Solving Lasserre relaxations of unconstrained binary quadratic optimization problems by an interior-point method

Soodeh Habibi, Michal Kočvara, Michael Stingl

Solving Lasserre relaxations of unconstrained binary quadratic optimization problems by an interior-point method

Soodeh Habibi[1], Michal Kočvara[1,2*] and Michael Stingl[3]

[1]School of Mathematics, University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK.
[2]Institute of Information Theory and Automation, Czech Academy of Sciences, Pod vodárenskou věží 4, Prague 8, 18208, Czech Republic.
[3]Applied Mathematics (Continuous Optimization), Friedrich-Alexander-Universität Erlangen-Nürnberg, Cauerstraße 11, Erlangen, 91058, Germany.

*Corresponding author(s). E-mail(s): m.kocvara@bham.ac.uk;
Contributing authors: s.habibi@bham.ac.uk; michael.stingl@fau.de;

Abstract

The aim of this paper is to solve linear semidefinite programs arising from higher-order Lasserre relaxations of unconstrained binary quadratic optimization problems. For this we use an interior point method with a preconditioned conjugate gradient method solving the linear systems. The preconditioner utilizes the low-rank structure of the solution of the relaxations. In order to fully exploit this, we need to re-write the moment relaxations. To treat the arising linear equality constraints we use an ℓ_1-penalty approach within the interior-point solver. The efficiency of this approach is demonstrated by numerical experiments with the MAXCUT and other randomly generated problems and a comparison with a state-of-the-art semidefinite solver and the ADMM method. As a by-product, we observe that the second-order relaxation is often high enough to deliver a globally optimal solution of the original problem.

Keywords: Binary quadratic optimization, Lasserre hierarchy, semidefinite optimization, interior-point methods, preconditioned conjugate gradients, MAXCUT problem.

MSC Classification: 90C22 , 90C51 , 65F08 , 74P05

1 Introduction

Unconstrained binary quadratic optimization problems (UBQP) represent a surprisingly wide class of important optimization problems; see, e.g., the comprehensive overview [1]. The famous MAXCUT problem is a typical representative of this class. It is thus not surprising that they attract a great deal of attention among algorithm and software developers. Some of the most efficient algorithms for finding a global solution of UBQP combine branch-and-bound techniques with semidefinite programming (SDP) relaxations to obtain good lower bounds. These are represented by software like Biq Mac [2], BiqCrunch [3] and BiqBin [4].

The SDP relaxations are typically based on Shor's relaxation [5] which is equivalent to the first-order Lasserre relaxation [6]. While higher-order relaxations would deliver much tighter lower bounds (if not exact solutions), the dimensions of the arising SDP problems are considered prohibitively large already for medium-sized UBQPs. For this reason, many authors proposed various techniques to strengthen the first-order relaxations; see, e.g., [7–9] and the references therein.

Our goal is to show that (at least) the second-order Lasserre relaxations are solvable by a specialized SDP software. We will also demonstrate that the second-order relaxations are, indeed, superior to the first-order ones and, in many cases, already deliver global solutions of the UBQP. This may allow one to avoid the branch-and-cut machinery, whose behaviour is known to be sometimes unpredictable, and replace it entirely by robust algorithms of known complexity.

Our aim is to solve UBQP

$$\min_{x\in\mathbb{R}^s} x^\top Q x \quad \text{subject to} \quad x_i \in \mathcal{B}\,, \quad i = 1,\dots,s \tag{1}$$

with a symmetric matrix $Q \in \mathbb{R}^{s\times s}$, where $\mathcal{B}$ is either the set $\{0,1\}$ or the set $\{-1,1\}$. *We do not assume any sparsity in* Q, it is a generally dense matrix. In order to find a global optimum, we use Lasserre hierarchy of SDP problems—relaxations—of growing dimension [6]. The SDP relaxations have the form

$$\begin{aligned} &\min_{y\in\mathbb{R}^n} q^\top y \\ &\text{subject to } M(y) := \sum_{i=1}^{n} y_i M_i - M_0 \succeq 0\,. \end{aligned} \tag{2}$$

Here M is a so-called moment matrix, a (generally) dense matrix of a very specific form. In particular, if the solution of (1) is unique and the order of the relaxation is big enough, then $\operatorname{rank} M(y^*) \in \{1,2\}$, where y^* is a solution of (2) (see Section 3 below).

These problems are known to be difficult to solve due to the quickly growing dimension of the problem with the order of the relaxation; see, e.g., [10]. Since we do not assume any sparsity in Q, we cannot use sparse techniques, such as those proposed in [11–13]. Instead, we propose to use our recently developed software *Loraine* [14]. Loraine uses a primal-dual predictor-corrector interior-point (IP)

method together with an iterative algorithm for the solution of the resulting linear systems. The iterative solver is a preconditioned Krylov-type method with a preconditioner utilizing low rank of the solution.

Several authors observed and confirmed by numerical experiments that SDP reformulation of UBQP (the first-order relaxation) can be efficiently solved by an SDP variant of the ADMM method, see [15]. We will show that not only this observation can be extended to higher-order relaxations but that inexact ADMM results can be efficiently used as a warm start for the IP algorithm in Loraine. This is due to the choice of the preconditioner used within the iterative solver in Loraine. Based on this observation, we will propose a new, *hybrid ADMM-Loraine algorithm* that, for a certain class of problems, will be superior to the single algorithms.

The paper is organized as follows. Section 2 introduces the low-rank SDP solver Loraine and the assumptions needed for its efficiency. In Section 3 we discuss the various forms of SDP relaxations of UBQP. Then, in Section 4 we briefly describe the ADMM algorithm for SDP and introduce the new hybrid algorithm, using an inexact ADMM result as a warm start for Loraine. The last Section 5 is devoted to numerical experiments using instances of the MAXCUT problems and randomly generated UBQPs.

Notation

We denote by $\mathbb{S}^m$, $\mathbb{S}^m_+$ and $\mathbb{S}^m_{++}$, respectively, the space of $m \times m$ symmetric matrices, positive semidefinite and positive definite matrices. The notation "svec" and "smat" refer to the symmetrized vectorization and its inverse operation, respectively. The symbol $\bullet$ denotes the Frobenius inner product of two matrices, $A \bullet B = \mathrm{trace}(A^\top B)$. Finally, the notation e_n (or just e) is used for the vector of all ones.

2 The solver Loraine

Loraine[1] is a general-purpose solver for any linear SDP developed by the authors and implemented in MATLAB and Julia. Compared to other general-purpose SDP software, it particularly targets at problems with low-rank solutions. To solve the arising systems of linear equations, it uses the preconditioned conjugate gradient method, as described in detail in [14]. The preconditioner, introduced in [16], is based on the assumption that the solution matrix has a small number of outlying eigenvalues. The user can choose between the direct and iterative solver, the type of preconditioner and the expected rank of the solution.

Loraine was developed for problems of the type

$$\min_{y \in \mathbb{R}^n,\, S \in \mathbb{S}^m,\, s_{\mathrm{lin}} \in \mathbb{R}^\nu} \; b^\top y \tag{3}$$

[1] github.com/kocvara/Loraine.m

$$\begin{aligned} \text{subject to } & \sum_{i=1}^{n} y_i A_i + S = C \\ & Dy + s_{\text{lin}} = d \\ & S \succeq 0,\ s_{\text{lin}} \geq 0 \end{aligned}$$

with the Lagrangian dual

$$\begin{aligned} \max_{X \in \mathbb{S}^m,\, x_{\text{lin}} \in \mathbb{R}^\nu} \quad & C \bullet X + d^\top x_{\text{lin}} \\ \text{subject to } A_i \bullet X + (D^\top x_{\text{lin}})_i &= b_i, \quad i = 1, \ldots, n \\ X \succeq 0,\ x_{\text{lin}} &\geq 0\,. \end{aligned} \tag{4}$$

In the following, we call (3) a problem of *primal form* and (4) a problem of *dual form*.

Loraine is efficient under the following assumptions:

Assumption 1 Problems (3),(4) are strictly feasible, i.e., there exist $X \in \mathbb{S}^m_{++}$, $x_{\text{lin}} \in \mathbb{R}^\nu_{++}$, $y \in \mathbb{R}^n$, $S \in \mathbb{S}^m_{++}$, $s_{\text{lin}} \in \mathbb{R}^\nu_{++}$, such that $A_i \bullet X + (D^\top x_{\text{lin}})_i = b_i$, $\sum_{i=1}^n y_i A_i + S = C$ and $Dy + s_{\text{lin}} = d$ (Slater's condition).

Assumption 2 Define the matrix $\mathbf{A} = [\operatorname{svec} A_1, \ldots, \operatorname{svec} A_n]$. We assume that any matrix-vector products with $\mathbf{A}$ and $\mathbf{A}^\top$ may each be computed in $\mathcal{O}(n)$ flops and memory.

Assumption 3 The inverse $(D^\top D)^{-1}$ exists and $(D^\top D)^{-1}$ together with the matrix-vector product with $(D^\top D)^{-1}$ may each be computed in $\mathcal{O}(n)$ flops and memory.

Assumption 4 The dimension of X is much smaller than the number of constraints in (4), i.e., $m \ll n$.

Assumption 5 Let X^* be the solution of (4). We assume that X^* has k outlying eigenvalues, i.e., that

$$(0 \leq)\ \lambda_1(X^*) \leq \cdots \leq \lambda(X^*)_{m-k} \ll \lambda(X^*)_{m-k+1} \leq \cdots \leq \lambda(X^*)_m\,,$$

where k is very small, typically smaller than 10 and, often, equal to 1. This includes the particular case when the rank of X^* is very small.

The last Assumption 5 is not satisfied by problem (2) so, in the next section, we will:

(i) re-write SDP relaxation (2) in the dual form (4) by introducing auxiliary variables and additional linear equality constraints;
(ii) treat the new linear equality constraints by ℓ_1-penalty approach, in order to replace equalities by inequalities, as required by the interior-point algorithm;
(iii) show that the matrix associated with the new linear inequality constraints is block diagonal with blocks of very small size and thus satisfies Assumption 2;
(iv) show that Assumption 4 is naturally satisfied for the re-written problem.

3 Forms of SDP relaxations

3.1 Lasserre hierarchy of moment problems

Let $\mathcal{B} = \{-1, 1\}$. By introducing a matrix variable $X = xx^\top$, problem (1) can be equivalently written as the following SDP problem with a rank constraint:

$$\begin{aligned} &\min_{X \in \mathbb{S}^s} Q \bullet X \\ &\text{subject to } X_{i,i} = 1, \quad i = 1, \ldots, s \\ &\qquad\qquad X \succeq 0 \\ &\qquad\qquad \operatorname{rank} X = 1\,. \end{aligned} \tag{5}$$

Remark 1 For $\mathcal{B} = \{0, 1\}$, we first use the substitution $\tilde{x} = 2x - 1$ to get a problem with objective function $\tilde{x}^\top Q\tilde{x} + 2e^\top Q\tilde{x}$ and constraints $\tilde{x}_i \in \{-1, 1\}$. Then we set $\widetilde{Q} = \begin{pmatrix} 0 & e^\top Q \\ Qe & Q \end{pmatrix} \in \mathbb{S}^{s+1}$, introduce a variable $\widetilde{X} \in \mathbb{S}^{s+1}$, $\widetilde{X} = \begin{pmatrix} 1 \\ \tilde{x} \end{pmatrix} \begin{pmatrix} 1 & \tilde{x}^\top \end{pmatrix}$ and solve problem (5) with Q, X replaced by $\widetilde{Q}, \widetilde{X}$.

In view of the above remark, *in the rest of the paper we will only consider the case* $\mathcal{B} = \{-1, 1\}$.

Now, to each unique component of the matrix $X = xx^\top$ we assign a variable y_k. Considering the constraints $X_{i,i} = 1$ and the symmetry of X, we thus have $y \in \mathbb{R}^r$ with $r = s(s-1)/2$. Replacing the variable X by y, the constraint $X \succeq 0$ can then be written as $\sum_{i=1}^r M_i y_i + I \succeq 0$, where $M_i \in \mathbb{S}^s$ are suitable matrices; for details, see [6].

By ignoring the rank constraint, we can now define the first-order Lasserre relaxation of (5) as the following *moment problem*:

$$\begin{aligned} &\min_{y \in \mathbb{R}^r} q^\top y \\ &\text{subject to } \sum_{i=1}^r M_i y_i + I \succeq 0\,, \end{aligned} \tag{6}$$

where $q = \operatorname{svec} Q$.

Example 1 For instance, for $s = 3$ we use the vector (monomial basis) $\begin{pmatrix} x_1, & x_2, & x_3 \end{pmatrix}^\top$; then $r = 3$ and

$$M_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \; M_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \; M_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

and the moment matrix $\sum_{i=1}^r M_i y_i + I$ is associated with the original variables as

$$(M(y) =) \sum_{i=1}^r M_i y_i + I = \begin{pmatrix} 1 & y_1 & y_2 \\ y_1 & 1 & y_3 \\ y_2 & y_3 & 1 \end{pmatrix} \longleftrightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}^\top = \begin{pmatrix} x_1^2 & x_1x_2 & x_1x_3 \\ x_1x_2 & x_2^2 & x_2x_3 \\ x_1x_3 & x_2x_3 & x_3^2 \end{pmatrix}.$$

The Lasserre hierarchy consists in adding higher-order monomials in the monomial basis and, therefore, the higher-order relaxation problems *will all be of the form* (6) only of growing dimension.

Example 2 The second-order relaxation in Example 1 will lead to a (6×6) moment matrix associated with the dyadic product of the vector of monomials of order up to two $(x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3)^\top$ with itself.

As some of the terms in the higher-order moment matrix are repeated and some can be simplified using the constraint $x_i^2 = 1$ (e.g., $x_1x_2^2$ is reduced to x_1) and thus even more terms are repeated, the number of variables in the higher-order relaxations is smaller than the number of the elements of the upper triangle of the matrix (see also Example 3 below).

Denote the order of the Lasserre relaxation by ω.

Definition 1 The *Lasserre relaxation of order* ω of the problem (1) is given by problem (6) with general dimensions n, m, replacing r, s, respectively; i.e., $y \in \mathbb{R}^n$ and $M_i \in \mathbb{S}^m$, $i = 1, \ldots, n$.

The dimensions n, m grow quickly with the order of the relaxation. In particular, we have

$$\frac{s^4}{24} \lessapprox n \leq 2^s - 1 \quad \text{and} \quad m = \sum_{i=1}^{\omega} \binom{s}{i}. \tag{7}$$

(The authors are not aware of any exact formula for n or the lower bound on n; the lower bound in (7) is our estimate based on numerical experiments. Obviously, the higher ω the bigger n until it reaches the "saturation point" $2^s - 1$.)

The dimension of the problem makes it very challenging for standard SDP solvers based on second-order methods, such as interior-point methods; see [10] and the numerical examples in the last section of this paper.

It was shown by Fawzi et al. [17] and Sakaue et al. [18], and numerically confirmed in [10], that for this type of problems, the sequence of these approximations is *finite* and the upper bound on the order of the relaxation to obtain exact solution of (1) is $\lceil n/2 \rceil$. Laurent [19] showed that this is also a lower bound for MAXCUT problems with unweighted complete graphs.

To determine whether a relaxation of order ω is exact, i.e., whether its solution is the solution of (1), we use the rank of the moment matrix; see [6, Thm. 6.19]. In particular, assuming that (1) has p global minimizers, the rank of the moment matrix will be less than or equal to p. Therefore, if we assume that (1) has a unique solution, then the moment matrix of the exact relaxation will be of rank one.

Remark 2 Notice that, when $\mathcal{B} = \{-1, 1\}$, every global minimizer x will have a "symmetric" counterpart $-x$, as there are only quadratic terms in the problem. In this case, by unique minimizer, we understand "unique up to the multiple by -1" and the rank of the exact relaxation will be two.

Above, we spoke about an "exact relaxation" and a "rank of the moment matrix". When solving the problem by an interior point method, we can only speak about a numerical rank defined below.

Definition 2 Consider $A \in \mathbb{S}^m_+$ with eigenvalues $\lambda_i(A)$ and with the largest eigenvalue $\lambda_{\max}(A)$. Let $\varepsilon > 0$. The *numerical rank* of A is defined as

$$r_\varepsilon(A) = |\mathcal{I}|, \quad \mathcal{I} = \left\{ i \in \{1, \dots, m\} \mid \frac{\lambda_i(A)}{\lambda_{\max}(A)} \geq \varepsilon \right\}.$$

Definition 3 For problems with a unique solution and $\mathcal{B} = \{-1, 1\}$, the relaxation of order ω is called *exact* if the numerical rank of the corresponding moment matrix is less than or equal to two.

3.2 Re-writing the problem

As mentioned in the Introduction, problem (6) is in the "wrong" form for our solver Loraine. While the solution $M^* := M(y^*)$ of (6) has low rank (in particular, $\operatorname{rank} M^* = 2$ when the solution of (1) is unique), Loraine expects the solution of an SDP problem in the *dual form* (4) to have low rank. Our next goal is thus to rewrite (6) in the dual form; in other words, to rewrite the dual to (6) in the *primal form* (3).

The dual to (6) is the problem

$$\begin{aligned} \max_{Z \in \mathbb{S}^m} \; & -I \bullet Z \\ \text{subject to } & M_i \bullet Z = q_i, \quad i = 1, \dots, n \\ & Z \succeq 0\,, \end{aligned}$$

and, by vectorizing the matrices as $z = \operatorname{svec} Z$ and $\mathbf{M} = (\operatorname{svec} M_1, \dots, \operatorname{svec} M_n)^\top$, $\mathbf{M} \in \mathbb{R}^{n \times \widetilde{n}}$, it can be further written as

$$\begin{aligned} \min_{z \in \mathbb{R}^{\widetilde{n}}} \; & (\operatorname{svec} I)^\top z \\ \text{subject to } & \operatorname{smat}(z) \succeq 0 \\ & \mathbf{M} z = q \end{aligned} \tag{8}$$

with $\widetilde{n} = m(m+1)/2$. Now, from the fact that the problem (8) is the dual problem to (6), we have the following lemma:

Lemma 1 *The vector y in (6) is the Lagrangian multiplier to $\mathbf{M}z = q$ in (8).*

Problem (8) is now almost in the right form for our solver: it is in the primal form and the dual solution is a low-rank matrix. Notice that, despite Q being possibly a dense matrix, the data in (8) are always very sparse. One obstacle remains—the linear equality constraints in (8). We are addressing it in the next section.

3.3 Equality constraints by ℓ_1-penalty

Interior-point methods have been designed for optimization problems with inequality constraints and cannot directly handle equality constraints. There are various ways how to overcome this, starting with writing equality as two inequalities; see [20] for an overview focused on SDP interior-point algorithms. In our algorithm, we treat the equality constraints by the following ℓ_1-penalty approach.

Introducing ℓ_1-penalty for the linear equality constraints in (8), we obtain the following problem

$$\begin{aligned} &\min_{z\in\mathbb{R}^{\tilde{n}}} (\operatorname{svec} I)^\top z + \mu\|\mathbf{M}z - q\|_1 \\ &\text{subject to } \operatorname{smat}(z) \succeq 0\,, \end{aligned} \tag{9}$$

with a penalty parameter $\mu > 0$. Recall that this penalty is exact, in the sense that there exists $\mu_0 > 0$ such that the solution of (9) with $\mu \geq \mu_0$ is equivalent to the solution of (8) (see, e.g., [21]).

We now introduce two new variables, $r \in \mathbb{R}^n$, $s \in \mathbb{R}^n$, satisfying

$$\mathbf{M}z - q = r - s, \quad r \geq 0,\ s \geq 0\,.$$

After substitution, (9) reads as

$$\begin{aligned} \min_{z\in\mathbb{R}^{\tilde{n}},\, r\in\mathbb{R}^n,\, s\in\mathbb{R}^n} \ & (\operatorname{svec} I)^\top z + \mu \sum_{i=1}^{n} (r_i + s_i) \\ \text{subject to } & \operatorname{smat}(z) \succeq 0 \\ & \mathbf{M}z - q = r - s \\ & r \geq 0,\ s \geq 0\,. \end{aligned} \tag{10}$$

Using the identity $r = \mathbf{M}z - q + s$ to eliminate variable r, we arrive at our final problem

$$\begin{aligned} \min_{z\in\mathbb{R}^{\tilde{n}},\, s\in\mathbb{R}^n} \ & (\operatorname{svec} I)^\top z + \mu \sum_{i=1}^{n} ((\mathbf{M}z - q)_i + 2s_i) \\ \text{subject to } & \operatorname{smat}(z) \succeq 0 \\ & \mathbf{M}z - q + s \geq 0 \\ & s \geq 0\,. \end{aligned} \tag{11}$$

Problem (11) is now in the primal form (3).

By Lemma 1, we know that y in (6) is the Lagrangian multiplier to the equality constraint in (8). The question is how to obtain y from (11), the ℓ_1-penalty reformulation of (8).

Lemma 2 *A solution y of (6) can be obtained from the solution of (11) as $y_i = \mu + \lambda_i,\ i = 1, \ldots, n$, in which μ is the penalty parameter at the solution and λ is the Lagrangian multiplier to the constraint $\mathbf{M}z - q + s \geq 0$.*

Proof We start with the optimality conditions of the problem (11). Using suitable matrices $\mathcal{E}_i \in \mathbb{S}^m$, we can write $\mathrm{smat}(z) = \sum_{i=1}^{\widetilde{n}} \mathcal{E}_i z_i$. The Lagrangian of (11) is then

$$\begin{aligned}\mathcal{L}_1(z, s, \Gamma, \lambda, \eta) = &- \left(f(z) + \mu \sum_{i=1}^{n} ((\mathbf{M}z - q)_i + 2s_i) \right) - \Gamma \bullet \sum_{i=1}^{\widetilde{n}} \mathcal{E}_i z_i \\ &- \sum_{i=1}^{n} \lambda_i \left(\sum_{j=1}^{\widetilde{n}} \mathbf{M}_{ij} z_j - q_i + s_i \right) - \sum_{i=1}^{n} \eta_i s_i,\end{aligned}$$

where $\Gamma \succeq 0$ and $\lambda, \eta \geq 0$ are the corresponding Lagrangian multipliers and $f(z) = (\mathrm{svec}\, I)^\top z$. By using the KKT conditions at the solution we have

$$\begin{aligned}&\nabla_z \mathcal{L}_1(z^*, s^*, \Gamma^*, \lambda^*, \eta^*) = 0 : \\ &- (\nabla f(z^*))_j - \mu \sum_{i=1}^{n} \mathbf{M}_{ij} - \sum_{i=1}^{n} \mathbf{M}_{ij} \lambda_i^* - \Gamma^* \bullet \mathcal{E}_j = 0, \quad j = 1, \ldots, \widetilde{n}\end{aligned}$$

and so

$$(\nabla f(z^*))_j = - \sum_{i=1}^{n} \mathbf{M}_{ij} (\mu + \lambda_i^*) - \Gamma^* \bullet \mathcal{E}_j, \quad j = 1, \ldots, \widetilde{n}\,. \tag{12}$$

Now, the Lagrangian of the problem (8) is

$$\mathcal{L}_2(z, \Gamma, y) = -f(z) - \Gamma \bullet \sum_{i=1}^{\widetilde{n}} \mathcal{E}_i z_i - \sum_{i=1}^{n} y_i \left(\sum_{j=1}^{\tilde{n}} \mathbf{M}_{ij} z_j - q_i \right),$$

where, as above, $\Gamma \succeq 0$ and $y \geq 0$ are the corresponding Lagrangian multipliers. From the KKT conditions, we have

$$\begin{aligned}&\nabla_z \mathcal{L}_2(z^*, \gamma^*, y^*) = 0 : \\ &- (\nabla f(z^*))_j - \Gamma^* \bullet \mathcal{E}_j - \sum_{i=1}^{n} \mathbf{M}_{ij} y_i^* = 0, \quad j = 1, \ldots, \widetilde{n}\end{aligned}$$

so

$$(\nabla f(z^*))_j = - \sum_{i=1}^{n} \mathbf{M}_{ij} y_i^* - \Gamma^* \bullet \mathcal{E}_j, \quad j = 1, \ldots, \widetilde{n}\,. \tag{13}$$

Thus, from (12) and (13), y can be written as

$$y_i = \mu + \lambda_i, \quad i = 1, \ldots, n \tag{14}$$

and we are done. □

The question remains how to initialize and update the penalty parameter μ in order to get an exact solution of problem (8) using our interior-point solver. There are essentially two options.

(A) For the current value of μ, fully solve the penalized problem (11) by Loraine, check the residuum of the equality constraint in (8) for the optimal solution of (11) and, if bigger than a prescribed threshold, increase μ and repeat this process.
(B) Incorporate penalty update within the interior-point solver, following, e.g., [22] and [23]. That is, run the IP algorithm with an initial μ and, as soon as the residuum of the equality constraint (8) gets bigger than a value dependent on the IP barrier parameter, increase μ and continue in IP iterations.

Option (A) is simple and robust but may also be expensive: we may need to solve several SDP problems to full optimality. Option (B) is more attractive from the efficiency point of view but is also more sensitive to the choice of parameters. We implemented (B) but had mixed experience with it so we returned to (A). The good news is that a class of BQP problems usually requires a single choice of μ in order to guarantee convergence to an optimal point with the norm of the residuum of the equality constraint smaller than a prescribed value; see the numerical experiments in the last section. Needless to say that one cannot start with a too big value of μ due to the potential ill-conditioning of the resulting SDP problem and resulting difficulties of the IP algorithm to solve the problem at all.

3.4 Loraine and problem (11)

Recall the definition of the matrix $\mathbf{M} = (\operatorname{svec} M_1, \ldots, \operatorname{svec} M_n)^T$ with matrices M_i introduced at the beginning of this section. We have the following result.

Lemma 3 *There exists a permutation matrix $P \in \mathbb{R}^{\widetilde{n} \times \widetilde{n}}$ such that $P\mathbf{M}^\top \mathbf{M} P^\top$ is a block diagonal matrix with small full blocks. In particular, $\mathbf{M}^\top \mathbf{M}$ is a sparse chordal matrix.*

Proof Every matrix M_i localizes the corresponding variables in the moment matrix $\sum_{i=1}^{n} M_i y_i + I$ in (6), whereas the elements of the moment matrix are either ones or the variables y_i; see also Example 1. As every M_i is associated with exactly one variable y_i, the nonzeros in this matrix are unique to the matrix. Therefore every column of matrix $\mathbf{M}$ will either contain all zeros or a single nonzero number one at the i-th position, $i = 1, \ldots, n$. We can now re-order the columns of $\mathbf{M}$ such that the first columns will only contain zeros, the next columns element 1 at the first position, the next ones element 1 at the second position, etc. This re-ordering will define the permutation matrix P. The claims follow. □

Example 3 Consider a problem with $s = 3$ and relaxation order $\omega = 2$. Utilizing the constraints $x_i^2 = 1$, we only consider variables associated with the monomial basis $b(x) =$

$(x_1,\ x_2,\ x_3,\ x_1x_2,\ x_2x_3,\ x_1x_3)$:

$$X = b(x)b^\top(x) = \begin{pmatrix} x_1^2 & x_1x_2 & x_1x_3 & x_1^2x_2 & x_1x_2x_3 & x_1^2x_3 \\ x_1x_2 & x_2^2 & x_2x_3 & x_1x_2^2 & x_2^2x_3 & x_1x_2x_3 \\ x_1x_3 & x_2x_3 & x_3^2 & x_1x_2x_3 & x_2x_3^2 & x_1x_3^2 \\ x_1^2x_2 & x_1x_2^2 & x_1x_2x_3 & x_1^2x_2^2 & x_1x_2^2x_3 & x_1^2x_2x_3 \\ x_1x_2x_3 & x_2^2x_3 & x_2x_3^2 & x_1x_2^2x_3 & x_2^2x_3^2 & x_1x_2x_3^2 \\ x_1^2x_3 & x_1x_2x_3 & x_1x_3^2 & x_1^2x_2x_3 & x_1x_2x_3^2 & x_1^2x_3^2 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & x_1x_2 & x_1x_3 & x_2 & x_1x_2x_3 & x_3 \\ x_1x_2 & 1 & x_2x_3 & x_1 & x_3 & x_1x_2x_3 \\ x_1x_3 & x_2x_3 & 1 & x_1x_2x_3 & x_2 & x_1 \\ x_2 & x_1 & x_1x_2x_3 & 1 & x_1x_3 & x_2x_3 \\ x_1x_2x_3 & x_3 & x_2 & x_1x_3 & 1 & x_1x_2 \\ x_3 & x_1x_2x_3 & x_1 & x_2x_3 & x_1x_2 & 1 \end{pmatrix}$$

i.e., variables $y \in \mathbb{R}^7$ corresponding to $(x_1, x_2, x_3, x_1x_2,\ x_2x_3, x_1x_3, x_1x_2x_3)$, the unique elements of X. The matrices M_i simplify to

$$M_1 = \begin{pmatrix} 0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1 \\ 0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0 \end{pmatrix},\ M_2 = \begin{pmatrix} 0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0 \\ 1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0 \end{pmatrix}, \ldots, M_7 = \begin{pmatrix} 0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&1 \\ 0&0&0&1&0&0&0 \\ 0&0&1&0&0&0&0 \\ 1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0 \end{pmatrix}$$

and thus

$$\mathbf{M} = \begin{pmatrix} 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&0&0 \end{pmatrix}$$

and

$$\mathbf{M}P^\top = \begin{pmatrix} 0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1 \end{pmatrix}$$

with a suitable permutation matrix P. Then $P\mathbf{M}^\top\mathbf{M}P^\top$ is block diagonal with block sizes 2 and 3.

Remark 3 The size of the blocks in the matrix $P\mathbf{M}^\top\mathbf{M}P^\top$ is given by the count of the corresponding variables in the upper triangle of the moment matrix. This number grows with the relaxation order.

Theorem 4 *Assume that (1) has a unique solution and that Lasserre hierarchy is exact for order $\omega \geq 1$. Then problem (11) corresponding to ω satisfies Assumptions 1–5.*

Proof A strictly feasible point for (11) can be found, e.g., by choosing z as a unit vector and s positive, arbitrarily large. Similarly, the moment problem (6) is trivially strictly feasible by choosing $y = 0$. Hence Assumption 1 holds. Every matrix $\mathcal{E}_i$ contains at most two nonzero elements, hence Assumption 2 is trivially satisfied. Assumption 3 holds by Lemma 3: because the matrix $\mathbf{M}^\top \mathbf{M}$ is chordal and sparse, sparse Cholesky factorization leads to zero fill-in and is thus very efficient. Assumption 4 is satisfied by construction of the problem, as the number of variables $\widetilde{n}$ is proportional to the square of the size of the matrix inequality m. Finally, under the assumption of unique solution to (1) and exact relaxation, the dual variable associated with the matrix inequality in (11) has rank at most two. Hence Assumption 5 is satisfied. □

4 ADMM for SDP relaxations

The Alternating Direction Method of Multipliers (ADMM) is a popular alternative to the interior-point methods for convex optimization problems. Its extension to semidefinite optimization was introduced in [15]. It proved to be efficient for certain SDP problems, among others many UBQPs ([15, Sec 4.3]). We thus briefly describe this algorithm and offer a numerical comparison with Loraine. Furthermore, we will introduce a new "hybrid" approach combining ADMM with Loraine.

4.1 The ADMM-SDP algorithm

The method consists of minimizing the augmented Lagrangian of the problem

$$\mathcal{L}_\rho(y, S, X) = -b^\top y + X \bullet \left(\sum_{i=1}^n y_i A_i + S - C \right) + \frac{1}{2\rho} \left\| \sum_{i=1}^n y_i A_i + S - C \right\|_F^2$$

alternately, first with respect to y, then with respect to S, and then updating X by

$$X^{(k+1)} = X^{(k)} + \frac{\sum_{i=1}^n y_i^{(k+1)} A_i + S^{(k+1)} - C}{\rho}.$$

While the minimization with respect to y leads to an unconstrained problem and the minimum is obtained by solving the first-order optimality system of (linear) equations, the minimization of $\mathcal{L}_\rho$ with respect to S can be formulated as a projection of the matrix $C - \sum_{i=1}^n y_i A_i - \rho X$ onto the semidefinite cone $\mathbb{S}^m_+$.

For $Z \in \mathbb{S}^m$, let $\mathcal{A}(Z)$ be a vector with elements $A_i \bullet Z$, $i = 1, \ldots, n$ and $[Z]_+$ the projection of Z on the semidefinite cone. With a penalty parameter $\rho > 0$ and a relaxation parameter $\sigma \in (0, \frac{1+\sqrt{5}}{2})$, the update of the primal-dual point (y, S, X) in the k-th iteration of ADMM is defined in Algorithm 1.

For details, in particular the choice and update of ρ, see [15]. In the numerical experiments below, we use our MATLAB implementation[2] of the algorithm.

[2] https://github.com/kocvara/ADMM_for_SDP

Algorithm 1 ADMM for SDP

Given an initial point $(y^{(0)}, S^{(0)}, X^{(0)})$, parameters ρ and σ, and a stopping parameter $\varepsilon_{\text{ADMM}}$.

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: Update y by $y^{(k+1)} = -(\mathbf{A}\mathbf{A}^\top)^{-1}\left(\rho(\mathcal{A}(X^{(k)}) - b) + \mathcal{A}(S^{(k)} - C)\right)$
3: Update S by $\widehat{S} = C - \sum_{i=1}^{n} y_i^{(k+1)} A_i - \rho X^{(k)}, \quad S^{(k+1)} = \left[\widehat{S}\right]_+$
4: Update X by $\widehat{X} = \frac{1}{\rho}\left(S^{(k+1)} - \widehat{S}\right), \quad X^{(k+1)} = (1-\sigma)X^{(k)} + \sigma\widehat{X}$
5: Update ρ
6: Check convergence
7: **end for**

4.2 ADMM as a warm starter for Loraine

When testing the ADMM algorithm with the MAXCUT problems considered in this paper, we have made the following two observations:

(i) The method is more efficient when solving the reformulated (bigger) problem (11) than the smaller one (6). Moreover, for problem (11), the convergence is very fast in the first iterations, before it slows down: the algorithm gets quickly a "good" approximation of the solution, both primal and dual.

(ii) Due to the nature of the ADMM method (projections on the semidefinite cone), the rank of the approximate solution is always "numerically exact"; in particular, the rank of $X^{(k)}$ is low when we are close enough to the solution.

The fact that a low-rank approximation of the solution can be obtained relatively quickly, together with the fact that ADMM is a primal-dual algorithm leads to the following idea: use a low-precision ADMM solution as a warm start for Loraine. Presuming that the ADMM approximation of X is already of the expected low rank (or very close to it), the preconditioner H_α will be extremely efficient during the remaining iterations of Loraine. We therefore propose the following *hybrid ADMM-Loraine* Algorithm 2.

Algorithm 2 ADMM-Loraine

1: Solve (11) by ADMM with stopping tolerance $\varepsilon_{\text{ADMM}}$. Save the primal-dual approximate solution $(y, S, X)_{\text{ADMM}}$.
2: Solve (11) by Loraine with
– initial point $(y, S, X)_{\text{ADMM}}$
– initial value of the stopping criterion for the CG method reduced to 10^{-6}.

Notice that, in order to get a primal-dual approximation of the solution, we have to solve the same problem formulation by ADMM and Loraine, i.e., formulation (11) with the ℓ_1-penalty and the same value of the penalty parameter μ, despite the fact that ADMM could directly handle the linear equality constraints.

5 Numerical experiments

In this section we report the results of our numerical experiments with relaxations of MAXCUT problems and randomly generated UBQPs. The presented algorithms will be compared with the state-of-the-art SDP solver MOSEK [24] that is relying on the interior-point method and uses a direct linear solver for the Schur complement equation.

The stopping criterion for our implementations of Loraine and ADMM was based on the DIMACS errors [25] that measure the (normalized) primal and dual feasibility, duality gap and complementary slackness. The algorithms were stopped when all these errors were below 10^{-6}. MOSEK was used with default settings.

All problems were solved on an iMac desktop computer with 3.6 GHz 8-Core Intel Core i9 and 64 GB 2667 MHz DDR4 using MATLAB R2022b.

5.1 MAXCUT problems

For the first set of numerical experiments, we use MAXCUT problems, the standard test problems for UBQP algorithms. Let Γ be an undirected n-node graph and let the arcs (i, j) be associated with nonnegative weights a_{ij}. The MAXCUT problem is formulated as the following UBQP:

$$\max_x \left\{ \frac{1}{4} \sum_{i,j=1}^{n} a_{ij}(1 - x_i x_j) \mid x_i^2 = 1,\ i = 1, \ldots, n \right\} \tag{15}$$

and can be thus solved by the techniques presented above. The solution of the SDP relaxation (2) is of rank two whenever the relaxation is exact and the solution to (15) is unique. (Recall that uniqueness of x means uniqueness up to the multiple by -1.)

Obviously, a complete unweighted graph (with $a_{ij} \in \{0, 1\}$) may have many "symmetric" solutions. Therefore we generated two sets of problems of increasing size. Firstly, to avoid the non-uniqueness, we generated undirected, weighted, generally complete graphs with weights randomly distributed between 0 and 12 with 20–50 nodes, using the MATLAB command `graph`. It turned out that for all these problems the relaxation order two in the Lasserre hierarchy is already high enough to deliver the optimal solution of the MAXCUT problem (15). Secondly, we generated a set of problems of the same sizes but for unweighted graphs. The dimensions of the corresponding SDP problems for relaxation order two are shown in Table 1.

We first present the results for the weighted problems, see Table 2. These problems are relatively "simple": the ADMM method is rather efficient, and MOSEK converges in a very small number of iterations, six to eight. However, the sheer size prevents MOSEK with a direct solver to solve larger problems. We present results for ADMM applied to both problem formulations, the original one (6) and the re-written one of larger dimension and with ℓ_1-penalty (11). Perhaps surprisingly, ADMM is more efficient for the latter problem, despite the bigger dimension. Moreover, unlike for ADMM applied to (6), the number of iterations of ADMM applied to (11) is almost independent of the problem size. The last four columns of Table 2 show

Table 1 Problems `MAXCUT-<n>`, relaxation order $\omega = 2$: dimensions for formulations (6) and (11)

problem	problem (6) variables	problem (6) matrix size	problem (11) variables	problem (11) matrix size	problem (11) lin. constr.
MAXCUT-20	6 196	211	28 561	211	12 390
MAXCUT-25	15 275	326	68 576	326	30 550
MAXCUT-30	31 930	466	140 741	466	63 860
MAXCUT-35	59 535	631	258 931	631	119 070
MAXCUT-40	102 090	821	439 521	821	204 180
MAXCUT-45	164 220	1 036	701 386	1 036	328 440
MAXCUT-50	251 175	1 276	1 065 901	1 276	502 350

the results for the hybrid ADMM-Loraine algorithm: we present the number of iterations of ADMM plus Loraine ('iter'), the time of ADMM ('timeA'), time of Loraine ('timeL') and the total time of the hybrid method. For these experiments, we have used the stopping tolerance of ADMM $\varepsilon_{\text{ADMM}} = 5 \cdot 10^{-3}$ (problems 20–25) and $\varepsilon_{\text{ADMM}} = 5 \cdot 10^{-4}$ (problems 30–50).

Table 2 Loraine, MOSEK, ADMM and ADMM-Loraine in weighted `MAXCUT-<n>` problems, relaxation order $\omega = 2$

MAXCUT problem	Loraine for (11) iter	Loraine for (11) CG it	Loraine for (11) time	MOSEK (6) time	ADMM for (6) iter	ADMM for (6) time	ADMM for (11) iter	ADMM for (11) time	ADMM-Loraine for (11) iter	ADMM-Loraine for (11) timeA	ADMM-Loraine for (11) timeL	ADMM-Loraine for (11) time
20	18	559	3.8	9	3615	10	3042	13	628+4	3.9	0.8	4.7
25	20	728	11	78	4732	33	2735	28	181+5	2.5	2.9	5.4
30	21	1032	28	607	6770	99	3537	80	795+5	20	5.8	26
35	23	2183	96	2911	5255	164	3030	126	863+4	42	11	53
40	27	2275	186	mem	9611	500	1280	92	914+7	73	132	205
45	25	2521	335		16901	1400	2639	358	755+4	104	33	137
50	24	2540	528		19521	2951	3296	745	727+5	162	58	220

To better demonstrate the behaviour of the hybrid method, in Figure 1 we show the output of both codes. We can see that the ADMM method, indeed, gets very quickly a good approximation of objective value, while Loraine uses the warm start very efficiently.

Next we will try to solve the unweighted MAXCUT problems. Recall that these problems often have nonunique solutions, in particular problems with almost dense graphs. Moreover, relaxation order $\omega = 2$ may not be high enough to obtain an exact solution. We thus cannot expect the preconditioner to be efficient in those cases. This is, indeed, demonstrated in Table 3. This table, in addition to columns identical to Table 2, shows also the rank of the solution of the order-2 relaxation. As expected, Loraine is less efficient for problems with higher solution rank, in particular problems `MAXCUT-30` and `MAXCUT-50`. Still, it can solve these problems reliably. Again, the hybrid method is superior for these problems. The convergence behaviour of the algorithms and their estimated complexity are further illustrated in Figure 2.

```
================================== A D M M  for  S D P ==================
Number of LMI Constraints:   1
Number of Variables: 1065901
Maximal Constraint Size: 1276
------------------------------------------------------------------------
 iter    p-infeas      d-infeas       d-gap         error       objective
------------------------------------------------------------------------
  100   0.00058292   0.08603023   0.00093126    0.086030   1981.22444814
  200   0.00018776   0.02554937   0.00034476    0.025549   2028.80448106
  300   0.00018959   0.01054250   0.00063882    0.010542   2037.35317708
  400   0.00010212   0.00991976   0.00025128    0.009920   2025.69453823
  500   0.00005383   0.00972192   0.00013810    0.009722   2038.21968191
  600   0.00002508   0.00891292   0.00001945    0.008913   2026.90960413
  700   0.00002609   0.00281354   0.00010411    0.002814   2037.33886004
  727   0.00002899   0.00047182   0.00012837    0.000472   2039.90970508
------------------------------------------------------------------------
Total ADMM iterations: 727; Final precision: 4.72e-04; CPU Time 162.59s
========================================================================

 *** Loraine v0.1 ***
 Number of variables: 1065901
 Matrix size(s)     :    1276
 Linear constraints :  502350
 *** IP STARTS
 it      objective      error   cg_iter   CPU/it
  1   2.03994277e+03  3.77e-03       36     9.39
  2   2.03999424e+03  2.32e-03       33     9.55
  3   2.03999941e+03  2.50e-04       20     7.71
  4   2.03999992e+03  3.84e-05        9     5.80
  5   2.03999999e+03  5.96e-06        8     5.54
*** Total CG iterations:    106
*** Total CPU time:         58.24 seconds
```

Figure 1 The printout of low-precision ADMM and warm-started Loraine for the weighted `MAXCUT-50`.

Table 3 Loraine, MOSEK, ADMM and ADMM-Loraine in unweighted `MAXCUT-<n>` problems, relaxation order $\omega = 2$

MAXCUT problem	Loraine for (11) iter	CG it	time	MOSEK (6) time	ADMM for (11) iter	time	ADMM-Loraine for (11) iter	timeA	timeL	time	rank
20	17	735	4.3	9	1981	8	738+7	3.4	1.5	4.9	2
25	18	858	14	82	805	9	328+4	3.5	2.8	6.3	4
30	23	6482	145	635	2218	53	566+8	13	135	148	57
35	21	2604	128	3357	5251	229	1222+8	48	15	63	2
40	22	3538	305	mem	7137	575	1855+7	142	35	177	2
45	22	3225	536		6794	845	1081+4	139	49	188	4
50	25	12599	2593		5723	1162	1677+5	361	365	726	13

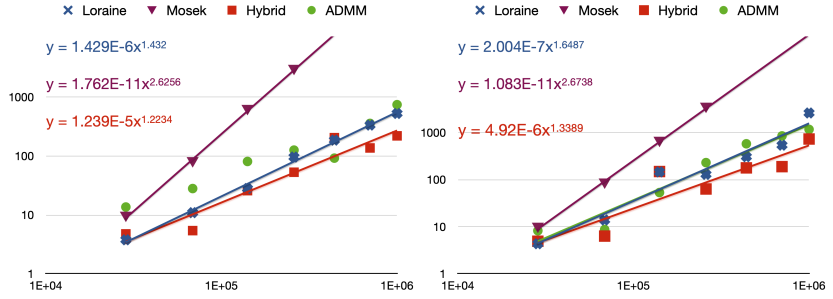

Figure 2 CPU times in log-log scale for weighted (left) and unweighted (right) MAXCUT problems. Loraine (blue), MOSEK (brown), ADMM (green) and hybrid ADMM-Loraine (red).

5.2 Randomly generated problems

We further conducted numerical experiments with matrices Q randomly generated. We are aware of the fact that problems with random data may not always be representative and may sometimes lead to false conclusions regarding algorithm behaviour. However, we believe that these results still well demonstrate the efficiency of our approach.

It has been observed in [10] that problems with rank-one matrix Q may require relaxation order of up to $\omega = \lceil s/2 \rceil$ to reach the exact solution; a typical example is $Q = ee^\top$. On the other hand, for problems with matrix Q of rank 3, $\omega = 2$ was always sufficient in experiments performed in ([10, Fig.4]). We have thus considered two main classes of problems:

(A) problems with $\mathcal{B} = \{-1, 1\}$ and with $\operatorname{rank} Q = 1$ generated by the following MATLAB code

```
rng(0); q = randn(s,1); Q = q*q';
```

(B) problems with $\mathcal{B} = \{-1, 1\}$ and with a full-rank indefinite Q generated by the following MATLAB code

```
rng(0); q = randn(s,1); Q = q*q';
for k=1:s-1
    if ceil(k/2)*2 == k
        q = randn(s,1); Q = Q - q*q';
    else
        q = randn(s,1); Q = Q + q*q';
    end
end
```

Apart from random numbers generated with normal distribution, we also performed tests with uniform distribution (function `rand`) and lognormal distribution (function `logncdf`); in both cases the results and conclusions were rather similar to the above choice and are thus not reported here.

The exactness of the relaxation was measured by the numerical rank of the dual solution to the matrix inequality in problem (11)—when the rank was equal to 1 or 2 (depending on the set $\mathcal{B}$) the relaxation order was considered sufficient; see Definition 3.

Remark 4 For Q constructed as in (B), relaxation order $\omega = 2$ was sufficient to get an exact solution of (1). This observation, though, cannot be extended to any full-rank matrix Q. For instance, for $Q = ee^\top + \operatorname{Diag}(d), d \in \mathbb{R}^s$, the lowest relaxation order will be $\omega = \lceil s/2 \rceil$. This is because $x_i^2 = 1$ and thus the diagonal elements of Q will be irrelevant in the optimization process and the resulting problem will be equivalent to that with a rank-one matrix $Q = ee^\top$.

5.2.1 Full-rank Q and relaxation order $\omega = 2$

Using the MATLAB code from point (B) above, we generated problems of growing dimension $s = 10 \ldots 50$ and solved the corresponding SDP relaxations for order $\omega = 2$. The dimensions of the generated problems are reported in Table 4; the table shows problem sizes for the original SDP relaxation (6) and for the re-written problem (11). The computational results are presented in Table 5 and clearly demonstrate

Table 4 Randomly generated UBQP problems, relaxation order $\omega = 2$: dimensions for formulations (6) and (11)

UBQP size	problem (6) variables	problem (6) matrix size	problem (11) variables	problem (11) matrix size	problem (11) lin. constraints
10	385	56	1596	56	770
15	1 940	121	7 381	121	3 880
20	6 195	211	22 366	211	12 390
25	15 275	326	53 301	326	30 550
30	31 930	466	108 811	466	63 860
35	59 535	631	199 369	631	119 070
40	102 090	821	337 431	821	204 180
45	164 220	1 036	537 166	1 036	328 440
50	251 175	1 276	814 726	1 276	502 350

the efficiency of Loraine. The computational complexity of Loraine (applied to (11)) and MOSEK and ADMM (applied to (6)) is further illustrated in Figure 3. We do not report on the hybrid ADMM-Loraine algorithm. That is because ADMM applied to problem (11) (as required by the hybrid method) appears to be much less efficient than for the MAXCUT problems and so the hybrid algorithm is not even competitive to ADMM applied to (6).

Table 5 Randomly generated UBQP problems, relaxation order $\omega = 2$: Loraine, MOSEK and ADMM

UBQP size	Loraine for (11) iter	Loraine for (11) CG iter	Loraine for (11) time	MOSEK for (6) iter	MOSEK for (6) time	ADMM for (6) iter	ADMM for (6) time
10	10	256	0.1	6	0.2	1006	0.4
15	10	725	1.3	5	1.1	1928	2.2
20	11	423	2.4	6	9.3	2888	9.7
25	13	294	5.5	7	81	5622	48
30	13	326	14	7	496	8498	147
35	15	530	43	memory		11672	396
40	14	730	106			14362	880
45	16	896	230			14320	1369
50	16	1114	431			21701	3251

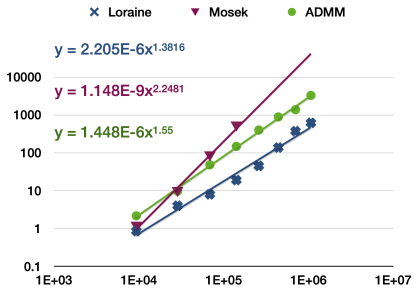

Figure 3 CPU times in log-log scale for random BQP problems. Loraine (blue), MOSEK (brown) and ADMM (green).

5.2.2 Rank-one Q and higher relaxation order

Let us recall from (7) the dependence of problem sizes on the relaxation order: for the original moment problem (6), we have

$$\frac{s^4}{24} \lessapprox n \leq 2^s - 1 \quad \text{and} \quad m = \sum_{i=1}^{\omega} \binom{s}{i}.$$

On the other hand, in the re-written problem (11) with ℓ_1-penalty term, we have $\widetilde{n}+n$ variables with $\widetilde{n} = m^2$, LMI of size m and $2n$ linear inequality constraints; see Table 6. This table, in particular, shows that the number of variables grows much

Table 6 Problem dimensions for formulations (6) and (11) as functions of the relaxation order; here s is the number of variables in BQP (1) and ω the relaxation order and $\frac{s^4}{24} \lessapprox n \leq 2^s - 1$

problem (6)		problem (11)		
variables	matrix size	variables	matrix size	lin. constraints
n	$\sum_{i=1}^{\omega} \binom{s}{i}$	$\left(\sum_{i=1}^{\omega} \binom{s}{i}\right)^2 + n$	$\sum_{i=1}^{\omega} \binom{s}{i}$	$2n$

more quickly for problem (11) than for the original problem (6) where it, eventually, reaches the finite limit $2^s - 1$. Therefore we cannot expect Loraine with iterative solver applied to (11) to be as efficient for higher-order relaxations as it is for $\omega = 2$. This is clearly demonstrated in Table 7 comparing Loraine with a direct solver applied to problem (6) (this code was slightly faster than MOSEK for these problems, due to direct handling of the rank-one data matrices) with Loraine with the iterative solver applied to problem (11).

Table 7 Randomly generated UBQP problem, $n = 9$, relaxation order $\omega = 2, \ldots, 5$: problem dimensions and CPU times for Loraine-direct and Loraine-iterative

ω	Loraine-direct for (6)			Loraine-iterative for (11)				
	vars	matrix size	time	vars	matrix size	lin constr	CG iter	time
2	255	46	0.09	1081	46	510	99	0.08
3	465	130	0.77	8515	130	930	203	0.97
4	510	256	2.87	32896	256	1020	302	15.7
5	511	382	7.04	73153	382	1022	701	129

6 Conclusions

Our numerical experiments demonstrate the ability of an interior point method with a specially designed solver of the linear system to solve higher-order Lasserre relaxations of UBQP. The approach is particularly efficient for relaxation order two which,

for most of the tested problems, was high enough to deliver either the exact solution of the UBQP or a good approximation of it.

We have also introduced a new, hybrid algorithm that uses an approximate solution obtained by ADMM as a warm start for the used interior point method. This algorithm is rather efficient for problems for which ADMM itself is relatively efficient, such as the MAXCUT problems. On the other hand, it may be inefficient once ADMM fails to converge quickly to an approximate solution.

Acknowledgments. This work has been supported by European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Actions, grant agreement 813211 (POEMA). The second author has been partly supported by the Czech Science Foundation through project No. 22-15524S.

References

[1] Kochenberger, G., Hao, J.-K., Glover, F., Lewis, M., Lü, Z., Wang, H., Wang, Y.: The unconstrained binary quadratic programming problem: a survey. Journal of Combinatorial Optimization **28**(1), 58–81 (2014)

[2] Rendl, F., Rinaldi, G., Wiegele, A.: A branch and bound algorithm for max-cut based on combining semidefinite and polyhedral relaxations. In: International Conference on Integer Programming and Combinatorial Optimization, pp. 295–309 (2007). Springer

[3] Krislock, N., Malick, J., Roupin, F.: Biqcrunch: A semidefinite branch-and-bound method for solving binary quadratic problems. ACM Transactions on Mathematical Software (TOMS) **43**(4), 1–23 (2017)

[4] Gusmeroli, N., Hrga, T., Lužar, B., Povh, J., Siebenhofer, M., Wiegele, A.: BiqBin: a parallel branch-and-bound solver for binary quadratic problems with linear constraints. ACM Transactions on Mathematical Software (TOMS) **48**(2), 1–31 (2022)

[5] Ben-Tal, A., Nemirovski, A.: Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications. SIAM, Philadelphia (2001)

[6] Lasserre, J.-B.: An Introduction to Polynomial and Semi-algebraic Optimization. Cambridge University Press, Cambridge (2015)

[7] Gvozdenović, N., Laurent, M.: The operator ψ for the chromatic number of a graph. SIAM Journal on Optimization **19**(2), 572–591 (2008)

[8] Ghaddar, B., Vera, J.C., Anjos, M.F.: A dynamic inequality generation scheme for polynomial programming. Mathematical Programming **156**(1), 21–57 (2016)

[9] Campos, J.S., Misener, R., Parpas, P.: Partial lasserre relaxation for sparse Max-Cut. Optimization and Engineering, 1–22 (2022)

[10] Kim, S., Kojima, M.: Binary quadratic optimization problems that are difficult to solve by conic relaxations. Discrete Optimization **24**, 170–183 (2017)

[11] Waki, H., Kim, S., Kojima, M., Muramatsu, M., Sugimoto, H.: Algorithm 883: SparsePOP—a sparse semidefinite programming relaxation of polynomial optimization problems. ACM Transactions on Mathematical Software (TOMS) **35**(2), 1–13 (2008)

[12] Wang, J., Magron, V., Lasserre, J.-B.: TSSOS: A moment-SOS hierarchy that exploits term sparsity. SIAM Journal on Optimization **31**(1), 30–58 (2021)

[13] Wang, J., Magron, V., Lasserre, J.B., Mai, N.H.A.: CS-TSSOS: Correlative and term sparsity for large-scale polynomial optimization. ACM Transactions on Mathematical Software **48**(4), 1–26 (2022)

[14] Habibi, S., Kočvara, M., Stingl, M.: Loraine—an interior-point solver for low-rank semidefinite programming. Optimization Methods and Software (2022). Submitted.

[15] Wen, Z., Goldfarb, D., Yin, W.: Alternating direction augmented Lagrangian methods for semidefinite programming. Mathematical Programming Computation **2**(3), 203–230 (2010)

[16] Zhang, R.Y., Lavaei, J.: Modified interior-point method for large-and-sparse low-rank semidefinite programs. 2017 IEEE 56th Annual Conference on Decision and Control (CDC) (2017)

[17] Fawzi, H., Saunderson, J., Parrilo, P.A.: Sparse sums of squares on finite abelian groups and improved semidefinite lifts. Mathematical Programming **160**(1), 149–191 (2016)

[18] Sakaue, S., Takeda, A., Kim, S., Ito, N.: Exact SDP relaxations with truncated moment matrix for binary polynomial optimization problems. In: Proceedings of the 5th International Conference on Continuous Optimization (ICCOPT) (2016)

[19] Laurent, M.: Lower bound for the number of iterations in semidefinite hierarchies for the cut polytope. Mathematics of Operations Research **28**(4), 871–883 (2003)

[20] Anjos, M.F., Burer, S.: On handling free variables in interior-point methods for conic linear optimization. SIAM Journal on Optimization **18**(4), 1310–1325 (2008)

[21] Di Pillo, G., Grippo, L.: Exact penalty functions in constrained optimization. SIAM Journal on Control and Optimization **27**(6), 1333–1360 (1989)

[22] Gould, N.I., Orban, D., Toint, P.L.: An interior-point ℓ1-penalty method for nonlinear optimization. Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England (2003)

[23] Curtis, F.E.: A penalty-interior-point algorithm for nonlinear constrained optimization. Mathematical Programming Computation **4**(2), 181–209 (2012)

[24] MOSEK ApS: The MOSEK Optimization Toolbox for MATLAB Manual. Version 10.0. (2022). http://docs.mosek.com/10.0/toolbox/index.html

[25] Mittelmann, H.D.: An independent benchmarking of SDP and SOCP solvers. Mathematical Programming **95**, 407–430 (2003)