Noname manuscript No.
(will be inserted by the editor)

Mining for diamonds — matrix generation algorithms for binary quadratically constrained quadratic problems

Enrico Bettiol · Immanuel Bomze · Lucas Létocart · Francesco Rinaldi · Emiliano Traversi

Received: date / Accepted: date

Abstract In this paper, we consider binary quadratically constrained quadratic problems and propose a new approach to generate stronger bounds than the ones obtained using the Semidefinite Programming relaxation. The new relaxation is based on the Boolean Quadric Polytope and is solved via a Dantzig-Wolfe Reformulation in matrix space. For block-decomposable problems, we extend the relaxation and analyze the theoretical properties of this novel approach. If overlapping size of blocks is at most two (i.e., when the sparsity graph of any pair of intersecting blocks contains either a cut node or an induced *diamond graph*), we establish equivalence to the one based on the Boolean Quadric Polytope. We conjecture that equivalence holds for any block structure with a chordal sparsity graph. The tailored decomposition algorithm in the matrix space is used for efficiently bounding sparsely structured problems. Preliminary numerical results show that the proposed approach yields very good bounds in reasonable time.

Keywords Binary Quadratically Constrained Quadratic Programs · Dantzig-Wolfe Reformulation · Boolean Quadric Polytope · Sparse Problem.

Mathematics Subject Classification (2010) 90C09 · 90C20 · 90C26 · 49M27

Enrico Bettiol
Fakultät für Mathematik, TU Dortmund University, Germany E-mail: enrico.bettiol@math.tu-dortmund.de

Immanuel Bomze
ISOR, VCOR & ds:UniVie, University of Vienna E-mail: immanuel.bomze@univie.ac.at

Lucas Létocart, Emiliano Traversi
LIPN, CNRS, (UMR7030), University Sorbonne Paris Nord E-mail: letocart@lipn.univ-paris13.fr, traversi@lipn.univ-paris13.fr

Francesco Rinaldi
Department of Mathematics "Tullio Levi-Civita", University of Padova E-mail: rinaldi@math.unipd.it

1 Introduction and literature review

A generic Binary Quadratically Constrained Quadratic problem (BQCQP) can be written in the following form:

$$
\begin{aligned}
\min\ & x^\top Q x && && \text{(1a)}\\
\text{s. t. }\ & x^\top A_i x \le b_i && \forall i \in \mathcal{I} && \text{(1b)}\\
& x \in \{0,1\}^n\,, && && \text{(1c)}
\end{aligned}
$$

where $\mathcal{I}$ is the index set of the constraints, while Q and A_i both are symmetric $n \times n$-matrices and $b_i \in \mathbb{R}$ for all $i \in \mathcal{I}$. No further assumptions on the matrices are required, in particular, the continuous relaxation of the problem can be non convex.

BQCQPs play an important role in the field of Mathematical Programming. They have applications in many different fields, such as Telecommunications, Finance, Biology, Energy, Robotics, just to cite a few of them (see [25], [26] for more details). The solution of BQCQPs is NP-hard and the solution techniques for BQCQPs draw heavily on both discrete and continuous theory and methodologies. For a detailed description of solution techniques for nonlinear problems and quadratic problems we refer the reader to [16] and [25].

The techniques proposed in this paper are closely related to conic optimization (more precisely semidefinite, copositive and completely positive optimization) on one side, and Dantzig-Wolfe Reformulation (DWR) and Column Generation (CG) on the other side.

DWR, firstly introduced in [17], is a well known technique used to obtain tight bounds for discrete problems. Its principle is to replace the feasibility region corresponding to a subset of constraints of the model by the convex hull of its extreme points. This is obtained through an inner representation that exploits the Minkowski-Weyl Theorem, a classic result in polyhedral theory saying that every polyhedron is the convex combination of its extreme points plus the conic combination of its extreme rays. CG is a technique frequently used for solving problems with a large (usually exponential) number of columns. CG is normally used for handling linear programs, but it can be applied to any nonlinear program, under some convexity assumptions. The fact that the application of DWR leads to reformulated problems with an exponential number of variables makes it natural to use CG to solve these reformulations effectively. Among the various articles and surveys on CG and DWR available in the literature, we point out [2], [18], [33] and [40].

In order to rewrite the BQCQP (1) in matrix form, we make use of the Hilbert product $\langle A, B\rangle = \operatorname{trace}(AB)$ and introduce the matrix variable X to represent all products of the original variables: $X_{ij} = x_i x_j$, $\forall i, j = 1, \ldots, n$.

Now, we can equivalently rewrite the original problem as follows:

$$\begin{aligned} \min\ & \langle Q, X\rangle && && \text{(2a)}\\ \text{s. t. }\ & \langle A_i, X\rangle \le b_i && \forall i \in \mathcal{I} && \text{(2b)}\\ & X = xx^\top && && \text{(2c)}\\ & x \in \{0,1\}^n\,. && && \text{(2d)} \end{aligned}$$

Several conic reformulations for quadratic programs are formulated on the extended (matrix) space obtained after adding an additional set of variables representing the products of the original variables. Since the constraint imposing an exact relation between the original and the new set of variables is non convex and difficult to deal with, a standard technique is relaxing this constraint, and obtain lower bounds to be used in a Branch-and-Bound framework.

SemiDefinite Programming (SDP) based relaxations, which require the matrix variable to lie in the cone of positive semidefinite matrices, represent a nice tool in this context. SDP is particularly interesting because it enjoys polynomial complexity and usually leads to relatively tight dual bounds (see, e.g., [35]). We refer the reader to [30] and [32] for a survey on SDP. Specific software has been developed for SDP, for example the commercial solver *MOSEK*, SeDuMi (see [38]), SDPT3 (see [39]), CSDP (see [11]) among others. The open source software *BiqCrunch* is a semidefinite-based solver for binary quadratic problems (see [37]). In this work we will present a new relaxation to provide even tighter bounds for problem (1).

Among the other types of conic relaxations, two important classes are represented by the copositive and completely positive relaxations, that respectively lead to optimization in the cone of *Copositive matrices* and *Completely positive matrices*. The subject of copositive programming was first introduced in [29]. Then, it has received increasing interest over the last few decades and a large number of works has been produced. For a description of copositive optimization, we start referring to [15] and to the survey [22]. Other papers which present reviews of recent advances in this field, along with applications, are [5] and [9]. The book [4] is also a useful source.

The first important result about copositive programming is given in [7]: the authors consider the so-called *Standard quadratic problem*, which is the minimization of a quadratic function over a simplex, and they prove that its completely positive relaxation is a reformulation of the original problem.

This first reformulation of a quadratic program in copositive form is followed by other results. A noticeable result is given in [13]. The author extended the first result to every linearly constrained quadratic problem with binary variables.

Solving a copositive problem remains obviously very hard. Typically, hierarchies of polyhedral cones are used to approximate the copositive cone. These are used, for instance, in [6] and in [12], while in [14] the author proposes a doubly non negative relaxation of the completely positive cone. A different approach is given in [8], where the authors propose a feasible direction heuristic to solve a problem in the completely positive cone. However, the initial point with a factorization is needed, which is trivial in some cases, but difficult in general.

Geometrical aspects of these cones have been extensively studied. The interior of the completely positive cone has been described in [23] and their result has been improved in [19]. Other important results are obtained in [20].

2 Boolean Quadric Polytope (BQP) relaxation

As mentioned above, it is well known that the SDP relaxation of the formulation given in (2) can be used to obtain strong dual bounds for problem (1). In this work, we present a seemingly novel relaxation of problem (2) that allows us to obtain dual bounds stronger than the ones obtained using the SDP relaxation. We intend to replace the constraint (2c) by letting the matrix X be a convex combination of rank-1 matrices X_p of the type $X_p = x_p x_p^\top$, where $x_p \in \{0,1\}^n$. The problem then takes the following form:

$$
\begin{aligned}
\min\ & \langle Q, X\rangle && && \text{(3a)}\\
\text{s. t. } & \langle A_i, X\rangle \leq b_i && \forall i \in \mathcal{I} && \text{(3b)}\\
& X = \sum_{p\in\mathscr{P}} \lambda_p X_p && && \text{(3c)}\\
& \sum_{p\in\mathscr{P}} \lambda_p = 1 && && \text{(3d)}\\
& \lambda_p \geq 0 && \forall p \in \mathscr{P}\,, && \text{(3e)}
\end{aligned}
$$

where $\mathscr{P} = \{p \in \mathbb{N} : x_p \in \{0,1\}^n\}$ is the index set of all possible binary extreme points x_p. Thus $\mathscr{P}$ is a finite set of size exponential in n, that is $|\mathscr{P}| = 2^n$.

It is interesting to investigate the relation between problem (3) and the well-known Boolean Quadric Polytope [34], the convex hull BQP^n of all binary rank-1 matrices $X_p \in \mathbb{R}^{n\times n}$.

Proposition 1 *The problem* (3) *is a relaxation of* (1)*, and its domain is the* n*-dimensional Boolean Quadric Polytope* BQP^n*.*

Proof It is a relaxation because all the solutions of the original problem (1) are achieved by our reformulation with $\lambda \in \{0,1\}^{|\mathscr{P}|}$, that is in the case when only one extreme point is considered (due to the constraint (3d)). Then, by the constraints (3c) and the constraints on the λ variables, we know that a solution X of (3) is a convex combination of rank-1 binary matrices X_p given by $X_p = x_p x_p^\top$, where x_p is binary, hence it is in BQP^n. □

We now introduce the notions of Copositive and Completely Positive matrices: let $A \in \mathcal{S}^n$. We say that A is *CoPositive* (COP) if $x^\top A x \geq 0$ for all $x \in \mathbb{R}^n_+$. On the other hand, we say that A is *ComPletely Positive* (CPP) if there exist k and $B \in \mathbb{R}_+^{n \times k}$ such that $A = BB^\top$. We notice that, since every point in BQP^n is a convex combination of doubly non-negative matrices, it is clearly in the CPP cone, which we denote by $\mathcal{C}^*$. Indeed, if we define $b_p := \sqrt{\lambda_p} x_p$, then (3c) becomes $X = \sum_{p \in \mathscr{P}} b_p b_p^\top$, which implies that $X \in \mathcal{C}^*$. Hence the bounded polyhedron BQP^n is strictly contained in the non-polyhedral cone $\mathcal{C}^*$ and thus the domain of (3) is typically strictly contained in that of the CPP relaxation. By consequence, the BQP relaxation is stronger than the CPP relaxation which in turn is stronger than the SDP relaxation, since the SDP cone contains the CPP cone.

Related works on an inner approximation of the CPP cone include [27,41]. While [41] offers theoretical error bounds for a general setting of discretization of the CPP cone (or a base of it), [27] takes a different approach which uses 2×2 blocks overlapping in at most one diagonal entry. This would correspond to a cut node in the specification graph. In the sequel, we will consider a more general case of diamond graphs; see Proposition 2.

2.1 Solving the BQP relaxation

In this section we present a way to solve (3) via Column Generation (CG) in the matrix space. CG starts with a subset of variables $\bar{\mathscr{P}} \subset \mathscr{P}$ and iteratively verifies (via an auxiliary problem called *Pricing problem*) if the solution obtained with the variables used so far is optimal or if it is potentially necessary to add additional variables taken from $\mathscr{P} \setminus \bar{\mathscr{P}}$.

Let RMP($\bar{\mathscr{P}}$) be the Restricted Master problem with variables $\bar{\mathscr{P}}$, that is problem (3) with $\mathscr{P}$ replaced by $\bar{\mathscr{P}}$. Given the extreme points $\{x_p x_p^\top : p \in \bar{\mathscr{P}}\}$, problem RMP($\bar{\mathscr{P}}$) can be written in the following form:

$$
\begin{aligned}
\min\ & \langle Q, X \rangle && && \text{(4a)}\\
\text{s. t. } & \langle A_i, X \rangle \le b_i && \forall i \in \mathcal{I} && \text{(4b)}\\
& X = \sum_{p \in \bar{\mathscr{P}}} \lambda_p \left(x_p x_p^\top \right) && && \text{(4c)}\\
& \sum_{p \in \bar{\mathscr{P}}} \lambda_p = 1 && && \text{(4d)}\\
& \lambda_p \ge 0 && \forall p \in \bar{\mathscr{P}}\,. && \text{(4e)}
\end{aligned}
$$

We can replace the constraints (4c) in the objective function and in the other constraints, thus obtaining a linear problem in the λ variables only. We also introduce the notation $X_p := x_p x_p^\top$. Hence we have:

$$
\begin{aligned}
\mathrm{RMP}(\bar{\mathscr{P}}) \qquad & \min \sum_{p\in\bar{\mathscr{P}}} \langle Q, X_p\rangle \lambda_p && && \text{(5a)}\\
& \text{s. t.} \sum_{p\in\bar{\mathscr{P}}} \langle A_i, X_p\rangle \lambda_p \le b_i && \forall i \in \mathcal{I} \quad [\rho] && \text{(5b)}\\
& \sum_{p\in\bar{\mathscr{P}}} \lambda_p = 1 && [\pi_0] && \text{(5c)}\\
& \lambda_p \ge 0 && \forall p \in \bar{\mathscr{P}}\,. && \text{(5d)}
\end{aligned}
$$

The dual of RMP($\bar{\mathscr{P}}$) reads as follows:

$$
\begin{aligned}
& \max\ b^\top \rho + \pi_0 && && \text{(6a)}\\
& \text{s. t.} \sum_{i\in\mathcal{I}} \langle A_i, X_p\rangle \rho_i + \pi_0 \le \langle Q, X_p\rangle && \forall p \in \bar{\mathscr{P}} && \text{(6b)}\\
& \rho \le 0\,, && && \text{(6c)}
\end{aligned}
$$

where $\rho \in \mathbb{R}^{|\mathcal{I}|}$ are the dual variables corresponding to the quadratic constraints and $\pi_0 \in \mathbb{R}$ is the dual variable corresponding to the equality constraint.

CG can be viewed as a dual cutting plane method. Once RMP($\bar{\mathscr{P}}$) is solved to optimality, the optimal dual variables ρ^*, π_0^* are available. The Pricing problem reduces to check if all the constraints (6b) are satisfied with $\rho = \rho^*, \pi = \pi_0^*$ for every point X_p as $p \in \mathscr{P} \setminus \bar{\mathscr{P}}$. If these constraints are valid for every $p \in \mathscr{P}$, then our primal feasible solution is also feasible for the dual of the master program, so it is optimal. Otherwise, there are points $X_p \in \mathscr{P} \setminus \bar{\mathscr{P}}$ that violate these constraints. The pricing problem consists of finding an extreme point that corresponds to a violated constraint (6b) (in other words, an extreme point with a negative reduced cost). To do so, the reduced cost is minimized for every point $X \in \mathscr{P}$. If the minimum is less than 0, we can add the corresponding constraint in the dual and the corresponding variable in the master, otherwise the algorithm terminates. The pricing problem takes the following form:

$$
\begin{aligned}
& \min\ \langle Q, X\rangle - \sum_{i\in\mathcal{I}} \langle A_i, X\rangle \rho_i^* - \pi_0^* && \text{(7a)}\\
& \text{s. t.}\ X = xx^\top && \text{(7b)}\\
& x \in \{0,1\}^n\,, && \text{(7c)}
\end{aligned}
$$

and can be rewritten in vector form:

$$
\begin{aligned}
& \min\ x^\top (Q - \sum_{i\in\mathcal{I}} \rho_i^* A_i) x - \pi_0^* && \text{(8a)}\\
& \text{s. t.}\ x \in \{0,1\}^n\,. && \text{(8b)}
\end{aligned}
$$

While the master program is linear, the pricing problem is an unconstrained binary quadratic program. If the original problem is convex, i.e., if the matrix Q is positive semidefinite and so are all the matrices A_i, then all the pricing problems are convex as well: indeed, the matrix in its objective function (8a) is sum of positive semidefinite matrices because of the non positivity of the dual variables ρ^*, see (6c).

2.2 Computational experiments with the BQP relaxation

When using this framework, we need to solve a sequence of linear master problems and unconstrained quadratic binary pricing problems to obtain the optimal value of the BQP-relaxation related to the original problem. In our implementation `Cplex` is used as solver for both the master and the pricing problems. From a technical point of view, to ensure feasibility in the first iterations of the algorithm, we initialize RMP($\bar{\mathscr{P}}$) with a set of *dummy* columns that makes the set of constraints feasible and with a sufficiently large cost. Since the most challenging task is solving the pricing problem, we use an *early stopping* strategy when dealing with it. We impose that the solver stops as soon as it finds a point with a negative reduced cost. If the algorithms finds it, then we can add this point to the set of extreme points for the master and proceed in the column generation algorithm. If there are no such points, this means that no other extreme point can be added so the algorithm has already found the optimum and it can stop.

It is worth noticing that the proposed methodology could be extended to solve instances with also continuous and general integer variables but in a first round of tests we focus on purely binary instances from the QPLIB library [25].

We compare the performance of our method with `BiqCrunch` [31], an open source SDP solver for binary quadratic programs. When running `BiqCrunch`, we consider two standard sets of parameters. The first set (*BC-bound*) is used to obtain the value of the SDP bound (see, e.g., [35]), without any additional inequality. The second set (*BC-cuts*) provides the value of the SDP bound with the addition of the so-called *triangle inequalities* [1]. The results are collected in Table 1. The first column contains the names of the instances. Then, for each of the `BiqCrunch` set of parameters and for our algorithm (BQP), two subcolumns represent the root node gap and the time, in seconds, spent to obtain it. The root node gap is calculated as the difference between the optimum and the value of the lower bound given by the relaxation, divided by the optimal value, in percentage.

The average gap exceeds 1000% for the basic SDP bound, it is equal to 132% for the SDP bound with triangle inequalities and to 76% for our method. The fact that the BQP relaxation provides a dual bound that is also stronger than the SDP relaxation with triangle inequalities is theoretically confirmed by the results in [34]. In terms of computing time, BQP is still competitive with BC with cuts. Taking into account that the pricing problem is solved with a generic solver, the results obtained are definitely encouraging. We might indeed

Instance	BC-bound		BC-cuts		BQP	
	T (s)	Gap (%)	T (s)	Gap (%)	T (s)	Gap (%)
QPLIB-0067	0	5	22	2	0	1
QPLIB-1976	27	433	193	371	7	368
QPLIB-2017	113	441	114	441	124	240
QPLIB-2029	180	562	180	562	1865	192
QPLIB-2036	220	740	220	740	185	313
QPLIB-2055	21	41	104	35	92	32
QPLIB-2060	36	42	655	33	153	32
QPLIB-2067	72	68	149	65	242	62
QPLIB-2073	57	18	1078	10	285	10
QPLIB-2085	85	33	2642	23	1066	23
QPLIB-2087	123	71	172	71	2935	56
QPLIB-2096	82	18	2679	11	1210	11
QPLIB-2357	16	13	46	0	3223	0
QPLIB-2359	74	11	54	2	2888	0
QPLIB-2512	2	428	117	120	6	100
QPLIB-2733	10	762	1006	178	19258	154
QPLIB-2957	78	> 1000	2392	357	11261	100
QPLIB-3307	5	798	1044	198	472	100
QPLIB-3413	33	> 1000	678	210	11	100
QPLIB-3587	5	791	109	104	2	100
QPLIB-3614	4	680	123	100	2	100
QPLIB-3714	2	101	20	0	1607	0
QPLIB-3751	2	100	20	0	7709	0
QPLIB-3757	482	18	344	37	8850	0
QPLIB-3762	1	17	2	0	1037	0
QPLIB-3775	5	100	25	0	32932	0
QPLIB-3803	12	33	57	0	5620	0
QPLIB-3815	2	29	41	6	1807	2
QPLIB-6647	1009	> 1000	11271	150	232	100
QPLIB-7127	646	> 1000	1662	> 1000	2750	0
average	95.1	> 1000	881.28	132	3623.48	76

Table 1: Gap and time, QPLIB instances.

obtain a significant speed-up in the procedure by using ad-hoc algorithms for unconstrained quadratic problems.

3 BQP block relaxation

Now we present a generalization of the relaxation proposed in the previous sections that exploits the sparsity of a problem by decomposing it into several blocks. The new block decomposition presents interesting links with the theory of matrix completion problems. We report some useful definitions below.

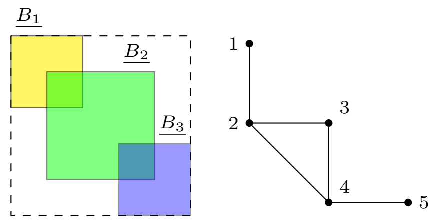
Definition 1 We define the *support* of a matrix $M \in \mathbb{R}^{m\times n}$ as

$$\mathrm{Supp}(M) := \{(p,q) \in \{1,\dots,m\} \times \{1,\dots,n\} : M_{pq} \neq 0\}.$$

Definition 2 Given $n \in \mathbb{N}$ and $k \leq n$, we define a *k-block sequence* in $\mathbb{R}^n$ the set $\{\underline{b_1},\dots,\underline{b_k}\}$, where the blocks $\underline{b_j} \subseteq \{1,\dots,n\}$ $\forall j \in \mathcal{J} := \{1,\dots,k\}$ and $\bigcup_{j\in\mathcal{J}} \underline{b_j} = \{1,\dots,n\}$. We indicate with $d_j = |\underline{b_j}|$ the size of each block. We also assume that no block is a subset of another one and that they are sorted according to the order of their first element.

$$\begin{pmatrix} * & * & 0 & 0 & 0 \\ * & * & * & * & 0 \\ 0 & * & * & * & 0 \\ 0 & * & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}$$

(a) Matrix



(b) Block structure (c) Sparsity graph

Fig. 1: Example of a block-decomposable matrix.
Nonzero entries are symbolized by a star $*$

Definition 3 Let $n, k \leq n \in \mathbb{N}$ and $J = \{1, \ldots, k\}$. Given a k-block sequence, *matrix blocks* are defined as $\underline{B_j} := \underline{b_j} \times \underline{b_j} \subseteq \{1, \ldots, n\}^2$ for all $j \in \mathcal{J}$. We define the set $\underline{\mathscr{B}_\mathcal{J}} := \bigcup_{j \in \mathcal{J}} \underline{B_j}$ as a *block structure* in $\mathbb{R}^{n \times n}$.

Definition 4 The *sparsity graph* of a block structure $\underline{\mathscr{B}_\mathcal{J}}$ in $\mathbb{R}^{n \times n}$ is a graph $G(V, E)$ with $V = \{1, \ldots, n\}$ and with edges on the block structure: $\{p, q\} \in E$ if and only if $(p, q) \in \underline{\mathscr{B}_\mathcal{J}}$.

According to the definition, every subgraph of the sparsity graph induced by the vertices of a block is complete. Hence, the sparsity graph of a block structure is given by the union of cliques. In Figure 1 we show, for a given matrix (a), its block structure (b) and its sparsity graph (c).

Definition 5 Given a matrix $M \in \mathbb{R}^{n \times n}$, an index set $\mathcal{J}$ with $|\mathcal{J}| \leq n$ and a block structure $\underline{\mathscr{B}_\mathcal{J}}$ in $\mathbb{R}^{n \times n}$, we say that M is *block-decomposable under* $\underline{\mathscr{B}_\mathcal{J}}$ if $\mathrm{Supp}(M) \subseteq \underline{\mathscr{B}_\mathcal{J}}$.

In general, several block structures are valid for a single matrix. For instance, every matrix is trivially decomposable in a structure consisting of just one $n \times n$ block. In this case $|\mathcal{J}| = 1$, but this means no decomposition. We are therefore interested in structures with $|\mathcal{J}| > 1$ blocks and potentially we seek a large value of $|\mathcal{J}|$. This means that the nonzero entries of the matrix fit in (possibly overlapping) smaller blocks. A special case is when the blocks are disjoint:

Definition 6 If all the blocks in $\underline{\mathscr{B}_\mathcal{J}}$ have pairwise empty intersection (that is $\{\underline{b_j}\}_{j \in \mathcal{J}}$ is a partition of $\{1, \ldots, n\}$) then the matrix is called *block-separable*.

Definition 7 A quadratic problem of the form (1) is called *block-decomposable* (resp. *block-separable*) if there exists a common block structure $\underline{\mathscr{B}_\mathcal{J}}$ under which all the matrices of the problem (Q and $A_i \; \forall i \in \mathcal{I}$) are block-decomposable (resp. block-separable).

When dealing with block-decomposable binary problems, we are interested in a relaxation of the problem that takes into account only the vertices of the

BQP for each block. Hence, we consider a much smaller number of extreme points than the one related to the problem in the original space. In this respect, our approach deviates significantly from Padberg's study [34] of the facet structure of sparse problems in a more general context.

It remains to show how to write the relaxation taking into account the intersections of the blocks, whether this is still a relaxation for the original problem and if it is equivalent to the one we introduced in Section 2. We will see that the problem of the equivalence is not trivial and it is related to a matrix completion problem.

The theory of matrix completion problems is well developed for semidefinite and completely positive completion. However, the BQP completion problem, to the best of our knowledge, has not been treated in depth, but it can be helpful and interesting, as we will see, because it is strictly related to binary quadratic problems.

In the rest of this section we assume to have a problem of the form (2) and a block structure $\underline{\mathscr{B}_{\mathcal{J}}}$ under which the problem is block-decomposable. We also introduce the following notation in order to split the parameters of the problem into blocks:

Definition 8 For every $j \in \mathcal{J}$ we define:

$$Q^j \in \mathbb{R}^{n\times n} : (Q^j)_{pq} := \begin{cases} Q_{pq} & \text{if } (p,q) \in \underline{B_j} \setminus (\underline{B_1} \cup \cdots \cup \underline{B_{j-1}}) \\ 0 & \text{otherwise,} \end{cases}$$
$$A_i^j \in \mathbb{R}^{n\times n} : (A_i^j)_{pq} := \begin{cases} (A_i)_{pq} & \text{if } (p,q) \in \underline{B_j} \setminus (\underline{B_1} \cup \cdots \cup \underline{B_{j-1}}) \\ 0 & \text{otherwise,} \end{cases}$$

for every $i \in \mathcal{I}$. We also use the following notation for the restriction to the blocks:

$$X^{\underline{B_j}} := X_{|\underline{B_j}} = \{X_{pq} : (p,q) \in \underline{B_j}\} \in \mathbb{R}^{d_j \times d_j} \quad \forall X \in \mathbb{R}^{n\times n},$$
$$x^{\underline{b_j}} := x_{|\underline{b_j}} = \{x_p : p \in \underline{b_j}\} \in \mathbb{R}^{d_j} \quad \forall x \in \mathbb{R}^n.$$

If we introduce variables $Y^j \in \mathbb{R}^{d_j \times d_j}$ for every block $j \in \mathcal{J}$, we can provide a relaxation of the original problem based on the blocks. In the following, with an abuse of notation we consider $Q^j = (Q^j)^{\underline{B_j}} \in \mathbb{R}^{d_j\times d_j}$ and $A_i^j = (A_i^j)^{\underline{B_j}} \in \mathbb{R}^{d_j\times d_j}$. With this notation we have

$$\langle Q, X\rangle = \sum_{j\in\mathcal{J}} \langle Q^j, X^{\underline{B_j}}\rangle, \qquad \langle A_i, X\rangle = \sum_{j\in\mathcal{J}} \langle A_i^j, X^{\underline{B_j}}\rangle, \quad \forall i \in \mathcal{I}.$$

Considering the variables Y^j, we can now write the following problem:

$$
\begin{aligned}
\min\ & \sum_{j\in\mathcal{J}} \langle Q^j, Y^j\rangle && && \text{(9a)}\\
\text{s. t. } & \sum_{j\in\mathcal{J}} \langle A_i^j, Y^j\rangle \le b_i && \forall i \in \mathcal{I} && \text{(9b)}\\
& (Y^j)_{\underline{B_j\cap B_h}} = (Y^h)_{\underline{B_j\cap B_h}} && \forall j,h \in \mathcal{J},\ j<h && \text{(9c)}\\
& Y^j = \sum_{l\in\mathscr{P}^j} \mu_l^j (y_l^j)(y_l^j)^\top && \forall j \in \mathcal{J} && \text{(9d)}\\
& \sum_{l\in\mathscr{P}^j} \mu_l^j = 1 && \forall j \in \mathcal{J} && \text{(9e)}\\
& \mu_l^j \ge 0 && \forall l \in \mathscr{P}^j, \forall j \in \mathcal{J}, && \text{(9f)}
\end{aligned}
$$

where $y_l^j \in \{0,1\}^{d_j}$ $\forall l = 1,\dots,2^{d_j}$, $\forall j \in \mathcal{J}$ are binary vector of the dimension of the corresponding block j. Here $\mathscr{P}^j$ are the index sets of all the possible extreme points $y_l^j \in \{0,1\}^{d_j}$ as $j \in \mathcal{J}$. With this relaxation we allow a convex combination of extreme points for every block, with the additional requirement, given by constraint (9c), that the intersections of blocks must be consistent.

Clearly, if we consider the trivial decomposition in one single n dimensional block, formulation (9) is the same as the one we proposed in (3). Moreover, the index sets $\mathscr{P}^j$, $j \in \mathcal{J}$ are still exponentially large, but their sizes depend on the size of the blocks: $|\mathscr{P}^j| = 2^{d_j}$. Hence, the total number of points is potentially reduced with respect to the one-block formulation, since $\sum_{j\in\mathcal{J}} |\mathscr{P}^j| \ll 2^n$.

Also formulation (9) can be solved via CG with a procedure similar to the one outlined in Section 2.1. The difference with respect to problem (7) is that now the pricing problem decomposes into one problem for each block and those problems are all independent. Furthermore, the size of these problems is the same as the size of the corresponding block, hence they are potentially smaller and therefore easier to solve than a single n-dimensional pricing problem.

4 Comparison to the original BQP relaxation — role of chordality

We now analyze the relations between the BQP relaxation (3) and the BQP block relaxation (9). Our aim is to study conditions that guarantee their equivalence. We will prove that the latter formulation always provides a lower bound for the former one. Moreover, we will see that proving the other implication is a matrix completion problem in BQP, defined in analogy to the PSD and CPP matrix completion problem as treated, e.g., in [4,21,24,28]. We will describe the problem and we will prove this second implication under some conditions. To this aim, we start with the case of only two overlapping blocks, then we will see how the results can be extended to the case of several blocks and analyze the conditions needed on the block structure to guarantee the result. It turns out that a crucial property is chordality of the sparsity graph. The smallest chordal graph relevant in this context is the diamond (K_4 with one edge removed), which gave rise to the title of our study.

4.1 Two overlapping blocks

Here we suppose that all the matrices Q, A_i of the problem are decomposable into two overlapping blocks. Without loss of generality we can assume that each of those two blocks has consecutive components. We introduce some specific notation for the two-block case. Let $\underline{B_j} = \underline{b_j} \times \underline{b_j}$ be the two blocks, $j = 1, 2$. Then let $\underline{B_c} = \underline{b_c} \times \underline{b_c} = \underline{B_1} \cap \underline{B_2}$ the intersection block. We indicate with r the dimension of $\underline{b_c}$; further, let $\underline{b_a} := \underline{b_1} \setminus \underline{b_c}$, let s be its dimension, and let $\underline{B_a} := \underline{b_a} \times \underline{b_a}$; similarly let $\underline{b_d} := \underline{b_2} \setminus \underline{b_c}$, let t be its dimension and let $\underline{B_d} := \underline{b_d} \times \underline{b_d}$. To each $p = 1, \dots, 2^n$ we can assign a triple of indices $\{k, l, m\} : k \in \{1, \dots, 2^{d_s}\}$, $l \in \{1, \dots, 2^{d_r}\}$ and $m \in \{1, \dots, 2^{d_t}\}$, each of them indicates the sub-vector of x_p in the corresponding sub-block. Hence we can replace λ_p and x_p in the one-block formulation with $\lambda_{k,l,m}$, and $x_{k,l,m}$, respectively. Similarly, considering the two-block formulation, to each $p = 1, \dots, 2^{d_1}$ we can assign a pair of indices $\{k, l\} : k \in \{1, \dots, 2^{d_s}\}$, $l \in \{1, \dots, 2^{d_r}\}$, and to each $p = 1, \dots, 2^{d_2}$ we can assign a pair of indices $\{l, m\} : l \in \{1, \dots, 2^{d_r}\}$ and $m \in \{1, \dots, 2^{d_t}\}$. And so, for $p = 1, \dots, 2^{d_1}$ we replace μ^1_p and y^1_p with $\mu_{k,l}$ and $y^1_{k,l}$ respectively, and for $p = 1, \dots, 2^{d_2}$ we replace μ^2_p and y^2_p with $\nu_{l,m}$ and $y^2_{l,m}$, respectively. We note that, given any triple $\{k, l, m\} : k \in \{1, \dots, 2^{d_s}\}$, $l \in \{1, \dots, 2^{d_r}\}$ and $m \in \{1, \dots, 2^{d_t}\}$, the vectors $x_{k,l,m}$, $y^1_{k,l}$, and $y^2_{l,m}$ are linked by the following relations: $(x_{k,l,m})^{\underline{b_1}} = y^1_{k,l}$, $(x_{k,l,m})^{\underline{b_2}} = y^2_{l,m}$, and $(y^1_{k,l})^{\underline{b_c}} = (y^2_{l,m})^{\underline{b_c}} =: y^c_l$.

Proving the equivalence is not a trivial task, even in the case of two blocks only. We start with the following.

Lemma 1 *Let (Y^1, μ, Y^2, ν) be a feasible solution to the block-formulation* (9)*. If*

$$\sum_{k=1}^{2^s} \mu_{k,l} = \sum_{m=1}^{2^t} \nu_{l,m} \tag{10}$$

holds for all $l = 1, \dots, 2^r$, then there exists an equivalent feasible solution (X, λ) to the one-block formulation (3)*, i.e. $X^{\underline{B_1}} = Y^1$ and $X^{\underline{B_2}} = Y^2$.*

Proof Under the given assumptions, we look for coefficients $\lambda_{k,l,m}$ that satisfy:

$$\sum_{m=1}^{2^t} \lambda_{k,l,m} = \mu_{k,l} \quad \forall k,\ \forall l \tag{11a}$$

$$\sum_{k=1}^{2^s} \lambda_{k,l,m} = \nu_{l,m} \quad \forall m,\ \forall l \tag{11b}$$

$$\lambda_{k,l,m} \geq 0 \quad \forall k,\ \forall l,\ \forall m. \tag{11c}$$

Indeed, if this holds, then clearly:

$$\sum_{k=1}^{2^s}\sum_{l=1}^{2^r}\sum_{m=1}^{2^t}\lambda_{k,l,m} = \sum_{k=1}^{2^s}\sum_{l=1}^{2^r}\mu_{k,l} = \sum_{l=1}^{2^r}\sum_{m=1}^{2^t}\nu_{l,m} = 1$$

and

$$X^{\underline{B_1}} = \sum_{k=1}^{2^s}\sum_{l=1}^{2^r}\sum_{m=1}^{2^t}\lambda_{k,l,m}(x_{k,l,m})^{\underline{b_1}}(x_{k,l,m})^{\underline{b_1}^\top} = \sum_{k=1}^{2^s}\sum_{l=1}^{2^r}\mu_{k,l}(y^1_{k,l})(y^1_{k,l})^\top = Y^1$$

$$X^{\underline{B_2}} = \sum_{l=1}^{2^r}\sum_{m=1}^{2^t}\sum_{k=1}^{2^s}\lambda_{k,l,m}(x_{k,l,m})^{\underline{b_2}}(x_{k,l,m})^{\underline{b_2}^\top} = \sum_{l=1}^{2^r}\sum_{m=1}^{2^t}\nu_{l,m}(y^2_{l,m})(y^2_{l,m})^\top = Y^2.$$

So, we just have to show that there exists a feasible solution for (11). But these are the constraints of a transportation problem for each fixed $l = 1, \dots, 2^r$. A classical result is that a transportation problem is feasible if and only if the sum of the right-hand-side of the first set of constraint equals the same sum in the second set of constraints, because both of them equal the global sum $\sum_{k,m}\lambda_{k,l,m}$. But in our case, this equality is exactly (10), so it holds by hypothesis and this concludes the proof. □

Remark 1 We notice that this lemma is also true when $r = 0$. Indeed, in this case both formulations (10) and (11) can be defined without the index l, and moreover, the hypotheses of the lemma are clearly always verified, since $\sum_{k=1}^{2^s}\mu_k = \sum_{m=1}^{2^t}\nu_m = 1$.

The hypotheses of Lemma 1 do not hold in general. The following proposition gives a result with some hypothesis on the size of the intersection.

Proposition 2 *Let (Y^1, μ, Y^2, ν) be a solution feasible to the multiple block formulation* (9)*. If the dimension of the intersection block is $r \le 2$, then there exists an equivalent feasible solution (X, λ) to the one-block formulation* (3)*, i.e. $X^{\underline{B_1}} = Y^1$ and $X^{\underline{B_2}} = Y^2$.*

Proof In this case, constraints (9c) become: $\sum_{k,l}\mu_{k,l}y^c_l y^{c\top}_l = \sum_{l,m}\nu_{l,m}y^c_l y^{c\top}_l$. If all the matrices $y^c_l y^{c\top}_l$ are linearly independent, equality (10) must hold, for each l. Among the matrices $y^c_l y^{c\top}_l$ there is always the null matrix, which is dependent on the others. However, if the nonzero matrices are linearly independent, that is, all of the matrices are affinely independent, condition (10) holds for all l such that $y^c_l y^{c\top}_l$ is nonzero. And since the sum of all coefficients is always 1, then by difference it holds also for the null matrix. If $r = 1$ or $r = 2$, it is easy to see that the matrices are affinely independent, so (10) holds, hence the result is true. Thanks to Remark 1, this result is proved also for $r = 0$. □

This result holds with $r \le 2$, but if $r > 2$, the number of $y^c_l y^{c\top}_l$ matrices is 2^r, thus exceeding the dimension of the space $r(r+1)/2$ plus 1. Hence, they are affinely dependent and condition (10) cannot be directly obtained. In order to better understand the equivalence statement, and the conditions that guarantee it, in the following section we consider the multiple block case.

$$\frac{1}{2}\begin{pmatrix} 1 & 1 & ? & \dots & ? & 0 \\ 1 & 1 & \ddots & \ddots & \ddots & ? \\ ? & 1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & ? \\ ? & \ddots & \ddots & \ddots & 1 & 1 \\ 0 & ? & \dots & ? & 1 & 1 \end{pmatrix}$$

Fig. 2: A not BQP completable matrix.

4.2 Multiple-block case and BQP completion

In the previous subsection we proved that, whenever we have a specific 2-block-decomposable problem (i.e., with intersection size $r \leq 2$), for any solution of the block relaxation (9) there is an equivalent solution of the BQP relaxation (3). In this section we assume that our problem is decomposable with respect to a more general block structure $\underline{\mathscr{B}_{\mathcal{J}}}$. The following proposition states that the inverse inclusion is always true: given any feasible solution of (3), an equivalent solution of (9) can be obtained. This means that multiple-block relaxation always provides a valid lower bound for the one-block relaxation.

Proposition 3 *Given a block structure $\underline{\mathscr{B}_{\mathcal{J}}}$, suppose that problem* (3) *is $\underline{\mathscr{B}_{\mathcal{J}}}$-decomposable. Given any feasible point for problem* (3)*, i.e. a feasible matrix X and the corresponding coefficients λ_p, $p = 1, \dots, 2^n$, then there exists a solution of* (9)*, given by Y^j and μ_l^j with $l = 1, \dots, 2^{d_j}$, $j \in \mathcal{J}$, such that $X^{\underline{B_j}} = Y^j$ $\forall j \in \mathcal{J}$.*

Proof We are given a matrix X and coefficients $\lambda_p \geq 0$ $\forall p$, $\sum_{p=1}^{2^n} \lambda_p = 1$, such that $X = \sum_{p=1}^{2^n} \lambda_p (x_p x_p^\top)$ with $x_p \in \{0,1\}^n$. We introduce the following notation. For every $j \in \mathcal{J}$, let $\bar{\underline{b_j}} = \{1, \dots, n\} \setminus \underline{b_j}$ the complement of the block $\underline{b_j}$ in $\{1, \dots, n\}$. To each $p = 1, \dots 2^n$ we can assign a couple of indices $\{l, m\}$: $l \in \{1, \dots 2^{d_j}\}$, $m \in \{1, \dots 2^{n-d_j}\}$. Let $y_l := (x_p)^{\underline{b_j}}$ and $z_m := (x_p)^{\bar{\underline{b_j}}}$ be the restrictions of x_p to $\underline{b_j}$ and $\bar{\underline{b_j}}$. We can hence rename x_p as $x_{l,m}$, λ_p as $\lambda_{l,m}$ and write: $X = \sum_{p=1}^{2^n} \lambda_p (x_p x_p^\top) = \sum_{l=1}^{2^{d_j}} \sum_{m=1}^{2^{n-d_j}} \lambda_{l,m} (x_{l,m}\, x_{l,m}^\top) \quad \forall j \in \mathcal{J}$.
Hence, for all $j \in \mathcal{J}$ we can define:

$$\mu_l^j := \sum_{m=1}^{2^{n-d_j}} \lambda_{l,m} \quad \forall l = 1, \dots, 2^{d_j}. \tag{12}$$

Clearly, $\mu_l^j \geq 0$ and $\sum_{l=1}^{2^{d_j}} \mu_l^j = \sum_{l=1}^{2^{d_j}} \sum_{m=1}^{2^{n-d_j}} \lambda_{l,m} = \sum_{p=1}^{2^n} \lambda_p = 1$. Moreover, for all $j \in \mathcal{J}$, it holds:

$$X^{\underline{B_j}} = \sum_{l=1}^{2^{d_j}} \sum_{m=1}^{2^{n-d_j}} \lambda_{l,m} (x_{l,m}\, x_{l,m}^\top)^{\underline{B_j}} = \sum_{l=1}^{2^{d_j}} (\sum_{m=1}^{2^{n-d_j}} \lambda_{l,m}) (y_l y_l^\top) = \sum_{l=1}^{2^{d_j}} \mu_l^j (y_l y_l^\top) \,.$$

So there is a feasible point for (9), equivalent to the solution of (3), where μ_l^j are given by (12) and Y^j are defined as: $Y^j := \sum_{l=1}^{2^{d_j}} \mu_l^j (y_l y_l^\top)$ as $j \in \mathcal{J}$. □

Hence, in the 2-blocks case with intersection size $r \leq 2$, we get the equivalence of the two formulations by combining Propositions 2 and 3.

We now try to extend the result of Proposition 2 to the case of general blocks. However, we show that it does not hold for any block structure and we notice that it can be stated in terms of a BQP matrix completion problem. For a detailed description of completion problems (specifically PSD and CPP completion) we refer to [4]. The first result we report resembles similar ones for PSD and CPP completion problems; it shows that not all the specification graphs are BQP completable (the proof is based on the example given in Figure 2).

Proposition 4 *If a graph is not chordal, then it is not BQP completable.*

Proof If the graph is not chordal, it contains a cycle of length $l \geq 4$ with no chords. Without loss of generality we suppose that the vertices of this cycle are the first ones. Any matrix with this specification graph, restricted to the first l entries, would be specified in the following entries: $\{i, i+1\}$ $\forall i = 1, \ldots, l-1$ and $\{1, l\}$ (and the symmetric elements, of course). This means that the block structure, restricted to these entries, is made up of l consecutive 2×2 blocks on the diagonal, and one 2×2 block connecting the first and the last entry of the cycle. Hence, we can always have the matrix shown in Figure 2 [4, Example 1.35], where the question marks correspond to unspecified elements. Indeed, every 2×2 matrix on the diagonal is given by $\frac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ and the 2×2 matrix restricted to elements $\{1, l\}$ is given by $\frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$ so they are convex combinations of 2×2 binary matrices of rank one. But it is known that the only PSD matrix with ones on the entries (i, j) s.t. $|i - j| \leq 1$ is the all-1 matrix [28, Lemma 6], hence the matrix in Figure 2 is not PSD completable. Since BQP is a subset of the semidefinite cone, this matrix is neither BQP completable. □

However, due to Proposition 2 and the properties of chordal graphs, we can state the following result:

Proposition 5 *If a graph G is chordal and the size of the intersection of any two maximal cliques of G is at most 2, then G is BQP completable.*

Proof We prove the statement by induction on the number n of maximal cliques. If there are only two maximal cliques, the result is given by Proposition 2. Now we suppose to have $n > 2$ maximal cliques and we assume by inductive hypothesis that the result holds true for $n - 1$ cliques. Since the graph G is chordal, there is a perfect elimination ordering (PEO) of its vertices [28]. Without loss of generality, we suppose that the vertices of G are sorted according to this ordering. We also sort the maximal cliques according to the order of their first vertex. We recall that, by definition of PEO, each vertex, together with its neighbours which follow it in the order, form a clique. In particular, all the neighbours of the first vertex belong to the first maximal clique C_1. We now consider the subgraph G' of G induced by the last $n-1$ maximal cliques $C_2 \cup \cdots \cup C_n$. By inductive hypothesis G' is BQP completable. Hence G is completable if G'' is completable, where G'' is obtained by adding to G all the edges which complete G'. In this way we now have two cliques: C_1 and $C_2 \cup \cdots \cup C_n$. We notice that $C_1 \cap (C_2 \cup \cdots \cup C_n) = C_1 \cap C_2$, again for the PEO property. Hence, if $|C_1 \cap C_2| \leq 2$, we can apply again Proposition 2 and conclude the proof. □

Hence by Proposition 3 we have the following corollary:

Corollary 1 *If a problem is decomposable under a block structure whose sparsity graph G is chordal and the size of the intersection of any two maximal cliques of G is at most 2, then Formulations* (3) *and* (9) *are equivalent.*

The crucial property is that any two cliques must have intersection size $r \leq 2$, that is that the sparsity graph restricted to them either is not connected ($r = 0$), or has a cut edge (in this case $r = 1$ and the graph is block-clique), or has an induced diamond graph ($r = 2$).

We already observed that $BQP^n \subset \mathcal{C}^* \subset \mathcal{S}_+$. However, since the CPP completable graphs are block-clique graphs, and the PSD completable graphs are chordal graphs, we have the following:

Corollary 2 *The class of BQP completable graphs is strictly larger than the class of CPP completable graphs and contained in that of PSD completable graphs.*

As a consequence, it is natural to investigate what can happen for more general chordal graphs. We propose the following conjecture:

Proposition 6 (Conjecture) *Chordal graphs are BQP completable.*

By combining this result with Proposition 4, we would get that a graph is BQP completable if and only if it is chordal. To the best of our knowledge, the conjecture stated above has not been studied yet. However, this result could be very useful, since it could allow to efficiently tackle specific sparse problems and get stronger bounds than SDP ones. Our preliminary experiments seem to support the conjecture; see below.

In order to prove the conjecture, we can restrict ourselves to a simpler two-block case, with both of the blocks having dimension $n-1$, and only the elements $\{1, n\}$ not covered by the blocks. This means that the dimension of the block intersection is $n-2$ and the dimension of both s and t (see notation in Section 4.1) is 1. This result is obtained by following the proof of the completion problem in [28]: the authors strongly exploit the structure and the properties of chordal graphs, showing that they can reduce to two blocks with only one missing entry. Under these assumptions they prove their result for PSD completion and they prove that they can extend it to every chordal graph, just using the properties coming from the structure of these graphs. We could try to prove the same result for BQP matrices, but it is not straightforward. We also notice that this result does not hold if the extreme matrices are general real nonnegative rank-1 matrices (this is the case of CPP completion, which only holds for block-clique graphs).

4.3 Computational experiments with the BQP block relaxation

From the computational point of view, formulation (9) might be better than formulation (3). The most important pros are the reduced number of extreme points, and the smaller size of the pricing problems (solving those problems was the most time consuming part in the one-block formulation). However, due to the high number of constraints related to the intersections between blocks, the master program gets more difficult (although still linear).

In our implementation of the CG for solving the BQP multiple-block relaxation, we again used dummy columns for the initialization of the master problem and an early stopping technique when solving it. Furthermore, in order to deal with the master problem, we implemented a *purging* technique for the columns: more precisely, every ten iterations, we remove 50% of the columns which have a positive reduced cost for more than ten iterations.

In general, the block structure might not be unique. Therefore, one important issue is how to build a block structure that fits the problem. We shall note that the (aggregate) sparsity graph of the problem is not chordal in general. To obtain such a property we need to add edges, resulting in what is called *chordal extension* of the graph (see [24] for an application of chordal extension to SDP). Computing the minimal (in terms of additional edges) chordal extension is a NP-hard problem, and heuristics are proposed to find a minimal extension quickly [24, 36]. In our preliminary experiments we used the heuristic proposed in [36] to obtain (almost) minimal chordal extensions to solve the QPLIB instances presented in Section 2.2. The computing times for the BQP block relaxation based on such chordal extension were always higher than the ones obtained from the single block BQP relaxation. The results seem to indicate that the minimal chordal extension is not always the one leading to the easiest possible block decomposition. Finding the best decomposition for a generic formulation is an extremely challenging task (see [3] for an application of automatic detection of decompositions for standard Dantzig-Wolfe Reformulation) and it goes beyond the scope of this paper. This is the reason why we decided to focus our computational experiments on a class of instances where the block decomposition is straightforward. We hence compared the performance obtained using formulation (9) and formulation (3) on the SONET instances. We consider a formulation of the SONET instances that has been introduced in [10]; some of these instances are also included in the QPLIB. Those instances have a block-diagonal quadratic objective function. Such a structure allows to easily exploit the block decomposition. In Table 2, we report the results of `BiqCrunch` with triangle inequalities (BC-cuts) and

Instance		BC-cuts		BQP				
				Single block		Multiple block		
Name	#	T (s)	Gap (%)	# Fails	T (s)	# Fails	T (s)	Gap (%)
ins.16	5	89	186	0	304	0	0.26	85
ins.17	5	171	205	1	7210	0	0.28	68
ins.18	5	74	232	0	1402	0	0.46	89
ins.19	5	53	274	2	15215	0	0.44	90
ins.20	5	67	175	4	28900	0	0.42	84
ins.21	5	85	340	4	28826	0	0.70	80
ins.22	5	109	397	2	14645	0	0.92	76
ins.23	3	139	212	2	32570	0	0.43	91
ins.24	5	171	> 1000	3	22156	0	1.34	88
ins.25	5	204	384	2	14542	0	1.52	78
average	5	116	362	2	16577	0	0.68	83

Table 2: Performance comparison on the SONET instances.

of the CG approach based on the single and multiple-block BQP relaxation (BQP, single or multiple block). We show the computational time to solve the instances and the final gap obtained. We also compared the number of failures between the single and multiple block formulations. For each size of instance, each line presents the number of instances and the average results. The table shows clear evidence that our approach is extremely good: thanks to the block decomposition we are able to obtain (in a very short time) a gap that is more than four times smaller than the one obtained using the SDP relaxation with triangle inequalities. Furthermore, with the multiple block formulation all the instances are solved, while the single block formulation reaches the time limit (10h) or the maximum number of iterations (10^5) in 20 out of 48 instances. We notice that the overall number of iterations to solve the relaxation is dramatically reduced with the multiple block formulation: indeed, on average just 10 iterations are needed with Formulation (9), while the single block relaxation generates more than 40000 iterations on average.

In order to better understand the difficulty caused by the intersections between blocks, we compared the two BQP formulations in instances with increasing intersection between blocks. More specifically, for each of the block-separable SONET instances, we increased the size of each block by a number r varying from 1 to 10 in such a way that every block has r elements in common with (at least) one of the subsequent blocks. We thus added randomly generated elements in the objective function to fill the increased blocks and solved each instance with the two BQP algorithms. We show the results with the plot in Figure 3: for every instance, with size r from 0 (the original instances) to 10, we show the ratio (in logarithmic scale) between the computational time of the multiple block and of the single block formulation to solve it. We notice that only one instance has not been solved by either algorithm, and does not appear in the plot; for every other case, we used the timelimit (10h) as value for every failure.

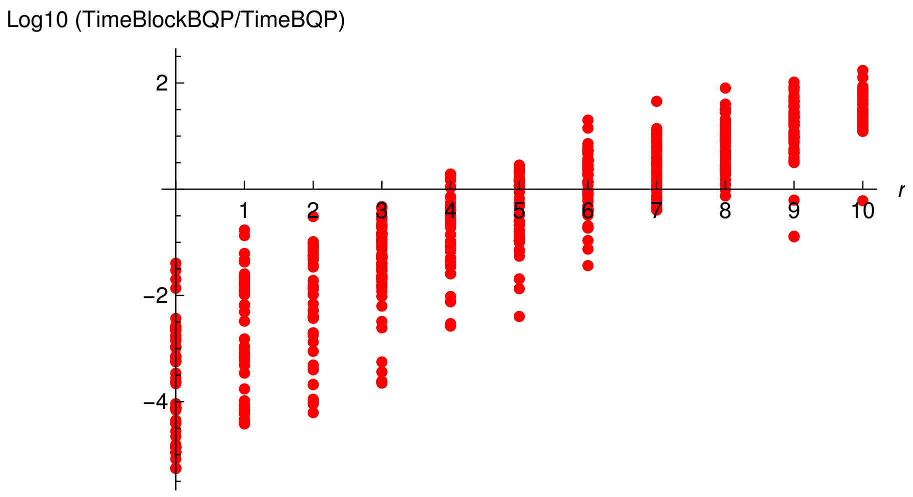


Fig. 3: Performance of block BQP on problems with different intersection size.

If the intersection between blocks is small, then the multiple block formulation is more efficient than the single block one: for separable problems it was already clear from Table 2, but Figure 3 shows that it is true even with some values of $r > 0$. However, if the intersection size grows, then the block formulation becomes heavier, because the master has more constraints, the pricing problems are larger, and more extreme points can be generated, so the number of iterations also grows. Hence, if $r \geq 6$, the single block formulation is preferable in a significant number of cases, at least in our examples.

We further notice that, for every instance solved by both algorithms, the objective function optimal values are equal, even with $r > 2$: this observation, given by around 500 instances with some random elements, may corroborate our conjecture.

5 Conclusions

In this work, we presented two relaxations for BQCQP that are stronger than the widely used SDP relaxation. We carefully analyzed the relation between the proposed relaxations and discussed the equivalence of the two formulations. We showed that the BQP block relaxation is always a relaxation of the BQP relaxation (while it is still unclear whether equivalence holds in general). By using the connection with a matrix completion problem, we identified problem classes in which this equivalence is ensured, and other classes where this is not guaranteed. We proved that the BQP completion is possible for problems with chordal graphs (if the maximal dimension of the intersection of blocks is 2), and we also proved that if the graph is not chordal, completion does not hold. A complete proof of the BQP completion result is still open. From a computational point of view, since both problems are exponential in size, we proposed a column generation algorithm to get a solution in reasonable time. The reported results show that the proposed formulations provide significantly tighter bounds than the ones obtained using the SDP relaxation in a reasonable amount of time. Moreover, if the problems have a block structure, the decomposition can be solved very quickly.

References

1. Barahona, F., Mahjoub, A.R.: On the cut polytope. Mathematical Programming **36**(2), 157–173 (1986)
2. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W., Vance, P.H.: Branch-and-price: Column generation for solving huge integer programs. Operations Research **46**(3), 316–329 (1998)
3. Bergner, M., Caprara, A., Ceselli, A., Furini, F., Lübbecke, M.E., Malaguti, E., Traversi, E.: Automatic Dantzig–Wolfe reformulation of mixed integer programs. Mathematical Programming **149**(1-2), 391–424 (2015)
4. Berman, A., Shaked-Monderer, N.: Completely positive matrices. World Scientific (2003)
5. Bomze, I.M.: Copositive optimization–recent developments and applications. European Journal of Operational Research **216**(3), 509–520 (2012)
6. Bomze, I.M., De Klerk, E.: Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. Journal of Global Optimization **24**(2), 163–185 (2002)
7. Bomze, I.M., Dür, M., De Klerk, E., Roos, C., Quist, A.J., Terlaky, T.: On copositive programming and standard quadratic optimization problems. Journal of Global Optimization **18**(4), 301–320 (2000)
8. Bomze, I.M., Jarre, F., Rendl, F.: Quadratic factorization heuristics for copositive programming. Mathematical Programming Computation **3**(1), 37–57 (2011)
9. Bomze, I.M., Schachinger, W., Uchida, G.: Think co(mpletely)positive! matrix properties, examples and a clustered bibliography on copositive optimization. Journal of Global Optimization **52**(3), 423–445 (2012)
10. Bonami, P., Nguyen, V.H., Klein, M., Minoux, M.: On the solution of a graph partitioning problem under capacity constraints. In: International Symposium on Combinatorial Optimization, pp. 285–296. Springer (2012)
11. Borchers, B.: CSDP, a C library for semidefinite programming. Optimization Methods and Software **11**(1-4), 613–623 (1999)
12. Bundfuss, S., Dür, M.: An adaptive linear approximation algorithm for copositive programs. SIAM Journal on Optimization **20**(1), 30–53 (2009)
13. Burer, S.: On the copositive representation of binary and continuous nonconvex quadratic programs. Mathematical Programming **120**(2), 479–495 (2009)
14. Burer, S.: Optimizing a polyhedral-semidefinite relaxation of completely positive programs. Mathematical Programming Computation **2**(1), 1–19 (2010)
15. Burer, S.: Copositive programming. In: Handbook on semidefinite, conic and polynomial optimization, pp. 201–218. Springer (2012)
16. Burer, S., Letchford, A.N.: Non-convex mixed-integer nonlinear programming: A survey. Surveys in Operations Research and Management Science **17**(2), 97–106 (2012)
17. Dantzig, G.B., Wolfe, P.: Decomposition principle for linear programs. Operations Research **8**(1), 101–111 (1960)
18. Desaulniers, G., Desrosiers, J., Solomon, M.M. (eds.): Column generation. Springer, US (2005)
19. Dickinson, P.J.: An improved characterisation of the interior of the completely positive cone. Electron. J. Linear Algebra **20**, 723–729 (2010)
20. Dickinson, P.J.: Geometry of the copositive and completely positive cones. Journal of Mathematical Analysis and Applications **380**(1), 377–395 (2011)

21. Drew, J.H., Johnson, C.R.: The completely positive and doubly nonnegative completion problems. Linear and Multilinear Algebra **44**(1), 85–92 (1998)
22. Dür, M.: Copositive programming–a survey. In: Recent advances in optimization and its applications in engineering, pp. 3–20. Springer (2010)
23. Dür, M., Still, G.: Interior points of the completely positive cone. Electron. J. Linear Algebra **17**, 48–53 (2008)
24. Fukuda, M., Kojima, M., Murota, K., Nakata, K.: Exploiting sparsity in semidefinite programming via matrix completion I: General framework. SIAM Journal on Optimization **11**(3), 647–674 (2001)
25. Furini, F., Traversi, E., Belotti, P., Frangioni, A., Gleixner, A., Gould, N., Liberti, L., Lodi, A., Misener, R., Mittelmann, H., Sahinidis, N., Vigerske, S., Wiegele, A.: QPLIB: a library of quadratic programming instances. Mathematical Programming Computation **11**(2), 237–265 (2019)
26. Gould, N.I., Toint, P.L.: A quadratic programming bibliography. Numerical Analysis Group Internal Report **1**, 32 (2000)
27. Gouveia, J., Pong, T.K., Saee, M.: Inner approximating the completely positive cone via the cone of scaled diagonally dominant matrices. Journal of Global Optimization **76**(2), 383–405 (2020)
28. Grone, R., Johnson, C.R., Sá, E.M., Wolkowicz, H.: Positive definite completions of partial hermitian matrices. Linear Algebra and its Applications **58**, 109–124 (1984)
29. Hall, M., Newman, M.: Copositive and completely positive quadratic forms. In: Mathematical Proceedings of the Cambridge Philosophical Society, vol. 59, pp. 329–339. Cambridge University Press (1963)
30. Helmberg, C.: Semidefinite programming. European Journal of Operational Research **137**(3), 461–482 (2002)
31. Krislock, N., Malick, J., Roupin, F.: Biqcrunch: A semidefinite branch-and-bound method for solving binary quadratic problems. ACM Transactions on Mathematical Software (TOMS) **43**(4), 32 (2017)
32. Laurent, M., Rendl, F.: Semidefinite programming and integer programming. Handbooks in Operations Research and Management Science **12**, 393–514 (2005)
33. Nemhauser, G.L.: Column generation for linear and integer programming. Optimization Stories **20**, 64 (2012)
34. Padberg, M.: The Boolean Quadric Polytope: some characteristics, facets and relatives. Mathematical Programming **45**(1-3), 139–172 (1989)
35. Rendl, F.: Semidefinite relaxations for integer programming. In: 50 Years of Integer Programming 1958-2008, pp. 687–726. Springer (2010)
36. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. SIAM Journal on computing **5**(2), 266–283 (1976)
37. Roupin, F.: From linear to semidefinite programming: an algorithm to obtain semidefinite relaxations for bivalent quadratic problems. Journal of Combinatorial Optimization **8**(4), 469–493 (2004)
38. Sturm, J.F.: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optimization Methods and Software **11**(1-4), 625–653 (1999)
39. Tütüncü, R.H., Toh, K.C., Todd, M.J.: Solving semidefinite-quadratic-linear programs using sdpt3. Mathematical programming **95**(2), 189–217 (2003)
40. Vanderbeck, F.: On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Operations Research **48**(1), 111–128 (2000)
41. Yıldırım, E.A.: Inner approximations of completely positive reformulations of mixed binary quadratic programs: a unified analysis. Optimization Methods and Software **32**(6), 1163–1186 (2017)