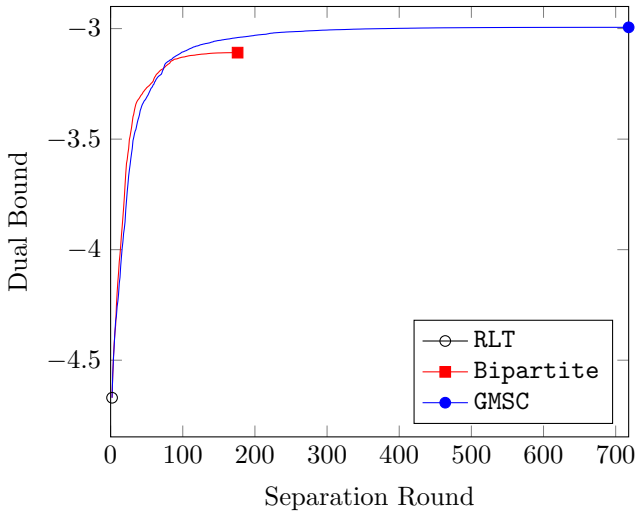


FIG. 3. *Evolution of the dual bound for instance* triangular_30_-10_0_-5__04_posDiag.

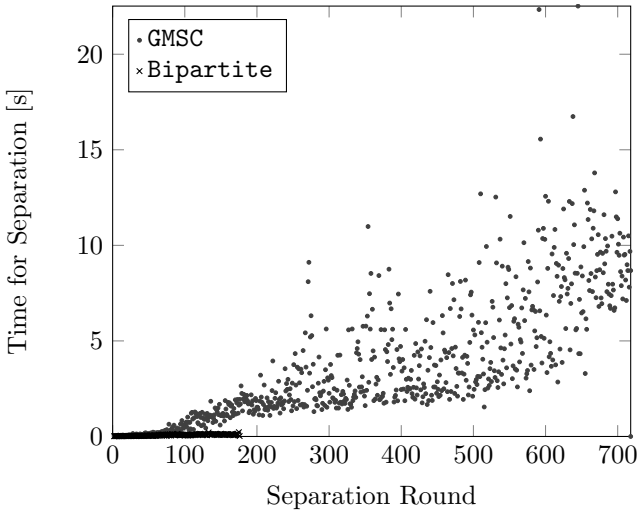


FIG. 4. *Separation time per round for instance* triangular_30_-10_0_-5__04_posDiag.

graphs is defined in terms of the partitions: let $\mathcal{M} = (M, \bar{M})$ and $\mathcal{N} = (N, \bar{N})$ be two partitions of the same set. Then, define the distance $d(\mathcal{M}, \mathcal{N})$ between the partitions by

$$d(\mathcal{M}, \mathcal{N}) = \min(|M \triangle N|, |M \triangle \bar{N}|),$$

where $\triangle$ is the symmetric difference. Note that $M \triangle N = \bar{M} \triangle \bar{N}$, so the above is well defined.

Figure 5 plots the minimum distance of every graph to all previous graphs for `Bipartite` (Figure 5(a)) and `GMSC` (Figure 5(b)). Specifically, for each round, the figure reports the minimum distance of every graph generated at the given round. Since `Bipartite` requires only 175 rounds to converge against the 717 required by

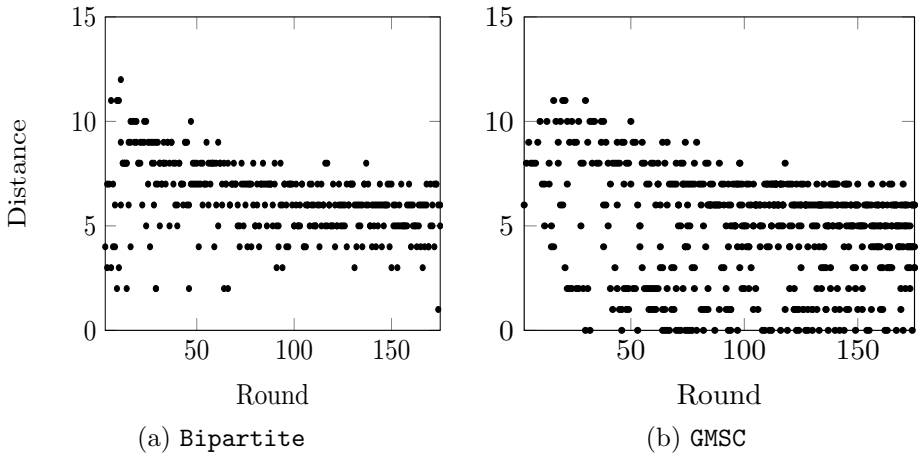


FIG. 5. *Distance to known graphs for instance* triangular_30_-10_0_-5_04_posDiag.

GMSC, the plot for GMSC is restricted to the first 175 rounds. The picture clearly shows that Bipartite tends to separate cuts associated with bipartite graphs that are more diverse with respect to GMSC. While for Bipartite most of the graphs have a distance of between 5 and 10 to the previously separated graphs, we see a lot of graphs that are very similar (e.g., a distance less than or equal to 2) to one of the previous graphs for GMSC. Frequently, we even separate from the same graph multiple times with different values of $\alpha$. This is problematic since the resulting cuts in this case are very similar. Indeed, the more diversity in the observed graphs, the more diverse the cuts are.

In order to overcome the main drawbacks of GMSC discussed above while trying to approximate the GMSC bipartite closure as much as possible, we have tried a number of heuristics to combine the separation of MSC and GMSC bipartite inequalities. Detailed results on these heuristic versions are reported in [27]. In this paper, we focus on two main ideas, denoted GraphPool and Hybrid.

In GraphPool, we adopt a heuristic approach to find violated GMSC bipartite inequalities without solving the corresponding separation problem (25). Namely, while separating MSC bipartite inequalities, we store the new graphs (i.e., the partitions) that get separated in a *graph pool*. After adding the MSC bipartite inequalities, we compute the GMSC bipartite inequalities from all graphs in the pool with respect to the current relaxation solution $(x^*, Y^*)$ and add the violated ones. This approach is expected to generate a lot of very similar cuts at every round, and thus we rely on a CPLEX cut purging to discard the ones that are not useful. The repeated separation of GMSC bipartite inequalities from the same graphs can be seen as a way to "update" the existing cuts based on the current relaxation solution to cutoff.

In Hybrid, we combine the exact separation of MSC and GMSC bipartite inequalities as follows. First, we optimize over the MSC bipartite closure (i.e., perform Bipartite) and then we separate up to 200 rounds of GMSC bipartite inequalities (i.e., perform up to 200 rounds of GMSC). Further, at each separation round of GMSC, we enforce a deterministic work limit as follows. Let $\tau_0$ be the deterministic time[4] needed to find the first violated cut, and $\Phi > 0$ be a scaling parameter. The additional

[4]CPLEX uses a deterministic measure of the work it performs, called *ticks*, and allows us to set proper deterministic time limits accordingly. See [1, 20] for details.

deterministic time to find the $i$th violated cut after finding $i-1$ of them is then limited by $\tilde{\tau}_i = \tau_0 \Phi^i$. In our computations, we chose $\Phi = 0.9$, so we allow less work on every iteration.

The results obtained with `GraphPool` and `Hybrid` are reported in the last columns of Tables 1 and 2 and show that both approaches are quite effective in reaching the simultaneous goals of closing almost as much gap as generalized MSC bipartite inequalities and a quite reasonable computational price.

**6.4. Branch-and-cut results.** Branch-and-cut results obtained with all the approaches discussed in subsection 6.3 are given in Table 3 for the case when $d = 30$ and in Table 4 for the case when $d = 50$. The tables have the same structure and report aggregated results on all the instances that can be solved to optimality by at least one of the considered approaches. For the case when $d = 30$, only 3 instances cannot be solved within the time limit of 2 hours, while 10 instances with $d = 50$ are not solved by any of the methods within the time limit of 6 hours. Interestingly, all the unsolved instances are generated with positive diagonal entries and distribution $(-10, -5, 0)$. This is not particularly surprising, since for those instances the $Q$-space relaxation (MC-StQP) is expected to be very weak. Indeed, the negative objective coefficients drive the relaxation variables $Y_{ij}$ toward $\min(x_i, x_j)$, which is typically much further away from the correct values of $x_i x_j$ than the opposite bound, which is zero. For example, taking $x_i = x_j = \frac{1}{n}$, the correct value would be $x_i x_j = \frac{1}{n^2}$, but the linearization variable takes the value $Y_{ij} = \frac{1}{n}$. Furthermore, on these instances the diagonal terms are not linearized, and thus only the weaker projected RLT inequalities can be separated instead of RLT equations.

Each of the tables gives separate results on all solved instances and on solved "hard" instances, where an instance of size $d = 30$ (resp., $d = 50$) is considered to be hard if it takes at least 30 seconds (resp., 300 seconds) to be solved with all compared approaches. For each of the tested methods and for each class of problems, the tables report the number of solved instances, the average computing time in seconds, the shifted geometric mean of the computing times (with a shift of 10 seconds), the average number of branch-and-bound nodes, the shifted geometric mean of the number of nodes (with a shift of 100 nodes), and the average percentage gap left at the root node. Time limits are accounted for in the computations on the running time, and out-of-memory errors (that only happen for `CPLEX`) are accounted as time limits. Computing the average and geometric mean of the number of nodes is problematic for instances that are not solved to optimality. To make a fair comparison, the calculations for the number of nodes only consider those instances where all solvers but `CPLEX` can solve within the time limit. For `CPLEX`, the number of nodes processed until running out of memory or time is used, and thus the reported numbers of nodes for `CPLEX` (which are already an order of magnitude higher than those for `RLT`) are underestimated.

The branch-and-cut results given in Tables 3 and 4 are consistent with the ones reported in subsection 6.3 for the root node. On the one hand, RLT inequalities appear to be fundamental for (StQP), since `RLT` clearly outperforms `CPLEX`. On the other hand, MSC and GMSC bipartite inequalities are also very effective. Indeed, `Bipartite` outperforms `RLT` on all the performance indicators reported in the tables (i.e., number of solved instances, computing time, and number of branch-and-bound nodes), while, in turn, both `GraphPool` and `Hybrid` provide a neat improvement over `Bipartite`, especially on the hard instances. This indicates that MSC bipartite inequalities are important on top of RLT inequalities and that a "clever" selection of GMSC bipartite inequalities is also important to improve over MSC bipartite

TABLE 3
*Branch-and-cut results on standard quadratic programming problems of size* 30.

| | Solved | Time [s] | | Nodes | | Root gap |
|---|---|---|---|---|---|---|
| | | Avg. | S. Geom. | Avg. | S. Geom. | Avg. |
| **All 117 instances solved by at least one** | | | | | | |
| `CPLEX` | 88 | 2394.6 | 503.4 | 238799.2 | 64871.3 | 982.6% |
| `RLT` | 110 | 471.9 | 23.2 | 17581.6 | 1975.2 | 67.3% |
| `Bipartite` | 117 | 245.6 | 17.3 | 7951.0 | 1005.4 | 21.8% |
| `GMSC` | 115 | 325.4 | 31.3 | 1420.0 | 468.8 | 11.2% |
| `GraphPool` | 117 | 103.7 | 14.5 | 2245.6 | 533.4 | 13.9% |
| `Hybrid` | 117 | 92.3 | 23.0 | 2060.4 | 465.3 | 11.4% |
| **15 hard instances (all more than 30 seconds)** | | | | | | |
| `CPLEX` | 2 | 6273.9 | 4616.2 | 442361.3 | 330591.4 | 1940.2% |
| `RLT` | 8 | 3561.3 | 1424.8 | 136804.2 | 80704.2 | 55.1% |
| `Bipartite` | 15 | 1850.7 | 630.3 | 62796.9 | 25174.6 | 21.5% |
| `GMSC` | 13 | 2391.6 | 726.5 | 8748.2 | 4481.3 | 14.4% |
| `GraphPool` | 15 | 757.4 | 333.8 | 15902.9 | 8837.0 | 17.0% |
| `Hybrid` | 15 | 608.9 | 241.7 | 14538.8 | 5613.0 | 15.5% |

TABLE 4
*Branch-and-cut results on standard quadratic programming problems of size* 50.

| | Solved | Time [s] | | Nodes | | Root gap |
|---|---|---|---|---|---|---|
| | | Avg. | S. Geom. | Avg. | S. Geom. | Avg. |
| **All 110 instances solved by at least one** | | | | | | |
| `CPLEX` | 31 | 17800.0 | 13063.7 | 334357.7 | 251486.3 | 1433.6% |
| `RLT` | 106 | 1697.4 | 167.7 | 22010.8 | 4621.9 | 61.6% |
| `Bipartite` | 110 | 602.2 | 111.2 | 5580.8 | 1668.4 | 21.5% |
| `GMSC` | 110 | 653.2 | 194.4 | 2641.9 | 675.2 | 10.9% |
| `GraphPool` | 110 | 248.6 | 85.4 | 2647.7 | 687.2 | 14.4% |
| `Hybrid` | 110 | 233.0 | 92.9 | 2568.6 | 585.4 | 11.6% |
| **16 hard instances (all more than 300 seconds)** | | | | | | |
| `CPLEX` | 6 | 14053.3 | 6814.8 | 163380.6 | 103019.4 | 1997.0% |
| `RLT` | 12 | 10190.3 | 4825.9 | 114540.0 | 75263.0 | 64.1% |
| `Bipartite` | 16 | 3530.8 | 2081.9 | 28236.0 | 20190.5 | 42.0% |
| `GMSC` | 16 | 2857.7 | 2012.7 | 13989.6 | 5300.7 | 35.6% |
| `GraphPool` | 16 | 1260.8 | 988.0 | 14369.9 | 5983.9 | 38.1% |
| `Hybrid` | 16 | 1152.1 | 858.0 | 14482.7 | 6138.1 | 37.4% |

inequalities. More precisely, although the number of problems solved to optimality is the same for `Bipartite`, `GraphPool` and `Hybrid`, separating GMSC bipartite inequalities yields a remarkable reduction in the number of nodes, which also yields a significant reduction in the computing times, as mentioned, especially on the hard instances.

In order to gather more insights on the branch-and-cut results, Figures 6 and 7 show Dolan–Moré performance profiles [15] on all solved instances and on solved hard instances, respectively. This time, instances of size $d = 30$ and $d = 50$ are considered together. In such plots every approach is compared to the virtual best of all approaches according to some performance measure, in our case the running time. To this end, for every $x \geq 1$, the fraction of instances where relative performance of the approach compared to the virtual best is at most $x$ is plotted. Consequently, higher values on the $y$-axis (for fixed $x$) and smaller values on the $x$-axis (for fixed $y$) are beneficial.

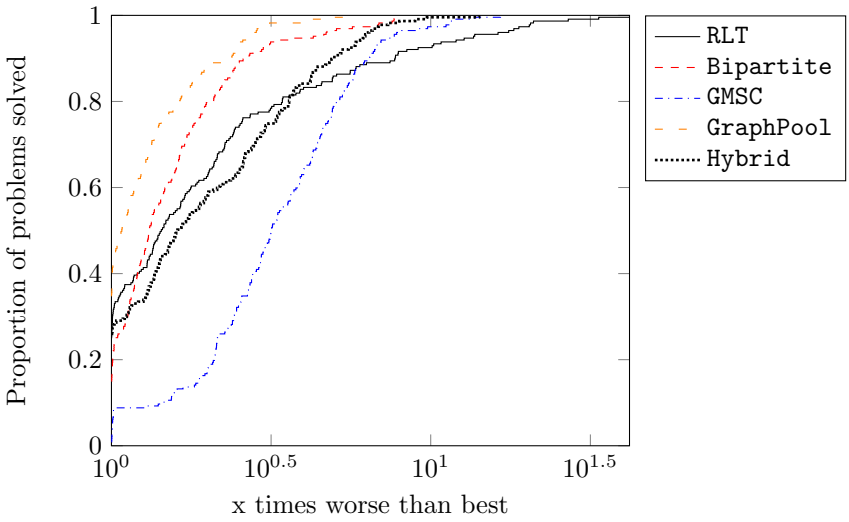


FIG. 6. *Dolan–Moré performance profile for all solved instances.*

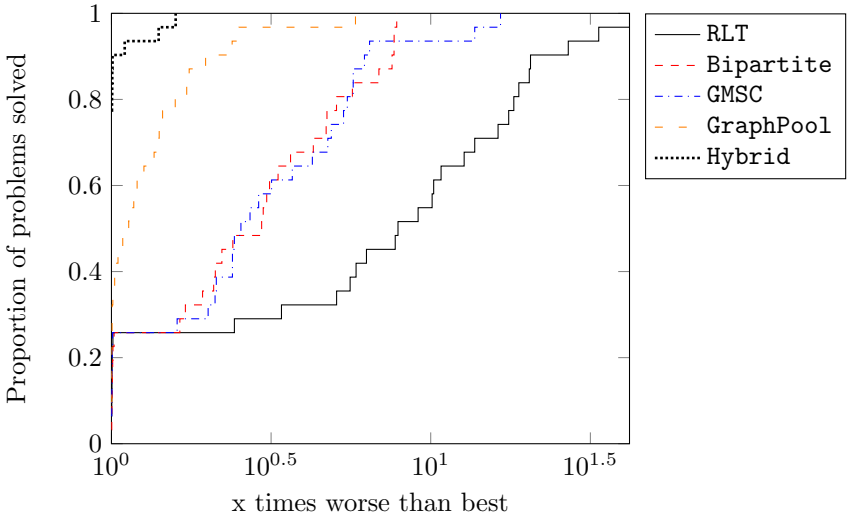


FIG. 7. *Dolan–Moré performance profile for "hard" solved instances.*

We omit the results obtained with default CPLEX from the plots because it is clearly dominated by all the other methods.

The performance profiles confirm the importance of MSC and GMSC bipartite inequalities. Even from these plots one can conclude that `Bipartite` outperforms `RLT`, while in turn `GraphPool` outperforms `Bipartite`. Finally, `GraphPool` appears to be the best approach if all (solved) instances are considered (Figure 6), while `Hybrid` becomes the best one if we instead restrict ourselves to hard instances only (Figure 7).

**6.5. Computational results on the instances from [28].** As mentioned previously, Scozzari and Tardella [28] proposed a combinatorial algorithm for (StQP)

TABLE 5
*Results on instances from* [28] *affected by RLT or (G)MSC inequalities.*

| | Problem_30x30_0.75 | | Problem_50x50_0.75 | | Root gap |
|---|---|---|---|---|---|
| | Time [s] | Nodes | Time [s] | Nodes | Avg. |
| `CPLEX` | 43200.0 | 1370290 | 43200.0 | 276080 | 4147.5% |
| `RLT` | 43.1 | 5130 | 7962.6 | 119634 | 50.6% |
| `Bipartite` | 12.2 | 1224 | 669.9 | 3239 | 11.2% |
| `GMSC` | 26.3 | 60 | 4344.6 | 413 | 13.7% |
| `GraphPool` | 14.9 | 494 | 513.6 | 440 | 9.2% |
| `Hybrid` | 31.5 | 63 | 254.0 | 440 | 8.8% |

and performed experiments on randomly generated instances, from which a subset of 14 instances has been published. The website mentioned in [28] is no longer active, so we have republished the instances at http://or.dei.unibo.it/library/msc. Since some of the instances have very large dimension (up to $d = 1000$), we impose a time limit of 12 hours.

Table 5 summarizes the results on this test set. Our callback is able to separate a cut at the root on 6 out of the 14 instances. Two instances (Problem_30x30_0.75 and Problem_50x50_0.75) are solved to optimality as soon as RLT inequalities are separated (i.e., in all configurations but `CPLEX`). Table 5 reports the time to optimality and number of nodes for these 2 instances. The last column shows the average root gap of those 6 instances on which our callback separates at least one cut.

The results are consistent with those of the previous sections. Although MSC and GMSC bipartite inequalities do not allow us to solve more instances than `RLT`, they greatly reduce the root gap, as well as the time to optimality and the number of nodes. As before, `GraphPool` and `Hybrid` show the best compromise between time needed for separation and impact on the root gap and overall solution time. Finally, note that the 14 instances are among the hardest that the combinatorial algorithm [28] can solve within one or two hours of time limit, so the reduced gaps are overall quite satisfactory.

**6.6. Comparison with QuadProgBB.** In this section we report a computational comparison between the `Hybrid` approach illustrated in subsections 6.3 and 6.4 and QuadProgBB [14]. The results for QuadProgBB[5] have been obtained on a cluster of servers with 16 Intel Xeon 5672 CPUs running at 3.20 GHz with 48 GB RAM and using Ubuntu 16.04.3 LTS as the operating system. Although the machines are different than those used for `Hybrid`, from a qualitative standpoint we can assume the computing times to be roughly comparable.

As discussed in subsection 2.3, QuadProgBB implements a spatial branch-and-bound based on a particularly strong relaxation that combines the strength of the SDP and RLT. In particular, as a direct consequence of Theorem 12, MSC and GMSC bipartite inequalities are implied by RLT-StQP and SDP, and therefore the root relaxation of QuadProgBB theoretically dominates the root relaxation of `Hybrid` obtained by separating RLT inequalities and MSC and GMSC bipartite inequalities on top of the $Q$-space relaxation (MC-StQP).

[5]QuadProgBB is implemented in MATLAB and freely available via https://github.com/sburer/QuadProgBB. In our experiments we used MATLAB R2014a and QuadProgBB commit `a02831e489d0d8b53eaafa06d5159578265a7ab9`.

TABLE 6

*Comparison between the* `Hybrid` *configuration and QuadProgBB with different values of maximum iterations.*

| | Size 30 | | Size 50 | |
|---|---|---|---|---|
| | Avg. root gap | B&B time | Avg. root gap | B&B time |
| **No outliers** | | | | |
| `Hybrid` | 4.47 | 15.65 | 4.01 | 77.71 |
| `QPBB Default` | 1.99 | 15.07 | 40.95 | 116.12 |
| `QPBB 10000it` | 0.00 | 11.91 | 1.12 | 59.31 |
| `QPBB 50000it` | 0.00 | 40.66 | 0.00 | 142.74 |
| **(−10, −5, 0), PosDiag** | | | | |
| `Hybrid` | 5.35 | 1569.06 | 6.48 | > 6h |
| `QPBB Default` | 0.83 | 18.82 | 82.17 | 483.32 |
| `QPBB 10000it` | 0.00 | 22.58 | 0.08 | 92.49 |
| `QPBB 50000it` | 0.00 | 51.46 | 0.00 | 156.93 |
| **(0, 5, 10), PosDiag** | | | | |
| `Hybrid` | 85.86 | 14.10 | 87.87 | 336.01 |
| `QPBB Default` | 661.20 | 120.80 | 2365.09 | 164.49 |
| `QPBB 10000it` | 9.53 | 28.38 | 84.38 | 98.65 |
| `QPBB 50000it` | 0.01 | 52.51 | 0.00 | 184.25 |

Table 6 reports aggregated results on the instances described in subsection 6.1. For each configuration tested and for each class of instances, the table reports the average percentage gap left at the root node and the geometric mean of the branch-and-bound computing time in seconds, calculated as in Tables 3 and 4. Based on the results, the instances are divided into three categories to highlight some outliers. The second part of the table gives the results on instances with distribution $(-10, -5, 0)$ and where all the diagonal elements of the $Q$ matrix are nonnegative; the third part of the table gives the results on instances with distribution $(0, 5, 10)$ and where all the diagonal elements of the $Q$ matrix are nonnegative, while the first part of the table (i.e., "no outliers") gives aggregated results on all other instances. Furthermore, since in some cases the default version of QuadProgBB is not able to solve the root relaxation completely, we considered three different settings of QuadProgBB obtained by changing the value of the `max_iter` option, which controls the maximum number of iterations allowed at each node. Precisely, `QPBB Default` gives the results obtained by running the default version of QuadProgBB, where `max_iter` is equal to $1,000$, while `QPBB 10000it` and `QPBB 50000it` report the results obtained by setting `max_iter` to $10,000$ and $50,000$, respectively.

The results in Table 6 confirm the theoretical dominance settled by Theorem 12. More precisely, the "no outliers" results show that, although the bound obtained by QuadProgBB is generally better than that of `Hybrid` (except in the case of `QPBB Default` and $d = 50$), our approach is still competitive in terms of the computing time to optimally solve the problem. For the instances in class $(-10, -5, 0)$, `Hybrid` has a slightly worse bound than the "no outliers" case, but definitely struggles in closing such a gap by exploring the branch-and-bound tree (`Hybrid` reaches the time limit on all instances with $d = 50$); thus, QuadProgBB significantly dominates in all versions. Conversely, the instances in class $(0, 5, 10)$ turn out to have a relevant root node gap for both `Hybrid` and QuadProgBB, the latter being able to recover a very good bound only with $50,000$ iterations. Nevertheless, such a gap can be effectively closed by all algorithms through branch-and-bound, with `Hybrid` being the fastest approach for $d = 30$.

**7. Computational results on quadratic knapsack instances.** The results in section 6 show that MSC and GMSC bipartite inequalities are extremely effective for (StQP). However, it is not obvious whether this result carries over to the extended version of such inequalities when applied to more general problems. To start investigating this question we considered the quadratic knapsack problem (QKP), which is a straightforward generalization of (StQP). In QKP one needs to maximize a quadratic objective function subject to a knapsack constraint $w^T x \leq c$, where $x$ is a vector of binary variables, $w$ are nonnegative weights, and $c$ is the nonnegative capacity. We considered the continuous relaxation of QKP because CPLEX reformulates the problem into a mixed-integer linear program if the variables $x$ are binary. Then, the knapsack constraint is the constraint used as a basis to formulate RLT inequalities and (generalized) MSC inequalities.

In our experiments we considered two sets of instances. The first set, referred to as QKP1, is generated by following an approach often used in the literature (see, e.g., [13, 17]). Therein, the instances are parameterized by their size $d$ and density $D$, i.e., the fraction of nonzero elements in the objective function. After sampling the nonzero elements, the objective coefficients $Q_{ij} = Q_{ji}$ are sampled uniformly from $[1, 100]$. The weights $w_i$ are sampled uniformly from $[1, 50]$, and the capacity $c$ from $[50, \sum_{i=1}^{d} w_i]$. As for (StQP), we used only fully dense objective matrices, i.e., $D = 1.0$, and we generated 150 instances with size $d = 30$. The second set of instances, referred to as QKP2 in the following, has the same structure (again 150 instances), but the objective matrices are sampled as described in subsection 6.1. Instances from both test sets are available at http://or.dei.unibo.it/library/msc.

Computations were done for the four configurations[6] `CPLEX`, `RLT`, `Bipartite`, and `Hybrid` that are defined as in subsection 6.3. In these experiments, another cutting plane technique already applied by CPLEX, namely Boolean quadric polytope (BQP) cuts [7, 19], has a substantial impact. We present results both with BQP cuts disabled, to get a fair comparison between closures, and with BQP cuts enabled, to evaluate the effect of combined cutting planes.

Tables 7 and 8 show aggregated results at the root node on instances QKP1 and QKP2, respectively. For each of the settings considered, the tables report the number of instances solved, the number of timeouts, the number of instances affected by each class of inequalities, and the average root gaps obtained. An instance is considered to be affected by a given class of inequalities if at least one cut from that class is separated.

As seen for (StQP), applying RLT inequalities is highly beneficial, in terms of both number of instances solved at the root and root gap reduction. The effect is more pronounced on the instances of type QKP1 (Table 7), where 144 out of 150 instances are affected by RLT, 19 additional instances are solved w.r.t. `CPLEX` when BQP cuts are disabled, and 104 when BQP cuts are enabled. With BQP cuts disabled, the average gap left is reduced from 5.65% to 0.29%; `Bipartite` can then close an additional 39.62% gap w.r.t. `RLT`, leaving only 0.11% on average and solving in total 39 instances at the root. Surprisingly, separating GMSC bipartite inequalities on top of `Bipartite` has almost no effect. Even though almost one-third of the instances are affected, the number of instances solved and the gap stay (almost) the same. When the various types of cuts are combined with BQP cuts, the overall comparison is similar, but the gap closed by all techniques is even more important. Remarkably,

[6] In this case, `GMSC` turns out to be too expensive computationally.

TABLE 7
*Root node results on quadratic knapsack instances QKP1.*

| | CPLEX | RLT | Bipartite | Hybrid |
|---|---|---|---|---|
| ***BQP disabled*** | | | | |
| **Average root gap [%]** | | | | |
| Gap left | 5.65 | 0.29 | 0.11 | 0.10 |
| Closed w.r.t. CPLEX root | — | 80.26 | 85.21 | 85.41 |
| Closed w.r.t. RLT | — | — | 39.62 | 40.84 |
| **Solved/timeout at the root** | | | | |
| Solved | 6 | 25 | 39 | 39 |
| Timeout | 0 | 0 | 0 | 12 |
| **Affected** | | | | |
| RLT/MSC/GMSC | 0/0/0 | 144/0/0 | 144/101/0 | 144/101/47 |
| ***BQP enabled*** | | | | |
| **Average root gap [%]** | | | | |
| Gap left | 5.65 | 0.02 | 0.00 | 0.00 |
| Closed w.r.t. CPLEX root | — | 94.80 | 95.74 | 95.74 |
| Closed w.r.t. RLT | — | — | 24.24 | 24.24 |
| **Solved/timeout at the root** | | | | |
| Solved | 6 | 110 | 144 | 143 |
| Timeout | 0 | 0 | 0 | 2 |
| **Affected** | | | | |
| RLT/MSC/GMSC | 0/0/0 | 144/0/0 | 144/103/0 | 144/101/5 |

Bipartite and Hybrid can solve almost all instances (with 6 and 7 unsolved instances at the root respectively).

Instances of type QKP2 (Table 8) have a much larger average root gap, but show similar phenomena. On both test sets, RLT and Bipartite contribute to the solution of several additional instances at the root. On the second test set, with BQP cuts disabled, the effect of Bipartite on the average root gap is very small, but the 16% gap is closed with respect to RLT. Even if Hybrid allows us to solve three more instances w.r.t. Bipartite, GMSC bipartite inequalities appear to be less effective than we observed for (StQP), where the impact on the root gap is much more marked. Again the combination with BQP cuts seems beneficial.

**8. Conclusion.** We studied new cutting planes for strengthening the linear relaxations that appear in the solution of nonconvex quadratic problems with linear constraints. The cutting planes exploit the relation between these problems and the maximum clique problem of a graph, in the spirit of the classical Motzkin–Straus theorem. In particular, we concentrated on bipartite graphs and we proposed two families of cutting planes, namely MSC bipartite inequalities and GMSC bipartite inequalities. By analyzing the relationship between the new cutting planes and the RLT inequalities, we showed that, interestingly, (i) MSC bipartite inequalities are not comparable with first level RLT inequalities, (ii) the derivation of GMSC bipartite inequalities generalizes both MSC bipartite inequalities and first level RLT, and (iii) GMSC bipartite inequalities are implied by the combination of the SDP inequality and RLT.

Our computational experiments showed that both MSC bipartite inequalities and GMSC bipartite inequalities allow us to get a significantly stronger bound than first-level RLT. In addition, we showed that a first implementation of the embedding of

TABLE 8
*Root node results on quadratic knapsack instances QKP2.*

| | CPLEX | RLT | Bipartite | Hybrid |
|---|---|---|---|---|
| ***BQP disabled*** | | | | |
| **Average root gap [%]** | | | | |
| Gap left | 124.86 | 118.44 | 116.40 | 115.74 |
| Closed w.r.t. CPLEX root | — | 22.69 | 28.14 | 29.12 |
| Closed w.r.t. RLT | — | — | 16.08 | 17.78 |
| **Solved/timeout at the root** | | | | |
| Solved | 24 | 32 | 43 | 46 |
| Timeout | 0 | 0 | 0 | 0 |
| **Affected** | | | | |
| RLT/MSC/GMSC | 0/0/0 | 94/0/0 | 94/72/0 | 94/72/73 |
| ***BQP enabled*** | | | | |
| **Average root gap [%]** | | | | |
| Gap left | 48.77 | 45.29 | 44.63 | 44.54 |
| Closed w.r.t. CPLEX root | — | 27.66 | 37.05 | 39.76 |
| Closed w.r.t. RLT | — | — | 22.39 | 26.48 |
| **Solved/timeout at the root** | | | | |
| Solved | 29 | 40 | 55 | 56 |
| Timeout | 0 | 0 | 0 | 21 |
| **Affected** | | | | |
| RLT/MSC/GMSC | 0/0/0 | 94/0/0 | 94/80/0 | 94/72/69 |

our cutting planes within CPLEX spatial branch-and-bound results in an exact solver that is reasonably competitive with the QuadProgBB solver using both SDP and RLT. While for pure (StQP) instances of the considered size an SDP-based branch and bound would probably be the most effective choice, on larger and heterogeneous instances, an LP-based algorithm has some competitive advantages in terms of computational tractability and ease of integration into state-of-the-art MILP technology. Thus, improving LP techniques, for example by cuts, is computationally relevant.

Some possible extensions of our approach would be to derive cuts corresponding to graphs with larger clique number (greater than two) and to exploit generalized versions of the Motzkin–Straus theorem [18].

**Acknowledgments.** The authors are indebted to Andrea Scozzari and Fabio Tardella for sharing their instances and their thoughts on standard quadratic programming, and to Samuel Burer for sharing his knowledge on copositive programming and QuadProgBB. The constructive comments of two anonymous referees are also warmly acknowledged.

## REFERENCES

[1] T. ACHTERBERG, D. JUNGLAS, AND R. WUNDERLING, *Deterministic Parallelization through Atomic Task Computation*, US Patent, December 3, 2013; available at https://www.google.co.uk/patents/US8601486.

[2] K. ANSTREICHER, *On convex relaxations for quadratically constrained quadratic programming*, Math. Program., 136 (2012), pp. 233–251.

[3] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, *Mixed-integer nonlinear optimization*, Acta Numer., 22 (2013), pp. 1–131.
[4] I. Bomze, M. Dür, E. de Klerk, C. Roos, A. Quist, and T. Terlaky, *On copositive programming and standard quadratic optimization problems*, J. Global Optim., 18 (2000), pp. 301–320.
[5] I. M. Bomze, *On standard quadratic optimization problems*, J. Global Optim., 13 (1998), pp. 369–387.
[6] I. M. Bomze, M. Locatelli, and F. Tardella, *New and old bounds for standard quadratic optimization: Dominance, equivalence and incomparability*, Math. Program., 115 (2008), pp. 31–64.
[7] P. Bonami, O. Günlük, and J. Linderoth, *Globally solving nonconvex quadratic programming problems with box constraints via integer programming methods*, Math. Program. Comput., 10 (2018), pp. 333–382.
[8] C. Brás, G. Eichfelder, and J. Júdice, *Copositivity tests based on the linear complementarity problem*, Comput. Optim. Appl., 63 (2016), pp. 461–493.
[9] S. Burer, *On the copositive representation of binary and continuous nonconvex quadratic programs*, Math. Program., 120 (2009), pp. 479–495.
[10] S. Burer, *Optimizing a polyhedral-semidefinite relaxation of completely positive programs*, Math. Program. Comput., 2 (2010), pp. 1–19.
[11] S. Burer and D. Vandenbussche, *Globally solving box-constrained nonconvex quadratic programs with semdefinite-based finite branch-and-bound*, Comput. Optim. Appl., 43 (2009), pp. 181–195.
[12] A. Caprara, M. Carvalho, A. Lodi, and G. J. Woeginger, *Bilevel knapsack with interdiction constraints*, INFORMS J. Comput., 28 (2016), pp. 319–333.
[13] A. Caprara, D. Pisinger, and P. Toth, *Exact solution of the quadratic knapsack problem*, INFORMS J. Comput., 11 (1999), pp. 125–137.
[14] J. Chen and S. Burer, *Globally solving nonconvex quadratic programming problems via completely positive programming*, Math. Program. Comput., 4 (2012), pp. 33–52.
[15] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.
[16] M. Dür, *Copositive programming: A survey,* in Recent Advances in Optimization and Its Applications in Engineering: The 14th Belgian-French-German Conference on Optimization, Springer, Berlin, 2010, pp. 3–20.
[17] G. Gallo, P. Hammer, and B. Simeone, *Quadratic knapsack problems*, in Combinatorial Optimization, M. Padberg, ed., Math. Program. Stud. 12, Springer, Berlin, 1980, pp. 132–149.
[18] L. Gibbons, D. Hearn, P. Pardalos, and M. Ramana, *Continuous characterizations of the maximum clique problem*, Math. Oper. Res., 22 (1997), pp. 754–768.
[19] IBM, *Boolean quadric polytope (BQP) cuts*, in CPLEX 12.6.3 User's Manual, http://www-01.ibm.com/support/knowledgecenter/SSSA5P_12.6.3/ilog.odms.cplex.help/CPLEX/UsrMan/topics/discr_optim/mip/cuts/28_BQPcuts.html (March 18, 2016).
[20] IBM, *Deterministic time limit*, in CPLEX 12.6.3 User's Manual, http://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.3/ilog.odms.cplex.help/CPLEX/Parameters/topics/DetTiLim.html (March 18, 2016).
[21] IBM CPLEX Optimizer, http://www.cplex.com.
[22] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, eds., The IBM Research Symposia Series, Springer, New York, 1972, pp. 85–103.
[23] A. Lodi, T. K. Ralphs, and G. J. Woeginger, *Bilevel programming and the separation problem*, Math. Program., 146 (2014), pp. 437–458.
[24] G. McCormick, *Computability of global solutions to factorable nonconvex programs: Part* I – *convex underestimating problems*, Math. Program., 10 (1976), pp. 147–175.
[25] R. Misener and C. A. Floudas, *GloMIQO: Global mixed-integer quadratic optimizer*, J. Global Optim., 57 (2013), pp. 3–50.
[26] T. S. Motzkin and E. G. Straus, *Maxima for graphs and a new proof of a theorem of Turán*, Canad. J. Math., 17 (1965), pp. 533–540.
[27] J. Schweiger, *Nonlinearity and uncertainty in energy networks*, Ph.D. thesis, Technische Universität Berlin, 2017.
[28] A. Scozzari and F. Tardella, *A clique algorithm for standard quadratic programming*, Discrete Appl. Math., 156 (2008), pp. 2439–2448.
[29] H. D. Sherali and W. P. Adams, *A hierarchy of relaxations between the continuous and*

*convex hull representations for zero-one programming problems*, SIAM J. Discrete Math., 3 (1990), pp. 411–430.

[30] H. D. Sherali and A. Alameddine, *A new reformulation-linearization technique for bilinear programming problems*, J. Global Optim., 2 (1992), pp. 379–410.

[31] M. Tawarmalani and N. V. Sahinidis, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*, Kluwer Academic, Boston, MA, 2002.