Comput Optim Appl (2014) 59:27–45
DOI 10.1007/s10589-013-9618-8

# Faster, but weaker, relaxations for quadratically constrained quadratic programs

**Samuel Burer · Sunyoung Kim · Masakazu Kojima**

Received: 24 April 2013 / Published online: 8 November 2013


**Abstract** We introduce a new relaxation framework for nonconvex quadratically constrained quadratic programs (QCQPs). In contrast to existing relaxations based on semidefinite programming (SDP), our relaxations incorporate features of both SDP and second order cone programming (SOCP) and, as a result, solve more quickly than SDP. A downside is that the calculated bounds are weaker than those gotten by SDP. The framework allows one to choose a block-diagonal structure for the mixed SOCP-SDP, which in turn allows one to control the speed and bound quality. For a fixed block-diagonal structure, we also introduce a procedure to improve the bound quality without increasing computation time significantly. The effectiveness of our framework is illustrated on a large sample of QCQPs from various sources.

**Keywords** Nonconvex quadratic programming · Semidefinite programming · Second-order cone programming · Difference of convex

The research of S. Burer was supported in part by NSF Grant CCF-0545514. The research of S. Kim was supported by NRF 2012-R1A1A2-038982 and NRF 2010-000-8784. The research of M. Kojima was partially supported by the Japan Science and Technology Agency (JST), the Core Research of Evolutionary Science and Technology (CREST) Research Project.

S. Burer (✉)
Department of Management Sciences, University of Iowa, Iowa City, IA, 52242-1994, USA
e-mail: samuel-burer@uiowa.edu

S. Kim
Department of Mathematics, Ewha W. University, 11-1 DaHyun-Dong, SuhDaeMoon-Gu, Seoul, Korea 120-750
e-mail: skim@ewha.ac.kr

M. Kojima
Research and Development Initiative & JST CREST, Chuo University, 1-13-27, Kasuga, Bunkyo-ku, Tokyo 112-8551 Japan
e-mail: kojimamasakazu@mac.com

## 1 Introduction

Nonconvex quadratically constrained quadratic programs (QCQPs) model many different types of optimization problems and are generally NP-hard. Semidefinite programming (SDP) relaxations can provide tight bounds [6, 11, 12], but they can also be expensive to solve by classical interior-point methods [14]. Many researchers have thus suggested various alternatives to interior-point methods for improving solution times [1, 2, 4, 8, 15, 19]. These efforts have been quite successful on many classes of SDP relaxations.

Others have studied different types of relaxations, for example, ones based on linear programming [10, 13, 18] or second-order cone programming (SOCP) [9, 17, 20]. Generally speaking, one would expect such relaxations to provide weaker bounds in less time compared to SDP relaxations. In fact, SOCP-based relaxations are often constructed as further relaxations of SDP relaxations. So, in a certain sense, one can see (as discussed in Sect. 2) that SOCP relaxations are never tighter than their SDP counterparts.

In this paper, we describe a "middle way" between SOCP and SDP relaxations. Our contribution is a framework for constructing mixed SOCP-SDP relaxations of QCQPs that allows one to balance the trade-off between solution time and bound quality. The key idea is a particular d.c. (difference-of-convex) strategy for further relaxing the linear constraints of an SDP relaxation of a QCQP. While related d.c. approaches have been studied previously, ours uniquely ensures a favorable block-diagonal structure on the resultant SOCP-SDP while simultaneously improving the bound quality. We illustrate the effectiveness of our approach on a large sample of QCQPs from various sources.

Our framework is not without drawbacks. Notably, there are several different choices one must make before applying the framework to a specific QCQP instance, and we do not know how to predict effectively the impact of these choices on the final solution time and bound quality. Indeed, it may be impossible to do so, but this is an interesting avenue for future research. For this paper, we simply illustrate the general behavior of our framework under various options for these choices.

### 1.1 Notation

In this paper, bold capital letters, such as $\boldsymbol{M}$, indicate matrices, and bold lowercase letters, such as $\boldsymbol{v}$, indicate vectors. The notation $\boldsymbol{A} \succeq \boldsymbol{B}$, $\boldsymbol{B} \preceq \boldsymbol{A}$ means that $\boldsymbol{A} - \boldsymbol{B}$ is positive semidefinite, and $\boldsymbol{A} \succ \boldsymbol{B}$, $\boldsymbol{B} \prec \boldsymbol{A}$ means $\boldsymbol{A} - \boldsymbol{B}$ is positive definite. The special matrices $\boldsymbol{I}$ and $\boldsymbol{O}$ are the identity matrix and all-zeros matrix, respectively. For matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ of the same size, the notation $\boldsymbol{M} \bullet \boldsymbol{N} := \text{trace}(\boldsymbol{M}^T \boldsymbol{N})$ is the matrix inner product, and for vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ of the same size, $\boldsymbol{v} \circ \boldsymbol{w}$ is the Hadamard product, i.e., component-wise product. For positive integer $n$, we define $[n] := \{1, \ldots, n\}$.

## 2 The problem and existing techniques

We study the QCQP

$$\begin{array}{ll} \text{minimize} & \boldsymbol{x}^T \boldsymbol{A}_0 \boldsymbol{x} + \boldsymbol{a}_0^T \boldsymbol{x} \\ \text{subject to} & \boldsymbol{x}^T \boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + \alpha_i \le 0 \quad (i = 1, \dots, m) \end{array} \tag{1}$$

where $\boldsymbol{A}_i \in \mathbb{S}^n$, $\boldsymbol{a}_i \in \mathbb{R}^n$, and $\alpha_i \in \mathbb{R}$ for all $i = 0, 1, \dots, m$. Let $F$ denote the feasible set of (1). The basic SDP relaxation of (1) is

$$\begin{array}{ll} \text{minimize} & \boldsymbol{A}_0 \bullet \boldsymbol{X} + \boldsymbol{a}_0^T \boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}_i \bullet \boldsymbol{X} + \boldsymbol{a}_i^T \boldsymbol{x} + \alpha_i \le 0 \quad (i = 1, \dots, m) \\ & \boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T. \end{array} \tag{2}$$

In some cases, it may be possible to strengthen the SDP relaxation—say, by first adding redundant quadratic constraints to (1) before deriving the SDP relaxation—but here we assume that (1) already contains all constraints of interest.

It may also be possible to construct SOCP relaxations of (1) in the original variable space. For example, such a relaxation could be represented as

$$\begin{array}{ll} \text{minimize} & \boldsymbol{x}^T \boldsymbol{B}_0 \boldsymbol{x} + \boldsymbol{b}_0^T \boldsymbol{x} \\ \text{subject to} & \boldsymbol{x}^T \boldsymbol{B}_l \boldsymbol{x} + \boldsymbol{b}_l^T \boldsymbol{x} + \beta_l \le 0 \quad (l \in L), \end{array} \tag{3}$$

where $L$ is an arbitrary index set and all $\boldsymbol{B}_0, \boldsymbol{B}_l \succeq \boldsymbol{O}$. More precisely, we say that (3) is an *SOCP relaxation of* (1) if $\boldsymbol{x} \in F$ implies that $\boldsymbol{x}$ is feasible for (3) and the inequality $\boldsymbol{x}^T \boldsymbol{B}_0 \boldsymbol{x} + \boldsymbol{b}_0^T x \le \boldsymbol{x}^T \boldsymbol{A}_0 \boldsymbol{x} + \boldsymbol{a}_0^T \boldsymbol{x}$ holds. In fact, it is well known that SOCPs can also be solved as SDPs. Specifically, Fujie and Kojima [3] prove that (3) is equivalent to the SDP

$$\begin{array}{ll} \text{minimize} & \boldsymbol{B}_0 \bullet \boldsymbol{X} + \boldsymbol{b}_0^T \boldsymbol{x} \\ \text{subject to} & \boldsymbol{B}_l \bullet \boldsymbol{X} + \boldsymbol{b}_l^T \boldsymbol{x} + \beta_l \le 0 \quad (l \in L) \\ & \boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T. \end{array} \tag{4}$$

The following proposition states the equivalence of (3) and (4) in terms of our current setting.

**Proposition 1** *Suppose the SOCP* (3) *is a valid relaxation of the QCQP* (1). *Then the SDP* (4) *is also a valid relaxation, and its optimal value equals that of* (3).

The upshot of Proposition 1 is that SOCP relaxations in $\boldsymbol{x}$ can never provide better bounds than SDP relaxations in $(\boldsymbol{x}, \boldsymbol{X})$.

Kim and Kojima [9] provided the first SOCP relaxation of (1) not relying on the variable $\boldsymbol{X}$. First, the authors assume without loss of generality that the objective of (1) is linear. This can be achieved, for example, by introducing a new variable $t \in \mathbb{R}$ and a new quadratic constraint $\boldsymbol{x}^T \boldsymbol{A}_0 \boldsymbol{x} + \boldsymbol{a}_0^T \boldsymbol{x} \le t$ and then minimizing $t$. Next, each $\boldsymbol{A}_i$ $(i = 1, \dots, m)$ is written as the difference of two carefully chosen positive semidefinite $\boldsymbol{A}_i^+, \boldsymbol{A}_i^- \succeq \boldsymbol{O}$, i.e., $\boldsymbol{A}_i = \boldsymbol{A}_i^+ - \boldsymbol{A}_i^-$, so that $i$-th constraint may be expressed equivalently as

$$\boldsymbol{x}^T \boldsymbol{A}_i^+ \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + \alpha_i \le \boldsymbol{x}^T \boldsymbol{A}_i^- \boldsymbol{x}.$$

Then, an auxiliary variable $z_i \in \mathbb{R}$ is introduced to represent $\boldsymbol{x}^T \boldsymbol{A}_i^- \boldsymbol{x}$ but also immediately relaxed as $\boldsymbol{x}^T \boldsymbol{A}_i^- \boldsymbol{x} \le z_i$ resulting in the convex system

$$\begin{aligned} \boldsymbol{x}^T \boldsymbol{A}_i^+ \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + \alpha_i &\le z_i \\ \boldsymbol{x}^T \boldsymbol{A}_i^- \boldsymbol{x} &\le z_i. \end{aligned}$$

Finally, $z_i$ must be bounded in some fashion, say as $z_i \le \mu_i \in \mathbb{R}$, or else the relaxation will in fact be useless. Bounding $z_i$ depends very much on the problem and the choice of $\boldsymbol{A}_i^+$, $\boldsymbol{A}_i^-$. In particular, [9] provides strategies for doing so.

Closely related approaches have recently been developed by Saxena et al. [17] and Zheng et al. [20]. In [17], the authors study the relaxation obtained by the following spectral splitting of $\boldsymbol{A}_i$:

$$\boldsymbol{A}_i = \left( \sum_{\lambda_{ij} > 0} \lambda_{ij} \boldsymbol{v}_{ij} \boldsymbol{v}_{ij}^T \right) - \left( \sum_{\lambda_{ij} < 0} |\lambda_{ij}| \boldsymbol{v}_{ij} \boldsymbol{v}_{ij}^T \right)$$

where $\{\lambda_{i1}, \dots, \lambda_{in}\}$ and $\{\boldsymbol{v}_{i1}, \dots, \boldsymbol{v}_{in}\}$ are the eigenvalues and eigenvectors of $\boldsymbol{A}_i$, respectively. The constraint $\boldsymbol{x}^T \boldsymbol{A}_i \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + \alpha_i \le 0$ can thus be reformulated as $\sum_{\lambda_{ij}>0} \lambda_{ij} (\boldsymbol{v}_{ij}^T \boldsymbol{x})^2 + \boldsymbol{a}_i^T \boldsymbol{x} + \alpha_i \le \sum_{\lambda_{ij}<0} |\lambda_{ij}| (\boldsymbol{v}_{ij}^T \boldsymbol{x})^2$. Assuming known lower and upper bounds on the entries of $\boldsymbol{x}$, the non-convex terms $(\boldsymbol{v}_{ij}^T \boldsymbol{x})^2$ for $\lambda_{ij} < 0$ can be relaxed via a secant approximation to derive a convex relaxation of the above constraint. The paper [20] employs similar ideas but further solves a secondary SDP over different splittings of $\boldsymbol{A}_i$ to improve the resultant SOCP relaxation quality.

## 3 A new relaxation framework

We are motivated by the idea that there should exist relaxations between the two extremes introduced in the previous section: SOCPs in only $\boldsymbol{x}$ versus SDPs in both $(\boldsymbol{x}, \boldsymbol{X})$. By "between," we mean that these hypothesized relaxations solve faster than SDPs while providing better bounds than SOCPs. In this section, we present a general construction, which provides these in-between relaxations.

### 3.1 A simple case

Before detailing the general construction in Sects. 3.2–3.5, we first introduce a specific case as a gentle introduction. For all $i = 0, 1, \dots, m$, define $\lambda_i := -\lambda_{\min}[\boldsymbol{A}_i]$ so that $\boldsymbol{A}_i + \lambda_i \boldsymbol{I} \succeq \boldsymbol{O}$. Then (1) is equivalent to

$$\begin{array}{ll} \text{minimize} & -\lambda_0 \boldsymbol{x}^T \boldsymbol{x} + \boldsymbol{x}^T (\boldsymbol{A}_0 + \lambda_0 \boldsymbol{I}) \boldsymbol{x} + \boldsymbol{a}_0^T \boldsymbol{x} \\ \text{subject to} & -\lambda_i \boldsymbol{x}^T \boldsymbol{x} + \boldsymbol{x}^T (\boldsymbol{A}_i + \lambda_i \boldsymbol{I}) \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + \alpha_i \le 0 \quad (i = 1, \dots, m) \end{array}$$

which has the following mixed SOCP-SDP relaxation:

$$\begin{array}{ll} \text{minimize} & -\lambda_0 \operatorname{trace}(\boldsymbol{X}) + \boldsymbol{x}^T (\boldsymbol{A}_0 + \lambda_0 \boldsymbol{I}) \boldsymbol{x} + \boldsymbol{a}_0^T \boldsymbol{x} \\ \text{subject to} & -\lambda_i \operatorname{trace}(\boldsymbol{X}) + \boldsymbol{x}^T (\boldsymbol{A}_i + \lambda_i \boldsymbol{I}) \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + \alpha_i \le 0 \quad (i = 1, \dots, m) \\ & \boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T. \end{array} \tag{5}$$

Notice that, other than the constraint $\boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T$, the only variables in $\boldsymbol{X}$ to appear in the objective and remaining constraints are the variables $\boldsymbol{X}_{jj}$. Said differently, the entries $\boldsymbol{X}_{jk}$ for $j \neq k$ are only relevant for the semidefinite constraint. In addition, one can see that, with $\boldsymbol{x}$ fixed, the diagonal entries of $\boldsymbol{X}$ can be made arbitrarily large to satisfy all constraints with $\lambda_i > 0$ as well as drive the objective to $-\infty$ if $\lambda_0 > 0$. So, in general, one should bound $\boldsymbol{X}_{jj}$ to form a sensible relaxation. For the sake of presentation, let us assume that each $\boldsymbol{x}_j$ is bounded in [0, 1] in (1) so that the constraints $\boldsymbol{X}_{jj} \le \boldsymbol{x}_j$ are valid for the SDP relaxation.

We remark that, instead of defining $\lambda_i := -\lambda_{\min}[\boldsymbol{A}_i]$ as above, another possibility would be to set $\lambda_i$ to $\max\{0, -\lambda_{\min}[\boldsymbol{A}_i]\}$, still ensuring $\boldsymbol{A}_i + \lambda_i \boldsymbol{I} \succeq \boldsymbol{O}$. In words, $\boldsymbol{A}_i + \lambda_i \boldsymbol{I}$ would equal $\boldsymbol{A}_i$ whenever $\boldsymbol{A}_i$ is already positive semidefinite. However, we have chosen the stated definition of $\lambda_i$ because it may lead to a stronger relaxation (5). For example, consider the inequality $\boldsymbol{x}^T\boldsymbol{x} - 1 \le 0$. Our definition of $\lambda_i$ yields the constraint $\text{trace}(\boldsymbol{X}) - 1 \le 0$, whereas the alternative would keep $\boldsymbol{x}^T\boldsymbol{x} - 1 \le 0$. Since $\boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T$ implies $\boldsymbol{x}^T\boldsymbol{x} \le \text{trace}(\boldsymbol{X})$, the former inequality is stronger than the latter.

The definition of $\lambda_i$ has another benefit. If the original problem (1) contains a strictly convex inequality constraint with $\boldsymbol{A}_i \succ \boldsymbol{O}$, the feasible region of (5) will be bounded without the assumption $\boldsymbol{x}_j \in [0, 1]$ and constraint $\boldsymbol{X}_{jj} \le \boldsymbol{x}_j$ as described above. For example, the spherical constraint $\boldsymbol{x}^T\boldsymbol{x} - 1 \le 0$ would be transformed to $\text{trace}(\boldsymbol{X}) - 1 \le 0$, thus bounding the feasible region of (5) in conjunction with $\boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T$. Still, in this paper, our preference is to use $\boldsymbol{x}_j \in [0, 1]$ and $\boldsymbol{X}_{jj} \le \boldsymbol{x}_j$ to establish boundedness.

Consider now the following proposition:

**Proposition 2** (Grone et al. [7]) *Given a vector* $\boldsymbol{x}$ *and scalars* $\boldsymbol{X}_{11}, \ldots, \boldsymbol{X}_{nn}$, *there exists a symmetric-matrix completion* $\boldsymbol{X}$ *of* $\boldsymbol{X}_{11}, \ldots, \boldsymbol{X}_{nn}$ *satisfying* $\boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T$ *if and only if* $\boldsymbol{X}_{jj} \ge \boldsymbol{x}_j^2$ *for all* $j = 1, \ldots, n$.

*Proof* This follows from the chordal arrow structure of the matrix

$$\begin{pmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \textbf{Diag}(\boldsymbol{X}_{11}, \ldots, \boldsymbol{X}_{nn}) \end{pmatrix},$$

where $\textbf{Diag}(\cdot)$ places its arguments in a diagonal matrix. We refer the reader to Grone et al. [7] and Fukuda et al. [4] for further details. □

In light of Proposition 2, problem (5) with additional bounding constraints $\boldsymbol{X}_{jj} \le \boldsymbol{x}_j$ is equivalent to

$$\begin{array}{ll} \text{minimize} & -\lambda_0 \sum_{j=1}^n \boldsymbol{X}_{jj} + \boldsymbol{x}^T(\boldsymbol{A}_0 + \lambda_0 \boldsymbol{I})\boldsymbol{x} + \boldsymbol{a}_0^T \boldsymbol{x} \\ \text{subject to} & -\lambda_i \sum_{j=1}^n \boldsymbol{X}_{jj} + \boldsymbol{x}^T(\boldsymbol{A}_i + \lambda_i \boldsymbol{I})\boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + \alpha_i \le 0 \\ & \quad (i = 1, \ldots, m) \\ & \boldsymbol{x}_j^2 \le \boldsymbol{X}_{jj} \le \boldsymbol{x}_j \quad (j = 1, \ldots, n). \end{array} \tag{6}$$

Compared to the SDP relaxation (2), which has $O(n^2)$ variables, problem (6) has only $O(n)$ variables and hence is much faster to solve. Of course, its bound on (1) should generally be weaker than the SDP bound.

It may actually be possible to improve the bound quality of (6) without changing its basic structure and computational complexity. Instead of splitting $\boldsymbol{A}_i$ into $\boldsymbol{A}_i + \lambda_i \boldsymbol{I} \succeq \boldsymbol{O}$ and $-\lambda_i \boldsymbol{I} \preceq 0$, we could more generally split it into $\boldsymbol{A}_i + \boldsymbol{D}_i \succeq \boldsymbol{O}$ and $-\boldsymbol{D}_i$, where $\boldsymbol{D}_i$ is a diagonal (not necessarily positive semidefinite) matrix. The resultant relaxation

$$\begin{array}{ll} \text{minimize} & -\sum_{j=1}^{n}[\boldsymbol{D}_0]_{jj}\boldsymbol{X}_{jj} + \boldsymbol{x}^T(\boldsymbol{A}_0 + \boldsymbol{D}_0)\boldsymbol{x} + \boldsymbol{a}_0^T\boldsymbol{x} \\ \text{subject to} & -\sum_{j=1}^{n}[\boldsymbol{D}_i]_{jj}\boldsymbol{X}_{jj} + \boldsymbol{x}^T(\boldsymbol{A}_i + \boldsymbol{D}_i)\boldsymbol{x} + \boldsymbol{a}_i^T\boldsymbol{x} + \alpha_i \le 0 \\ & (i = 1, \ldots, m) \\ & \boldsymbol{x}_j^2 \le \boldsymbol{X}_{jj} \le \boldsymbol{x}_j \quad (j = 1, \ldots, n), \end{array} \tag{7}$$

could provide a better bound than (6) if the $\boldsymbol{D}_i$ are chosen intelligently, but it should still require roughly the same amount of time to solve.

This naturally leads to the idea of optimizing the choice of $\{\boldsymbol{D}_i\}$ so as to improve (or maximize) the lower bound coming from (7). However, this appears to be a difficult problem. Not only does it require the $O(nm)$ variables $\{\boldsymbol{D}_i\}$ and the $O(m)$ positive-semidefinite constraints $\boldsymbol{A}_i + \boldsymbol{D}_i \succeq \boldsymbol{O}$, it is also nonconvex since $\boldsymbol{D}_i$ appear in the constraints of (7). So, rather than trying to directly optimize $\{\boldsymbol{D}_i\}$, it would be better just to solve the SDP (2).

Still, we might be able to optimize $\{\boldsymbol{D}_i\}$ heuristically. In particular, suppose that $\{\boldsymbol{D}_i\}$ and $\{\widehat{\boldsymbol{D}}_i\}$ are two valid choices such that $[\boldsymbol{D}_i]_{jj} > [\widehat{\boldsymbol{D}}_i]_{jj}$ for all $i = 0, 1 \ldots, m$ and $j = 1, \ldots, n$. Then the conditions $\boldsymbol{x}_j^2 \le \boldsymbol{X}_{jj}$ guarantee that replacing $\boldsymbol{D}_i$ by $\widehat{\boldsymbol{D}}_i$ in (7) yields—loosely speaking—a tighter feasible region with a uniformly higher objective function. In other words, if we start with $\boldsymbol{D}_i$ and reduce its diagonal entries, while still maintaining $\boldsymbol{A}_i + \boldsymbol{D}_i \succeq \boldsymbol{O}$, then the optimal value of (7) will increase (more precisely, *may* increase). In a sense, we would like to make $\boldsymbol{A}_i + \boldsymbol{D}_i$ "less positive semidefinite" by reducing its diagonal entries as a means to improve the bound. This could be done independently and heuristically for each $i$ by looping over the index $j$ to reduce each $[\boldsymbol{D}_i]_{jj}$ separately. At each step of the loop, we would solve the subproblem of minimizing $[\boldsymbol{D}_i]_{jj}$ subject to $\boldsymbol{A}_i + \boldsymbol{D}_i \succeq \boldsymbol{O}$, while keeping all other entries of $\boldsymbol{D}_i$ fixed. Such a one-variable SDP should be quickly solvable.

In the following Sects. 3.2–3.5, we formalize all of the above ideas and apply them in a more general block-diagonal case.

### 3.2 The mixed SOCP-SDP relaxation

Given a positive integer $r \le n$, let $\mathcal{C} := \{C_1, \ldots, C_r\}$ be a partition of the indices $[n]$. We assume without loss of generality that $\mathcal{C}$ satisfies $\max(C_k) \le \min(C_{k+1})$ for all $k = 1, \ldots, r-1$. In words, $C_1$ consists of the first few indices in $[n]$, $C_2$ consists of the next few, and so on. We say that a symmetric matrix $\boldsymbol{D}$ is $\mathcal{C}$*-block diagonal* if $\boldsymbol{D}_{jk} = 0$ whenever $j$ and $k$ are members of different sets in $\mathcal{C}$. That is, $\boldsymbol{D}$ is block diagonal with blocks $\boldsymbol{D}_{C_1C_1}, \ldots, \boldsymbol{D}_{C_rC_r}$.

Based on the partition $\mathcal{C}$, we construct a mixed SOCP-SDP relaxation of (1) as follows. For each $i = 0, 1, \ldots, m$, let $\boldsymbol{D}_i$ be a $\mathcal{C}$-block diagonal matrix satisfying $\boldsymbol{A}_i + \boldsymbol{D}_i \succeq \boldsymbol{O}$. This yields the following problem equivalent to (1), where we assume

$\boldsymbol{x} \in [0, 1]^n$ as above:

$$\begin{array}{ll} \text{minimize} & -\boldsymbol{x}^T \boldsymbol{D}_0 \boldsymbol{x} + \boldsymbol{x}^T (\boldsymbol{A}_0 + \boldsymbol{D}_0) \boldsymbol{x} + \boldsymbol{a}_0^T \boldsymbol{x} \\ \text{subject to} & -\boldsymbol{x}^T \boldsymbol{D}_i \boldsymbol{x} + \boldsymbol{x}^T (\boldsymbol{A}_i + \boldsymbol{D}_i) \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + \alpha_i \le 0 \quad (i = 1, \ldots, m) \\ & \boldsymbol{x} \circ \boldsymbol{x} \le \boldsymbol{x}. \end{array}$$

Its mixed SOCP-SDP relaxation is

$$\begin{array}{ll} \text{minimize} & -\boldsymbol{D}_0 \bullet \boldsymbol{X} + \boldsymbol{x}^T (\boldsymbol{A}_0 + \boldsymbol{D}_0) \boldsymbol{x} + \boldsymbol{a}_0^T \boldsymbol{x} \\ \text{subject to} & -\boldsymbol{D}_i \bullet \boldsymbol{X} + \boldsymbol{x}^T (\boldsymbol{A}_i + \boldsymbol{D}_i) \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + \alpha_i \le 0 \quad (i = 1, \ldots, m) \\ & \operatorname{diag}(\boldsymbol{X}) \le \boldsymbol{x} \\ & \boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T. \end{array} \tag{8}$$

Relaxation (8) has the property that, due to the block structure of $\boldsymbol{D}_i$, the only portions of $\boldsymbol{X}$ appearing in the objective and linear constraints are the blocks $\boldsymbol{X}_{C_1C_1}, \ldots, \boldsymbol{X}_{C_rC_r}$. The other entries of $\boldsymbol{X}$ are only relevant for the semidefiniteness constraint $\boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T$. The following proposition allows us to eliminate all entries of $\boldsymbol{X}$ except for the blocks $\boldsymbol{X}_{C_kC_k}$:

**Proposition 3** (Grone et al. [7]) *Given a vector $\boldsymbol{x}$ and symmetric blocks $\boldsymbol{X}_{C_1C_1}, \ldots, \boldsymbol{X}_{C_rC_r}$, there exists a symmetric-matrix completion $\boldsymbol{X}$ of $\boldsymbol{X}_{C_1C_1}, \ldots, \boldsymbol{X}_{C_rC_r}$ satisfying $\boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T$ if and only $\boldsymbol{X}_{C_kC_k} \succeq \boldsymbol{x}_{C_k}\boldsymbol{x}_{C_k}^T$ for all $k = 1, \ldots, r$.*

*Proof* This follows from the chordal arrow structure of the matrix

$$\begin{pmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \mathbf{Diag}(\boldsymbol{X}_{C_1C_1}, \ldots, \boldsymbol{X}_{C_rC_r}) \end{pmatrix},$$

where $\mathbf{Diag}(\cdot)$ places its arguments in a diagonal matrix. We refer the reader to Grone et al. [7] and Fukuda et al. [4] for further details. □

In light of the Proposition 3, problem (8) is equivalent to

$$\begin{array}{ll} \text{minimize} & -\sum_{k=1}^r [\boldsymbol{D}_0]_{C_kC_k} \bullet \boldsymbol{X}_{C_kC_k} + \boldsymbol{x}^T (\boldsymbol{A}_0 + \boldsymbol{D}_0) \boldsymbol{x} + \boldsymbol{a}_0^T \boldsymbol{x} \\ \text{subject to} & -\sum_{k=1}^r [\boldsymbol{D}_i]_{C_kC_k} \bullet \boldsymbol{X}_{C_kC_k} + \boldsymbol{x}^T (\boldsymbol{A}_i + \boldsymbol{D}_i) \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + \alpha_i \le 0 \\ & \quad (i = 1, \ldots, m) \\ & \boldsymbol{X}_{jj} \le \boldsymbol{x}_j \quad (j = 1, \ldots, n) \\ & \boldsymbol{X}_{C_kC_k} \succeq \boldsymbol{x}_{C_k}\boldsymbol{x}_{C_k}^T \quad (k = 1, \ldots, r). \end{array} \tag{9}$$

It is important to keep in mind that only the portions $\boldsymbol{X}_{C_kC_k}$ of the original $\boldsymbol{X}$ actually remain in the problem. We say that (9) has a $\mathcal{C}$*-block structure*.

Problem (9) is the "in-between" relaxation that we propose to solve. However, it remains to choose the splitting of $\boldsymbol{A}_i$ intelligently. To make the notation of the next subsections easier, we apply the linear change of variables $\boldsymbol{B}_i := \boldsymbol{A}_i + \boldsymbol{D}_i$ so that $\boldsymbol{B}_i \succeq \boldsymbol{O}$ and $\boldsymbol{B}_i - \boldsymbol{A}_i$ is $\mathcal{C}$-block diagonal. We also rewrite (9) as

$$\begin{array}{ll} \text{minimize} & (\boldsymbol{A}_0 - \boldsymbol{B}_0) \bullet \boldsymbol{X} + \boldsymbol{x}^T \boldsymbol{B}_0 \boldsymbol{x} + \boldsymbol{a}_0^T \boldsymbol{x} \\ \text{subject to} & (\boldsymbol{A}_i - \boldsymbol{B}_i) \bullet \boldsymbol{X} + \boldsymbol{x}^T \boldsymbol{B}_i \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + \alpha_i \le 0 \quad (i = 1, \ldots, m) \\ & \operatorname{diag}(\boldsymbol{X}) \le \boldsymbol{x} \\ & \boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T \end{array} \tag{10}$$

in terms of $\boldsymbol{B}_i$ and the full matrix $\boldsymbol{X}$. We stress that the use of $\boldsymbol{X}$ is only to simplify the presentation. In computation, the alternative yet equivalent representation in terms of $\boldsymbol{X}_{C_k C_k}$ would be employed to exploit the $\mathcal{C}$-block structure of (10).

### 3.3 How to choose the matrices $\boldsymbol{B}_i$

We now discuss how to choose the matrices $\boldsymbol{B}_i$ intelligently. Consistent with the discussion in the previous subsection, $\boldsymbol{B}_i$ is a feasible choice as long as it is an element of

$$\mathcal{F}(\boldsymbol{A}_i) := \left\{\boldsymbol{M} \succeq \boldsymbol{O} : \boldsymbol{A}_i - \boldsymbol{M} \text{ is } \mathcal{C}\text{-block diagonal}\right\}.$$

One idea would be to choose the collection $\{\boldsymbol{B}_i\}$ that results in the tightest bound from (10). In other words, $\{\boldsymbol{B}_i\}$ could be chosen as an optimal solution of

$$\begin{array}{ll} \text{maximize} & f(\boldsymbol{B}_0, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_m) \\ \text{subject to} & \boldsymbol{B}_i \in \mathcal{F}(\boldsymbol{A}_i) \quad (i = 0, \ldots, m) \end{array}$$

where $f(\boldsymbol{B}_0, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_m)$ is the optimal value of (10). However, the objective of this problem is non-convex, and so the problem is likely to require as much time to solve as the SDP relaxation (2). It is then not worth the effort since it cannot even produce a better bound than (2) by Proposition 1. So the choice of the $\boldsymbol{B}_i$'s must balance the cost to compute them with the resultant bound quality.

As a compromise, consider the following observation:

**Proposition 4** *For each $i = 0, 1, \ldots, m$, suppose $\boldsymbol{B}_i, \widehat{\boldsymbol{B}}_i \in \mathcal{F}(\boldsymbol{A}_i)$ satisfy $\boldsymbol{B}_i \succeq \widehat{\boldsymbol{B}}_i$. Then*

$$f(\boldsymbol{B}_0, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_m) \le f(\widehat{\boldsymbol{B}}_0, \widehat{\boldsymbol{B}}_1, \ldots, \widehat{\boldsymbol{B}}_m).$$

*Proof* We must show that the optimal value of (10) cannot deteriorate when all $\boldsymbol{B}_i$ are replaced by $\widehat{\boldsymbol{B}}_i$. To see this, let $(\boldsymbol{x}, \boldsymbol{X})$ be feasible for the problem with $\widehat{\boldsymbol{B}}_i$. Note that $\boldsymbol{B}_i \succeq \widehat{\boldsymbol{B}}_i$ and $\boldsymbol{X} \succeq \boldsymbol{x}\boldsymbol{x}^T$ imply $(\boldsymbol{B}_i - \widehat{\boldsymbol{B}}_i) \bullet (\boldsymbol{X} - \boldsymbol{x}\boldsymbol{x}^T) \ge 0$, and so, for all $i$, the inequalities

$$\begin{aligned} &(\boldsymbol{A}_i - \boldsymbol{B}_i) \bullet \boldsymbol{X} + \boldsymbol{x}^T \boldsymbol{B}_i \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} \\ &\quad \le (\boldsymbol{A}_i - \boldsymbol{B}_i) \bullet \boldsymbol{X} + \boldsymbol{x}^T \boldsymbol{B}_i \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} + (\boldsymbol{B}_i - \widehat{\boldsymbol{B}}_i) \bullet \left(\boldsymbol{X} - \boldsymbol{x}\boldsymbol{x}^T\right) \\ &\quad = (\boldsymbol{A}_i - \widehat{\boldsymbol{B}}_i) \bullet \boldsymbol{X} + \boldsymbol{x}^T \widehat{\boldsymbol{B}}_i \boldsymbol{x} + \boldsymbol{a}_i^T \boldsymbol{x} \end{aligned}$$

imply that $(\boldsymbol{x}, \boldsymbol{X})$ is feasible for (10) with no higher objective value. □

Proposition 4 suggests that a reasonable choice of $\{\boldsymbol{B}_i\}$ should at least have the property that each $\boldsymbol{B}_i$ is a "minimally positive semidefinite" member of $\mathcal{F}(\boldsymbol{A}_i)$. In the next subsection, we deal with the issue of finding such a minimal member of $\mathcal{F}(\boldsymbol{A}_i)$ for each $i$.

### 3.4 Minimal and minimum elements

The previous subsection has discussed choosing good matrices $\{\boldsymbol{B}_i\}$ to form (10). Our idea is to choose a "minimal" element in the set $\mathcal{F}(\boldsymbol{A}_i)$. Here, we discuss this issue formally and computationally. We also discuss the related idea of "minimum" elements. The results in this subsection are general and will be applied specifically to choose $\{\boldsymbol{B}_i\}$ in Sect. 3.5.

Given a nonempty, closed, convex subset $\mathcal{T}$ of the positive semidefinite matrices, a member $\boldsymbol{T} \in \mathcal{T}$ is called *minimal* if $\{\boldsymbol{M} \in \mathcal{T} : \boldsymbol{T} \succeq \boldsymbol{M}\} = \{\boldsymbol{T}\}$. In other words, $\boldsymbol{T}$ is minimal if there exists no $\boldsymbol{M}$ in $\mathcal{T}$ distinct from $\boldsymbol{T}$ such that $\boldsymbol{M} \preceq \boldsymbol{T}$. Furthermore, $\boldsymbol{T} \in \mathcal{T}$ is called *minimum* if $\boldsymbol{T} \preceq \boldsymbol{M}$ for all $\boldsymbol{M} \in \mathcal{T}$. It is not difficult to see that every $\mathcal{T}$ has at least one minimal element and that $\mathcal{T}$ has a minimum element $\boldsymbol{T}$ if and only if $\boldsymbol{T}$ is the unique minimal element. We let Minimal$(\mathcal{T})$ denote the set of minimal elements in $\mathcal{T}$ and minimum$(\mathcal{T})$ denote the minimum element if it exists.

*Example 3.1* Let

$$\mathcal{T} := \left\{ \begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix} \in \mathbb{S}^2_+ : t_{12} = 1 \right\} = \left\{ \begin{pmatrix} t_{11} & 1 \\ 1 & t_{22} \end{pmatrix} \in \mathbb{S}^2 : \begin{array}{c} t_{11} > 0, t_{22} > 0 \\ t_{11}t_{22} \geq 1 \end{array} \right\}.$$

One can show

$$\begin{aligned} \text{Minimal}(\mathcal{T}) &= \left\{ \begin{pmatrix} t_{11} & 1 \\ 1 & t_{22} \end{pmatrix} \in \mathbb{S}^2 : \begin{array}{c} t_{11} > 0, t_{22} > 0 \\ t_{11}t_{22} = 1 \end{array} \right\} \\ &= \left\{ \begin{pmatrix} t_{11} & 1 \\ 1 & t_{11}^{-1} \end{pmatrix} \in \mathbb{S}^2 : t_{11} > 0 \right\}. \end{aligned}$$

In this case, there are multiple minimal elements and hence no minimum element. We also note that the rank of all minimal elements is small, namely 1.

By solving an SDP, it is always possible to calculate some $\boldsymbol{T} \in \text{Minimal}(\mathcal{T})$. Choose any positive definite matrix $\boldsymbol{C}$, and let $\boldsymbol{T}$ be an optimal solution of $\min\{\boldsymbol{C} \bullet \boldsymbol{X} : \boldsymbol{X} \in \mathcal{T}\}$. To see that $\boldsymbol{T}$ is minimal, assume on the contrary that $\boldsymbol{X} \in \mathcal{T}$ satisfies $\boldsymbol{T} \neq \boldsymbol{X} \preceq \boldsymbol{T}$. Then $\boldsymbol{C} \bullet (\boldsymbol{X} - \boldsymbol{T}) < 0$, contradicting the optimality of $\boldsymbol{T}$.

However, in this paper we do not suggest solving an SDP to get a minimal element in $\mathcal{T}$ because—at least for our application—doing so would require as much time as solving (2). So we suggest a different technique that will work quickly in our specialized setting. Accordingly, let us focus on specific sets $\mathcal{T}$ relevant to this paper. Given $\boldsymbol{A} \in \mathbb{S}^n$, define

$$\mathcal{F}(\boldsymbol{A}) := \left\{ \boldsymbol{M} \succeq \boldsymbol{O} : \boldsymbol{A} - \boldsymbol{M} \text{ is } \mathcal{C}\text{-block diagonal} \right\}.$$

One can think of $\boldsymbol{A}$ as generically representing a matrix $\boldsymbol{A}_i$ in (1), although $\boldsymbol{A}$ is not necessarily limited in this way. We would like to compute some $\boldsymbol{B} \in \text{Minimal}(\mathcal{F}(\boldsymbol{A}))$ without solving an SDP. Our approach is proposed as Algorithm 2 below.

We start with an observation that $\mathcal{F}(\boldsymbol{A})$ does not change even if $\boldsymbol{A}$ is shifted by a $\mathcal{C}$-block diagonal matrix:

**Proposition 5** *Given $\boldsymbol{A} \in \mathbb{S}^n$, let $\Delta$ be any $\mathcal{C}$-block diagonal matrix. Then $\mathcal{F}(\boldsymbol{A}) = \mathcal{F}(\boldsymbol{A} + \Delta)$.*

*Proof* This follows from the fact that $\boldsymbol{A} - \boldsymbol{M}$ is $\mathcal{C}$-block diagonal if and only if $(\boldsymbol{A} + \Delta) - \boldsymbol{M}$ is. □

This proposition will give us some useful flexibility. To calculate $\boldsymbol{B} \in \text{Minimal}(\mathcal{F}(\boldsymbol{A}))$, Algorithm 2 will first shift $\boldsymbol{A}$ by a $\mathcal{C}$-block diagonal matrix $\Delta$ such that $\boldsymbol{A} + \Delta \succeq \boldsymbol{O}$ and then calculate $\boldsymbol{B} \in \text{Minimal}(\mathcal{F}(\boldsymbol{A} + \Delta))$. The choice of the shift $\Delta$ may have an effect on the final calculated $\boldsymbol{B}$, and we will discuss different choices of $\Delta$ in Sect. 3.5.

Algorithm 2 will make use of a specialized subroutine (Algorithm 1). Abusing notation, let an input $\boldsymbol{A} \succeq \boldsymbol{O}$ to this subroutine be fixed, and for each $k = 1, \ldots, r$, define the following restriction of $\mathcal{F}(\boldsymbol{A})$, which depends on the subset $C_k$ of $[n]$:

$$\mathcal{F}_k(\boldsymbol{A}) := \left\{\boldsymbol{O} \preceq \boldsymbol{M} \preceq \boldsymbol{A} : \boldsymbol{A} - \boldsymbol{M} \text{ is } C_k\text{-block diagonal}\right\}$$

where a matrix is defined to be *$C_k$-block diagonal* if all entries outside of its $C_k C_k$ block are zero. We present Algorithm 1 for calculating $\boldsymbol{B} = \text{minimum}(\mathcal{F}_k(\cdot))$. Lemma 1 and Proposition 6 establish the correctness of Algorithm 1.

---

**Algorithm 1** Calculate the minimum element of $\mathcal{F}_k(\boldsymbol{A})$ for $\boldsymbol{A} \succeq \boldsymbol{O}$

---

**Input:** $C_k \subseteq [n]$, $\boldsymbol{A} \succeq \boldsymbol{O}$, $p := \text{rank}(\boldsymbol{A})$, $\boldsymbol{L} \in \mathbb{R}^{n \times p}$ s.t. $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$ if $p \geq 1$.

**Output:** $\boldsymbol{B} = \text{minimum}(\mathcal{F}_k(\boldsymbol{A}))$ $q := \text{rank}(\boldsymbol{B}) \leq p$, $\widehat{\boldsymbol{L}} \in \mathbb{R}^{n \times q}$ s.t. $\boldsymbol{B} = \widehat{\boldsymbol{L}}\widehat{\boldsymbol{L}}^T$ if $q \geq 1$.

1: If $p = 0$ or $C_k = [n]$, then return $\boldsymbol{B} = \boldsymbol{O}$.
2: Define $S := \text{span}\{\boldsymbol{L}_{i\cdot}^T : i \notin C_k\} \subseteq \mathbb{R}^p$. Let $q := \dim(S)$.
3: Construct $\boldsymbol{V} \in \mathbb{R}^{p \times q}$ with columns forming an orthonormal basis of $S$.
4: Return $\widehat{\boldsymbol{L}} := \boldsymbol{L}\boldsymbol{V}$ and $\boldsymbol{B} := \widehat{\boldsymbol{L}}\widehat{\boldsymbol{L}}^T$.

---

**Lemma 1** *Given $\boldsymbol{M}$ such that $\boldsymbol{M}\boldsymbol{M}^T \preceq \boldsymbol{I}$, suppose $\boldsymbol{N}$ has orthonormal columns with* $\text{range}(\boldsymbol{M}) \subseteq \text{range}(\boldsymbol{N})^\perp$. *Then $\boldsymbol{M}\boldsymbol{M}^T + \boldsymbol{N}\boldsymbol{N}^T \preceq \boldsymbol{I}$.*

*Proof* Let $\boldsymbol{W}$ be a matrix of orthonormal columns spanning $\text{range}(\boldsymbol{N})^\perp$. Then $\text{range}(\boldsymbol{M}) \subseteq \text{range}(\boldsymbol{W})$ and there exists $\boldsymbol{H}$ such that $\boldsymbol{M} = \boldsymbol{W}\boldsymbol{H}$.

We claim $\boldsymbol{H}\boldsymbol{H}^T \preceq \boldsymbol{I}$, where $\boldsymbol{I}$ is of the appropriate size. If not, then there exists $\boldsymbol{x}$ such that $\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{x} > \boldsymbol{x}^T\boldsymbol{x}$. Defining $\boldsymbol{v} := \boldsymbol{W}\boldsymbol{x}$, we see

$$\begin{aligned} \boldsymbol{v}^T\boldsymbol{M}\boldsymbol{M}^T\boldsymbol{v} &= (\boldsymbol{W}\boldsymbol{x})^T\left(\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{W}^T\right)(\boldsymbol{W}\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{x} > \boldsymbol{x}^T\boldsymbol{x} \\ &= \boldsymbol{x}^T\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{x} = \boldsymbol{v}^T\boldsymbol{v}, \end{aligned}$$

which contradicts $\boldsymbol{M}\boldsymbol{M}^T \preceq \boldsymbol{I}$.

Next, for arbitrary $\boldsymbol{v}$, write $\boldsymbol{v} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{N}\boldsymbol{y}$; note that $\boldsymbol{v}^T\boldsymbol{v} = \boldsymbol{x}^T\boldsymbol{x} + \boldsymbol{y}^T\boldsymbol{y}$. Then

$$\boldsymbol{v}^T\left(\boldsymbol{M}\boldsymbol{M}^T + \boldsymbol{N}\boldsymbol{N}^T\right)\boldsymbol{v} = (\boldsymbol{W}\boldsymbol{x} + \boldsymbol{N}\boldsymbol{y})^T\left(\boldsymbol{W}\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{W}^T + \boldsymbol{N}\boldsymbol{N}^T\right)(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{N}\boldsymbol{y})$$

$$
\begin{aligned}
&= \boldsymbol{x}^T \boldsymbol{H}\boldsymbol{H}^T \boldsymbol{x} + \boldsymbol{y}^T \boldsymbol{y} \\
&= \boldsymbol{x}^T \big(\boldsymbol{H}\boldsymbol{H}^T - \boldsymbol{I}\big)\boldsymbol{x} + \boldsymbol{v}^T \boldsymbol{v} \\
&\le \boldsymbol{v}^T \boldsymbol{v}.
\end{aligned}
$$

This proves the result. □

**Proposition 6** *Algorithm 1 correctly calculates* $\boldsymbol{B} = \text{minimum}(\mathcal{F}_k(\boldsymbol{A}))$ *when* $\boldsymbol{A} \succeq \boldsymbol{O}$.

*Proof* Clearly, $\boldsymbol{B} \succeq \boldsymbol{O}$. We next show $\boldsymbol{B} \preceq \boldsymbol{A}$. If $p = 0$ or $C_k = [n]$ so that $\boldsymbol{B} = \boldsymbol{O}$, then $\boldsymbol{B} \preceq \boldsymbol{A}$ is obvious. Otherwise, if $1 \le p \le r$, then $\boldsymbol{A} - \boldsymbol{B} = \boldsymbol{L}\boldsymbol{L} - \widehat{\boldsymbol{L}}\widehat{\boldsymbol{L}}^T = \boldsymbol{L}(\boldsymbol{I}_r - \boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{L}$. This is positive semidefinite because $\boldsymbol{V}\boldsymbol{V}^T \preceq \boldsymbol{I}_r$ from Lemma 1 with $(\boldsymbol{M}, \boldsymbol{N}) = (\boldsymbol{0}, \boldsymbol{V})$.

We now argue that $\boldsymbol{A} - \boldsymbol{B}$ is $C_k$-block diagonal. If $p = 0$ or $C_k = [n]$ so that $\boldsymbol{B} = \boldsymbol{O}$, then this is clear. If $1 \le p \le r$, then note that $\boldsymbol{V}\boldsymbol{V}^T \in \mathbb{R}^{r \times r}$ serves as the orthogonal projection matrix from $\mathbb{R}^r$ onto the subspace $S$. Since $\boldsymbol{L}_{i\cdot}^T \in S$ for each $i \notin C_k$, we see that $\boldsymbol{V}\boldsymbol{V}^T \boldsymbol{L}_{\cdot i}^T = \boldsymbol{L}_{\cdot i}^T$ for $i \notin C_k$. Therefore, $(i, j) \notin C_k \times C_k$ ensures

$$
\boldsymbol{A}_{ij} - \boldsymbol{B}_{ij} = \boldsymbol{A}_{ij} - \boldsymbol{L}_{i\cdot}\boldsymbol{V}\boldsymbol{V}^T \boldsymbol{L}_{j\cdot}^T = \boldsymbol{A}_{ij} - \boldsymbol{L}_{i\cdot}\boldsymbol{L}_{j\cdot}^T = 0,
$$

i.e., $\boldsymbol{A} - \boldsymbol{B}$ is $C_k$-block diagonal.

The preceding two paragraphs establish $\boldsymbol{B} \in \mathcal{F}_k(\boldsymbol{A})$. We now prove that $\boldsymbol{B}$ is in fact minimum in $\mathcal{F}_k(\boldsymbol{A})$. That is, we prove $\boldsymbol{B} \preceq \boldsymbol{Z}$ for all $\boldsymbol{Z} \in \mathcal{F}_k(\boldsymbol{A})$. So let $\boldsymbol{Z} \in \mathcal{F}_k(\boldsymbol{A})$ be arbitrary. If $\boldsymbol{Z} = \boldsymbol{A}$, we already know $\boldsymbol{B} \preceq \boldsymbol{Z}$. Likewise, if $\boldsymbol{B} = \boldsymbol{O}$, the result is clear. So assume $p \ge 1$ and $q := \text{rank}(\boldsymbol{A} - \boldsymbol{Z}) \ge 1$.

Since $\boldsymbol{O} \preceq \boldsymbol{A} - \boldsymbol{Z}$, there exists a rank-$q$ $\boldsymbol{F} \in \mathbb{R}^{n \times q}$ such that $\boldsymbol{A} - \boldsymbol{Z} = \boldsymbol{F}\boldsymbol{F}^T$. Hence, $\boldsymbol{Z} = \boldsymbol{A} - \boldsymbol{F}\boldsymbol{F}^T = \boldsymbol{L}\boldsymbol{L}^T - \boldsymbol{F}\boldsymbol{F}^T \succeq \boldsymbol{O}$. It follows that $\text{null}(L^T) \subseteq \text{null}(F^T)$, which implies $\text{range}(F) \subseteq \text{range}(L)$. So there exists rank-$q$ $\boldsymbol{G} \in \mathbb{R}^{r \times q}$ such that $\boldsymbol{F} = \boldsymbol{L}\boldsymbol{G}$. We also see that, for all $i \notin C_k$, the equation $0 = [\boldsymbol{A} - \boldsymbol{Z}]_{ii} = \|\boldsymbol{F}_{i\cdot}\|^2 = \|\boldsymbol{L}_{i\cdot}\boldsymbol{G}\|^2$ implies that each column of $\boldsymbol{G}$ lies in $S^\perp$, the orthogonal complement of the linear subspace $S$. Thus, we have $\boldsymbol{Z} - \boldsymbol{B} = \boldsymbol{L}\boldsymbol{L}^T - \boldsymbol{F}\boldsymbol{F}^T - \widehat{\boldsymbol{L}}\widehat{\boldsymbol{L}}^T = \boldsymbol{L}(\boldsymbol{I}_r - \boldsymbol{G}\boldsymbol{G}^T - \boldsymbol{V}\boldsymbol{V}^T)\boldsymbol{L}^T$, where $\boldsymbol{I}_r - \boldsymbol{G}\boldsymbol{G}^T - \boldsymbol{V}\boldsymbol{V}^T \succeq \boldsymbol{O}$ by Lemma 1 with $(\boldsymbol{M}, \boldsymbol{N}) = (\boldsymbol{G}, \boldsymbol{V})$. □

We are now ready to present Algorithm 2, based on the subroutine Algorithm 1, for computing a minimal element of $\mathcal{F}(\boldsymbol{A})$ for arbitrary $\boldsymbol{A} \in \mathbb{S}^n$. Note that Algorithm 1 requires a factorization of its input and provides a factorization of its output, which is useful for repeatedly calling Algorithm 1 within Algorithm 2. Theorem 1 establishes the correctness of Algorithm 2 via Lemma 2.

**Lemma 2** *Let* $\boldsymbol{A} \in \mathbb{S}^n$ *and* $k \in \{1, \ldots, r\}$. *If* $\boldsymbol{B} \in \mathcal{F}(\boldsymbol{A})$, *then* $\mathcal{F}_k(\boldsymbol{B}) \subseteq \mathcal{F}(\boldsymbol{A})$.

*Proof* Let $\boldsymbol{M} \in \mathcal{F}_k(\boldsymbol{B})$, i.e., $\boldsymbol{O} \preceq \boldsymbol{M} \preceq \boldsymbol{B}$ and $\boldsymbol{B} - \boldsymbol{M}$ is $C_k$-block diagonal. Since $\boldsymbol{B} \in \mathcal{F}(\boldsymbol{A})$, we also know $\boldsymbol{A} - \boldsymbol{B}$ is $\mathcal{C}$-block diagonal. Hence $\boldsymbol{B} \succeq \boldsymbol{O}$ and $\boldsymbol{A} - \boldsymbol{M} = (\boldsymbol{A} - \boldsymbol{B}) + (\boldsymbol{B} - \boldsymbol{M})$ is $\mathcal{C}$-block diagonal. So $\boldsymbol{M} \in \mathcal{F}(\boldsymbol{A})$. □

**Theorem 1** *Algorithm 2 correctly calculates* $\boldsymbol{B} \in \text{Minimal}(\mathcal{F}(\boldsymbol{A}))$.

**Algorithm 2** Calculate a minimal element of $\mathcal{F}(\boldsymbol{A})$

**Input:** $\boldsymbol{A} \in \mathbb{S}^n$.
**Output:** $\boldsymbol{B} \in \text{Minimal}(\mathcal{F}(\boldsymbol{A}))$.
1: Choose $\mathcal{C}$-block diagonal $\Delta$ such that $\boldsymbol{A} + \Delta \succeq \boldsymbol{O}$.
2: Initialize $\boldsymbol{B}^0 := \boldsymbol{A} + \Delta$.
3: **for** $k = 1, 2, \ldots, r$ **do**
4: Calculate $\boldsymbol{B}^k := \text{minimum}(\mathcal{F}_k(\boldsymbol{B}^{k-1}))$ via Algorithm 1.
5: **end for**
6: Set $\boldsymbol{B} := \boldsymbol{B}^r$.

*Proof* We first argue that $\boldsymbol{B} \in \mathcal{F}(\boldsymbol{A} + \Delta)$. Let $\boldsymbol{A} + \Delta =: \boldsymbol{B}^0, \boldsymbol{B}^1, \ldots, \boldsymbol{B}^r =: \boldsymbol{B}$ be the sequence generated by Algorithm 2. Clearly $\boldsymbol{B}^0 \in \mathcal{F}(\boldsymbol{A} + \Delta)$. For induction, assume $\boldsymbol{B}^{k-1} \in \mathcal{F}(\boldsymbol{A} + \Delta)$. Then $\boldsymbol{B}^k \in \mathcal{F}_k(\boldsymbol{B}^{k-1}) \subseteq \mathcal{F}(\boldsymbol{A} + \Delta)$ by Lemma 2. So $\boldsymbol{B}^r \in \mathcal{F}(\boldsymbol{A} + \Delta)$.

Next we prove that $\mathcal{F}_k(\boldsymbol{B}) = \{\boldsymbol{B}\}$ for all $k = 1, \ldots, r$. Fix $k$. If $\boldsymbol{B} = \boldsymbol{O}$ then the assertion holds easily. For $\boldsymbol{B}^r =: \boldsymbol{B} \neq \boldsymbol{O}$, we assume on the contrary the existence of $\boldsymbol{Z} \in \mathcal{F}_k(\boldsymbol{B}^r)$ with $\boldsymbol{Z} \neq \boldsymbol{B}^r$. Define $\widetilde{\boldsymbol{B}} := \boldsymbol{B}^k - (\boldsymbol{B}^r - \boldsymbol{Z})$. Then

$$\boldsymbol{B}^{k-1} \succeq \boldsymbol{B}^k \succeq \widetilde{\boldsymbol{B}} = \left(\boldsymbol{B}^k - \boldsymbol{B}^{k+1}\right) + \left(\boldsymbol{B}^{k+1} - \boldsymbol{B}^{k+2}\right) + \cdots + \left(\boldsymbol{B}^{r-1} - \boldsymbol{B}^r\right) + \boldsymbol{Z} \succeq \boldsymbol{O}$$

and the fact that $\boldsymbol{B}^r - \boldsymbol{Z} = \boldsymbol{B}^k - \widetilde{\boldsymbol{B}}$ is $C_k$-block diagonal implies also that $\boldsymbol{B}^{k-1} - \widetilde{\boldsymbol{B}} = (\boldsymbol{B}^{k-1} - \boldsymbol{B}^k) + (\boldsymbol{B}^k - \widetilde{\boldsymbol{B}})$ is $C_k$-block diagonal. Hence, $\widetilde{\boldsymbol{B}} \in \mathcal{F}_k(\boldsymbol{B}^{k-1})$ and $\boldsymbol{B}^k \neq \widetilde{\boldsymbol{B}} \preceq \boldsymbol{B}^k$, but this contradicts the fact that $\boldsymbol{B}^k = \text{minimum}(\mathcal{F}_k(\boldsymbol{B}^{k-1}))$.

We are now ready to prove $\boldsymbol{B} \in \text{Minimal}(\mathcal{F}(\boldsymbol{A} + \Delta))$. Assume on the contrary that there exists $\widetilde{\boldsymbol{B}} \in \mathcal{F}(\boldsymbol{A} + \Delta)$ such that $\boldsymbol{B} \neq \widetilde{\boldsymbol{B}} \preceq \boldsymbol{B}$. Let $\boldsymbol{D} := \boldsymbol{B} - \widetilde{\boldsymbol{B}}$. Then $\boldsymbol{D}$ is nonzero, positive semidefinite, and $\mathcal{C}$-block diagonal. So $\boldsymbol{D}_{C_k C_k} \succeq \boldsymbol{O}$ is nonzero for some $k$. Let $\boldsymbol{D}_k$ be the $C_k$-block diagonal matrix with block $\boldsymbol{D}_{C_k C_k}$, and define $\boldsymbol{Z} := \boldsymbol{B} - \boldsymbol{D}_k$. Then we see that $\boldsymbol{Z} \in \mathcal{F}_k(\boldsymbol{B})$, and so $\boldsymbol{Z} = \boldsymbol{B}$ by the previous paragraph. However, this is a contradiction.

Finally, we know $\boldsymbol{B} \in \text{Minimal}(\mathcal{F}(\boldsymbol{A} + \Delta)) = \text{Minimal}(\mathcal{F}(\boldsymbol{A}))$ by Proposition 5. □

### 3.5 Our practical choice of matrices $\boldsymbol{B}_i$

Section 3.3 argued that each $\boldsymbol{B}_i$ should be a minimal member of $\mathcal{F}(\boldsymbol{A}_i)$ in order to improve the bound gotten from the mixed SOCP-SDP relaxation (10), and Sect. 3.4 presented Algorithm 2 to compute a member of $\text{Minimal}(\mathcal{F}(\boldsymbol{A}_i))$. We will see in Sect. 4 that, as intended, Algorithm 2 computes $\boldsymbol{B}_i$ efficiently.

Given input $\boldsymbol{A}_i$, Algorithm 2 relies in step 1 on the choice of a $\mathcal{C}$-block diagonal shift $\Delta_i$ such that $\boldsymbol{A}_i + \Delta_i \succeq \boldsymbol{O}$, so we now discuss choices for $\Delta_i$. Of course, there are many possible choices, but it is important that the choice be made quickly so that the overall time to construct and solve (10) is less than directly solving the SDP (2). We suggest and study two choices of $\Delta_i$, both of which are based on the spectral decomposition, which is reasonably quick to compute for the sizes of problems that we consider in Sect. 4.

Before stating our choices of $\Delta_i$, we first need some definitions. Given any symmetric matrix $\boldsymbol{M}$, define $\rho(\boldsymbol{M}) := -\lambda_{\min}[\boldsymbol{M}]$ so that $\boldsymbol{M} + \rho(\boldsymbol{M})\boldsymbol{I} \succeq \boldsymbol{O}$. Also decompose $\boldsymbol{M}$ into the sum of two matrices $\boldsymbol{M}(\mathcal{C})$ and $\overline{\boldsymbol{M}}(\mathcal{C}) := \boldsymbol{M} - \boldsymbol{M}(\mathcal{C})$ such that $\boldsymbol{M}(\mathcal{C})$ is $\mathcal{C}$-block diagonal and $\overline{\boldsymbol{M}}(\mathcal{C})$ has nonzeros only in the complementary positions. In other words, the blocks of $\boldsymbol{M}(\mathcal{C})$ are precisely $\boldsymbol{M}_{C_k C_k}$ $(k = 1, \ldots, r)$.

Our first choice for $\Delta_i$ takes diagonal $\Delta_i := \rho(\boldsymbol{A}_i)\boldsymbol{I}$ and applies Algorithm 2 to $\boldsymbol{A}_i + \Delta_i \succeq \boldsymbol{O}$. This choice was also discussed at the beginning of Sect. 3. We call it the *first shift*.

Our second choice is motivated by the observation that, in a certain sense, the choice of $\boldsymbol{B}_i \in \text{Minimal}(\mathcal{F}(\boldsymbol{A}_i))$ should not depend on the $\mathcal{C}$-block diagonal portion $\boldsymbol{A}_i(\mathcal{C})$ of $\boldsymbol{A}_i$, where $\boldsymbol{A}_i = \boldsymbol{A}_i(\mathcal{C}) + \overline{\boldsymbol{A}}_i(\mathcal{C})$ as defined above. This is because the set $\mathcal{F}(\boldsymbol{A}_i)$ may be defined equivalently as $\{\boldsymbol{M} \succeq \boldsymbol{O} : \overline{\boldsymbol{M}}(\mathcal{C}) = \overline{\boldsymbol{A}}_i(\mathcal{C})\}$, i.e., it may be defined in a way that does not depend on $\boldsymbol{A}_i(\mathcal{C})$. Proposition 5 further supports this observation because $\mathcal{F}(\boldsymbol{A}_i) = \mathcal{F}(\overline{\boldsymbol{A}}_i(\mathcal{C}))$ as $\boldsymbol{A}_i$ and $\overline{\boldsymbol{A}}_i(\mathcal{C})$ differ by $\boldsymbol{A}_i(\mathcal{C})$. So, loosely speaking, our second choice of shift is to apply the first shift to $\overline{\boldsymbol{A}}_i(\mathcal{C})$. More precisely, define

$$\Delta_i := \rho\big(\overline{\boldsymbol{A}}_i(\mathcal{C})\big)\boldsymbol{I} - \boldsymbol{A}_i(\mathcal{C})$$

so that $\boldsymbol{A}_i + \Delta_i = \overline{\boldsymbol{A}}_i(\mathcal{C}) + \rho(\overline{\boldsymbol{A}}_i(\mathcal{C}))\boldsymbol{I} \succeq \boldsymbol{O}$. We call this the *second shift*.

## 4 Computational results

In this section, we describe our computational experience with the framework introduced and described in Sect. 3. Our first goal is to verify that our technique—including preprocessing time—can speed up the solution of the mixed SOCP-SDP (10) compared to solving the SDP (2). Of course, the resultant bound is weaker, and so our second goal is to quantify this bound loss. We test the framework on a set of interesting instances from the literature, and we also test the two types of shifts (*first* and *second*) introduced in Sect. 3.5.

One important detail when applying the framework to a particular instance is the choice of partition $\mathcal{C} := \{C_1, \ldots, C_r\}$ of $[n]$. Clearly the precise structure of the partition will have a big impact on the solution time of (10) as well as the resultant bound. As we describe in the next subsection, we take a straightforward approach for choosing $\mathcal{C}$, one which depends only on the value of $n$ of an instance. Although this approach does not take full advantage of the data $(\boldsymbol{A}_i, \boldsymbol{a}_i, \alpha_i)$, we feel this approach allows us to study the basic features of our framework.

### 4.1 The instances and relaxations

We collected a total of 400 instances of (1) from the literature, which consisted of three groups: (i) 199 instances of the maximum cut (MaxCut) problem coming from [8] and [16] (21 instances of the *Gset* library and 178 instances of the *BiqMac* library, respectively); (ii) 64 instances of binary quadratic programming (BinQP) coming from the *BiqMac* library [16]; and (iii) 36 instances from GlobalLib [5] having bounded feasible sets. In particular, all instances had between $n = 16$ and $n = 800$

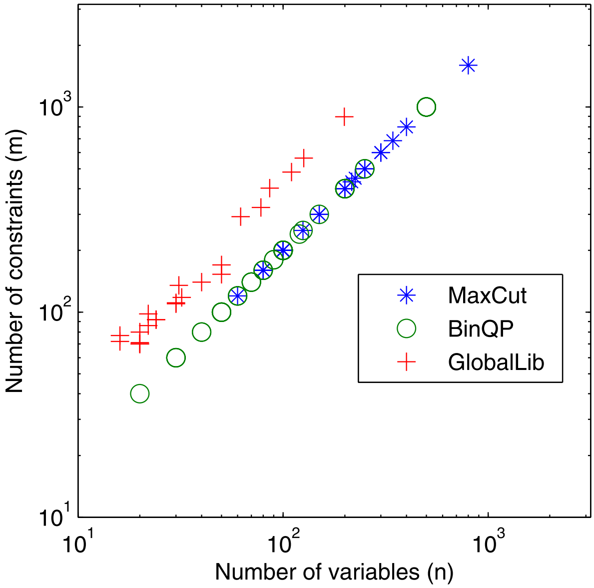



**Fig. 1** The sizes of the 400 MaxCut, BinQP, and GlobalLib instances. Some single points depict multiple instances of the same size

variables. The instance sizes (number of variables and constraints) are depicted in Fig. 1 on a log-log scale. We note that all instances were formulated precisely in the standard form of (1). For example, quadratic equations were split into two quadratic inequalities. In addition, we made sure to bound the diagonal of $X$ as shown in (10).

To test the framework, we next created partitions $\mathcal{C}$ as utilized in Sect. 3. Specifically, for each QCQP instance, we created and tested 4 different partitions. The first partition consists simply of all variables in a single set $C_1 = \{1, \ldots, n\}$, which yields an instance of (10) that is equivalent to (2); this is our "base case" partition. The second partition is a refinement of the first gotten by (approximately) halving the first partition:

$$C_1 = \left\{1, \ldots, \left\lceil \frac{n}{2} \right\rceil \right\}, \; C_2 = \left\{ \left\lceil \frac{n}{2} \right\rceil + 1, \ldots, n \right\}.$$

Likewise, the third partition approximately halves the second, and the fourth approximately halves the fourth. For example, if $n = 16$ for a QCQP instance, then we would create the 4 partitions

$$\begin{aligned}
&C_1 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\};\\
&C_1 = \{1, 2, 3, 4, 5, 6, 7, 8\}, \quad C_2 = \{9, 10, 11, 12, 13, 14, 15, 16\};\\
&C_1 = \{1, 2, 3, 4\}, \quad C_2 = \{5, 6, 7, 8\}, \quad C_3 = \{9, 10, 11, 12\},\\
&\quad C_4 = \{13, 14, 15, 16\};\\
&C_1 = \{1, 2\}, \quad C_2 = \{3, 4\}, \quad C_3 = \{5, 6\}, \quad C_4 = \{7, 8\}, \quad C_5 = \{9, 10\},\\
&\quad C_6 = \{11, 12\}, \quad C_7 = \{13, 14\}, \quad C_8 = \{15, 16\}.
\end{aligned}$$

For each QCQP, every partition gives an instance of (10). So our 400 chosen instances of (1) yield a total of 1,600 instances of (10). While our choice of partitions is somewhat arbitrary, it is designed to provide a variety of sizes of partitions that we investigate. It would be interesting to choose the partition using, for example, a clever heuristic to improve the subsequent bound, but we do not do so here.

### 4.2 Algorithm variants and implementation

The 1,600 instances of (10) described in the previous subsection are solved by several algorithm variants that we describe now. The variants are based on two design choices, each with two options, yielding a total of four algorithm variants.

The first design choice is the type of shift employed, either *first* or *second*, as described in Sect. 3.5. The second design choice is whether to continue running Algorithm 2 after the shift in step 1 is made. In other words, we can terminate Algorithm 2 immediately after step 1 and simply use $\boldsymbol{B}_i := \boldsymbol{A}_i + \Delta_i$ to create the SOCP-SDP relaxation (10). We denote the four variants as *1N* (first shift without continuing Algorithm 2), *1Y* (first shift with the full Algorithm 2), *2N* (second shift without continuing Algorithm 2), and *2Y* (second shift with the full Algorithm 2). The letters *N* and *Y* are meant to indicate "no" and "yes" for the full Algorithm 2.

The purpose of these four variants is to isolate the effects of different aspects of our framework. For example, by comparing 1N with 1Y (or 2N with 2Y), we can determine if calculating a minimal element $\boldsymbol{B}_i \in \text{Minimal}(\mathcal{F}(\boldsymbol{A}_i))$ has added benefits over simply constructing the relaxation (10) with a $\mathcal{C}$-block structure after the shift. Further, comparing 1N with 2N (or 1Y with 2Y), we can determine the different effects of the two types of shifts. Finally, we may also hope to find the best overall choice of the four algorithm variants.

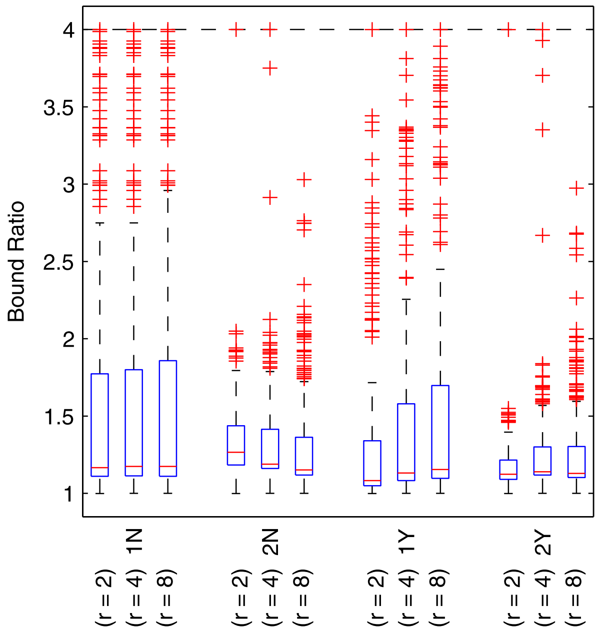
For each of the four variants and any of the 1,600 instances, the mixed SOCP-SDP (10) is converted to standard block-diagonal form derived from the $\mathcal{C}$-block structure and Proposition 3. This involves adding just a few extra variables and constraints (at most $n$). The resultant problems are solved with a default installation of SeDuMi on an Ubuntu Linux computer having an Intel Core 2 Quad CPU running at 2.4 GHz with 4 MB cache and 4 GB RAM. Even though the CPU has multiple cores, we limit Matlab to using at most one core.

### 4.3 Comparisons

We now compare the four different algorithm variants on the 400 instances of (2), each of which gives rise to four instances of (10) based on different partitions $\mathcal{C} := \{C_1, \ldots, C_r\}$. Recall that, for each instance, the four partitions have $r = 1, 2, 4, 8$ blocks, respectively. When $r = 1$, we have the "base case," which is equivalent to solving the SDP relaxation (2).

Consider a single instance of (2) solved by a single variant (1N, 1Y, 2N, or 2Y) for the four block values $r = 1, 2, 4, 8$. This gives rise to four lower bounds $\{b_r : r = 1, 2, 4, 8\}$ on the optimal value of (2) and four computation times $\{t_r : r = 1, 2, 4, 8\}$. In particular, $t_r$ is the total time to apply our framework including calculating the shift for each $i$, applying the full Algorithm 2 for each $i$ (as required), and setting up and

**Fig. 2** Box plots for the bound ratios $\beta_r$ over all pairs of algorithm variants (1N, 2N, 1Y, 2Y) and block values ($r = 2, 4, 8$)




solving (10). We compare our framework to the base-case SDP by calculating the six ratios

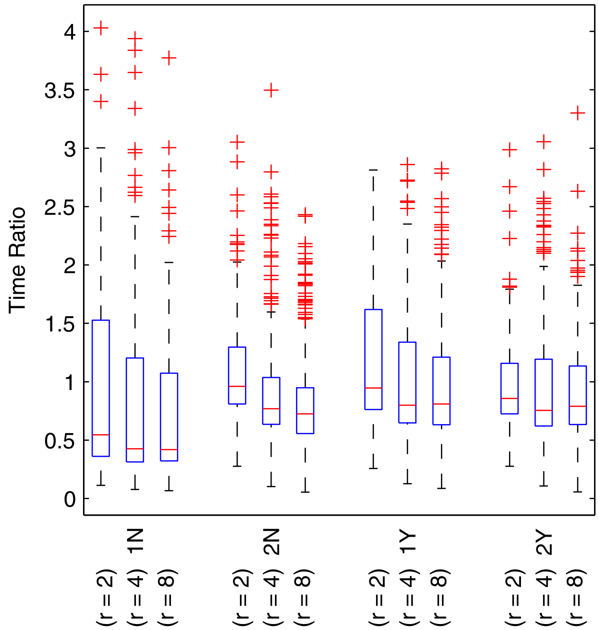
$$\beta_r := \frac{b_r}{b_1} \quad (r = 2, 4, 8) \quad \text{and} \quad \tau_r := \frac{t_r}{t_1} \quad (r = 2, 4, 8).$$

(Note that all instances had $b_1 \leq -0.01$ so that the denominator defining $\beta_r$ was sufficiently far from 0 and so that each $b_r \leq -0.01$ and each $\beta_r \geq 1$.) By examining these ratios $\beta_r$ and $\tau_r$, we can compare our framework's bounds and times to the SDP bounds and times on a standardized, relative scale. For example, if $\beta_r = 1.05$ and $\tau_r = 0.90$, it means that our framework degraded the bound 5 % but took 10 % less time.

For each combination of an algorithm variant and a block value $r \in \{2, 4, 8\}$, Fig. 2 displays a box plot of the bound ratios $\beta_r$ gotten over all 400 instances of (2). In a similar fashion, Fig. 3 shows the time ratios $\tau_r$. The plots were created using Matlab, and we point out the visual features that explain how to read the plots. For each box plot, the blue box spans the 25th and 75th percentiles of the data, and the red line within the blue box indicates the median (50th percentile). The vertical, black, dashed lines that extend from the blue box and terminate with horizontal, black "whiskers" show the rest of the data, which is not considered to be outliers. According to Matlab's defaults, a data point is considered to be an outlier if it is smaller than $p_{25} - 1.5(p_{75} - p_{25})$ or larger than $p_{75} + 1.5(p_{75} - p_{25})$, where $p_{25}$ and $p_{75}$ are the 25th and 75th percentiles, i.e., edges of the blue box. Outliers are indicted by red plus signs. In addition, Fig. 2 also uses Matlab's "extreme mode," which collapses the outliers beyond a certain point onto a single, dashed horizontal line to save space.

We first discuss the bound ratios in Fig. 2 and make several observations:

**Fig. 3** Box plots for the time ratios $\tau_r$ over all pairs of algorithm variants (1N, 2N, 1Y, 2Y) and block values ($r = 2, 4, 8$)




- For each of the variants 1N and 1Y, we see that the bound ratio generally worsens as $r$ increases, while for 2N and 2Y, the bound ratio generally improves or stays the same. This indicates that, irrespective of the full use of Algorithm 2, the second shift is better for improving or maintaining the bound quality as $r$ increases.

  Actually, for 1N, the decreasing bounds are expected because the $(r + 1)$-st relaxation is itself a relaxation of the $r$-th relaxation. This is true since the $\mathcal{C}$ for $r + 1$ is a finer partition than the $\mathcal{C}$ for $r$ and since the first shift is not dependent on the partition $\mathcal{C}$. In contrast, for 2N, the second shift is able to counteract the expected loss of bound due to the finer partition.
- Note, however, that for a fixed $r$, neither shift dominates the other. For example, when $r = 2$, 1N provides better bounds than 2N, but when $r = 8$, the bounds provided by 1N are worse.
- For each value of $r$, the bound ratios for 1N are worse than for 1Y. Likewise, for each value of $r$, the bound ratios for 2N are worse than for 2Y. This indicates that, irrespective of the type of shift, the full use of Algorithm 2 keeps the bound quality closer to that of the SDP relaxation.
- Of the four variants, 2Y (second shift with full Algorithm 2) keeps the bounds consistently low for all values of $r$. As such, it seems to be the best performing variant overall.

We next discuss the time ratios in Fig. 3:

- Generally speaking, for each variant, increasing $r$ results in relaxations that are faster to solve, and the median time ratios are well below the value of 1, which means that our framework is generally faster than solving the SDP directly.
- However, there is considerable variation with many individual ratios above 1. The smallest ratios are near 0.5, meaning that our framework is, roughly speaking, at

most twice as fast as the SDP. In particular, we do not see a full order-of-magnitude speed up.

- Variant 1N is clearly the fastest variant, while the other three variants require roughly the same amount of time.

In both figures, we also see an interesting trend regarding the variation of the data as depicted by the heights of the blue boxes. Variants 2N and 2Y clearly exhibit less variation than 1N and 1Y. The second shift appears to be largely responsible for this decrease in variation, but even in Fig. 2 for the bounds, the variation in 2Y is a bit less than the variation in 2N, which shows a slight improvement due to the full Algorithm 2.

While the comparisons just made using Figs. 2 and 3 are imperfect—especially due to the overall variation seen in the bound and time ratios due to the outliers—the general trends indicate that our framework can provide faster relaxations but with a corresponding loss in bound quality. It appears that variant 2Y, which uses the second shift and Algorithm 2, achieves the best bound quality relative to the SDP relaxation, while taking about the same amount of time as variants 2N and 1Y. (Variant 1N is noticeably faster, but its bounds are less reliable.)

## 5 Conclusions

When attempting to solve difficult QCQPs, the bounds provided by SDP relaxations can be tight and hence quite useful, but they can also be time consuming to calculate. This paper has presented a framework for constructing mixed SOCP-SDPs that provide faster, but weaker, bounds. Our framework is unique compared to other related approaches in the literature as it allows one to control the solution speed of the SOCP-SDP (via its block structure) while simultaneously working to improve the bound quality via the idea of minimum and minimal elements, which respect the block structure. We found that the combination of the second shift and the full Algorithm 2 (variant 2Y) performed the best overall.

There are many avenues to improve our framework. In Sect. 4, we tested simplistic choices for the partition $\mathcal{C} := \{C_1, \ldots, C_r\}$ that do not take into account the actual data $\{(\boldsymbol{A}_i, \boldsymbol{a}_i, \alpha_i)\}$. We did so with the intent of just testing the basic behavior of our framework, but it would be very interesting to design heuristics that choose $\mathcal{C}$ intelligently to preserve the bound quality even beyond the second shift and the full use of Algorithm 2.

Another way to extend our approach is to allow $\mathcal{C}$ to be a covering of $[n]$ rather than just a partition, that is, to allow the elements $C_k$ to overlap. As long as we can extend Proposition 3 to this case, the framework will extend easily, and extending Proposition 3 relies in turn on the chordal-graph structure of the matrix

$$\begin{pmatrix} 1 & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{Y} \end{pmatrix},$$

where $\boldsymbol{Y}_{C_k C_k} = \boldsymbol{X}_{C_k C_k}$ for all $k = 1, \ldots, r$ and zero otherwise (see again [7]). The flexibility of choosing overlapping $\boldsymbol{C}_k$ should allow further preservation of the bound without dramatic increases in the computation time.

## References


1. Burer, S., Monteiro, R.: A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Math. Program., Ser. B **95**, 329–357 (2003)
2. de Klerk, E., Sotirov, R.: Exploiting group symmetry in semidefinite programming relaxations of the quadratic assignment problem. Math. Program., Ser. A **122**(2), 225–246 (2010)
3. Fujie, T., Kojima, M.: Semidefinite programming relaxation for nonconvex quadratic programs. J. Glob. Optim. **10**(4), 367–380 (1997)
4. Fukuda, M., Kojima, M., Murota, K., Nakata, K.: Exploiting sparsity in semidefinite programming via matrix completion. I. General framework. SIAM J. Optim. **11**(3), 647–674 (2000)
5. See the website: www.gamsworld.org/global/globallib/globalstat.htm
6. Goemans, M., Williamson, D.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM **42**, 1115–1145 (1995)
7. Grone, R., Johnson, C., Sá, E., Wolkowicz, H.: Positive definite completions of partial Hermitian matrices. Linear Algebra Appl. **58**, 109–124 (1984)
8. Helmberg, C., Rendl, F.: A spectral bundle method for semidefinite programming. SIAM J. Optim. **10**, 673–696 (2000)
9. Kim, S., Kojima, M.: Second order cone programming relaxation of nonconvex quadratic optimization problems. Optim. Methods Softw. **15**(3–4), 201–224 (2001)
10. Kojima, M., Tunçel, L.: Cones of matrices and successive convex relaxations of nonconvex sets. SIAM J. Optim. **10**(3), 750–778 (2000)
11. Lasserre, J.B.: Global optimization with polynomials and the problem of moments. SIAM J. Optim. **11**(3), 796–817 (2001)
12. Laurent, M., Rendl, F.: Semidefinite programming and integer programming. In: Karen Aardal, R.W., Nemhauser, G. (eds.) Handbook on Discrete Optimization, pp. 393–514. Elsevier, Amsterdam (2005)
13. Lovász, L., Schrijver, A.: Cones of matrices and set-functions and 0-1 optimization. SIAM J. Optim. **1**, 166–190 (1991)
14. Monteiro, R.D.C.: First- and second-order methods for semidefinite programming. Math. Program. **97**, 209–244 (2003)
15. Nakata, K., Fujisawa, K., Fukuda, M., Kojima, M., Murota, K.: Exploiting sparsity in semidefinite programming via matrix completion. II. Implementation and numerical results. Math. Program., Ser. B **95**(2), 303–327 (2003). Computational semidefinite and second order cone programming: the state of the art
16. Rendl, F., Rinaldi, G., Wiegele, A.: Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. Math. Program. **121**(2), 307 (2010)
17. Saxena, A., Bonami, P., Lee, J.: Convex relaxations of non-convex mixed integer quadratically constrained programs: projected formulations. Math. Program. (2010). doi:10.1007/s10107-010-0340-3
18. Sherali, H.D., Adams, W.P.: A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems. Kluwer, Dordrecht (1997)
19. Zhao, X., Sun, D., Toh, K.: A Newton-CG augmented Lagrangian method for semidefinite programming. SIAM J. Optim. **20**, 1737–1765 (2010)
20. Zheng, X., Sun, X., Li, D.: Nonconvex quadratically constrained quadratic programming: best d.c. decompositions and their sdp representations. J. Glob. Optim. **50**, 695–712 (2011)