A new branch-and-bound algorithm for standard quadratic programming problems

G. Liuzzi*, M. Locatelli†, V. Piccialli‡

Abstract

In this paper we propose convex and LP bounds for Standard Quadratic Programming (StQP) problems and employ them within a branch-and-bound approach. We first compare different bounding strategies for StQPs in terms both of the quality of the bound and of the computation times. It turns out that the polyhedral bounding strategy is the best one to be used within a branch-and-bound scheme. Indeed, it guarantees a good quality of the bound at the expense of a very limited computation time. The proposed branch-and-bound algorithm performs an implicit enumeration of all the KKT (stationary) points of the problem. We compare different branching strategies exploiting the structure of the problem. Numerical results on randomly generated problems (with varying density of the underlying convexity graph) are reported which show the effectiveness of the proposed approach, in particular in limiting the growth of the number of nodes in the branch-and-bound tree as the density of the underlying graph increases.

Keywords: Standard Quadratic Programming; branch-and-bound; polyhedral bound.

1 Introduction

The problem

$$\begin{aligned} \min \quad & \tfrac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ & \mathbf{x} \in \Delta_n, \end{aligned} \tag{1}$$

where a quadratic function is minimized over the n-dimensional unit simplex

$$\Delta_n = \{\mathbf{x} \in \mathbb{R}^n_+ \ : \ \sum_{i=1}^n x_i = 1\},$$

is called a Standard Quadratic Programming one (StQP in what follows). The attention to these problems was brought by Bomze (see [4]). He remarks that quadratic resource allocation problems (see, e.g., [16]) and mean/variance portfolio selection problems (see also [3, 20, 21]) can be reformulated as StQPs . Moreover, a relevant combinatorial problem such as the maximum clique one (see [22]) and its weighted version (see [12]) can both be reformulated as StQPs. StQPs are polynomially solvable when the Hessian matrix $\mathbf{Q}$ is semidefinite positive (convex case) or semidefinite negative (the minimum is attained at one of the n vertices of the unit simplex). But the previously mentioned reformulation of maximum clique problems as StQPs proves that in the indefinite case StQPs are NP-hard. Moreover, as a consequence of an inapproximability result

*Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" (IASI), Consiglio Nazionale delle Ricerche (CNR), Via dei Taurini 19, 00185 Rome, Italy. e-mail: giampaolo.liuzzi@iasi.cnr.it

†Dipartimento di Ingegneria dell'Informazione, Università di Parma, Via G.P. Usberti, 181/A, 43124 Parma, Italy. e-mail: marco.locatelli@unipr.it

‡Dipartimento Di Ingegneria Civile e Ingegneria Informatica, Università di Roma Tor Vergata, Via del Politecnico, 1, 00133 Roma, Italy. e-mail: veronica.piccialli@uniroma2.it

proven in [15] for the maximum clique problem, it also turns out that StQP problems do not admit a FPTAS (Fully Polynomial Time Approximation Scheme) unless $\mathcal{NP} = \mathcal{ZPP}$, where $\mathcal{ZPP}$ denotes the set of problems which can be solved by a randomized algorithm whose average running time is polynomial. In [6] it has been proven that StQPs admit a PTAS, but its computational effort, though polynomial, turns out to be impractical even for relatively low precision requirements.

Many bounds have been suggested in the literature about StQPs. A description and a comparison of many such bounds can be found in [8]. These range from low-quality but very cheap to high-quality but very expensive ones. Cheap bounds are based on the analysis of the entries of matrix $\mathbf{Q}$ and vector $\mathbf{c}$ and can be computed in $O(n^2)$ time. More expensive bounds are those computed by the solution of Semidefinite Programming (SDP) problems. These include Nowak's bound [23] and the best bound based on DC (Difference-of-Convex) decompositions [1]. A source of many bounds for StQP problems was the observation [5] that these problems can be reformulated as problems over the cone of copositive matrices or over the dual cone of completely positive matrices. In particular, many hierarchies of SDP bounds which converge to the optimal solution of (1) have been proposed (see [11, 14, 17, 24]). The first bound of these hierarchies is the well known copositive bound, which in the context of maximum clique problems is also known as Lovàsz-Schrijver bound (see [19, 26]). The following bounds in the hierarchies improve the copositive one but, unfortunately, though polynomial, the computational requirements of these bounds make them impractical even at low dimension. Upper bounds for StQPs based on the solution of inner polyhedral approximations of the cone of completely positive matrices have been studied in [25, 29].

While the literature abounds of lower bounds for StQP problems, only few branch-and-bound approaches have been proposed for their exact solution. To the authors' knowledge these include a branch-and-bound algorithm proposed in [7] where lower bounds are computed through DC decompositions of the objective function (but computational experiments are not reported there), and a branch-and-bound approach proposed in [27] where cheap $O(n^2)$ lower bounds are employed and an implicit enumeration is carried on over the set of cliques of the so called *convexity graph* of (1) which will be defined in Section 2.

In this paper we consider a lower bound for problem (1) proposed in [18]. The computation of the bound requires the solution of a convex optimization problem and is based on the decomposition of the quadratic function into a sum of quadratic functions for which it is possible to compute the convex envelope over the unit simplex. After some preliminary observations about StQPs in Section 2, the bound, called in what follows `Conv` bound, will be described in Section 3. In the same section a lower bound based on a polyhedral underestimation of the objective function of the convex optimization problem will be proposed. The bound can be computed by solving a linear programming problem and will be called in what follows `LP` bound. The experiments will show that the quality of this bound is only slightly inferior with respect to the `Conv` bound, while the computing times are largely inferior. In Section 4 we will describe a branch-and-bound approach for the solution of problem (1). In Section 5 we will make two different sets of experiments. In the first set of experiments, described in Section 5.1, we compare, in terms of quality and computing times, the `Conv` and `LP` bounds with two bounds based on the solution of semidefinite programming problems, namely the bound based on difference-of-convex decompositions, called `DC` bound in what follows, and the copositive bound, called `Cop` bound in what follows. In Section 5.2 we compare the results of our branch-and-bound approach with those reported in [27]. In particular, we will show that our approach is very effective in limiting the growth of the branch-and-bound tree and, thus, performs well when the density of the underlying convexity graph (defined in Section 2 and strictly related to the density of the Hessian of the quadratic function) is large. Conclusions and possible future developments are outlined in Section 6.

2 Preliminary observations about StQPs

In this section we summarize some results about StQP problems which will be employed later on. The following observation, whose simple proof can be found, e.g., in [18], can be made about StQP problems.

Observation 2.1 *For any StQP problem (1):*

(i) matrix $\mathbf{Q}$ *and vector* $\mathbf{c}$ *can always be re-defined in such a way that* $Q_{ii} = 0$ *for all* $i = 1, \ldots, n$*, i.e., all diagonal elements of the Hessian matrix* $\mathbf{Q}$ *can be taken equal to 0;*

(ii) if the diagonal condition above holds, then all $Q_{ij} > 0$ *can be set equal to 0, since at any optimal solution of the StQP problem,* $Q_{ij} > 0$ *implies that at least one among* x_i *and* x_j *is certainly equal to 0.*

Therefore, from now on, we will always assume that the diagonal entries of $\mathbf{Q}$ are null and that the off-diagonal entries are either null or negative. A graph $G = (V, E)$ can be associated to problem (1), where

$$V = \{1, \ldots, n\}, \quad E = \{(i, j) \ : \ i, j \in \{1, \ldots, n\}, \ Q_{ij} < 0\}.$$

This graph, introduced in [27], is called *convexity graph* of problem (1). In [27] the following observation is proved.

Observation 2.2 *The optimal solution of (1) must be attained in the relative interior of a face*

$$F_C = \{\mathbf{x} \in \Delta_n \ : \ x_i = 0, \ i \notin C\}, \tag{2}$$

where $C \subseteq V$ *is a clique over the convexity graph* G*. This implies that, given any independent set* $I \subseteq V$ *in the convexity graph* G*, at most one* x_i*,* $i \in I$*, is strictly positive at the optimal solution of (1).*

The KKT conditions for problem (1) are

$$\begin{aligned} &\mathbf{q}_i\mathbf{x} + c_i - \lambda = \mu_i \quad && i = 1, \ldots, n \\ &\mu_i \geq 0 && i = 1, \ldots, n, \end{aligned}$$

where $\mathbf{q}_i$ is the i-th row of matrix $\mathbf{Q}$, λ is the Lagrange multiplier of the constraint $\sum_{i=1}^{n} x_i = 1$, and μ_i, $i = 1, \ldots, n$, is the Lagrange multiplier of the constraint $x_i \geq 0$. Of course, the search for an optimal solution of (1) can be restricted to KKT points. We will make use of the following observation about the KKT points of (1).

Observation 2.3 *Let* $(\mathbf{x}, \lambda, \boldsymbol{\mu})$ *be a KKT point of (1). If* $x_i > 0$*, then for each* $j \neq i$

$$\mathbf{q}_j\mathbf{x} + c_j \geq \mathbf{q}_i\mathbf{x} + c_i. \tag{3}$$

Moreover

$$\frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x} \geq \frac{1}{2}\left[\mathbf{q}_i\mathbf{x} + c_i + \mathbf{c}^T\mathbf{x}\right]. \tag{4}$$

Proof. Since $x_i > 0$, then $\mu_i = 0$ so that

$$\mathbf{q}_i\mathbf{x} + c_i = \lambda.$$

Then, (3) follows from $\mu_j \geq 0$. Moreover, we have

$$x_j\mathbf{q}_j\mathbf{x} \geq (c_i + \mathbf{q}_i\mathbf{x})x_j - c_jx_j \quad \forall j,$$

so that

$$\frac{1}{2}\sum_{j=1}^{n} x_j\mathbf{q}_j\mathbf{x} \geq \frac{1}{2}(c_i + \mathbf{q}_i\mathbf{x})\sum_{j=1}^{n} x_j - \frac{1}{2}\sum_{j=1}^{n} c_jx_j,$$

from which (4) follows. □

3 Bounds for StQP based on convex and linear programming problems

In order to introduce the `Conv` and `LP` bound, we first need to recall a result proved in [18].

3.1 Convex envelope and convex polyhedral underestimator for a special quadratic function

In this section we briefly recall the convex underestimator proposed in [18] for the quadratic part of the objective function in (1). We first need to consider a quadratic function

$$f^i(\mathbf{x}) = \sum_{j \neq i} A_{ij} x_i x_j,$$

whose convexity graph is a star with center the node corresponding to variable x_i, and for which we assume that

$$0 \geq A_{i1} \geq A_{i2} \geq \cdots \ldots \geq A_{in}. \tag{5}$$

We denote by

$$A'_{t_i} = -A_{in} > \cdots > A'_1 = -A_{i1},$$

the t_i *distinct* values $-A_{ij}$, $j \in \{1, \ldots, n\}$, $j \neq i$. We also set $A'_{t_i+1} = +\infty$ and $A'_0 = 0$. For each $r = 1, \ldots, t_i + 1$, we introduce the set of indices

$$L^i_r = \{j \ : \ j \in \{1, \ldots, n\},\ j \neq i \ \text{ and } \ -A_{ij} < A'_r\}, \ \ U^i_r = \{1, \ldots, n\} \setminus (L^i_r \cup \{i\}).$$

We define, for each $r \in \{1, \ldots, t_i\}$, the three polyhedral subregions of the unit simplex

$$\begin{aligned}
\Gamma^i_r &= \left\{\mathbf{x} \in \Delta_n \ : \ \sqrt{A'_{r-1}}(1 - \textstyle\sum_{j \in L^i_r} x_j) \leq \sum_{j \in U^i_r} \sqrt{-A_{ij}} x_j \leq \sqrt{A'_r}(1 - \sum_{j \in L^i_r} x_j)\right\} \\
\Gamma^{i\,-}_r &= \left\{\mathbf{x} \in \Delta_n \ : \ \sqrt{A'_{r-1}}(1 - \textstyle\sum_{j \in L^i_r} x_j) \geq \sum_{j \in U^i_r} \sqrt{-A_{ij}} x_j\right\} \\
\Gamma^{i\,+}_r &= \left\{\mathbf{x} \in \Delta_n \ : \ \textstyle\sum_{j \in U^i_r} \sqrt{-A_{ij}} x_j \geq \sqrt{A'_r}(1 - \sum_{j \in L^i_r} x_j)\right\}.
\end{aligned}$$

For some $\mathbf{x} \in \Delta_n$, we introduce:

- for each $r \in \{1, \ldots, t_i + 1\}$, the affine functions

$$\gamma^i_r(\mathbf{x}) = -A'_{r-1}\left(1 - \sum_{j \in L^i_r} x_j\right) + 2\sqrt{A'_{r-1}}\left(\sum_{j \in U^i_r} \sqrt{-A_{ij}} x_j\right) + \sum_{j \in U^i_r} A_{ij} x_j;$$

- for each $r \in \{1, \ldots, t_i\}$, the functions

$$h^i_r(\mathbf{x}) = \begin{cases} \gamma^i_r(\mathbf{x}) & \text{if } \mathbf{x} \in \Gamma^{i\,-}_r \\ \frac{\left(\sum_{j \in U^i_r} \sqrt{-A_{ij}} x_j\right)^2}{1 - \sum_{j \in L^i_r} x_j} + \sum_{j \in U^i_r} A_{ij} x_j & \text{if } \mathbf{x} \in \Gamma^i_r \\ \gamma^i_{r+1}(\mathbf{x}) & \text{if } \mathbf{x} \in \Gamma^{i\,+}_r. \end{cases}$$

In [18] the following result has been proved.

Theorem 3.1 *The convex envelope of f^i over the unit simplex is*

$$conv_{f^i, \Delta_n}(\mathbf{x}) = \max_{r=1,\ldots,t_i} h^i_r(\mathbf{x}),$$

or, equivalently,

$$conv_{f^i, \Delta_n}(\mathbf{x}) = h^i_r(\mathbf{x}) \quad \textit{for } \ \mathbf{x} \in \Gamma^i_r, \quad r = 1, \ldots, t_i.$$

We also observe that $\gamma_r^i(\mathbf{x})$ is the supporting hyperplane of $conv_{f^i,\Delta_n}$ over the hyperplane

$$\sqrt{A'_{r-1}}(1 - \sum_{j \in L_r^i} x_j) = \sum_{j \in U_r^i} \sqrt{-A_{ij}} x_j,$$

while $\gamma_{r+1}^i(\mathbf{x})$ is the supporting hyperplane of $conv_{f^i,\Delta_n}$ over the hyperplane

$$\sum_{j \in U_r^i} \sqrt{-A_{ij}} x_j = \sqrt{A'_r}(1 - \sum_{j \in L_r^i} x_j).$$

Thus, the following corollary holds.

Corollary 3.1 *Function*

$$g^i(\mathbf{x}) = \max_{r=1,\ldots,t_i+1} \gamma_r^i(\mathbf{x}),$$

is a polyhedral convex underestimator of f^i.

Now we are ready to present the `Conv` and `LP` bounds.

3.2 `Conv` and `LP` bounds for StQP

The quadratic part of the objective function for problem (1) can be decomposed as follows

$$\sum_{i=1}^{n} f^i(\mathbf{x}),$$

where, for each $i = 1, \ldots, n$,

$$f^i(\mathbf{x}) = \frac{1}{2} \sum_{j=1,\ j \neq i}^{n} Q_{ij} x_i x_j.$$

By setting $A_{ij} = \frac{1}{2} Q_{ij}$ and, possibly, by re-ordering the variables in such a way that (5) is fulfilled, we are able to employ the results in Section 3.1 to derive the convex envelope of f^i over the unit simplex. Thus, a convex underestimator (not necessarily the convex envelope) for the quadratic part of the objective function for problem (1) is

$$\sum_{i=1}^{n} conv_{f^i,\Delta_n}(\mathbf{x}).$$

Thus, the `Conv` bound can be defined as follows

$$\begin{array}{rcll} \ell_{\texttt{Conv}}(\mathbf{Q}, \mathbf{c}) & = & \min & \displaystyle\sum_{i=1}^{n} conv_{f^i,\Delta_n}(\mathbf{x}) + \mathbf{c}^T \mathbf{x} \\ & & & \mathbf{x} \in \Delta_n. \end{array} \tag{6}$$

By exploiting Corollary 3.1, we are able to derive the `LP` bound

$$\begin{array}{rcll} \ell_{\texttt{LP}}(\mathbf{Q}, \mathbf{c}) & = & \min & \displaystyle\sum_{i=1}^{n} y_i + \mathbf{c}^T \mathbf{x} \\ & & & \mathbf{x} \in \Delta_n \\ & & & y_i \geq \gamma_r^i(\mathbf{x}) \qquad i = 1, \ldots, n,\ r = 1, \ldots, t_i + 1. \end{array} \tag{7}$$

Of course, the `LP` bound is weaker with respect to the `Conv` bound but, as we will see through some experiments, the difference is very mild. A possible way to make the `LP` bound closer to the `Conv` bound is the following. Let $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \Delta_n$ and for each $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, k\}$,

let $\alpha_i^j(\mathbf{x})$ be the supporting hyperplane of $conv_{f^i,\Delta_n}$ at $\mathbf{x}_j$. Then, a stronger LP bound is the following

$$\begin{array}{rcll} \ell_{\texttt{LP}-\texttt{CUT}_k}(\mathbf{Q},\mathbf{c}) & = & \min & \sum_{i=1}^n y_i + \mathbf{c}^T\mathbf{x} \\ & & & \mathbf{x} \in \Delta_n \\ & & & y_i \geq \gamma_r^i(\mathbf{x}) \qquad i = 1,\ldots,n,\ r = 1,\ldots,t_i+1 \\ & & & y_i \geq \alpha_i^j(\mathbf{x}) \qquad i = 1,\ldots,n,\ \ j = 1,\ldots,k. \end{array} \tag{8}$$

4 Branch-and-bound approach

The idea that led us to the definition of our algorithm is to carry out an *implicit* enumeration of all the (finitely many) KKT solutions of Problem (1). Such an implicit enumeration is performed by means of a branch-and-bound algorithm exploiting Observations 2.2 and 2.3. Note that branching rules based on KKT conditions have already been employed in the literature about QP problems (see, e.g., [2, 10]).

We denote by P_0 problem (1), i.e., the root node of the branch-and-bound tree, and L_0 its lower bound. We also denote by z the global upper bound and initialize it with the best value obtained by means of a certain number of local minimizations (we perform 10 local minimizations in the experiments). At each node of the B&B tree, we associate a vector $\mathbf{s} \in \{-1, 0, 1\}^n$ of variable statuses. Specifically,

$$s_i = \begin{cases} -1 & \text{implies } x_i \geq 0 \\ 0 & \text{implies } x_i = 0 \\ 1 & \text{implies } x_i > 0. \end{cases}$$

Then, at the root note, $\mathbf{s} = -\mathbf{1}$.

Algorithm 1 `Branch and Bound Algorithm for StQP`

```
1: procedure StQP-B&B(Q,c)
2:     Π = {(P_0, L_0)}, z = +∞
3:     while Π ≠ ∅ do
4:         (P, L) ← select(Π)
5:         Π ← Π \ {(P, L)}
6:         (C, z) ← branch&bound(P)
7:         update(Π, C)
       return the global upper bound z
```

Π is the set of open problems, $\mathcal{C}$ is the set of children (with their respective lower bounds) of a given open problem P. As concerns the three procedures that are used within Algorithm 1:

- the `select` procedure picks an open problem (among those in Π) according to some criterion, which we shall define when discussing the selection strategy;
- the `branch&bound` procedure carries out two tasks. On the one hand, given an open problem P, it generates, by means of the branching strategy described below, a set of children. On the other hand, it computes the lower and upper bounds of the children. This can be done according to two different strategies: (1) lower and upper bounds for problem P are computed and they are inherited by the children; (2) lower and upper bounds are computed for each child anew. Finally the procedure possibly updates the global upper bound z;
- the `update` procedure, on the basis of the list of problems $\mathcal{C}$ produced by the `branch&bound` procedure, updates the set of currently open problems Π. Namely, it adds to Π the new problems in $\mathcal{C}$ and it (possibly) fathoms from Π those problems with a lower bound greater than or equal to the global upper bound.

Bounding strategy We compute a lower bound at a given node by solving problem (7) where we add the constraints deriving from the corresponding node status $\mathbf{s}$. More specifically,

i) for every i such that $s_i = 0$, the x_i variable is fixed to 0;

ii) for every i such that $s_i = 1$, the constraints (3) are added; furthermore, exploiting the bound on the objective function (4) the following constraint is added

$$\sum_{j=1}^{n} y_j \geq \frac{1}{2} \left[\mathbf{q}_i \mathbf{x} + c_i + \mathbf{c}^T \mathbf{x} \right];$$

finally, for every $j \neq i$ such that $Q_{ij} = 0$ and $s_j = -1$, variable x_j is fixed to 0; should s_j be equal to 1, the node can be fathomed since no optimal solution of Problem (1) can have both x_i and x_j greater than zero when $Q_{ij} = 0$ (see Observation 2.1).

Remark 4.1 *Note that whenever the matrix Q is sparse, fixing $s_i = 1$ implies fixing many other variables to zero, due to the high number of zero elements on row i, so that the size of the LP problem reduces substantially.*

Branching strategy Given a generic node of the B&B tree, the branching procedure exploits Observation 2.2. More precisely, since the optimal solution of the problem $\mathbf{x}^*$ identifies a clique over the convexity graph G, given any independent set I in G, at most one variable x_i^*, $i \in I$, can be strictly positive. Hence, we exploit this observation by defining a (possibly) non-binary branching procedure. In particular, given an independent set I in G, we generate $|I| + 1$ children with statuses such that:

- for every $k \in I$, a child is generated with $s_k = 1$ and $s_i = 0$ for all $i \in I$, $i \neq k$;
- a further child is generated with $s_k = 0$, for all $k \in I$.

Note that, for every $k \notin I$, children's status s_k is inherited by the father.
We now describe how we actually compute the independent set I used for branching. To this aim, let $\bar{V} \subseteq V$ be the subset of nodes of graph G associated with variables whose statuses are equal to -1 at the current node, and let us denote by $G[\bar{V}]$ the subgraph of G induced by the set $\bar{V}$. Moreover, let $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ be the optimal solution of the relaxation at the current node.
For every node $i \in \bar{V}$, let

$$\eta_i = \bar{y}_i - \bar{\mathbf{x}}^\top \mathbf{q}_i \bar{x}_i \leq 0, \text{ for each } i \in \bar{V},$$

and π be the permutation corresponding to the ascending ordering of η_i's. We recall that the value η_i represents an estimate of the quality of the polyhedral bound on the star graph induced by node $i \in \bar{V}$. Hence, the lower this value, the more promising the node is to improve the bound. We compute an independent set I on $G[\bar{V}]$ by using Algorithm 2 (which is the greedy algorithm proposed in [30]) using as ordering of the nodes in $\bar{V}$ the above defined π.

Algorithm 2 `Greedy Algorithm for MIS`

```
1: procedure GreedyMIS(G = (V, E), π)
2:     if |V| = 0 then
3:         return ∅
4:     else
5:         Let v be the first node in V according to π
6:         V' ← V \ ({v} ∪ {u ∈ V : (u, v) ∈ E})
7:         return {v} ∪ GreedyMIS(G[V'], π)
```

We further process the independent set I computed by Algorithm 2 and define the following subset of nodes $T \subset I$,

$$T = \left\{ i \in I : |\eta_i| \geq 1.5\, \bar{\eta}_I \right\}, \quad \text{with} \quad \bar{\eta}_I = \frac{1}{|I|} \sum_{i \in I} |\eta_i|.$$

This amounts to choosing only the more promising nodes, i.e., those having a gap η_i sufficiently away from zero. By using set T, we generate $|T| + 1$ children as described above.
Note that this branching strategy implies that, if the matrix is sparse, thanks to Remark 4.1, the computation of the first $|T|$ bounds should be fast, since many variables are fixed to zero.

Selection strategy We implemented two well-known strategies. A *best bound* and a *depth first* strategy. Note that, when using the best bound strategy, the `branch&bound` procedure is required to compute the lower and upper bounds for all of the children. On the contrary, when the depth first strategy is implemented, the `branch&bound` procedure computes lower and upper bounds for the father node while children nodes inherit those values.

5 Computational experiments

Before discussing the computational results, we make the following premise. Although the maximum (weight) clique problem can be reformulated as an StQP, we are not going to use this problem for our tests. Indeed, its solution through the StQP continuous reformulation can hardly beat combinatorial approaches. In fact, the literature offers very efficient algorithms even for the *explicit* enumeration of all maximal cliques in a graph (see, e.g., [28]). In principle, these algorithms could also be employed for the solution of the general StQP problem (1). Indeed, as observed in Section 2, the optimal solution of (1) is attained in the relative interior of a face F_I of the unit simplex, defined in (2), and such that I is a maximal clique of the convexity graph for the problem. However, in this case some additional, nontrivial computational effort needs to be done for each maximal clique. First, the stationary point of the restriction of the objective function of (1) over F_I needs to be computed. In view of the KKT conditions, that requires the solution of the following linear system

$$\begin{cases} \mathbf{Q}_I \mathbf{x}_I + \mathbf{c}_I - \lambda = 0 \\ \mathbf{e}_I \mathbf{x}_I = 1, \end{cases} \tag{9}$$

where the subscript I denotes the restriction to the entries belonging to rows and columns inside the set I. Next, the objective function at the solution of the system above needs to be evaluated. The performance of such an enumeration technique tends to degrade as the density of the convexity graph increases, both because of the increase of the number of maximal cliques, and because of the increase of the size of the maximal cliques and, consequently, of the linear systems (9). In [27] a cheap $O(n^2)$ bound is introduced within the framework of a B&B approach in order to make some *implicit* enumeration. However, the quality of the bound does not allow for a strong reduction of the number of nodes to be explored and, again, the performance tends to degrade as the density of the convexity graph increases. In the approach proposed in this paper the `LP` bounds we employed are more costly but they allow for a quite effective implicit enumeration. For instance, in a random instance with $n = 100$ and density of the convexity graph equal to 0.9, the number of maximal cliques exceeds $3 \cdot 10^8$, but the number of nodes explored by the proposed approach is only 1,371 with a computing time of approximately 1,100 seconds. We will further discuss the performance of the proposed B&B approach in Section 5.2. Before doing that, we will compare the proposed bounds with two bounds which require the solution of SDP problems, namely the difference-of-convex (`DC`) bound, and the copositive (`Cop`) bounds.

5.1 Comparison with other bounds for StQP

First, we briefly recall how the copositive and DC bound are computed through the solution of semidefinite programs. The copositive bound can be computed as follows

$$\begin{aligned} \ell_{\texttt{Cop}}(\mathbf{Q},\mathbf{c}) \;=\; \min \quad & \tfrac{1}{2}\mathbf{Q_c} \bullet \mathbf{X} \\ & \mathbf{E} \bullet \mathbf{X} = 1 \\ & \mathbf{X} \succeq \mathbf{O} \\ & \mathbf{X} \geq \mathbf{O}. \end{aligned}$$

where:

- $\mathbf{Q_c} = \mathbf{Q} + \mathbf{c}\mathbf{e}^T + \mathbf{e}\mathbf{c}^T$;
- $\mathbf{e}$ is the vector with all entries equal to 1, while $\mathbf{E} = \mathbf{e}\mathbf{e}^T$ is the matrix with all entries equal to 1;
- $\mathbf{X} \succeq \mathbf{O}$ means that $\mathbf{X}$ is a semidefinite matrix;
- $\mathbf{X} \geq \mathbf{O}$ means that all the entries of $\mathbf{X}$ are nonnegative;
- $\bullet$ denotes the matrix inner product.

The DC bound can be computed as follows

$$\begin{aligned} \ell_{\texttt{DC}}(\mathbf{Q},\mathbf{c}) \;=\; \min \quad & \tfrac{1}{2}\mathbf{Q_c} \bullet \mathbf{X} \\ & \mathbf{E} \bullet \mathbf{X} = 1 \\ & \mathbf{X}\mathbf{e} \geq \mathbf{0} \\ & \mathit{Diag}(\mathbf{X}\mathbf{e}) - \mathbf{X} \succeq \mathbf{O} \\ & \mathbf{X} \succeq \mathbf{O}, \end{aligned}$$

where $\mathit{Diag}(\mathbf{X}\mathbf{e})$ denotes the diagonal matrix whose diagonal entries are given by vector $\mathbf{X}\mathbf{e}$. In [1] it has been proved that $\ell_{\texttt{DC}}(\mathbf{Q},\mathbf{c}) \leq \ell_{\texttt{Cop}}(\mathbf{Q},\mathbf{c})$.
In Table 1 we compare, both in terms of quality and in terms of computational time, the bounds $\ell_{\texttt{Cop}}$ and $\ell_{\texttt{DC}}$ with the bounds described in Section 3.2, namely the convex bound $\ell_{\texttt{Conv}}$ and the LP bounds $\ell_{\texttt{LP}}$ and $\ell_{\texttt{LP}-\texttt{CUT}_2}$, where the latter is the bound computed by solving the LP problem (8) with $k = 2$. We used CSDP [9] for computing both $\ell_{\texttt{Cop}}$ and $\ell_{\texttt{DC}}$, while we used Ipopt [31] for computing $\ell_{\texttt{Conv}}$, and GUROBI [13] for computing the two polyhedral bounds $\ell_{\texttt{LP}}$ and $\ell_{\texttt{LP}-\texttt{CUT}_2}$. In comparing the computational times, we stress that our machine has 4 cores. Both CSDP and GUROBI exploit all the cores, whereas Ipopt only uses one core. Hence, the time used by Ipopt should be appropriately scaled. As expected, the tighter bound is the copositive one, but it is the most expensive in terms of computational time. The worst bound is the DC bound, while the convex bound and the polyhedral ones have a comparable behaviour in terms of quality of the bound (as shown in Table 2), but the computational time for the two polyhedral ones is two order of magnitude smaller than the one of the copositive bound and one order of magnitude smaller than the DC bound (that is always significantly worse in terms of quality). By looking at these two tables, it emerges that the two polyhedral bounds represent the best options for a branch and bound algorithm aiming at solving large problems. Furthermore, looking carefully at Table 2, the improvement in the bounds obtained by adding cuts does not justify the increase in the computational time, especially for dense graphs. For this reason, our final choice was to use the $\ell_{\texttt{LP}}$ bound within the proposed branch-and-bound approach.

n(density)	$\ell_{\texttt{Cop}}$		$\ell_{\texttt{DC}}$		$\ell_{\texttt{LP}}$				$\ell_{\texttt{LP}-\texttt{CUT}_2}$				$\ell_{\texttt{Conv}}$	
	sol.time	value	sol.time	value	build time	sol.time	tot time	value	build time	sol.time	tot time	value	sol.time	value
100(0.25)	119.58	-5.47	41.35	-19.47	1.40	0.02	1.43	-7.15	1.81	0.10	1.91	-7.10	457.88	-7.10
100(0.25)	115.24	-5.85	46.33	-19.70	0.65	0.04	0.69	-7.16	0.83	0.14	0.97	-7.12	185.23	-7.11
100(0.25)	111.97	-5.56	44.72	-19.81	0.72	0.03	0.76	-7.07	0.90	0.14	1.04	-6.99	58.09	-6.99
100(0.25)	110.01	-5.62	42.95	-19.81	0.79	0.04	0.83	-6.92	0.97	0.14	1.11	-6.88	114.37	-6.88
100(0.25)	114.34	-5.69	46.03	-19.98	0.68	0.04	0.72	-7.23	0.86	0.14	1.00	-7.19	470.82	-7.19
100(0.25)	126.11	-6.07	45.23	-19.56	0.74	0.03	0.78	-7.19	0.92	0.14	1.06	-7.14	115.92	-7.14
100(0.5)	121.24	-6.28	46.17	-23.07	1.45	0.07	1.53	-7.66	1.79	0.29	2.08	-7.64	214.68	-7.64
100(0.5)	120.48	-6.14	44.25	-23.07	1.46	0.07	1.54	-7.53	1.80	0.27	2.07	-7.51	120.56	-7.51
100(0.5)	124.30	-6.28	46.61	-23.26	1.41	0.13	1.55	-7.57	1.77	0.36	2.12	-7.56	52.59	-7.55
100(0.5)	126.41	-6.28	42.11	-23.39	1.55	0.08	1.63	-7.66	1.90	0.29	2.19	-7.64	60.37	-7.64
100(0.5)	125.49	-6.27	43.38	-23.06	1.35	0.12	1.47	-7.54	1.70	0.43	2.13	-7.53	96.26	-7.53
100(0.5)	124.42	-6.18	43.82	-23.36	1.58	0.07	1.65	-7.67	1.92	0.28	2.20	-7.65	70.96	-7.65
100(0.75)	139.32	-6.63	43.19	-22.29	2.65	0.15	2.80	-7.92	3.23	0.54	3.77	-7.92	307.69	-7.92
100(0.75)	119.79	-6.57	41.32	-21.88	2.85	0.24	3.10	-7.82	3.39	0.77	4.16	-7.81	164.65	-7.81
100(0.75)	131.40	-6.54	44.19	-22.09	2.98	0.25	3.23	-7.83	3.52	0.89	4.40	-7.82	51.54	-7.82
100(0.75)	127.98	-6.63	44.83	-22.49	2.35	0.23	2.58	-7.94	2.89	0.76	3.65	-7.93	91.08	-7.93
100(0.75)	134.69	-6.56	47.70	-21.80	2.46	0.25	2.71	-7.82	3.00	0.78	3.78	-7.82	121.82	-7.81
100(0.75)	129.42	-6.81	47.77	-21.94	3.27	0.25	3.53	-7.89	3.81	0.81	4.62	-7.88	188.64	-7.88

Table 1: Bounds comparison. 'build time' denotes the CPU time used by GUROBI to build the LP problem.

n(density)	$\frac{\ell_{\texttt{Conv}} - \ell_{\texttt{LP}}}{\lvert\ell_{\texttt{Conv}}\rvert}$	$\frac{\ell_{\texttt{Conv}} - \ell_{\texttt{LP}-\texttt{CUT}_2}}{\lvert\ell_{\texttt{Conv}}\rvert}$	$\frac{\ell_{\texttt{LP}-\texttt{CUT}_2} - \ell_{\texttt{LP}}}{\lvert\ell_{\texttt{LP}-\texttt{CUT}_2}\rvert}$
100(0.25)	0.705 %	0.056 %	0.649 %
100(0.25)	0.580 %	0.037 %	0.542 %
100(0.25)	1.091 %	0.039 %	1.052 %
100(0.25)	0.668 %	0.037 %	0.631 %
100(0.25)	0.527 %	0.027 %	0.499 %
100(0.25)	0.608 %	0.028 %	0.580 %
100(0.5)	0.211 %	0.013 %	0.197 %
100(0.5)	0.250 %	0.019 %	0.231 %
100(0.5)	0.269 %	0.016 %	0.253 %
100(0.5)	0.272 %	0.018 %	0.254 %
100(0.5)	0.187 %	0.012 %	0.175 %
100(0.5)	0.214 %	0.014 %	0.200 %
100(0.75)	0.105 %	0.006 %	0.099 %
100(0.75)	0.134 %	0.009 %	0.125 %
100(0.75)	0.113 %	0.007 %	0.106 %
100(0.75)	0.145 %	0.009 %	0.137 %
100(0.75)	0.096 %	0.006 %	0.089 %
100(0.75)	0.073 %	0.005 %	0.068 %

Table 2: Comparison among the convex bound, the polyhedral bound and the polyhedral including cuts.

5.2 Numerical experimentation with the branch-and-bound approach

We implemented our branch and bound algorithm using Julia v0.4.5 and ran our experiments on a 2.7 GHz Intel I5 PC, with 32GB of RAM. GUROBI and Ipopt have been compiled with version 5.3.1 of the GCC library. As test problems, we used the set of random problems used in [27]. We also generated some more problems by following the procedure described in [23], which is the same used to generate test problems of [27]. For each couple (number of variables, density), we generated five random instances beside the one provided to us by Prof. A. Scozzari and also employed in [27].
Both the code and the instances are freely available for download at the URL `http://www.iasi.cnr.it/~liuzzi/StQP`.
Doing the experimentation, we ran the algorithm using all possible combinations of branching and selection strategies. Namely, we implemented both binary and n-ary branching strategies and best-bound and depth-first selection strategies. These four combinations are denoted by

- BBB, Binary branching with Best-Bound selection strategy;
- BDF, Binary branching with Depth-First selection strategy;
- NBB, N-ary branching with Best-Bound selection strategy;
- NDF, N-ary branching with Depth-First selection strategy.

Table 3: Results on random instances. The symbol '-' means that the algorithm did not complete because of an out of memory error

n(density)	BBB		BDF		NBB		NDF	
	# nodes	time	# nodes	time	# nodes	time	# nodes	time
100(0.25)	215	4.73	215	4.80	203	2.81	203	2.88
100(0.25)	129	3.44	129	3.49	128	2.16	128	2.20
100(0.25)	217	5.16	257	5.88	203	3.15	223	3.51
100(0.25)	139	3.79	189	4.72	163	2.57	212	3.27
100(0.25)	197	4.81	197	4.56	181	2.78	203	3.12
100(0.25)	91	2.56	91	2.59	94	1.60	94	1.62
100(0.25) avg.	164.67	4.08	179.67	4.34	**162**	**2.5**	177.17	2.77

continued on next page

Table 3 – continued from previous page

n(density)	BBB		BDF		NBB		NDF	
	# nodes	time	# nodes	time	# nodes	time	# nodes	time
100(0.5)	399	15.25	571	20.48	377	11.14	395	11.86
100(0.5)	331	13.39	349	14.12	323	9.80	328	10.23
100(0.5)	357	17.74	357	18.44	362	13.17	362	13.50
100(0.5)	361	14.94	431	17.16	353	11.11	499	14.99
100(0.5)	255	11.62	269	12.05	221	7.98	310	10.15
100(0.5)	545	20.97	573	22.60	461	14.10	484	14.95
100(0.5) avg.	374.67	15.65	425	17.47	**349.5**	**11.22**	396.33	12.61
100(0.75)	931	71.19	973	74.66	947	63.90	1009	69.24
100(0.75)	629	65.64	909	69.74	590	55.61	683	62.46
100(0.75)	981	96.82	1659	148.88	927	82.33	1374	116.23
100(0.75)	767	67.04	767	69.84	729	56.13	729	57.83
100(0.75)	633	58.45	1003	82.31	608	50.25	765	59.75
100(0.75)	485	54.29	527	46.85	401	41.91	463	37.85
100(0.75) avg.	737.67	68.9	973	82.05	**700.33**	**58.35**	837.16	67.23
200(0.25)	235	23.09	235	22.73	184	11.14	184	11.56
200(0.25)	419	35.50	501	39.53	388	20.95	453	24.22
200(0.25)	347	31.32	407	33.71	293	16.73	457	24.15
200(0.25)	533	42.37	619	45.93	515	26.15	532	27.67
200(0.25)	245	24.80	267	25.10	197	11.88	224	13.25
200(0.25)	359	30.41	359	30.41	313	16.68	313	17.34
200(0.25) avg.	356.33	31.25	398	32.9	**315**	**17.25**	360.5	19.7
200(0.5)	737	181.28	753	186.44	646	116.23	651	120.51
200(0.5)	1181	286.96	1279	301.64	1146	209.57	1187	218.69
200(0.5)	645	185.41	645	185.79	593	123.19	593	126.74
200(0.5)	1201	291.28	1641	358.82	1204	219.16	1217	225.26
200(0.5)	757	204.77	757	207.54	698	135.64	698	140.06
200(0.5)	623	175.11	627	176.47	575	118.07	755	142.72
200(0.5) avg.	857.33	220.8	950.33	236.12	**810.33**	**153.64**	850.16	162.33
200(0.75)	3741	2945.64	3747	2865.71	3889	2737.11	4334	2806.54
200(0.75)	3741	2914.88	3747	2842.91	3889	3052.65	4334	2783.15
200(0.75)	3593	2734.23	3975	2915.36	3638	2456.67	4195	2698.10
200(0.75)	2455	2086.92	3667	2750.65	2424	1858.77	2958	2013.54
200(0.75)	3149	2360.05	3195	2387.27	3291	2134.93	3415	2212.32
200(0.75)	2457	2004.03	2461	2012.76	2551	1768.29	2650	1834.61
200(0.75) avg.	**3189.33**	2507.63	3465.33	2629.11	3280.33	**2334.74**	3647.67	2391.38
500(0.25)	-	-	-	-	2017	2424.57	2140	1227.41
500(0.25)	-	-	-	-	1417	2959.50	1758	1033.43
500(0.25)	-	-	-	-	2017	2071.88	2708	1476.48
500(0.25)	-	-	-	-	1982	2488.60	2411	1357.10
500(0.25)	-	-	-	-	2951	1762.73	3205	1739.36
500(0.25)	-	-	-	-	1888	1068.54	1967	1137.46
500(0.25) avg.	-	-	-	-	**2045.33**	2129.3	2364.83	**1328.54**

In Table 3 we report for each instance and each strategy both the number of nodes of the branch-and-bound tree and the computing time. It emerges that for almost all instances the best combination of branching and selection strategy is the NBB. For problems 500(0.25), BBB and BDF are not able to solve the problems because of an out of memory error. Indeed, we note that in our implementation at each node we store the LP problem inherited from the father node, and then we only add the constraints deriving by the branching rule. This strategy however, is only possible if the number of open problems is limited with respect to the available memory, because when the dimension of the problem increases (i.e., more than 200) there are memory issues, see, e.g., the missing results for the two binary strategies on problems with $n = 500$ and density 0.25 in Table 3. For this reason, when the dimension is higher than 200, it is convenient to rebuild the LP problem whenever a node is created.

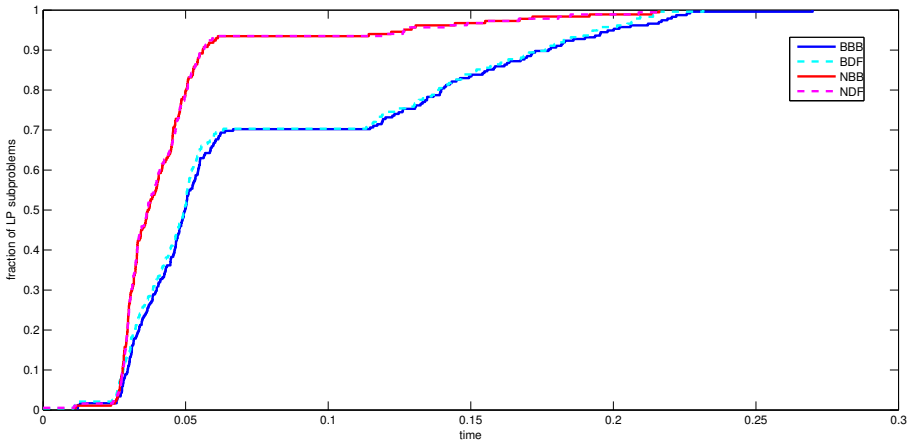

Figure 1: Percentage of LP problems solved within time t

n(density)	BBB		BDF		NBB		NDF	
	# nodes	time	# nodes	time	# nodes	time	# nodes	time
500(0.25)	2109	2170.12	2417	2263.99	2017	1265.28	2140	1334.62
500(0.25)	1503	1877.80	1507	1728.14	1417	928.63	1758	1128.51
500(0.25)	2119	2055.20	2559	2545.55	2017	1253.75	2708	1649.93
500(0.25)	2083	2077.85	2209	2315.42	1982	1261.75	2411	1504.78
500(0.25)	3087	2709.23	3177	2972.38	2951	1813.30	3205	1935.97
500(0.25)	1995	1990.26	2033	2200.00	1888	1186.47	1967	1231.75
500(0.25) avg.	2149.33	2146.74	2317	2337.58	**2045.33**	**1284.86**	2364.83	1464.26
500(0.5)	9721	35927.09	9721	35597.08	9499	26024.75	9499	25948.24
500(0.5)	6339	25194.37	6471	25594.24	5957	17733.04	6708	18836.66
500(0.5)	10353	37291.99	10557	38130.69	10055	27179.82	10552	28433.87
500(0.5)	10103	36033.43	10111	37194.76	9818	29087.96	10131	27007.95
500(0.5)	4877	20932.02	6263	24701.34	4513	13934.77	5789	16558.47
500(0.5) avg.	6898.83	25896.5	7187.17	26869.7	**6640.33**	**18993.4**	7113.17	19464.2

Table 4: Results obtained by rebuilding the LP subproblem to compute the bound at each node.

To further motivate the better performances of the n-ary branching, we draw in Figure 1, the fraction of LP problems solved within time t on a problem with $n = 200$ and $d = 0.25$. More in detail, we report the solution time on the x-axis and the number of problems solved within that time for each strategy on the y-axis. As expected, the n-ary branching allows to solve more problems where the variables are fixed to be positive, resulting in faster LPs since many variables are then fixed to zero.

In Table 4 we report results of the four version of our algorithm when the LP problems are not stored at every node of the branch and bound tree. As it can be seen, sometimes (see, for instance, 500(0.25) with strategy NBB), it may be convenient in terms of time even when it is not mandatory (see Table 4).

Considering the results reported in Table 1 of [27], we are able to fill most of the gaps in the Exact algorithm column. Indeed, our method suffers less of the increase in the density of the matrix, allowing us to solve exactly problems with larger size and/or density. More in detail, we can solve problems of size 200 with density 0.75, and of size 500 with density 0.5.

In Table 5 we report the computing times for NBB and for the approach proposed in [27], normalized by the time required by the latter to solve the problem with $n = 100$ and density 0.25 (which is, thus, equal to 1). Of course, the times of the two algorithms cannot be directly compared since the experiments have been run on different processors (a 2GHz P4 for what concerns [27]). However, the important observation that one can draw from these results is that, both at $n = 100$

n	0.25		0.5		0.75	
	NBB	[27]	NBB	[27]	NBB	[27]
100	1.61	1.00	6.40	29.05	36.72	1021.03
200	6.40	8.47	66.79	502.71	1573.05	-

Table 5: Time comparison between NBB and [27]

and at $n = 200$, the increase of the computing times with the density of the underlying convexity graph is much slower for NBB with respect to the approach proposed in [27]. Thus we are able to claim that the proposed approach scales better than the approach in [27] as the density increases.

6 Conclusions

In this paper we proposed a new branch-and-bound approach for the solution of StQP problems which performs an implicit enumeration of the KKT points of the StQP problem.
As concerns the bound computation, we tested different bounding strategies. The computational experiments showed that the best compromise between quality of the bound and speed of computation is obtained by using a polyhedral underestimator of the more complex convex bound, which is based on the decomposition of the quadratic function into a sum of quadratic functions. Such a decomposition is based on a decomposition of the underlying convexity graph of the StQP into star graphs for which the convex envelope over the unit simplex can be analytically computed.
We also tested different branching strategies and showed that the more promising one is a non-binary one based on the computation of a suitable independent set of the convexity graph.
The approach has been tested on instances randomly generated according to [23] and on instances used in the literature, see, e.g., [27]. The experiments showed that the non-binary branching strategy with a selection criterion based on the best bound (NBB), gives the best results allowing to exactly solve problems up to $n = 500$ with density of the convexity graph 0.5.
As a final note, we compared our best algorithm (i.e., NBB) with the exact approach proposed in [27]. We showed that NBB outperforms [27] in terms of scalability with respect to density of the convexity graph.
A possible direction for future research regards different way of decomposing the (underlying convexity graph of the) quadratic function in order to further improve the bound.

Acknowledgements

We thank professors A. Scozzari and F. Tardella for kindly providing us their instances of StQP problems.

References

[1] Anstreicher, K., Burer, S., D.C. Versus Copositive Bounds for Standard QP, *Journal of Global Optimization*, 33, 299-312 (2005)

[2] Audet, C., Hansen, P., Jaumard, B., and Savard, G., A symmetrical linear maxmin approach to disjoint bilinear programming, *Mathematical Programming*, 85 , 573–592 (1999)

[3] Best, M. J., B. Ding, Global and local quadratic minimization, *Journal of Global Optimization*, 10, 77–90 (1997)

[4] I.M. Bomze, On standard quadratic optimization problems, *Journal of Global Optimization*, 13, 369-387 (1998)

[5] Bomze, I.M., Duer, M., de Klerk, E., Roos, C., Quist, A.R., Terlaky, T., On copositive programming and Standard Quadratic optimization problems, *Journal of Global Optimization*, 18, 301-320 (2000)

[6] Bomze, I.M., De Klerk, E., Solving standard quadratic optimization problems via linear, semidefinite and copositive programming, *Journal of Global Optimization*, 24, 163–185 (2002)

[7] Bomze, I.M., Branch-and-bound approaches to standard quadratic optimization problems, *Journal of Global Optimization*, 22, 17-37 (2002)

[8] Bomze, I.M., Locatelli, M., Tardella, F., New and old bounds for standard quadratic optimization: dominance, equivalence and incomparability, *Mathematical Programming*, 115, 31–64 (2008)

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

[10] Burer, S., Vandenbussche, D., A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations, *Mathematical Programming*, 113 , 259–282 (2008)

[11] de Klerk, E., Pasechnik, D.V., Approximation of the stability number of a graph via copositive programming, *SIAM Journal of Optimization*, 12, 875–892 (2002)

[12] L.E. Gibbons, D.W. Hearn, P.M. Pardalos, M.V. Ramana, Continuous characterizations of the maximum clique problem, *Mathematics of Operations Research*, 22, 754-768 (1997)

[13] Gurobi Optimization, Inc., Gurobi Optimizer Reference Manual (2015).

[14] Gvozdenovič, N., Laurent, M. (2005), "Semidefinite bounds for the stability number of a graph via sums of squares of polynomials, *Mathematical Programming*, 110, 145-173 (2007)

[15] Hastad, J., Clique is hard to approximate within $|V|^{1-\varepsilon}$, *Acta Mathematica*, 182, 105-142 (1999)

[16] Ibaraki, T., Katoh, N., *Resource Allocation Problems: Algorithmic Approaches*, MIT Press, Cambridge (1988)

[17] Lasserre, J.B., Global optimization with polynomials and the problem of moments, *SIAM Journal on Optimization*, 11, 796–817 (2001)

[18] Locatelli, M., Convex envelopes of some quadratic functions over the n-dimensional unit simplex, to appear in *SIAM Journal on Optimization*, 25, 589—621 (2015)

[19] Lovàsz, L. , On the Shannon capacity of a graph, *IEEE Transactions on Information Theory*, 25, 1–7 (1979)

[20] Markowitz, H.M., Portfolio selection, *Journal of Finance*, 7, 77-91 (1952)

[21] Markowitz, H.M., The general mean-variance portfolio selection problem, In: Howison, S.D., Kelly, F.P.,Wilmott, P. (eds.) *Mathematical Models in Finance*, vol. 93-99. Chapman & Hall, London (1995)

[22] Motzkin, T. S., Straus, E. G., Maxima for graphs and a new proof of a theorem of Turán, *Canadian Journal of Mathematics*, 17, 533–540 (1965)

[23] Nowak, I., A new semidefinite programming bound for indefinite quadratic forms over a simplex, *Journal of Global Optimization*, 14, 357–364 (1999)

[24] Pena, J., Vera, J., Zuluaga, L., Computing the stability number of a graph via linear and semidefinite programming, *SIAM Journal on Optimiztaion*, 18, 87-105 (2007)

[25] Sagol, G., Yildirim, E.A., Analysis of copositive optimization based linear programming bounds on standard quadratic optimization, *Journal of Global Optimization*, 63, 37-59 (2015)

[26] Schrijver, A., A comparison of the Delsarte and Lovasz bounds, *IEEE Transactions on Information Theory*, 25, 425–429 ,(1979)

[27] Scozzari, A., Tardella, F., A clique algorithm for standard quadratic programming, *Discrete Applied Mathematics*, 156, 2439-2448 (2008)

[28] Tomita, E., Tanaka, A., Takahashi, H., The worst-case time complexity for generating all maximal cliques and computational experiments, *Theoretical Computer Science*, 363, 28-42 (2006)

[29] Yildirim, E.A., On the accuracy of uniform polyhedral approximations of the copositive cone, *Optimization Methods and Software*, 27, 155-173 (2012)

[30] Blelloch, G.E., Fineman, J.T., Shun, J., Greedy Sequential Maximal Independent Set and Matching Are Parallel on Average, in *Proceedings of the Twenty-fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2012, Pittsburgh)*, 308-317 (2012)

[31] W achter, A. Biegler, L. T., On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming, *Mathematical Programming*, 106(1), 25–57 (2006).