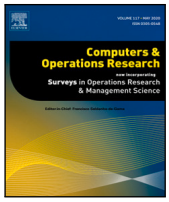
Computers & Operations Research 142 (2022) 105735

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor



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

Enrico Bettiol [a], Immanuel Bomze [b,*], Lucas Létocart [c], Francesco Rinaldi [d], Emiliano Traversi [c]

[a] Fakultät für Mathematik, TU Dortmund, Germany
[b] ISOR, VCOR & ds:UniVie, Universität Wien, Austria
[c] LIPN, UMR CNRS 7030, Université Sorbonne Paris Nord, France
[d] Dipartimento di Matematica, Università degli Studi di Padova, Italy

ARTICLE INFO

MSC:
90C09
90C20
90C26
49M27

Keywords:
Binary quadratic problem
Dantzig–Wolfe Reformulation
Boolean Quadric Polytope
Sparse problem

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 prove that this equivalence does not hold if the sparsity graph is not chordal and 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.

1. Introduction and literature review

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

$$\min \; x^\top Q x \tag{1a}$$

$$\text{s. t. } x^\top A_i x \le b_i \qquad \forall i \in \mathcal{I} \tag{1b}$$

$$x \in \{0,1\}^n, \tag{1c}$$

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. Linear terms $c^\top x$ can be integrated into objective and constraints by exploiting $x_i^2 = x_i$ from (1c).

BQCQPs play an important role in the optimization domain. They have applications in many different fields, such as Telecommunications, Finance, Biology, Energy, Robotics, just to cite a few of them (see Furini et al., 2019, Gould and Toint, 2000 for more details). The solution of BQCQPs is NP-hard and the optimization procedures for BQCQPs draw heavily on both discrete and continuous theory and methodologies. For a detailed description of solution techniques for nonlinear, especially quadratic, problems we refer the reader to Burer and Letchford (2012) and Furini et al. (2019).

The methods 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 Dantzig and Wolfe (1960), is a well-known technique used to obtain tight bounds for discrete problems. Its principle is to replace the feasible 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 techniques are frequently used for solving problems with a large (usually exponential) number of columns. Typically, CG is used for handling linear optimization problems, but it can be applied to any nonlinear optimization problem, 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 efficiently. Among various articles and surveys on CG and DWR available in the literature, we point out (Barnhart et al., 1998; Desaulniers et al., 2005; Nemhauser, 2012) and (Vanderbeck, 2000).

* Corresponding author.
E-mail address: immanuel.bomze@univie.ac.at (I. Bomze).

https://doi.org/10.1016/j.cor.2022.105735
Received 26 April 2021; Received in revised form 3 December 2021; Accepted 26 January 2022
Available online 12 February 2022

In order to rewrite the BQCQP (1) in matrix form, we make use of the Hilbert product $\langle A, B\rangle = \text{trace}(AB)$ and introduce the matrix variable X to represent all products of the original variables: $X_{ij} = x_i x_j$, for all $\{i, j\} \subseteq [1 : n]$. Here and in the sequel, for two integers a, b, we denote by $[a : b]$ the set of all integers k with $a \le k \le b$.

Now we can equivalently rewrite the original problem as follows:

$$\min \ \langle Q, X\rangle \tag{2a}$$

$$\text{s. t. } \langle A_i, X\rangle \le b_i \qquad \forall i \in \mathcal{I} \tag{2b}$$

$$X = xx^\top \tag{2c}$$

$$x \in \{0, 1\}^n. \tag{2d}$$

Several conic reformulations for quadratic optimization problems are formulated on the extended (matrix) space, obtained after adding additional 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 powerful tool in this context. SDP is particularly interesting because it enjoys polynomial complexity and usually leads to relatively tight dual bounds (see, e.g., Rendl, 2010). We refer the reader to Helmberg (2002) and Laurent and Rendl (2005) for a survey on SDP. Specific software has been developed for SDP, for example the commercial solver `MOSEK`, or `SeDuMi` (Sturm, 1999), `SDPT3` (Tütüncü et al., 2003), or `CSDP` (Borchers, 1999), among others. The open-source software `BiqCrunch` is a SDP-based solver for binary quadratic problems (Roupin, 2004). In this work we will present a new relaxation of problem (1) yielding bounds even tighter than those provided by SDP.

SDP-based techniques are also used in polynomial optimization problems, where objective function and constraints can be polynomials. In this context, hierarchies of semidefinite relaxations (see Lasserre, 2006) are used to provide bounds for the original problem. However, these approaches are computationally heavy, and often some sparsity of the problem should be exploited. Interesting results in this sense are given by Bienstock and Muñoz (2018) and Wainwright and Jordan (2004) where the authors obtain more efficient reformulations of the problem by exploiting the concept of *treewidth* of the graphs. However, these results can be exploited under strong assumptions on the sparsity of the problem, and in this work we provide a reformulation for BQCQPs under more general sparsity conditions. In Ahmadi et al. (2017), the authors make use of CG to solve approximations of the SDP relaxations of polynomial optimization problems and of a discrete optimization problem, although without sparsity exploitation.

Among other types of conic relaxations, two important classes are represented by the copositive and completely positive relaxations, that lead to optimization over the mutually dual cones of *Copositive matrices* and *Completely positive matrices*, respectively; see Hall and Newman (1963) for an early account. Over the last few decades, optimization over these cones has received increasing interest and a large number of works has been produced. For a description of copositive optimization, we start referring to Burer (2012) and to the survey (Dür, 2010). Other papers which present reviews of recent advances in this field, along with applications, are Bomze (2012) and Bomze et al. (2012). The book (Berman and Shaked-Monderer, 2003) is also a useful source. A very recent account with an emphasis on binary QPs is (Dür and Rendl, 2021).

The first important result about copositive optimization is given in Bomze et al. (2000): the authors consider the so-called *Standard Quadratic Problem*, which amounts to minimization of a quadratic function over a simplex, and they prove that its completely positive relaxation and its copositive dual is an exact reformulation of the original problem.

This first reformulation result of a quadratic optimization problem in copositive form was then significantly extended, under some mild additional assumptions, to every linearly constrained quadratic problem with binary variables in Burer (2009), and soon followed by other results.

Solving a copositive problem remains quite hard. Typically, hierarchies of polyhedral cones are used to approximate the copositive cone. These are used, for instance, in Bomze and De Klerk (2002) and in Bundfuss and Dür (2009), while in Burer (2010) the author proposes a doubly non-negative relaxation of the completely positive cone. A different approach is given in Bomze et al. (2011), 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 Dür and Still (2008) and their result has been improved in Dickinson (2010). Other important results are obtained in Dickinson (2011).

Outline. We now briefly outline the structure and the contributions of this work. We first introduce a formulation for binary QCQPs that provides a bound provably stricter than the SDP bound; then we propose a column generation method to compute this bound, and we demonstrate significant improvement w.r.t. the SDP bound on some benchmark instances; this is done in Section 2. Furthermore, in Section 3, we provide an ad-hoc relaxation for sparse problems that turns out to be computationally very efficient when embedded in our column generation method. Moreover, in Section 4, we show the theoretical relationship between these relaxations: we prove that the two formulations are equivalent for what we call "diamond-block-clique" sparsity graphs, thus extending the results from the literature. The existing results were indeed limited – to the best of our knowledge – to block-clique sparsity graphs. In addition, we prove that this equivalence does not hold for non-chordal graphs, and we conjecture that it holds for all chordal graphs; we support our conjecture with theoretical and numerical results. The final Section 5 presents preliminary but encouraging computational results which show that the proposed method is effective to efficiently solve sparsely structured problems.

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:

$$\min \ \langle Q, X\rangle \tag{3a}$$

$$\text{s. t. } \langle A_i, X\rangle \le b_i \qquad \forall i \in \mathcal{I} \tag{3b}$$

$$X = \sum_{p\in\mathcal{P}} \lambda_p X_p \tag{3c}$$

$$\sum_{p\in\mathcal{P}} \lambda_p = 1 \tag{3d}$$

$$\lambda_p \ge 0 \qquad \forall p \in \mathcal{P}, \tag{3e}$$

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

It is interesting to investigate the relation between problem (3) and the well-known Boolean Quadric Polytope (Padberg, 1989), the convex hull BQP^n of all binary rank-1 matrices $X_p \in \mathbb{R}^{n\times n}$.

Proposition 2.1. *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 $\lambda \in \{0,1\}^{|\mathcal{P}|}$, 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 $A \in S^n$. We say that A is *COPositive* (COP) if $x^\top Ax \geq 0$ 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 C^*. Indeed, if we define $b_p := \sqrt{\lambda_p}\, x_p$, $p \in \mathcal{P}$, and $B = [b_1, \dots, b_k] \in \mathbb{R}^{n\times k}_+$ with $k = |\mathcal{P}|$, then (3c) becomes $X = \sum_{p\in\mathcal{P}} b_p b_p^\top = BB^\top$, showing $X \in C^*$. Hence the bounded polyhedron BQP^n is strictly contained in the non-polyhedral cone 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 (Gouveia et al., 2020; Yıldırım, 2017). While Yıldırım (2017) offers theoretical error bounds for a general setting of discretization of the CPP cone (or a base of it), Gouveia et al. (2020) 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 specification graphs with an induced diamond graph (see Section 4).

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 $\bar{\mathcal{P}} \subset \mathcal{P}$ 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 $\mathcal{P} \setminus \bar{\mathcal{P}}$.

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

$$\min \ \langle Q, X\rangle \tag{4a}$$
$$\text{s. t. } \langle A_i, X\rangle \leq b_i \qquad \forall i \in \mathcal{I} \tag{4b}$$
$$X = \sum_{p\in\bar{\mathcal{P}}} \lambda_p \left(x_p x_p^\top\right) \tag{4c}$$
$$\sum_{p\in\bar{\mathcal{P}}} \lambda_p = 1 \tag{4d}$$
$$\lambda_p \geq 0 \qquad \forall p \in \bar{\mathcal{P}}\,. \tag{4e}$$

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:

$$\text{RMP}(\bar{\mathcal{P}}) \qquad \min \ \sum_{p\in\bar{\mathcal{P}}} \langle Q, X_p\rangle \lambda_p \tag{5a}$$
$$\text{s. t. } \sum_{p\in\bar{\mathcal{P}}} \langle A_i, X_p\rangle \lambda_p \leq b_i \qquad \forall i \in \mathcal{I} \quad [\rho] \tag{5b}$$
$$\sum_{p\in\bar{\mathcal{P}}} \lambda_p = 1 \qquad [\pi_0] \tag{5c}$$
$$\lambda_p \geq 0 \qquad \forall p \in \bar{\mathcal{P}}\,. \tag{5d}$$

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

$$\max \ b^\top \rho + \pi_0 \tag{6a}$$
$$\text{s. t. } \sum_{i\in\mathcal{I}} \langle A_i, X_p\rangle \rho_i + \pi_0 \leq \langle Q, X_p\rangle \qquad \forall p \in \bar{\mathcal{P}} \tag{6b}$$
$$\rho \leq 0\,, \tag{6c}$$

where $\rho \in \mathbb{R}^{|\mathcal{I}|}$ are the dual variables corresponding to the constraints (5b) 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($\bar{\mathcal{P}}$) 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 across $p \in \mathcal{P} \setminus \bar{\mathcal{P}}$. If these constraints are valid for every $p \in \mathcal{P}$, then our primal feasible solution is also feasible for the dual of the master problem, so it is optimal. Otherwise, there are $p \in \mathcal{P} \setminus \bar{\mathcal{P}}$ where X_p 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_p as $p \in \mathcal{P}$. 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:

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

and can be rewritten in vector form:

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

While the reduced master problem RMP($\bar{\mathcal{P}}$) is linear, the pricing problem is an unconstrained binary quadratic problem. 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, we used `Cplex` (IBM, 2017), version 12.8, as solver for both the master and the pricing problems. We run all our comparison tests in single thread on a Intel Xeon CPU E5-2650 v3 2.30 GHz processor with 64 GB RAM.

From a technical point of view, to ensure feasibility in the first iterations of the algorithm, we initialize RMP($\bar{\mathcal{P}}$) 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

Table 1
Gap and time, QPLIB instances.

Instance	Density	BC-bound		BC-cuts		BQP	
		T (s)	Gap (%)	T (s)	Gap (%)	T (s)	Gap (%)
QPLIB-0067	0.88	0	5	22	2	0	1
QPLIB-1976	0.07	27	433	193	371	7	368
QPLIB-2017	0.05	113	441	114	441	124	240
QPLIB-2029	0.05	180	562	180	562	1865	192
QPLIB-2036	0.05	220	740	220	740	185	313
QPLIB-2055	0.21	21	41	104	35	92	32
QPLIB-2060	0.20	36	42	655	33	153	32
QPLIB-2067	0.19	72	68	149	65	242	62
QPLIB-2073	0.18	57	18	1078	10	285	10
QPLIB-2085	0.16	85	33	2642	23	1066	23
QPLIB-2087	0.16	123	71	172	71	2935	56
QPLIB-2096	0.15	82	18	2679	11	1210	11
QPLIB-2357	0.08	16	13	46	0	3223	0
QPLIB-2359	0.04	74	11	54	2	2888	0
QPLIB-2512	0.77	2	428	117	120	6	100
QPLIB-2733	0.89	10	762	1006	178	19 258	154
QPLIB-2957	0.60	78	>1000	2392	357	11 261	100
QPLIB-3307	0.61	5	798	1044	198	472	100
QPLIB-3413	0.09	33	>1000	678	210	11	100
QPLIB-3587	0.13	5	791	109	104	2	100
QPLIB-3614	0.13	4	680	123	100	2	100
QPLIB-3714	0.32	2	101	20	0	1607	0
QPLIB-3751	0.32	2	100	20	0	7709	0
QPLIB-3757	0.01	482	18	344	37	8850	0
QPLIB-3762	0.28	1	17	2	0	1037	0
QPLIB-3775	0.33	5	100	25	0	32 932	0
QPLIB-3803	0.14	12	33	57	0	5620	0
QPLIB-3815	0.03	2	29	41	6	1807	2
QPLIB-6647	0.07	1009	>1000	11 271	150	232	100
QPLIB-7127	0.07	646	>1000	1662	>1000	2750	0
Shifted geometric mean	0.19	37.8	186	218.77	68	561.5	31

variables but in a first round of tests we focus on purely binary instances from the QPLIB library (Furini et al., 2019).

We compare the performance of our method with `BiqCrunch` (Krislock et al., 2017), an open source SDP solver for binary quadratic optimization problems. 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., Rendl, 2010), 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* (Barahona and Mahjoub, 1986). The results are collected in Table 1. The first column contains the names of the instances and the second column contains the densities, calculated as the number of variables with non-zero quadratic coefficient over the total number of variables (which is $n(n+1)/2$ since the problems are symmetric). Then, for each of the `BiqCrunch` set of parameters and for our algorithm (BQP), two sub-columns 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 73% for our method. Since the values have high variance, we replaced the average (arithmetic mean) by the shifted geometric mean to illustrate the differences in a better way. We used a shift parameter of 0.1 for the densities and 10 for both the time and the gap values. These values also show that the gap obtained by the BQP relaxation is approximately half of the gap of the SDP relaxation obtained with BiqCrunch, even though with a larger computational time. 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 Padberg (1989). 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 may indeed obtain a significant speed-up in the procedure by using ad-hoc algorithms for unconstrained quadratic problems (8).

As we want to measure the quality of our bound and solution method, we compare our results to those obtained with linearizing the quadratic terms. In order to do so, we use the solver `Cplex` and solve the problem with linearization, stopping the procedure firstly at the root node (`Cplex`-r.n.), and secondly as soon as a valid lower bound not smaller than the BQP bound is found (`Cplex`-bd). These results appear for the QPLIB instances in Table 2. We impose a time limit of 10 h in these tests, and the instances that are not solved in the time limit are marked with an asterisk (*).

In this table we can see that the root node gap of the linearization-based method used by the solver `Cplex` is worse than the BQP gap. However, `Cplex` obtains, on most instances, a better bound in shorter time; on a few instances it even closes the gap, but on some other instances our method is faster, and in two instances in particular `Cplex` does not reach the same bound as our method in a time limit of 10 h. This means that for general non-structured problems our method is not always efficient; however, as we will see in the next section, for structured problems the situation can change drastically.

3. Exploiting sparsity by 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 3.1. We define the *support* of a matrix $M \in \mathbb{R}^{m\times n}$ as

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

Definition 3.2. Given $n \in \mathbb{N}$ and $k \in [1:n]$, we define a *k-block sequence* in $\mathbb{R}^n$ the set $\{\underline{b_1},\dots,\underline{b_k}\}$, where the blocks $\underline{b_j} \subseteq [1:n]$ $\forall j \in \mathcal{J} := [1:k]$ and $\bigcup_{j\in\mathcal{J}} \underline{b_j} = [1:n]$, with the property that no block is

Table 2
Comparisons with Cplex, QPLIB instances.

Instance	Density	BQP		Cplex-r.n.		Cplex-bd
		T (s)	*Gap*(%)	T (s)	*Gap*(%)	T (s)
QPLIB-0067	0.88	0.2	1	0.5	1	0.3
QPLIB-1976	0.07	6.8	368	0.3	447	32.5
QPLIB-2017	0.05	124.4	240	0.4	262	20.8
QPLIB-2029	0.05	1865.5	192	0.4	207	20.4
QPLIB-2036	0.05	185.1	313	0.8	351	135.3
QPLIB-2055	0.21	92.4	32	1.2	41	24
QPLIB-2060	0.20	153	32	1.7	42	41
QPLIB-2067	0.19	242.2	62	1.8	69	24.7
QPLIB-2073	0.18	285.4	10	3	20	*36000.0
QPLIB-2085	0.16	1065.5	23	4.3	33	3403
QPLIB-2087	0.16	2935.3	56	5.4	68	60.4
QPLIB-2096	0.15	1209.6	11	6.1	20	*36 000.0
QPLIB-2357	0.08	3222.8	0	0.2	68	528.9
QPLIB-2359	0.04	2888.2	0	0.2	43	258.3
QPLIB-2512	0.77	6.3	100	0.1	100	0
QPLIB-2733	0.89	19 258.1	154	1.1	100	0.1
QPLIB-2957	0.60	11 261	100	1.9	100	0.2
QPLIB-3307	0.61	472.4	100	2	100	0.1
QPLIB-3413	0.09	10.5	100	0.3	100	0
QPLIB-3587	0.13	2.4	100	0.1	100	0
QPLIB-3614	0.13	1.9	100	0.1	100	0
QPLIB-3714	0.32	1607	0	0	100	133
QPLIB-3751	0.32	7708.6	0	0.1	100	579.7
QPLIB-3757	0.01	8850	0	0.7	31	421.3
QPLIB-3762	0.28	1036.8	0	0.1	98	84.1
QPLIB-3775	0.33	32 931.6	0	0.1	100	4600.2
QPLIB-3803	0.14	5620.1	0	0.2	163	814.3
QPLIB-3815	0.03	1807.3	1	0	48	24.7
QPLIB-6647	0.07	232.2	100	0.4	100	0
QPLIB-7127	0.07	2750.3	0	1.6	0	0.1
Shifted geometric mean	0.19	561.5	31	1.07	73	49.9

a subset of another one. We indicate with $d_j = |b_j|$ the size of each block, and assume that they are sorted according to the order of their first element.

Definition 3.3. Let $k \in [1:n]$ and $\mathcal{J} = [1:k]$. Given a k-block sequence $\{b_1, \dots, b_k\}$, *matrix blocks* are defined as $B_j := b_j \times b_j \subseteq [1:n]^2$ for all $j \in \mathcal{J}$. We define the set $\mathscr{B}_\mathcal{J} := \bigcup_{j\in\mathcal{J}} B_j$ as a *block structure* in $\mathbb{R}^{n\times n}$.

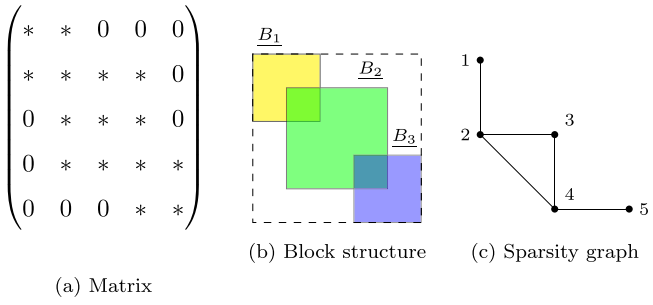
Definition 3.4. The *sparsity graph* of a block structure $\mathscr{B}_\mathcal{J}$ in $\mathbb{R}^{n\times n}$ is a graph $G(V, E)$ with $V = [1:n]$ and with edges on the block structure: $\{p, q\} \in E$ if and only if $(p, q) \in \mathscr{B}_\mathcal{J}$.

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 Fig. 1 we show, for a given matrix (a), its block structure (b) and its sparsity graph (c).

Definition 3.5. Given a matrix $M \in \mathbb{R}^{n\times n}$, an index set $\mathcal{J}$ with $|\mathcal{J}| \le n$ and a block structure $\mathscr{B}_\mathcal{J}$ in $\mathbb{R}^n$, we say that M is *block-decomposable under* $\mathscr{B}_\mathcal{J}$ if $\mathrm{Supp}(M) \subseteq \mathscr{B}_\mathcal{J}$.

In general, for a single matrix several block structures are valid. 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 3.6. If all the blocks in $\mathscr{B}_\mathcal{J}$ have pairwise empty intersection (that is $\{b_j\}_{j\in\mathcal{J}}$ is a partition of $[1:n]$) then the matrix is called *block-separable*.



(a) Matrix (b) Block structure (c) Sparsity graph

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

Definition 3.7. A quadratic problem of the form (1) is called *block-decomposable* (resp. *block-separable*) if there exists a common block structure $\mathscr{B}_\mathcal{J}$ 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 (Padberg, 1989) 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.

Observe that a common sparsity structure in the problem data (zero objective and constraint coefficients at same entry positions) allow for

free entry values of the feasible solutions X at this position, so the sparsity graph of the problem corresponds to the specification graph of X.

The theory of matrix completion problems is well developed for several matrix classes, including the semidefinite (PSD) and completely positive (CPP) matrices. 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 $\mathscr{B}_{\mathcal{J}}$ 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 3.8. 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:

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

and similarly, for the restrictions to intersections of two blocks:

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

For every block $j \in \mathcal{J}$, we introduce variables $Y^j \in \mathbb{R}^{n\times n}$ which can have non-zero entries in the indices of block j and are 0 elsewhere. These matrices Y^j represent the elements of $X_{|j}$. With these variables, we can provide a relaxation of the original problem based on the blocks.

Example 3.1. Suppose to have the following objective function:

$$\min \; X_{1,1} + 2X_{1,2} + 2X_{2,2} + 4X_{2,3} + X_{3,3}$$

and linear constraints (which do not affect the sparsity pattern). In this case, $Q = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 2 \\ 0 & 2 & 1 \end{pmatrix}$, and it is decomposable in two blocks $\underline{B_1} = \{1,2\}^2$ and $\underline{B_2} = \{2,3\}^2$ which intersect in the element 2. The sparsity graph has 3 nodes and the two edges $\{1,2\}$ and $\{2,3\}$. According to the above definitions, we have:

$$Q^1 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad Q^2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 2 & 1 \end{pmatrix},$$
$$X_{|1} = \begin{pmatrix} X_{1,1} & X_{1,2} \\ X_{2,1} & X_{2,2} \end{pmatrix} \quad X_{|2} = \begin{pmatrix} X_{2,2} & X_{2,3} \\ X_{3,2} & X_{3,3} \end{pmatrix}.$$

In the following, with an abuse of notation we consider $Q^j = (Q^j)_{|j} \in \mathbb{R}^{d_j\times d_j}$ and $A_i^j = (A_i^j)_{|j} \in \mathbb{R}^{d_j\times d_j}$. With this notation we have

$$\langle Q, X\rangle = \sum_{j\in\mathcal{J}} \langle Q^j, X_{|j}\rangle \quad \text{and} \quad \langle A_i, X\rangle = \sum_{j\in\mathcal{J}} \langle A_i^j, X_{|j}\rangle \quad \text{for all } i \in \mathcal{I}.$$

With the same abuse of notation we will also write $\langle Q^j, Y^j\rangle$ and $\langle A_i^j, Y^j\rangle$, considering the restrictions of these matrices to the indices in block j. Considering the variables Y^j, we can now write the following problem (where $(Y^j)_{|j\wedge h}$ indicates the restriction of Y^j to the components in $\underline{B_j} \cap \underline{B_h}$):

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

where $y_l^j \in \{0,1\}^{d_j}$, $l \in \mathscr{P}^j$, are binary vectors of the dimension of the corresponding block where $\mathscr{P}^j = [1 : 2^{d_j}]$ are the index sets of all possible such y_l^j, all $j \in \mathcal{J}$. 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 block of size n, 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 — theory

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 Berman and Shaked-Monderer (2003), Drew and Johnson (1998), Fukuda et al. (2001) and Grone et al. (1984). 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 size of $\underline{b_c}$; further, let $\underline{b_a} := \underline{b_1}\setminus\underline{b_c}$, let s be its size, 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 size and let $\underline{B_d} := \underline{b_d}\times\underline{b_d}$. To each $p \in [1 : 2^n]$ we can assign an index triple $(k,l,m) \in [1 : 2^s]\times[1 : 2^r]\times[1 : 2^t]$, each of them indicating 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 $p \in [1 : 2^{d_1}]$ we can assign an index pair $(k,l) \in [1 : 2^s]\times[1 : 2^r]$, and to each $p \in [1 : 2^{d_2}]$ we can assign an index pair $(k,l) \in [1 : 2^r]\times[1 : 2^t]$. And

so, for $p \in [1:2^{d_1}]$ we replace μ^1_p and y^1_p with $\mu_{k,l}$ and $y^1_{k,l}$ respectively, and for $p \in [1:2^{d_2}]$ 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 $(k,l,m) \in [1:2^s] \times [1:2^r] \times [1:2^t]$, the vectors $x_{k,l,m}$, $y^1_{k,l}$, and $y^2_{l,m}$ are linked by the following relations: $(x_{k,l,m})_{|1} = y^1_{k,l}$, $(x_{k,l,m})_{|2} = y^2_{l,m}$, and $(y^1_{k,l})_{|c} = (y^2_{l,m})_{|c} =: y^c_l$.

Example 4.1. With the objective function in Example 3.1, assuming only linear constraints for simplicity, we have that $n = 3$, $B_c = B_1 \cap B_2 = \{2\}$, $b_a = \{1\}$, $b_d = \{3\}$, hence $r = s = t = 1$, and each of the 8 binary vectors $x_p \in \{0,1\}^3, p \in [1:8]$, can be written as $x_{k,l,m}$, with $\{k,l,m\} \subseteq \{1,2\}$. Similarly, the vectors $y^1_p, y^2_p \in \{0,1\}^2$, with $p \in [1:4]$, can be represented by $y^1_{k,l}$ and $y^2_{l,m}$ respectively, with $\{k,l,m\} \subseteq \{1,2\}$.

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

Lemma 4.1. *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 $l \in [1:2^r]$, then there exists an equivalent feasible solution (X, λ) to the one-block formulation (3), i.e. $X_{|1} = Y^1$ and $X_{|2} = Y^2$.

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

$$X_{|1} = \sum_{k=1}^{2^s}\sum_{l=1}^{2^r}\sum_{m=1}^{2^t} \lambda_{k,l,m}(x_{k,l,m})_{|1}(x_{k,l,m})^\top_{|1} = \sum_{k=1}^{2^s}\sum_{l=1}^{2^r} \mu_{k,l}(y^1_{k,l})(y^1_{k,l})^\top = Y^1$$

$$X_{|2} = \sum_{l=1}^{2^r}\sum_{m=1}^{2^t}\sum_{k=1}^{2^s} \lambda_{k,l,m}(x_{k,l,m})_{|2}(x_{k,l,m})^\top_{|2} = \sum_{l=1}^{2^r}\sum_{m=1}^{2^t} \nu_{l,m}(y^2_{l,m})(y^2_{l,m})^\top = Y^2.$$

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 $l \in [1:2^r]$. 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 constraints 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 4.1. 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 4.1 do not hold in general. The following proposition gives a result with some hypothesis on the size of the intersection.

Proposition 4.1. *Let (Y^1, μ, Y^2, ν) be a solution feasible to the multiple block formulation (9). If the size of the intersection block is $r \leq 2$, then there exists an equivalent feasible solution (X, λ) to the one-block formulation (3), i.e. $X_{|1} = Y^1$ and $X_{|2} = Y^2$.*

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

Fig. 2. A not BQP completable matrix.

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 $r \in \{1,2\}$, it is easy to see that the matrices are affinely independent, so (10) holds, hence the result is true. Thanks to Remark 4.1, this result is proved also for $r = 0$. □

This result holds with $r \leq 2$, 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, we consider the multiple block case in the following section.

4.2. Multiple-block case and BQP-completion

In the previous subsection we proved that, whenever we have a specific 2-block-decomposable problem 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 $\mathscr{B}_\mathcal{J}$. The following proposition states that the reverse 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 4.2. *Given a block structure $\mathscr{B}_\mathcal{J}$, suppose that problem (3) is $\mathscr{B}_\mathcal{J}$-decomposable. Given any feasible point for problem (3), i.e. a feasible matrix X and the corresponding coefficients λ_p with $p \in [1:2^n]$, then there exists a solution of (9), given by Y^j and μ^j_l with $(l,j) \in [1:2^{d_j}] \times \mathcal{J}$, such that $X_{|j} = Y^j$ for all $j \in \mathcal{J}$.*

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 $\bar{b}_j = [1:n] \setminus b_j$ the complement of the block b_j in $[1:n]$. To each $p \in [1:2^n]$ we can assign an index pair $(l,m) \in [1:2^{d_j}] \times [1:2^{n-d_j}]$. Let $y_l := (x_p)_{|j}$ and $z_m := (x_p)_{|\bar{j}}$ be the restrictions of x_p to, respectively, b_j and $\bar{b}_j$. We can hence rename x_p as $x_{l,m}$, λ_p as $\lambda_{l,m}$ and write: $X = \sum_{p=1}^{2^n} \lambda_p (x_p x_p^\top) = \sum_{l=1}^{2^{d_j}} \sum_{m=1}^{2^{n-d_j}} \lambda_{l,m} (x_{l,m} x_{l,m}^\top)$ for all $j \in \mathcal{J}$.

Hence, for all $j \in \mathcal{J}$ we can define:

$$\mu^j_l := \sum_{m=1}^{2^{n-d_j}} \lambda_{l,m} \quad \text{for all } l \in [1:2^{d_j}]. \tag{12}$$

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:

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

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 4.1 and 4.2.

We now try to extend the result of Proposition 4.1 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-completion problem. For a detailed description of completion problems (specifically PSD- and CPP-completion) we refer to Berman and Shaked-Monderer (2003). 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 Fig. 2).

Proposition 4.3. *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), $(i,i+1)$, and $(i+1,i)$ for all $i \in [1:l-1]$, and as well (l,l), $(l,1)$ and $(1,l)$. 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 Fig. 2 (Berman and Shaked-Monderer, 2003, 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 (Grone et al., 1984, Lemma 6), hence the matrix in Fig. 2 is not PSD-completable. Since BQP is a subset of the semidefinite cone, this matrix is neither BQP-completable. □

We now introduce a definition that will be useful for the remainder of this section.

Definition 4.1. A graph G is called *diamond-block-clique* if it is chordal and the size of the intersection of any two maximal cliques of G is at most 2.

Remark 4.2. We note that this definition is an extension of the definition of block-clique graphs (given, e.g., in Berman and Shaked-Monderer, 2003) because block-clique graphs are connected chordal graphs in which the size of intersection of two maximal cliques is at most one. On the other hand, any graph whose maximal cliques have intersection of size 2 has an induced diamond graph, hence the name. We also note that a diamond-block-clique graph may be disconnected.

Due to Proposition 4.1 and the properties of chordal graphs, we can now have the following result:

Proposition 4.4. *Every diamond-block-clique graph 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 4.1. 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 (Grone et al., 1984). 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 neighbors which follow it in the order, form a clique. In particular, all the neighbors 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 BQP-completable if G'' is BQP-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 4.1 and conclude the proof. □

Hence by Proposition 4.2 we have the following corollary:

Corollary 4.1. *If a problem is decomposable under a block structure whose sparsity graph is diamond-block-clique, then formulations (3) and (9) are equivalent.*

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

Corollary 4.2. *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, in conjunction with Proposition 4.3:

Conjecture 4.1. *A graph is BQP-completable if and only if it is chordal.*

To the best of our knowledge, this conjecture 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.

If we want to prove this conjecture, from Proposition 4.3 we should only prove that chordal graphs are BQP-completable. Moreover, we could take advantage of similarities with the proof of the PSD-completion problem. In particular, in Grone et al. (1984) the authors make use of the following property of chordal graphs: if G is a chordal graph on n vertices, there exists a sequence of graphs $G_0 = G, G_1, \ldots, G_h = K_n$ where each G_i is chordal and is obtained by the predecessor by adding only one edge. By exploiting this result, we can derive, as the authors of Grone et al. (1984) do for the PSD case, that proving our statement reduces to proving it for graphs with just one missing edge: that is, where the partial matrix has only two blocks, both blocks have size $n-1$, and only the elements $(1,n)$ and $(n,1)$ are not covered by the blocks. Unfortunately, proving this two-block result is currently an open question for BQP-completion.

5. Computational experiments with the BQP block relaxation

From the computational point of view, formulation (9) may be better than formulation (3). The most important advantages 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 problem gets more difficult (although still linear).

Table 3
Performance comparison on the SONET instances.

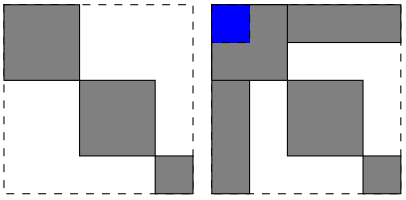
Instance			$BC-cuts$		BQP				
					Single block		Multiple block		
Name	#	Density	T (s)	Gap (%)	# Fails	T (s)	# Fails	T (s)	Gap (%)
ins.16	5	0.07	89	186	0	304	0	0.26	85
ins.17	5	0.07	171	201	1	13	0	0.28	64
ins.18	5	0.06	74	232	0	1402	0	0.46	89
ins.19	5	0.06	53	280	2	1358	0	0.44	88
ins.20	5	0.06	67	175	4	500	0	0.42	84
ins.21	5	0.06	86	348	4	128	0	0.7	70
ins.22	5	0.05	109	396	2	409	0	0.92	61
ins.23	3	0.05	139	330	2	25 710	0	0.43	82
ins.24	5	0.05	171	>1000	3	1389	0	1.34	75
ins.25	5	0.05	204	417	2	237	0	1.52	55
Average	5	0.06	115	394	2	3171	0	0.68	75

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 had a positive reduced cost for more than ten consecutive iterations.

In general, the block structure may 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 Fukuda et al., 2001 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 (Fukuda et al., 2001; Rose et al., 1976). In our preliminary experiments we used the heuristic proposed in Rose et al. (1976) 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 Bergner et al., 2015 for an application of automatic detection of decompositions for standard DWR) 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 Bonami et al. (2012); some of these instances are also included in the QPLIB library. Those instances have a block-diagonal quadratic objective function. Such a structure allows to easily exploit the block decomposition. In Table 3, we report the results of `BiqCrunch` with triangle inequalities (BC-cuts) and 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, the average density, 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 (10 h) 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 40 000 iterations on average.

Table 4
Comparisons with `Cplex`, SONET instances.

Instance			BQP		`Cplex`-r.n.		`Cplex`-bd
	#	Density	T (s)	Gap (%)	T (s)	Gap (%)	$T(s)$
ins.16	5	0.07	0.26	85	0.26	98	5.5
ins.17	5	0.07	0.28	64	0.22	96	21.1
ins.18	5	0.06	0.46	89	0.2	108	5
ins.19	5	0.06	0.44	88	0.23	99	4.8
ins.20	5	0.06	0.42	84	0.53	98	58.2
ins.21	5	0.06	0.7	70	0.74	87	22.5
ins.22	5	0.05	0.92	61	0.74	87	32.3
ins.23	3	0.05	0.43	82	0.48	88	7.8
ins.24	5	0.05	1.34	75	0.54	102	12.7
ins.25	5	0.05	1.52	55	1.09	82	56.2
Average	5	0.06	0.68	75	0.5	95	23.2



(a) separable blocks (b) block-arrowhead

Fig. 3. Sparsity pattern of block-separable instance and block-arrowhead modification.

As for the QPLIB instances, we measure the quality of our algorithm by comparison with the performance of `Cplex` with linearization, at the root node (`Cplex`-r.n.), or after the BQP bound is found (`Cplex`-bd) documented in Table 4.

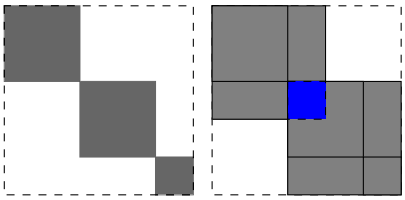
We notice that the improvement given by the block decomposition allows our method to obtain the bound in 0.7 s on average, while the linearization of `Cplex` obtains the same bound in more than 20 s.

In order to test our methods on problems with a more general chordal block structure, we produced some new instances with the so-called *block-arrowhead* sparsity structure: for a fixed parameter r, we added to the SONET instances random coefficients to fill the first r rows and columns of the quadratic objective function matrix, thus obtaining that the first r variables are in common between all blocks. Fig. 3 depicts the obtained structure. We considered ten different values for $r \in [1:10]$, for each of the original SONET instances, thus having 480 block-arrowhead type instances.

We compared our method, with single-block (SB) or multiple-block (MB) formulation, with the one based on the linearization obtained by `Cplex`, at the root node (`Cplex`-r.n.), or after the BQP bound is found

Table 5
Results and comparisons with Cplex, block-arrowhead instances.

r	Density	BQP			Cplex-r.n.		Cplex – *bd*
		SB. T (s)	$MBT(s)$	Gap (%)	T (s)	Gap (%)	T (s)
1	0.07	(16) 7021	1.1	7	0.5	137	153.5
2	0.08	(15) 5719	1.7	8	0.6	122	173.3
3	0.08	(11) 8730	4.7	12	0.7	127	73.8
4	0.09	(13) 7138	10.0	20	0.8	147	104.8
5	0.10	(16) 4375	19.2	23	0.9	156	124.9
6	0.11	(9) 5456	41.8	28	0.8	170	230.8
7	0.12	(13) 4993	84.9	40	0.9	198	210.5
8	0.13	(10) 6499	156.0	51	0.9	225	240.2
9	0.14	(11) 5372	351.3	57	1.0	242	725.3
10	0.15	(13) 5340	694.9	66	1.0	263	731.2
Average	0.11	6081	136.5	31	0.8	179	276.8



(a) separable blocks (b) chain of blocks

Fig. 4. Sparsity pattern of a block-separable instance and its chain modification.

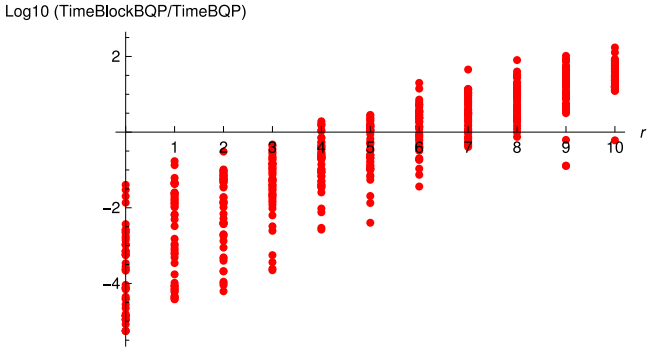


Fig. 5. Performance of block BQP on problems with different intersection size.

(Cplex-bd). Table 5 contains the results, where in each row we present the average of the results for 48 instances, for each value of intersection size r. The number of failures in the time limit of 10 h is written in brackets and the averages are computed only on the solved instances.

We notice that the multiple-block formulation is much more efficient on these instances, too: it indeed allows to solve all instances within a few minutes, while with the single-block formulation some instances are not solved at all within the time limit of 10 h. We further notice that the single-block formulation is faster than the multiple-block formulation in only 38 instances out of 480, all with $r \geq 8$. This is mainly due to the constraints (9c) that slow the multiple-block formulation down as the intersection size r gets large. Moreover, the gap obtained by the multiple-block formulation is much better than the gap of the root node of Cplex, and the same gap is obtained by Cplex in a much larger time, especially if r is small.

Finally, in order to better understand the difficulty caused by the intersections between blocks, we compared the two BQP formulations in another set of instances with increasing intersection between blocks.

More specifically, for each of the block-separable SONET instances, we increase the size of the blocks by a fixed number r so that each block has r elements in common with one or more consecutive blocks. If the subsequent blocks have less than r components in total, we merge all of them in a single block and eliminate the others. Thus we have a *chain* of intersecting blocks, yielding a chordal sparsity graph. We depict this block structure in Fig. 4. We thus added randomly generated coefficients in the objective function to fill the increased blocks, and solved each instance with the two BQP algorithms. We considered ten different values for $r \in [1:10]$, for each of the original SONET instance, thus having 480 instances in total. We show the results with the plot in Fig. 5: 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 time limit (10 h) as value for every failure.

If the intersection between blocks is small, then the multiple block formulation is more efficient than the single block one: for separable problems and for the instances with block-arrowhead sparsity type it was already clear from Tables 3 and 5, but Fig. 5 shows that it is true even with these instances, for some values of $r > 0$. We note that the block structure of these instances is harder to handle than the one in the block-arrowhead instances: although both types of instances have intersection between pairs of blocks of same size, and similar number of added coefficients w.r.t. the original SONET instances, in the block-arrowhead case, all intersections involve the same first r variables, while for these last instances all intersections involve different variables.

We note that the block formulation becomes heavier with increasing intersection size, because the master problem 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. A comparison with Cplex, as for the previous cases, provides similar results. In particular, our algorithm is most suitable when the intersection of the blocks are small.

We further notice that, for every instance solved by both the single-block and the multiple-block algorithms, the optimal objective values are equal, even with $r > 2$: this observation, given by almost 1000 instances with some random elements, may corroborate our conjecture.

6. 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 to matrix completion theory, 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 size of block intersection is 2), and we also proved that if the graph is not chordal, BQP-completion does

not hold. While for larger block intersections, a proof of the BQP-completion result is still due, our empirical findings suggest validity of the conjecture that all chordal graphs are BQP-completable, but also that this result is particularly relevant for small intersection sizes in terms of efficiency. 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.

CRediT authorship contribution statement

Enrico Bettiol: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Immanuel Bomze:** Methodology, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization, Supervision. **Lucas Létocart:** Conceptualization, Methodology, Formal analysis, Writing – original draft, Writing – review & editing, Supervision. **Francesco Rinaldi:** Methodology, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization, Supervision. **Emiliano Traversi:** Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – original draft, Visualization, Supervision.

Acknowledgments

The authors are indebted to the Editorial Team (Editor-in-Chief Francisco Saldanha da Gama and Associate Editor Alice E. Smith) as well as to two Referees whose thoughtful comments and constructive suggestions helped us to significantly improve the paper.

References

Ahmadi, A.A., Dash, S., Hall, G., 2017. Optimization over structured subsets of positive semidefinite matrices via column generation. Discrete Optim. 24, 129–151.
Barahona, F., Mahjoub, A.R., 1986. On the cut polytope. Math. Program. 36 (2), 157–173.
Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W., Vance, P.H., 1998. Branch-and-price: column generation for solving huge integer programs. Oper. Res. 46 (3), 316–329.
Bergner, M., Caprara, A., Ceselli, A., Furini, F., Lübbecke, M.E., Malaguti, E., Traversi, E., 2015. Automatic Dantzig–Wolfe reformulation of mixed integer programs. Math. Program. 149 (1–2), 391–424.
Berman, A., Shaked-Monderer, N., 2003. Completely Positive Matrices. World Scientific.
Bienstock, D., Muñoz, G., 2018. LP formulations for polynomial optimization problems. SIAM J. Optim. 28 (2), 1121–1150.
Bomze, I.M., 2012. Copositive optimization — recent developments and applications. European J. Oper. Res. 216 (3), 509–520.
Bomze, I.M., De Klerk, E., 2002. Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. J. Global Optim. 24 (2), 163–185.
Bomze, I.M., Dür, M., De Klerk, E., Roos, C., Quist, A.J., Terlaky, T., 2000. On copositive programming and standard quadratic optimization problems. J. Global Optim. 18 (4), 301–320.
Bomze, I.M., Jarre, F., Rendl, F., 2011. Quadratic factorization heuristics for copositive programming. Math. Program. Comput. 3 (1), 37–57.
Bomze, I.M., Schachinger, W., Uchida, G., 2012. Think co(mpletely)positive! matrix properties, examples and a clustered bibliography on copositive optimization. J. Global Optim. 52 (3), 423–445.
Bonami, P., Nguyen, V.H., Klein, M., Minoux, M., 2012. On the solution of a graph partitioning problem under capacity constraints. In: International Symposium on Combinatorial Optimization. Springer, pp. 285–296.
Borchers, B., 1999. CSDP, a C library for semidefinite programming. Optim. Methods Softw. 11 (1–4), 613–623.
Bundfuss, S., Dür, M., 2009. An adaptive linear approximation algorithm for copositive programs. SIAM J. Optim. 20 (1), 30–53.
Burer, S., 2009. On the copositive representation of binary and continuous nonconvex quadratic programs. Math. Program. 120 (2), 479–495.
Burer, S., 2010. Optimizing a polyhedral-semidefinite relaxation of completely positive programs. Math. Program. Comput. 2 (1), 1–19.
Burer, S., 2012. Copositive programming. In: Handbook on Semidefinite, Conic and Polynomial Optimization. Springer, pp. 201–218.
Burer, S., Letchford, A.N., 2012. Non-convex mixed-integer nonlinear programming: a survey. Surv. Oper. Res. Manag. Sci. 17 (2), 97–106.
Dantzig, G.B., Wolfe, P., 1960. Decomposition principle for linear programs. Oper. Res. 8 (1), 101–111.
Desaulniers, G., Desrosiers, J., Solomon, M.M. (Eds.), 2005. Column Generation. Springer, US.
Dickinson, P.J., 2010. An improved characterisation of the interior of the completely positive cone. Electron. J. Linear Algebra 20, 723–729.
Dickinson, P.J., 2011. Geometry of the copositive and completely positive cones. J. Math. Anal. Appl. 380 (1), 377–395.
Drew, J.H., Johnson, C.R., 1998. The completely positive and doubly nonnegative completion problems. Linear Multilinear Algebra 44 (1), 85–92.
Dür, M., 2010. Copositive programming — a survey. In: Recent Advances in Optimization and Its Applications in Engineering. Springer, pp. 3–20.
Dür, M., Rendl, F., 2021. Conic optimization: a survey with special focus on copositive optimization and binary quadratic problems. EURO J. Comput. Optim. 9, 100021.
Dür, M., Still, G., 2008. Interior points of the completely positive cone. Electron. J. Linear Algebra 17, 48–53.
Fukuda, M., Kojima, M., Murota, K., Nakata, K., 2001. Exploiting sparsity in semidefinite programming via matrix completion I: General framework. SIAM J. Optim. 11 (3), 647–674.
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., 2019. QPLIB: a library of quadratic programming instances. Math. Program. Comput. 11 (2), 237–265.
Gould, N.I., Toint, P.L., 2000. A quadratic programming bibliography. Numer. Anal. Group Intern. Rep. 1, 32.
Gouveia, J., Pong, T.K., Saee, M., 2020. Inner approximating the completely positive cone via the cone of scaled diagonally dominant matrices. J. Global Optim. 76 (2), 383–405.
Grone, R., Johnson, C.R., Sá, E.M., Wolkowicz, H., 1984. Positive definite completions of partial Hermitian matrices. Linear Algebra Appl. 58, 109–124.
Hall, M., Newman, M., 1963. Copositive and completely positive quadratic forms. Math. Proc. Camb. Phil. Soc. 59 (2), 329–339.
Helmberg, C., 2002. Semidefinite programming. European J. Oper. Res. 137 (3), 461–482.
IBM, 2017. Cplex (version 12.8).
Krislock, N., Malick, J., Roupin, F., 2017. Biqcrunch: a semidefinite branch-and-bound method for solving binary quadratic problems. ACM Trans. Math. Softw. 43 (4), 32.
Lasserre, J.B., 2006. Convergent SDP-relaxations in polynomial optimization with sparsity. SIAM J. Optim. 17 (3), 822–843.
Laurent, M., Rendl, F., 2005. Semidefinite programming and integer programming. Handbooks Oper. Res. Management Sci. 12, 393–514.
Nemhauser, G.L., 2012. Column generation for linear and integer programming. Optim. Stories 20, 64.
Padberg, M., 1989. The boolean quadric polytope: some characteristics, facets and relatives. Math. Program. 45 (1–3), 139–172.
Rendl, F., 2010. Semidefinite relaxations for integer programming. In: 50 Years of Integer Programming 1958-2008. Springer, pp. 687–726.
Rose, D.J., Tarjan, R.E., Lueker, G.S., 1976. Algorithmic aspects of vertex elimination on graphs. SIAM J. Comput. 5 (2), 266–283.
Roupin, F., 2004. From linear to semidefinite programming: an algorithm to obtain semidefinite relaxations for bivalent quadratic problems. J. Combin. Optim. 8 (4), 469–493.
Sturm, J.F., 1999. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim. Methods Softw. 11 (1–4), 625–653.
Tütüncü, R.H., Toh, K.-C., Todd, M.J., 2003. Solving semidefinite-quadratic-linear programs using SDPT3. Math. Program. 95 (2), 189–217.
Vanderbeck, F., 2000. On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Oper. Res. 48 (1), 111–128.
Wainwright, M.J., Jordan, M.I., 2004. Treewidth-Based Conditions for Exactness of the Sherali-Adams and Lasserre Relaxations. Technical Report 671, University of California, Berkeley.
Yıldırım, E.A., 2017. Inner approximations of completely positive reformulations of mixed binary quadratic programs: a unified analysis. Optim. Methods Softw. 32 (6), 1163–1186.