MADAM: A PARALLEL EXACT SOLVER FOR MAX-CUT BASED ON SEMIDEFINITE PROGRAMMING AND ADMM

Timotej Hrga∗, Janez Povh∗†

∗ University of Ljubljana, Faculty of Mechanical Engineering, Slovenia
† Institute of Mathematics, Physics and Mechanics Ljubljana, Slovenia
email: timotej.hrga@lecad.fs.uni-lj.si, janez.povh@lecad.fs.uni-lj.si

Abstract
We present MADAM, a parallel semidefinite based exact solver for Max-Cut, a problem of finding the cut with maximum weight in a given graph. The algorithm uses branch and bound paradigm that applies alternating direction method of multipliers as the bounding routine to solve the basic semidefinite relaxation strengthened by a subset of hypermetric inequalities. The benefit of the new approach is less computationally expensive update rule for the dual variable with respect to the inequality constraints. We provide theoretical convergence of the algorithm, as well as extensive computational experiments with this method, to show that our algorithm outperformes current state-of-the-art approaches. Furthermore, by combining algorithmic ingredients from the serial algorithm we develop an efficient distributed parallel solver based on MPI.

Keywords: semidefinite programming, alternating direction method of multipliers, maximum cut problem, parallel computing

1 Introduction

1.1 Motivation
Max-Cut problem is a classical NP-hard optimization problem [15, 22] on graphs with quadratic objective function and unconstrained binary variables. During the last decades it has attracted the interest of many researchers, from its theoretical and algorithmic perspective to its applicability in different fields, for instance mathematics, physics, and computer science [24, 27, 33]. Several exact algorithmic approaches have been proposed in the literature, including the solvers BiqMac [33] and BiqCrunch [25], which are among the best solvers and are based on semidefinite programming.

Lasserre [26] has proved that Max-Cut problem can be considered as a canonical model of linearly constrained linear and quadratic 0/1 programs. The transformation is based on the exact penalty approach and was further explored and advanced in a new paper by Gusmeroli and Wiegele [18]. Even for instances of moderate size, it is considered a computational challenge to solve the Max-Cut to optimality. In practice we typically solve such problems only approximately by using a heuristic or an approximation algorithm [12, 16]. However, to compare these algorithms and evaluate their performance, we still require the optimum solutions. Considering all this, solving increasingly large instances of Max-Cut to optimality on parallel computers is highly needed in scientific computing.

1.2 Problem formulation and notations
The central problem that we consider is the Max-Cut problem, which can be defined as follows. For a given undirected graph \( G = (V,E) \) on \( n = |V| \) vertices and with edge weights \( w_e \) for \( e \in E \), the Max-Cut problem asks to find a bipartition of the vertices such that the sum of the weights of the edges across the bipartition is maximum. Let \( A = (a_{ij}) \) denote the weighted adjacency matrix with \( a_{ij} = a_{ji} = w_e \) for edge \( e = \{i,j\} \in E \) and zero otherwise. Encoding
the partitions by vectors \( z = \{0, 1\}^n \), we obtain the following unconstrained binary quadratic optimization problem formulation for Max-Cut:

\[
\text{maximize } \sum_{i,j} a_{ij} z_i (1 - z_j) = z^T L_0 z \quad \text{subject to } z \in \{0, 1\}^n,
\]

where \( L_0 \) is the Laplacian matrix of the graph defined by \( L_0 = \text{diag}(A e) - A \). By \( e \) we denoted the vector of all ones. Note that due to the symmetry of the problem we can fix the last element of vector \( z \) to zero and thus removing the last row and column of \( L_0 \), obtaining the matrix \( \hat{L}_0 \in \mathbb{R}^{(n-1) \times (n-1)} \).

The inner product on the space of symmetric matrices is given by \( X \cdot Y = \langle X, Y \rangle = \text{tr}(XY) = \sum_{i,j} X_{ij} Y_{ij} \) and the associated Frobenious norm is defined by \( \|X\|_F = \sqrt{\langle X, X \rangle} = \sqrt{\sum_{i,j} X_{ij}^2} \).

Using the property of inner product \( z^T L_0 z = \langle L_0, zz^T \rangle \) we can reformulate problem (1) as:

\[
\text{maximize } \left[ \begin{array}{cc}
\hat{L}_0 & 0 \\
0 & 0 \\
\end{array} \right] \left[ \begin{array}{c}
zz^T \\
z^T \\
z \\
1
\end{array} \right]
\]

subject to \( z \in \{0, 1\}^{n-1} \).

We increased the dimension of the problem by one, since the last column of the solution matrix of a semidefinite relaxation will be used for determining the next branching variable.

In the following, we denote by Diag\( (v) \) the diagonal matrix having \( v \) on its diagonal and \( \text{diag}(X) \) denotes the vector obtained by extracting the main diagonal from the matrix \( X \). For given symmetric matrices \( A_i, i = 1, \ldots, m \), let \( A : \mathbb{S}_n \to \mathbb{R}^m \) denote the linear operator mapping \( n \times n \) symmetric matrices to \( \mathbb{R}^m \) with \( A(X)_i = \langle A_i, X \rangle \). Its adjoint is well known to be \( A^T(y) = \sum_{i=1}^m y_i A_i \). We denote the projection of some symmetric matrix \( X \) onto the positive semidefinite cone by \( X_+ \) and its projection onto the negative semidefinite cone by \( X_- \). It is well known that both projections can be computed from the spectral decomposition of \( X = SAS^T \), see, for instance [21]. Let \( S = (S_+, S_-) \) be a partition of the eigenvectors in \( S \) according to positive and negative eigenvalues \( \Lambda = (\Lambda_+, \Lambda_-) \). Then

\[
X = S_+ \Lambda_+ S_+^T + S_- \Lambda_- S_-^T = X_+ + X_-.
\]

1.3 Related work and our contribution

Over the decades many methods have been proposed for finding exact solutions of Max-Cut. Some of them are linear programming based methods [10, 27], which work particularly well when the underlying graph is sparse. Other algorithms combine semidefinite programming with polyhedral approach [19] to strengthen the basic SDP relaxation with cutting planes. Two of them, the BiqMac [33] and BiqCrunch [25] solvers, turned out in the last decade to be the best performing solvers for Max-Cut, also compared to commercial solvers. Both solvers utilize branch and bound (B&B) paradigm, however, the distinction is in using different algorithms for solving the underlying SDP relaxation. Furthermore, they do not use any parallelization. For other recent computational approaches the reader is referred to [31].

Recently, we have developed with other authors a solver BiqBin [17] for the class of binary quadratic problems with linear constraints, which includes Max-Cut problem. This solver uses exact penalty approach to reformulate every instance of binary quadratic problem into an instance of Max-Cut problem and then solve it using enhanced version of BiqMac, where the underlying SDP relaxations are tight due to inclusion of hypermetric inequalities. Additionally, the B&B part of BiqBin was improved compared to BiqMac and the solver was fully parallelized. Extensive numerical evidence shows that BiqBin outperforms BiqMac, BiqCrunch, GUROBI and SCIP on the Max-Cut instances and also on some classes of binary quadratic problems.
The main contribution of this paper is introduction of the parallel exact solver MADAM for Max-Cut, based on the alternating direction method of multipliers (ADMM) which was introduced in [37]. More precisely, we

- Adapt ADMM to solve SDP relaxations of Max-Cut with a subset of hypermetric inequalities included and provide a convergence proof. More specifically, we strengthen the basic SDP relaxation with triangle, pentagonal and heptagonal inequalities. The key idea in solving the underlying relaxation is to introduce a slack variable in order to eliminate solving a quadratic program over the nonnegative orthant when computing the dual variable corresponding to inequality constraints. Instead a sparse system of linear equations is solved and projection onto nonnegative orthant is performed.

- We propose a rounding procedure which rounds the (nearly feasible) solution of the dual problem obtained by ADMM to feasible solutions and therefore provides a valid upper bound for the optimum value of Max-Cut.

- We develop a new B&B algorithm for Max-Cut which exploits the valid upper bounds derived by rounding the nearly feasible solution of ADMM. The B&B is parallelized based on load coordinator-worker paradigm using Message Passing Interface (MPI) library. Scalability results for MADAM show good scaling properties when running on a supercomputer.

- We develop a C implementation of all modules and name it MADAM – Max-Cut Alternating Direction Augmented Lagrangian Method. The source code is available at

  \[https://github.com/HrgaT/MADAM.\]

MADAM is tested against BiqMac, BiqCrunch and BiqBin on several Max-Cut instances from BiqMac library. Numerical results show that MADAM outperforms other approaches. The reason is that our adaption of ADMM has a better balance between the time needed to solve the SDP relaxation and the quality of the solution, compared to other solvers.

The main difference between MADAM, BiqMac and BiqBin is that MADAM applies ADMM to solve the underlying SDP relaxations, while BiqMac and BiqBin use the bundle method. BiqMac strengthens the basic SDP relaxation with triangle inequalities, whereas BiqBin and MADAM include also a subset of pentagonal and heptagonal inequalities, which make these relaxations harder to solve. Using ADMM in MADAM implies that in each B&b node we have a (nearly) feasible solution for the SDP relaxation available, which is not the case with the bundle method applied to the partial Lagrangian dual. This gives us more information on how to generate feasible solutions and how to branch. Furthermore, the obtained SDP bounds are tighter, and consequently the size of the explored B&B tree is smaller compared to other solvers. Parallelization of B&B in MADAM is very similar to parallelization of B&B in BiqBin, since both are based on load coordinator-workers paradigm and use MPI library for communication between the processes.

The rest of this paper is organized as follows. Section 2 deals with semidefinite relaxations of Max-Cut used in this paper. We also describe the class of cutting planes that we use. Description of other solution approaches that are based on semidefinite programming is given in Section 3. In Section 4 we outline our new ADMM algorithm, discuss implementation details and provide proof of convergence. Details on how proposed ADMM method is used within B&B framework are given in Section 5, while Section 6 presents numerical results of the serial algorithm. In Section 7 we give insights on how parallelization improves performance of MADAM solver. We conclude with computational experience of our method and present scalability results.
2 Semidefinite relaxation of Max-Cut

Semidefinite program (SDP) optimizes a linear objective function of the matrix over the intersection of the cone of positive semidefinite matrices with an affine space [40]. Compared to linear programming, SDP relaxations have been shown to provide tighter bounds on optimum value for many combinatorial optimization problems. In particular for the Max-Cut problem, Goemans and Williamson [16] achieve the celebrated 0.879 approximation ratio using SDP relaxation.

Since we will consider semidefinite relaxation of (2), it is more appropriate to work with \(-1/1\) variables. We apply the change of variables between \(z \in \{0, 1\}^{n-1}\) and \(x \in \{-1, 1\}^{n-1}\), defined by \(z = \frac{1}{2}(x + e)\). This can be written in matrix form as

\[
\begin{bmatrix}
  z \\
  1
\end{bmatrix} = \begin{bmatrix}
  \frac{1}{2}I & \frac{1}{2}e \\
  0 & 1
\end{bmatrix} \begin{bmatrix}
  x \\
  1
\end{bmatrix} = U \begin{bmatrix}
  x \\
  1
\end{bmatrix}
\]

and therefore

\[
\begin{bmatrix}
  zz^T \\
  z^T \\
  1
\end{bmatrix} = U \begin{bmatrix}
  xx^T & x \\
  x^T & 1
\end{bmatrix} U^T.
\]

Observe that for any \(x \in \{-1, 1\}^{n-1}\) the matrix \(xx^T\) is positive semidefinite and its diagonal is equal to vector of all ones. Using the above transformation this allows us to write Max-Cut problem in \(-1/1\) variables as

\[
\begin{align*}
\text{maximize} & \quad \langle L, X \rangle \\
\text{subject to} & \quad \text{diag}(X) = e \\
& \quad X = \begin{bmatrix}
  xx^T & x \\
  x^T & 1
\end{bmatrix}
\end{align*}
\]

or equivalently

\[
\begin{align*}
\text{maximize} & \quad \langle L, X \rangle \\
\text{subject to} & \quad \text{diag}(X) = e \\
& \quad X \succeq 0 \\
& \quad \text{rank}(X) = 1
\end{align*}
\tag{MC}
\]

where

\[
L = U^T \begin{bmatrix}
  \widetilde{L}_0 & 0 \\
  0 & 0
\end{bmatrix} U = \frac{1}{4} \begin{bmatrix}
  \widetilde{L}_0 & \frac{\widetilde{L}_0 e}{e^T \widetilde{L}_0} \\
  \frac{\widetilde{L}_0 e}{e^T \widetilde{L}_0} & \frac{\widetilde{L}_0 e}{e^T \widetilde{L}_0}
\end{bmatrix} \in \mathbb{R}^{n \times n}.
\]

By dropping the rank-one constraint we obtain the basic SDP relaxation

\[
\begin{align*}
\text{maximize} & \quad \langle L, X \rangle \\
\text{subject to} & \quad \text{diag}(X) = e \\
& \quad X \succeq 0
\end{align*}
\tag{3}
\]

Interior-point methods (IPM) turned out to be the most prominent algorithms for solving SDPs. There exist several numerical packages to solve semidefinite programs, e.g., CSDP [2], MOSEK [30], SeDuMi [35] and SDPT3 [36]. Many of them are based on some variant of the primal-dual path following interior-point method, see for example [20].

Despite their expressive power we are limited to solving small and middle size SDPs in practice. As the size of problems and number of constraints grow, interior-point methods show poor scalability for large-scale problems. However, for special cases like the relaxation [3],
interior-point method tailored for this problem scales well, since the matrix determining the Newton direction can be efficiently constructed [20].

The bound from (3) is not strong enough to be successfully used within a branch and bound framework in order to solve larger Max-Cut problems to optimality. By adding additional equality or inequality constraints, known as cutting planes, which are valid for all feasible solutions of (MC), we strengthen the upper bound. One such class of cutting planes, called triangle inequalities, is obtained as follows. Observe that for an arbitrary triangle with vertices $i < j < k$ in the graph, any partition of vertices cuts either 0 or 2 of its edges. Moving to $\{-1, 1\}$ model this leads to

\[-X_{ij} - X_{ik} - X_{jk} \leq 1,\]
\[-X_{ij} + X_{ik} + X_{jk} \leq 1,\]
\[X_{ij} - X_{ik} + X_{jk} \leq 1,\]
\[X_{ij} + X_{ik} - X_{jk} \leq 1,\]

(see [19] and [33]). The inequalities are collected as $B_{\text{TRI}}(X) \leq e$ and we end up with the following SDP relaxation:

\[
\begin{align*}
\text{maximize} & \quad \langle L, X \rangle \\
\text{subject to} & \quad \text{diag}(X) = e \\
& \quad B_{\text{TRI}}(X) \leq e \\
& \quad X \succeq 0.
\end{align*}
\]

(MCSDP)

To exploit a stronger relaxation of (MCSDP), the bound is further strengthened by using pentagonal and heptagonal inequalities, which belong to the family of hypermetric inequalities valid for any matrix $X$ from the convex hull of rank-one matrices $xx^T$, for $x \in \{-1, 1\}^n$. For every integer vector $b$ such that $e^T b$ is odd, the inequality $|x^T b| \geq 1$ holds for all $x \in \{-1, 1\}^n$. Therefore we have $\langle bb^T, xx^T \rangle \geq 1$. Hypermetric inequality specified by the vector $b$ is the inequality

\[\langle bb^T, X \rangle \geq 1.\]

We consider the subset of hypermetric inequalities generated by choosing $b$ with $b_i \in \{-1, 0, 1\}$ and by fixing the number of non-zero entries to 3, 5, or 7. In this cases, we obtain triangle, pentagonal, and heptagonal inequalities, respectively, and collect them as $B(X) \leq e$. We obtain the following strengthening of (MCSDP):

\[
\begin{align*}
\text{maximize} & \quad \langle L, X \rangle \\
\text{subject to} & \quad \text{diag}(X) = e \\
& \quad B(X) \leq e \\
& \quad X \succeq 0.
\end{align*}
\]

(MCHYP)

We now describe the routine for separating the proposed cutting planes. There are $4\binom{n}{3}$ triangle inequalities, but for given $X$ we can enumerate all of them and identify the most violated ones. Due to a large number of higher order hypermetric inequalities, the separation of pentagonal and heptagonal inequalities is done heuristically. We use the idea proposed in BiqBin, where the separation problem is reformulated as a quadratic assignment problem of the form

\[
\begin{align*}
\text{minimize} & \quad \langle H, PP^T \rangle \\
& \quad P \in \Pi,
\end{align*}
\]

where $\Pi$ is the set of all $n \times n$ permutation matrices and $H$ determines the type of pentagonal or heptagonal inequality specified by the vector $b$. The problem is then approximately solved by using simulated annealing to obtain inequality with potentially large violation. The reader is referred to [17] for more details.
3 Other solution approaches based on semidefinite programming

In this section we summarise other approaches for solving Max-Cut problem to optimality, especially the BiqMac [33], BiqBin [17] and BiqCrunch [25] solvers, which are widely considered as the most powerful solvers for unconstrained binary quadratic problems.

3.1 BiqMac and BiqBin

The starting point of BiqMac is the strengthened SDP relaxation (MCSDP). By dualizing only the triangle inequality constraints, the authors obtain the convex nonsmooth partial dual function

\[
f(\gamma) = e^T \gamma + \max_{\text{diag}(X) = e, X \succeq 0} \langle L - B^T_{\text{tri}}(\gamma), X \rangle,
\]

where \( \gamma \) is the nonnegative dual variable. Evaluating the dual function and computing a subgradient amounts to solving an SDP of the form (3), which can be efficiently computed using an interior-point method tailored for this problem. The approximate minimizer of the dual problem

\[
\begin{align*}
\text{minimize} & \quad f(\gamma) \\
\text{subject to} & \quad \gamma \geq 0
\end{align*}
\]

is then computed using the bundle method [23]. In BiqBin the bound is further strengthened by also dualizing pentagonal and heptagonal inequalities, i.e. the bundle method is used to minimize the partial dual function of (MCHYP).

3.2 BiqCrunch

Due to diagonal constraint in (MCSDP) it can easily be proven that all feasible matrices of this problem satisfy the condition \( n^2 - \|X\|_F^2 \geq 0 \), in which the equality holds if and only if the matrix has rank 1. The idea of BiqCrunch is to add a multiple of this quadratic regularization term to the objective function. They obtain the following regularized problem:

\[
\begin{align*}
\text{maximize} & \quad \langle L, X \rangle + \frac{\alpha}{2} \left( n^2 - \|X\|_F^2 \right) \\
\text{subject to} & \quad \text{diag}(X) = e \\
& \quad B_{\text{tri}}(X) \leq e \\
& \quad X \succeq 0.
\end{align*}
\]

By using quadratic regularization the simplified dual problem

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2\alpha} \left\| \left[ \text{Diag}(y) + B^T_{\text{tri}}(z) - L \right] \right\|_F^2 + e^T y + e^T z + \frac{\alpha}{2} n^2 \\
\text{subject to} & \quad y \text{ free}, \quad z \geq 0
\end{align*}
\]

produces a family of upper bounds dependent on the penalty parameter \( \alpha \). Note that the objective function is convex and differentiable, see [24]. The function value and gradient are evaluated by computing partial spectral decomposition. This enables that for fixed \( \alpha \) the dual function is minimized using L-BFGS-B algorithm [5] from the family of quasi-Newton methods. Reducing the penalty parameter \( \alpha \) increases the tightness of the upper bound. This is due to decreased impact of regularization term in the primal objective function of (4).
4 Alternating direction method of multipliers

Several algorithmic alternatives to interior-point methods have been proposed in the literature to solve semidefinite programs \[4,13,29\]. Alternating direction method of multipliers (ADMM) has been studied over the last decades due to its wide range of applications in a number of areas and capability of solving large-scale problems \[3,32,37\]. ADMM is an algorithm that solves convex optimization problems by breaking them into smaller pieces which are easier to handle. This is achieved by alternating optimization of augmented Lagrangian function. The method deals with the following problems with only two blocks of functions and variables:

\[
\begin{align*}
\text{minimize} & \quad f_1(x_1) + f_2(x_2) \\
\text{subject to} & \quad A_1x_1 + A_2x_2 = b \\
& \quad x_1 \in X_1, \quad x_2 \in X_2.
\end{align*}
\]

(5)

Let

\[
L_\rho(x_1, x_2, y) = f_1(x_1) + f_2(x_2) + y^T(A_1x_1 + A_2x_2 - b) + \frac{\rho}{2} ||A_1x_1 + A_2x_2 - b||_2^2
\]

be the augmented Lagrangian function of (5) with the Lagrange multiplier \(y\) and a penalty parameter \(\rho > 0\). Then ADMM consists of iterations

\[
\begin{align*}
x_1^{k+1} &= \arg \min_{x_1 \in X_1} L_\rho(x_1, x_2^k, y^k) \\
x_2^{k+1} &= \arg \min_{x_2 \in X_2} L_\rho(x_1^{k+1}, x_2, y^k) \\
y^{k+1} &= y^k + \rho \left( A_1x_1^{k+1} + A_2x_2^{k+1} - b \right).
\end{align*}
\]

(6)

For details the reader is referred to \[3\] and references therein.

ADMM has already been successfully applied for solving semidefinite relaxations of combinatorial optimization problems. The Boundary Point Method \[32\] is an augmented Lagrangian method applied to the dual SDP in standard form. In the implementation the alternating optimization is used since in the inner loop only one iteration is performed. The method is currently one of the best algorithms for computing the theta number of a graph \[28\]. However, the drawback of the method is that it can only solve equality constrained semidefinite programs. Wen et al. \[37\] present ADMM for general semidefinite programs with equality and inequality constraints. Compared to our approach the drawback of their method is in solving the quadratic program over nonnegative orthant when computing the dual variable corresponding to inequality constraints. Furthermore, they use multi-block ADMM for which the proof of convergence is not presented.

4.1 ADMM method for (MCHYP)

By introducing the slack variable \(s\) the problem (MCHYP) is equivalent to

\[
\begin{align*}
\text{maximize} & \quad \langle L, X \rangle \\
\text{subject to} & \quad \text{diag}(X) = e \\
& \quad B(X) + s = e \\
& \quad X \succeq 0, \quad s \geq 0.
\end{align*}
\]

(7)
One can easily verify that its dual problem can be written as

\[
\begin{align*}
\text{minimize} & \quad e^T y + e^T t \\
\text{subject to} & \quad L - \text{Diag}(y) - B^T(t) + Z = 0 \\
& \quad u - t = 0 \\
& \quad y, t \text{ free}, \ Z \succeq 0, \ u \geq 0,
\end{align*}
\]

where \( y \) and \( t \) are dual variables associated respectively with the equality constraints, whereas \( u \) and \( Z \) are dual multipliers to the conic constraints.

Let \( m \) denote the number of hypermetric inequalities. For fixed \( \rho > 0 \) consider the augmented Lagrangian \( L_\rho \) for (8):

\[
\begin{align*}
L_\rho(y, t, Z, u; X, s) & = e^T y + e^T t + \langle L - \text{Diag}(y) - B^T(t) + Z, X \rangle + \\
& \quad \frac{\rho}{2} \| L - \text{Diag}(y) - B^T(t) + Z \|^2_F + \langle u - t, s \rangle + \frac{\rho}{2} \| u - t \|^2 \\
& = e^T y + e^T t + \frac{\rho}{2} \| L - \text{Diag}(y) - B^T(t) + Z + X/\rho \|^2_F + \\
& \quad \frac{\rho}{2} \| u - t + s/\rho \|^2_2 - \frac{1}{2\rho} \| X \|^2_F - \frac{1}{2\rho} \| s \|^2_2.
\end{align*}
\]

Alternating direction method of multipliers for problem (8) consists of alternating minimization of \( L_\rho \) with respect to one variable while keeping others fixed to get \( y, Z, t \) and \( u \). Then the primal variables \( X \) and \( s \) are updated using the following rules:

\[
\begin{align*}
y^{k+1} & = \arg \min_{y \in \mathbb{R}^n} L_\rho(y, \mathbf{t}^k, Z^k, u^k; X^k, s^k) \quad (9a) \\
t^{k+1} & = \arg \min_{t \in \mathbb{R}^m} L_\rho(y^{k+1}, t, Z^k, u^k; X^k, s^k) \quad (9b) \\
Z^{k+1} & = \arg \min_{Z \succeq 0} L_\rho(y^{k+1}, t^{k+1}, Z, u^k; X^k, s^k) \quad (9c) \\
u^{k+1} & = \arg \min_{s \in \mathbb{R}^k_+} L_\rho(y^{k+1}, t^{k+1}, Z^{k+1}, u; X^k, s^k) \quad (9d) \\
X^{k+1} & = X^k + \rho \left( L - \text{Diag}(y^{k+1}) - B^T(t^{k+1}) + Z^{k+1} \right) \quad (9e) \\
s^{k+1} & = s^k + \rho \left( u^{k+1} - t^{k+1} \right) \quad (9f)
\end{align*}
\]

Let us closely look at the subproblems (9a)–(9f). The first order optimality condition for problem (9a) is

\[
\nabla_y L_\rho = e + \rho y - \rho \text{diag}(L - B^T(t) + Z + X/\rho) = 0.
\]

Hence the computation of \( y \) is trivial

\[
y = -\frac{e}{\rho} + \text{diag}(L - B^T(t) + Z + X/\rho).
\]

Similarly for (9b), we compute the gradient with respect to \( t \)

\[
\nabla_t L_\rho = e - s - \rho(u - t) - \rho B(L - \text{Diag}(y) - B^T(t) + Z + X/\rho) = 0.
\]

Therefore \( t \) is the solution of the following linear system

\[
(BB^T + I)t = B(L - \text{Diag}(y) + Z + X/\rho) + u + \frac{s - e}{\rho}.
\]
Note that the matrix $BB^T + I$ is sparse and positive definite. Hence the above linear system can be efficiently solved using sparse Cholesky factorization.

By defining $M = L - \text{Diag}(y) - B^T(t) + X/\rho$, the subproblem (9c) can be formulated as

$$\arg\min_{Z \succeq 0} ||M + Z||_F^2,$$

whose solution

$$Z = (-M)_+ = -M_- \tag{10}$$

is the projection of the matrix $-M$ onto the positive semidefinite cone.

The subproblem (9d) can be written as

$$\arg\min_{u \geq 0} ||u - t + s/\rho||_2^2.$$

It asks for the nonnegative vector that is closest to $v = t - s/\rho$. Hence the solution is

$$u = \max(0, v) = v_+,$$

the nonnegative part of vector $v$.

After all variables for the dual problem (8) have been updated in alternating manner, we compute primal variables $X$ and $s$ using the expressions (9e) and (9f). As already observed in the Boundary Point Method [32], the update rule for $X$ can be simplified and also computed from the spectral decomposition of matrix $M$. Using (9e) we get

$$X^{k+1} = X^k + \rho \left( L - \text{Diag}(y^{k+1}) - B^T(t^{k+1}) + Z^{k+1} \right)$$

$$= \rho \left( L - \text{Diag}(y^{k+1}) - B^T(t^{k+1}) + X^k/\rho + Z^{k+1} \right)$$

$$= \rho(M - M_-) = \rho M_+.$$

Similarly, we can simplify the formula for (9f)

$$s^{k+1} = s^k + \rho(u^{k+1} - t^{k+1})$$

$$= \rho \left( u^{k+1} - (t^{k+1} - s^k/\rho) \right)$$

$$= \rho(v_+ - v) = -\rho v_-.$$

Hence by construction the matrix $X$ is positive semidefinite and the vector $s$ is nonnegative.

The overall complexity of one iteration of the method is solving a linear system with matrix $BB^T + I$ and computing the partial eigenvalue decomposition of $M$. Compared to interior-point methods where coefficient matrix changes in each iteration, matrix $BB^T + I$ remains constant throughout the algorithm and its factorization can be cached at the beginning to efficiently solve linear system in each iteration.

Difference between our algorithm, described in Algorithm 1, and the method proposed by Wenn et al. [37] is in the update rule of the dual variable $t$ corresponding to inequality constraints. The authors directly apply ADMM on the dual of (MCHYP). In our approach we introduced slack variable $s$ resulting in an unconstrained optimization problem for variable $t$ in (9d). This reduced the overall complexity of the algorithm by eliminating the need to solve a convex quadratic program of order $m$ over nonnegative orthant, especially since multiple hypermetric inequalities are added to strengthen to bound. Instead a sparse system of linear equations is solved and projection onto nonnegative orthant is used.
4.2 Implementation

The above update rules ensure that during the algorithm nonnegativity of vectors $u$ and $s$, and conic constraints for matrices $X$ and $Z$ are maintained, as well as complementarity conditions $u^T s = 0$ and $ZX = 0$. Hence, once the primal and dual feasibility are reached, the method converged to the optimal solution. To measure the accuracy of primal and dual feasibility we use

$$r_P = \frac{\| \text{diag}(X) - e \|_2 + \| \max (B(x) - e, 0) \|_2}{1 + \sqrt{n}},$$

$$r_D = \frac{\| L - \text{Diag}(y) - B^T(t) + Z \|_F + \| u - t \|_2}{1 + \| L \|_F}.$$ 

We terminate our algorithm when $\max\{r_P, r_D\} < \varepsilon$, for prescribed tolerance $\varepsilon > 0$.

The performance of the method is dependent on the choice of the penalty parameter $\rho$. Numerical experiments show that for the problems we considered the starting value of $\rho = 1$ or $\rho = 1.6$ is a good choice and the value is dynamically tuned during the algorithm in order to improve the practical convergence. Simple strategy to adjust the value of $\rho$ is observing the residuals:

$$\rho^{k+1} = \begin{cases} 
\tau \rho^k & \text{if } \log \left( \frac{r_D}{r_P} \right) > \mu \\
\rho^k / \tau & \text{if } \log \left( \frac{r_P}{r_D} \right) > \mu \\
\rho^k & \text{otherwise,}
\end{cases} \quad (11)$$

for some parameters $\mu$ and $\tau$. In our numerical tests we have used $\mu = 0.5$ and $\tau = 1.001$. The idea behind this penalty parameter update scheme is to try to keep the primal and dual residual norms in the same order of magnitude as they both converge to zero.

The computational time of our ADMM method is essentially determined by the number of partial eigenvalue decompositions and efficiency of sparse Cholesky solver, since these are the most computationally expensive steps. For obtaining positive eigenvalues and corresponding eigenvectors we use LAPACK \cite{lapack} routine \texttt{DSYEV}. For factoring the matrix $BB^T + I$ and then performing backsolves to get $t$ we use sparse direct solver from CHOLMOD \cite{cholmod}, a high performance library for sparse Cholesky factorization. CHOLMOD is part of the SuiteSparse linear algebra package \cite{sparse}.

In the following two subsections we elaborate on two potential issues when using ADMM and how to resolve them. These are obtaining safe upper bound which can be used within B&B algorithm and the convergence of multi-block ADMM.

4.3 Safe upper bound

To safely use the proposed upper bound within B&B algorithm we need a certificate that the value of the dual function is indeed a valid upper bound for the original problem \cite{kyng2016}. To achieve this the quadruplet $(y, t, Z, u)$ has to be dual feasible, i.e. by assigning $t \leftarrow u$, the equation

$$L - \text{Diag}(y) - B^T(u) + Z = 0$$

has to be satisfied. However, since we only approximately solve primal-dual pair of semidefinite programs to some precision, the dual feasibility is not necessarily reached when the algorithm terminates. Note that the variable $y$ is unconstrained, whereas the conic conditions on $u$ and $Z$ are satisfied by construction. In the following we describe the post-processing step we do after each computation of the bound, provided the pruning condition is satisfied.
Proposition 1. Let $y$, $t$, $u$ and $Z$ be the output variables computed with iteration scheme (9a) – (9f). Let $\lambda_{\min}$ denote the smallest eigenvalue of matrix $\hat{Z} := \text{Diag}(y) + B^T(u) - L$. If $\lambda_{\min} \geq 0$, then the value $e^T y + e^T u$ is a valid upper bound for problem (1). Otherwise the value $e^T \hat{y} + e^T u$, where $\hat{y} = y - \lambda_{\min} e$, provides an upper bound on the largest cut. Furthermore, it always holds that the value $e^T \hat{y} + e^T u$ is larger than $e^T y + e^T u$.

Proof. After ADMM method terminates and by assigning $t \leftarrow u$, the quadruplet $(y, t, Z, u)$ satisfies all the constraints of (8) within machine accuracy, only the linear constraint

$$L - \text{Diag}(y) - B^T(u) + Z = 0$$

may not be satisfied. By replacing $Z$ with $\hat{Z}$ we satisfy this linear constraint, but the positive semidefiniteness of $\hat{Z}$ might be violated. If the smallest eigenvalue $\lambda_{\min}$ of $\hat{Z}$ is nonnegative, then new quadruplet $(y, t, \hat{Z}, u)$ is feasible for (8) and the value $e^T y + e^T u$ provides an upper bound on the size of the largest cut.

If $\lambda_{\min}$ is negative, we adjust the matrix $\hat{Z}$ to be positive semidefinite by using

$$\hat{Z} \leftarrow \hat{Z} - \lambda_{\min} I.$$ 

To maintain dual equality constraints, we correct the unconstrained variable as $\hat{y} = y - \lambda_{\min} e$. It is obvious that the value $e^T \hat{y} + e^T u = e^T y + e^T u - n\lambda_{\min}$ is larger than $e^T y + e^T u$. \qed

The last statement of Proposition 1 guarantees that the computation of correction vector $\hat{y}$ is needed, only when we can prune the B&B node with the bound $e^T y + e^T u$, because the post-processing step in the case of a negative eigenvalue produces the bound which is always larger.

We now summarize the ADMM-based algorithm for solving SDP relaxation (MCHYP) of Max-Cut. In the view of the previous observations, the algorithm is described in Algorithm 1.\[\]

4.4 Convergence of the method

It has been recently shown [6] that multi-block ADMM is not necessarily convergent. In the following theorem we prove that due to the special structure of operators in the case of semidefinite relaxation of Max-Cut we can achieve convergent scheme by reducing it to a 2-block method.

Theorem 1. The sequence $\{(X^k, s^k, y^k, t^k, Z^k, u^k)\}$ generated by Algorithm 1 from any starting point $(X^0, s^0, y^0, t^0, Z^0, u^0)$ converges to a solutions $(X^*, s^*)$, $(y^*, t^*, Z^*, u^*)$ of primal-dual pair of semidefinite programs (7) and (8).

Proof. The convergence of our multi-block ADMM is guaranteed due to the orthogonality relations of the operators diag and $\mathcal{B}$ and their adjoints:

$$\mathcal{B} \left(\text{Diag}(y)\right) = 0, \quad \text{diag} \left(\mathcal{B}^T(t)\right) = 0,$$

for any vectors $y \in \mathbb{R}^n$ and $t \in \mathbb{R}^m$. (12)

In this case the multi-block ADMM reduces to a special case of the original method [3]. To see this, note that orthogonality in (12) implies that the first order optimality conditions for variables $y$ and $t$ in (9a) and (9b) reduce to

$$\nabla_y L_\rho = e + \rho y - \rho \text{diag}(L + Z + X/\rho) = 0$$

$$\nabla_t L_\rho = e - s - \rho(u - t) - \rho \mathcal{B}(L - \mathcal{B}^T(t) + Z + X/\rho) = 0,$$

meaning that $y$ and $t$ are independent and are jointly minimized by regarding $(y, t)$ as one variable. Similarly, update rules for variables $Z$ and $u$, as well as for primal pair $X$ and $s,$
Algorithm 1: ADMM method for solving SDP relaxation (MCHYP) of Max-Cut

Output: Upper bound on the largest cut of a graph.

Select $\rho > 0$, $\varepsilon > 0$;

$k = 0$, $u^k = 0$, $s^k = 0$, $X^k = 0$, $Z^k = 0$;

Compute sparse Cholesky factorization of matrix $BB^T + I$;

for $k = 0, 1, \ldots$ do

Compute $y^{k+1} = -\frac{e}{\rho} + \text{diag}(L + Z^k + X^k/\rho)$;

Solve for $t^{k+1}$ by using cached factorization:

$v = \rho^{-1} - s^k/\rho$;

$M = L - \text{Diag}(y^{k+1}) - B^T(t^{k+1}) + X^k/\rho$;

$Z^{k+1} = -M_-$;

$u^{k+1} = v_+;

X^{k+1} = \rho M_+$;

$s^{k+1} = -\rho v_-$;

Compute $\delta = \max\{r_p, r_D\}$;

if $\delta < \varepsilon$ then break;

Update penalty parameter $\rho$ according to (11).

Post-processing:

$\hat{Z} = \text{Diag}(y) + B^T(t) - L$;

Compute the smallest eigenvalue $\lambda_{\text{min}}(\hat{Z})$;

if $\lambda_{\text{min}}(\hat{Z}) \geq 0$ then return $e^Ty + e^Tu$;

else return $e^Ty + e^Tu - n\lambda_{\text{min}}(\hat{Z})$;

ensure that they can also be jointly minimized. By separately regarding $(Z, u)$ and $(X, s)$ as one variable, the iterate scheme (9a) – (9f) can be written as:

$$
(y^{k+1}, t^{k+1}) = \arg \min_{(y, t) \in \mathbb{R}^n \times \mathbb{R}^m} L_{\rho} \left( (y, t), (Z^k, u^k); (X^k, s^k) \right)
$$

$$
(Z^{k+1}, u^{k+1}) = \arg \min_{(Z, u) \in S^n_+ \times \mathbb{R}^m} L_{\rho} \left( (y^{k+1}, t^{k+1}), (Z, u); (X^k, s^k) \right)
$$

$$
(X^{k+1}, s^{k+1}) = (X^k, s^k) + \rho \left( L - \text{Diag}(y^{k+1}) - B^T(t^{k+1}) + Z^{k+1}, u^{k+1} - t^{k+1} \right).
$$

Thus the convergence of Algorithm 1 is implied by the analysis in [37], that looks at 2-block ADMM as a fixed point method.

5 Branch and bound

The prominent bounds obtained by using ADMM method for solving (MCHYP) motivated us to use this method within a branch and bound framework. We call the outcome of this work MADAM solver – Max-Cut Alternating Direction Augmented Lagrangian Method.

The Branch and Bound algorithm (B&B) is one of the standard enumerative methods for computing the global optimum of a NP-hard problem. It consists of the following ingredients:

- the bounding procedure, which provides for each instance of a problem an upper bound for the optimum value;
- the branching procedure, which splits the current problem into more problems of smaller dimensions by fixing some variables;
- a heuristic for generating feasible solution providing a lower bound.

**Bounding routine.** The bounding routine of **MADAM** applies the ADMM algorithm, described in Algorithm 1 which for a given set of triangle, pentagonal and heptagonal inequalities minimizes the dual function \( \mathcal{S} \). In order to obtain tight upper bound we use the cutting-plane approach where multiple hypermetric inequalities are iteratively added and purged after each computation of upper bound. First, the optimal solution of the basic semidefinite relaxation \( \mathcal{S} \) is computed using interior-point method. Then we separate the initial set of triangle inequalities. After the minimizer of problem \( \mathcal{S} \) is obtained, we purge all inactive constraints and new violated triangles are added. The problem with updated set of inequalities is solved and the process is iterated as long as the decrease of the upper bound is sufficiently large. After separating the most violated triangle inequalities, we add pentagonal and heptagonal inequalities in order to further decrease the upper bound. We monitor the maximum violation of triangle inequalities \( r_{\text{TRI}} \) and as soon as the number is sufficiently small, we use the heuristic from Section 2 to add some strongly violated pentagonal inequalities to the relaxation. Similarly, as the maximum violation of pentagonal inequalities \( r_{\text{PENT}} \) drops below some threshold, new heptagonal inequalities are separated and added to the relaxation. In our numerical tests, we have used the thresholds \( r_{\text{TRI}} < 0.2 \) and \( r_{\text{PENT}} < 0.4 \) and each time separated at most \( 10 \cdot n \) violated triangle inequalities. For pentagonal and heptagonal inequalities different strategy is used. The initial number of these cutting planes is small, but each time the node is not pruned, this number is increased by 200. A description of our bounding routine is in Algorithm 2.

Algorithm 2: Semidefinite bounding procedure of **MADAM**

1. solve the basic sdp relaxation \( \mathcal{S} \);
2. separate initial set of triangle inequalities;
3. while upper bound decreases significantly do
   - use ADMM method to obtain approximate maximizer \( X \) of \( \mathcal{MCHYP} \);
   - remove hypermetric inequalities that are not active;
   - add new hypermetric inequalities that are violated;

In order to improve the performance of **MADAM** we can stop the bounding routine, when we detect that we will not be able to prune the current B&B node. BiqCrunch terminates the bounding routine if the difference between consecutive bounds is less than some prescribed parameter and if the gap between upper and lower bound is still large. In **MADAM** we borrow an idea from BiqMac. After some cutting plane iterations, we make a linear forecast to decide whether it is worth doing more iterations. If the gap cannot be closed, we terminate the bounding routine, branch the current node, and start evaluating new subproblems.

**Branching rules.** In **MADAM** we use two branching strategies which are based on the approximate solution matrix \( X \) obtained from Algorithm 1. Once the ADMM method terminates, the last column of \( X \), with exception of the last element, is extracted. Due to the diagonal constraint and positive semidefiniteness of feasible matrices of \( \mathcal{MCHYP} \) all the entries lie in the interval \([−1, 1]\). By using the transformation \( z = \frac{1}{2}(x + e) \), a vector with entries in \([0, 1]\) is obtained. We can follow different strategies based on the information we get from \( z \). Similarly to BiqCrunch and BiqBin, the decision on which variable \( z_i \) to branch on is based on the following two strategies:

- most-fractional: we branch on the vertex \( i \) for which the variable \( z_i \) is closest to 0.5;
ii least-fractional: we branch on the vertex \( i \) for which the variable \( z_i \) is furthest from 0.5;

With this we create two new subproblems, one where \( z_i \) is fixed to 0 and the other where it is fixed to 1. The subproblems correspond to nodes in the B&B tree and are added to the priority queue of unexplored problems. Priority is based on the upper bound computed by the ADMM method. When selecting the next subproblem, a node with the worst upper bound is evaluated first.

**Generating feasible solutions.** For generating high quality feasible solutions of Max-Cut problem we use the optimum solution \( X \) of \( \text{(MCHYP)} \) in Goemans-Williamson rounding hyperplane technique [16]. By using factorization \( X = V^T V \) with column vectors \( v_i \) of \( V \) and selecting some random vector \( r \), one side of the cut is obtained as \( \{ i \mid v_i^T r \geq 0 \} \). The cut vector \( x \) obtained from this heuristic is then further improved by flipping the vertices. Note that in our case the factorization of \( X \) is already available from the bounding routine, which computes an eigenvalue decomposition of matrix \( M \). To summarize, for generating good cuts we use the following scheme:

i Use the Goemans-Williamson rounding hyperplane technique to generate cut vector \( x \) from minimizer of \( \text{(MCHYP)} \).

ii The cut \( x \) is locally improved by checking all possible moves of a single vertex to the opposite partition block.

The process is repeated several times with different random vectors \( v \), due to its low computational cost (in practice we repeat it \( n \) times, where \( n \) is the size of the input graph). Numerical experiments show that by using the proposed heuristic the optimum solution is usually found already in the root node.

**Strategy for faster enumeration of the B&B tree.** To obtain tight upper bounds on maximum cuts we strengthen the basic SDP relaxation with a subset of hypermetric inequalities. Adding these cutting planes to the model and solving the relaxation in each B&B node is computationally expensive and not necessary, especially if one can not prune the node. We use the strategy proposed in BiqBin, where we check (before including the hypermetric inequalities) whether we will be able to prune the current node or whether it is better to branch.

Firstly, in the root node we cache the bound \( \text{OPT}_{\text{SDP}} \) of the basic SDP relaxation (3) and compute the bound \( \text{OPT}_{\text{HYP}} \) by iteratively adding violated hypermetric inequalities. Let

\[
diff = \text{OPT}_{\text{SDP}} - \text{OPT}_{\text{HYP}}
\]

denote the difference between optimal values of both relaxations and let \( lb \) denote the current lower bound. Secondly, at all other nodes, we only compute the basic SDP relaxation. If the condition

\[
\text{OPT}_{\text{SDP}} \leq lb + \diff + 1
\]

(13) is satisfied, meaning we are close enough to the lower bound, we estimate that by adding the cutting planes to compute the tighter bound \( \text{OPT}_{\text{HYP}} \) we will prune the node. This idea helps efficiently traverse the B&B tree and we only invest time into the bounding routine when it is really needed. Numerical experiments show that overall this strategy produces more B&B nodes than necessary, but the performance of the algorithm is immensely improved, since in the first few levels of the B&B tree only the basic SDP relaxation needs to be computed.
6 Numerical results - serial algorithm

We compare our serial version of MADAM with three state-of-the-art semidefinite based exact solvers for Max-Cut, BiqMac [33], BiqCrunch [25] and BiqBin [17]. All solvers are written in C programming language. Several graphs available in the BiqMac library [39], with 100 vertices were used to test the algorithms. For BiqMac and BiqCrunch we used their default settings, except each instance is solved with both branching rules that the solvers provide. The one that produced faster time of execution is used in the tables presenting numerical results. The most-fractional rule is set as the default strategy for BiqBin and MADAM solver. We have run the algorithm with different penalty parameter values $\rho$ found in the literature [37]. The initial value of penalty parameter is set to 1, while for dense graphs from families $w05$, $w09$, $pw05$ and $pw09$ this value was decreased to 1, since we found this works well in our code. In each table we report the needed number of B&B nodes and running time to solve the problem to optimality.

All the computations were done on a cluster consisting of 24-core nodes featuring two Intel Xeon E5-2680V3 running at 2.5 GHz with 64 GB of RAM. The code is compiled against OpenBLAS and LAPACK. Observe that our solver completely outperforms other approaches. For many instances we significantly reduce the running time of the current best solver. The reason for efficiency is simplicity and speed of the proposed bounding routine.

| Problem    | BiqMac | BiqCrunch | BiqBin | MADAM - serial |
|------------|--------|-----------|--------|----------------|
|            | Time   | Nodes     | Time   | Nodes          | Time   | Nodes |
| g05_100.0  | 555.16 | 531       | 355.58 | 329            | 166.93 | 269   | 98.33  | 195   |
| g05_100.1  | 3547.17| 3643      | 1933.51| 1751           | 955.21 | 964   | 494.10 | 705   |
| g05_100.2  | 115.87 | 127       | 125.57 | 95             | 52.76  | 83    | 40.07  | 43    |
| g05_100.3  | 1308.85| 1215      | 602.59 | 643            | 324.56 | 655   | 129.60 | 497   |
| g05_100.4  | 71.03  | 69        | 44.35  | 33             | 18.30  | 15    | 9.71   | 11    |
| g05_100.5  | 116.16 | 129       | 111.12 | 109            | 37.37  | 49    | 28.63  | 31    |
| g05_100.6  | 177.22 | 193       | 139.52 | 99             | 51.05  | 67    | 29.52  | 47    |
| g05_100.7  | 332.35 | 337       | 282.33 | 329            | 145.12 | 121   | 75.31  | 73    |
| g05_100.8  | 291.28 | 275       | 198.62 | 163            | 72.40  | 73    | 35.78  | 67    |
| g05_100.9  | 321.10 | 277       | 232.88 | 229            | 104.52 | 149   | 47.34  | 101   |

Table 1: CPU times (s) and B&B nodes to solve g05 problems.

7 Parallel algorithm

In this section we describe how the algorithmic ingredients from the serial B&B algorithm are combined into a parallel solver which utilizes distributive memory parallelism. OpenMP (Open Multi-Processing) and MPI (Message Passing Interface) are two widely used paradigms available for parallel computing [8,14]. Algorithms based on B&B can be parallelized on different levels. For example, shared memory parallelization using OpenMP can be applied to speed up the computation of bounds, and secondly, exploration of different branches of B&B tree can be done in parallel using distributive scheme. In this case several workers, each with its own memory, are evaluating B&B nodes concurrently and using MPI to share important informations via messages. This is the approach we have taken.

Load coordinator–worker paradigm with distributed work pools is applied, in which one process becomes the master process carefully managing the status of each worker, while remaining processes concurrently explore branches of the B&B tree. Each worker has its own local priority queue of subproblems and work is shared when one of them becomes idle. The master process keeps track of the status of each worker and acts as a load coordinator receiving messages and,
Table 2: CPU times (s) and B&B nodes to solve pm problems.

| Problem | BiqMac | BiqCrunch | BiqBin | MADAM - serial |
|---------|--------|-----------|--------|----------------|
|         | Time   | Nodes     | Time   | Nodes          | Time   | Nodes          |
| pm1d_100.0 | 917.88  | 867       | 821.70 | 1011          | 317.62 | 507           | 138.36 | 287           |
| pm1d_100.1 | 2191.20 | 2113      | 1302.31| 1073          | 510.81 | 611           | 267.08 | 383           |
| pm1d_100.2 | 1458.67 | 1357      | 1003.45| 837           | 369.62 | 707           | 185.09 | 585           |
| pm1d_100.3 | 298.95  | 285       | 273.41 | 281          | 103.10 | 141           | 51.05  | 81            |
| pm1d_100.4 | 1188.23 | 1113      | 756.02 | 607          | 294.30 | 473           | 142.01 | 329           |
| pm1d_100.5 | 260.25  | 255       | 294.19 | 211          | 90.58  | 73            | 57.65  | 61            |
| pm1d_100.6 | 248.40  | 217       | 182.62 | 127          | 72.72  | 117           | 36.80  | 59            |
| pm1d_100.7 | 80.00   | 109       | 104.40 | 103          | 36.80  | 93            | 25.80  | 17            |
| pm1d_100.8 | 107.25  | 97        | 57.18  | 39           | 21.66  | 23            | 12.79  | 21            |
| pm1d_100.9 | 382.85  | 347       | 294.32 | 243          | 112.32 | 135           | 55.55  | 97            |
| pm1s_100.0 | 13.87   | 7         | 16.78  | 9            | 2.78   | 1             | 2.22   | 1             |
| pm1s_100.1 | 105.45  | 83        | 45.86  | 31           | 23.83  | 31           | 12.43  | 15            |
| pm1s_100.2 | 8.47    | 3         | 8.63   | 5            | 1.94   | 1             | 1.12   | 1             |
| pm1s_100.3 | 104.52  | 73        | 51.86  | 39           | 34.47  | 59           | 20.83  | 33            |
| pm1s_100.4 | 73.84   | 51        | 24.32  | 17           | 11.37  | 15           | 8.60   | 7             |
| pm1s_100.5 | 14.67   | 7         | 7.60   | 7            | 2.24   | 1             | 1.39   | 1             |
| pm1s_100.6 | 71.03   | 35        | 19.11  | 15           | 11.52  | 17           | 6.13   | 7             |
| pm1s_100.7 | 0.99    | 1         | 0.88   | 1            | 0.76   | 1             | 0.56   | 1             |
| pm1s_100.8 | 6.27    | 3         | 2.20   | 1            | 1.53   | 1             | 0.95   | 1             |
| pm1s_100.9 | 8.52    | 3         | 1.74   | 1            | 2.14   | 1             | 2.65   | 1             |

Based on their content, acts in appropriate manner. In order to distinguish between different workers, each process is given a fundamental identifier, called rank of the worker.

At the start of the algorithm the master process reads and broadcasts the graph instance to the workers. It is important that every process has the knowledge of the original graph, since construction of subproblems via branching or via received MPI messages is done based on this information. All the data about the B&B nodes is encoded as MPI structure, which is used in communication between different workers in order to efficiently exchange and construct the subproblems.

Next, the load coordinator evaluates the root node and distributes the current best lower bound. After the bounding step two new subproblems are generated, which are then sent to the first two idle processes. Afterwards, load coordinator monitors the status of workers (idle or busy), counts the number of B&B nodes, sends the ranks of free workers and distributes the best solution found so far.

After initialization phase the load coordinator waits for different types of messages sent by the workers. Firstly, if the worker’s local queue is empty, the message is sent informing the master that the process is idle and it can receive further work. Secondly, during the algorithm working processes generate multiple candidates for optimal solutions. The master node is keeping the track of current best optimal value and solution. When a new solution is received, the value is compared, updated if necessary and distributed back during communication phase. And thirdly, the master process helps the computing processes with work sharing by sending messages containing the ranks of idle workers. After a worker computes lower and upper bound, it compares these values to see if this branch of B&B tree can be safely pruned or further branching is needed and construction of new subproblems takes place. In the latter case, request message is sent to the load coordinator, asking for idle processes to share with one of the new generated subproblems and/or subproblems left in the queue from previous branching processes. If no idle worker is available, the generated subproblems remain in the worker’s queue and the work continues locally. Otherwise, subproblems are encoded and sent
| Problem   | BiqMac | BiqCrunch | BiqBin | MADAM - serial |
|-----------|--------|-----------|--------|----------------|
| w01_100.0| 12.31  | 2.50      | 2.45   | 1.24           |
| w01_100.1| 11.80  | 2.65      | 3.05   | 6.69           |
| w01_100.2| 146.67 | 39.01     | 23.23  | 20.71          |
| w01_100.3| 27.46  | 7.13      | 6.99   | 2.42           |
| w01_100.4| 2.08   | 1.10      | 1.52   | 0.85           |
| w01_100.5| 15.80  | 6.17      | 2.86   | 2.19           |
| w01_100.6| 2.62   | 1.16      | 1.88   | 0.88           |
| w01_100.7| 15.31  | 7.27      | 8.52   | 2.92           |
| w01_100.8| 1.22   | 1.07      | 1.04   | 0.98           |
| w01_100.9| 6.40   | 1.38      | 2.01   | 1.40           |
| w05_100.0| 735.17 | 524.39    | 194.08 | 114.82         |
| w05_100.1| 182.31 | 157.68    | 69.11  | 41.82          |
| w05_100.2| 137.86 | 104.17    | 43.36  | 27.14          |
| w05_100.3| 532.43 | 383.18    | 155.61 | 120.45         |
| w05_100.4| 921.26 | 564.00    | 240.62 | 168.03         |
| w05_100.5| 969.22 | 532.27    | 199.36 | 150.84         |
| w05_100.6| 185.23 | 120.55    | 53.42  | 41.32          |
| w05_100.7| 138.66 | 81.11     | 39.22  | 28.79          |
| w05_100.8| 799.63 | 514.31    | 194.68 | 150.44         |
| w05_100.9| 98.13  | 58.88     | 23.88  | 17.70          |
| w09_100.0| 474.91 | 393.72    | 129.50 | 119.88         |
| w09_100.1| 2588.66| 2447.42   | 829.93 | 635.93         |
| w09_100.2| 1043.74| 787.44    | 289.20 | 222.66         |
| w09_100.3| 1242.63| 1098.87   | 336.90 | 263.81         |
| w09_100.4| 575.05 | 575.57    | 194.62 | 138.72         |
| w09_100.5| 35.84  | 17.29     | 9.86   | 10.64          |
| w09_100.6| 81.73  | 110.93    | 27.36  | 35.35          |
| w09_100.7| 318.65 | 287.99    | 87.26  | 76.59          |
| w09_100.8| 174.95 | 188.65    | 87.18  | 76.31          |
| w09_100.9| 504.25 | 358.77    | 119.36 | 113.38         |

Table 3: CPU times (s) and B&B nodes to solve $w$ problems.
| Problem      | BiqMac Time | BiqMac Nodes | BiqCrunch Time | BiqCrunch Nodes | BiqBin Time | BiqBin Nodes | MADAM - serial Time | MADAM - serial Nodes |
|--------------|-------------|--------------|----------------|-----------------|-------------|--------------|--------------------|---------------------|
| pw01_100.1   | 72.27       | 35           | 28.61          | 11              | 3.98        | 1            | 4.23               | 1                   |
| pw01_100.2   | 49.88       | 27           | 23.15          | 11              | 12.41       | 15           | 6.28               | 5                   |
| pw01_100.3   | 61.59       | 27           | 32.13          | 15              | 16.77       | 19           | 12.27              | 9                   |
|              | 6.54        | 3            | 1.32           | 1               | 1.99        | 1            | 1.27               | 1                   |
|              | 7.74        | 3            | 1.47           | 1               | 1.95        | 1            | 1.29               | 1                   |
|              | 32.68       | 15           | 11.44          | 7               | 5.05        | 1            | 5.31               | 1                   |
| pw01_100.7   | 47.78       | 23           | 30.52          | 11              | 12.06       | 11           | 8.40               | 5                   |
|              | 1.40        | 1            | 0.97           | 1               | 1.12        | 1            | 0.89               | 1                   |
| pw01_100.9   | 193.16      | 155          | 86.91          | 45              | 20.31       | 19           | 18.61              | 27                  |
| pw05_100.0   | 1657.05     | 1553         | 1155.51        | 999             | 487.22      | 1205         | 274.04             | 629                 |
|              | 481.93      | 427          | 384.57         | 327             | 162.50      | 243          | 101.95             | 145                 |
| pw05_100.1   | 709.90      | 687          | 454.90         | 359             | 194.99      | 269          | 125.52             | 203                 |
|              | 132.51      | 121          | 129.83         | 93              | 40.96       | 47           | 30.94              | 33                  |
| pw05_100.4   | 793.85      | 777          | 585.08         | 411             | 222.35      | 263          | 133.66             | 179                 |
|              | 190.30      | 175          | 123.76         | 85              | 51.56       | 73           | 33.14              | 47                  |
| pw05_100.6   | 1401.88     | 1385         | 976.62         | 711             | 312.49      | 355          | 217.01             | 269                 |
| pw05_100.7   | 386.94      | 357          | 276.98         | 179             | 86.87       | 71           | 57.71              | 45                  |
|              | 98.62       | 121          | 101.21         | 49              | 25.15       | 17           | 23.05              | 13                  |
| pw05_100.8   | 323.26      | 307          | 321.72         | 327             | 113.98      | 129          | 65.91              | 97                  |
| pw09_100.0   | 579.02      | 527          | 457.12         | 323             | 171.56      | 227          | 107.15             | 163                 |
| pw09_100.1   | 821.02      | 753          | 706.71         | 543             | 248.79      | 311          | 153.76             | 231                 |
| pw09_100.2   | 292.42      | 275          | 248.28         | 175             | 88.34       | 109          | 46.98              | 127                 |
| pw09_100.3   | 153.66      | 153          | 170.37         | 113             | 52.44       | 39           | 34.71              | 27                  |
| pw09_100.4   | 394.53      | 373          | 310.72         | 213             | 109.10      | 147          | 73.68              | 93                  |
| pw09_100.5   | 489.02      | 519          | 513.91         | 321             | 295.02      | 145          | 129.74             | 111                 |
| pw09_100.6   | 390.50      | 337          | 278.82         | 241             | 113.95      | 131          | 72.65              | 77                  |
| pw09_100.7   | 1010.22     | 963          | 824.16         | 589             | 271.59      | 293          | 182.11             | 193                 |
| pw09_100.8   | 305.12      | 295          | 287.91         | 171             | 134.71      | 75           | 86.93              | 59                  |
| pw09_100.9   | 260.41      | 255          | 193.98         | 119             | 72.06       | 69           | 68.10              | 57                  |

Table 4: CPU times (s) and B&B nodes to solve pw problems.
to available idle workers. This is also where exchange of the current best lower bound happens.

When all the workers become idle (all local queues are empty), master process sends message to finish and the algorithm terminates.

Lastly, we elaborate on how efficient distribution of the subproblems to the workers is achieved and how the idle time is reduced. If the number of available workers is large, we need to reach a certain depth of the B&B tree in order for the processes to receive some work. Until this happens in the algorithm, workers are idle and resources are wasted. To fully exploit all the available resources we need a strategy for the worker processes to start evaluating the nodes of B&B tree as soon as possible. This is again where the strategy using the variable diff from Section 5 benefits the algorithm. After computation of the upper bound in the root node, the load coordinator distributes the value diff to the workers. When the first two idle processes evaluate the generated subproblems, typically the value of the basic SDP relaxation is such that condition (13) is not satisfied. Thus, on the first few levels of the B&B tree, the workers compute only the basic SDP bound to faster evaluate the node and idle processes quickly receive the generated subproblems.

7.1 Numerical results - parallel algorithm

We present numerical results comparing serial with parallel version of MADAM solver. As a Message Passing Interface library we use Open MPI to exchange the data between the computing cores. Among the graphs that are used to test the sequential algorithm, we consider only those for which the solver needs more than 200 seconds to compute the optimal solution. We solve the same problem but with increasing number of cores and compare times against the serial algorithm. The results are reported in Table 5.

For the hardest instance w09_100.1 for the serial solver we also present scalability plot to demonstrate the efficiency of parallel version. Due to increased number of workers, the computational time drops, however, larger number of workers also increases communication costs, since multiple request are sent to the load coordinator. This is shown in Figure 1 as the curve diverges from ideal scaling curve.

![Table 5](image)

Table 5: Numerical results obtained with parallel version of MADAM.

Next, we use rudy [34], a machine independent graph generator, to construct new Max-Cut instances with 150 vertices, similarly to the ones in BiqMac library. The parameters for generating the graphs can be found in [38] in Appendix B. Table 6 presents numerical results obtained by applying parallel solver using 240 cores for all instances. We report the maximum cut value and the number of B&B nodes, as well as the time needed to solve the problem to optimality. Furthermore, the number in parantheses next to the nodes specifies the maximum number of used workers. This number is the upper bound on the number of cores we should reserve in order to fully use available resources. Any number of cores beyond makes some of the workers idle and the resources are wasted.
Figure 1: The plots depict scalability of MADAM solver for the w09_100.1 problem. Due to increased number of workers, the computational time drops, however, larger number of workers also increases communication costs which results in the divergence of the curve from the ideal scaling curve (right).
### Table 6: Numerical results obtained with parallel version of MADAM for graphs with 150 vertices.

#### Problem | MADAM - parallel | Problem | MADAM - parallel | Problem | MADAM - parallel
---|---|---|---|---|---
| Solution | Time | Nodes | Solution | Time | Nodes | Solution | Time | Nodes |
| w05_150.0 | 3008 | **252.95** | 3008 | **252.95** | 3008 | **252.95** | 3008 | **252.95** | 3008 | **252.95** |
| w05_150.1 | 3032 | **135.74** | 2900 | **70.88** | 252.95 | 3008 | **252.95** | 3008 | **252.95** | 3008 | **252.95** |
| w05_150.2 | 2900 | **70.88** | 7139 | **23827** | 2900 | **70.88** | 7139 | **23827** | 2900 | **70.88** |
| w05_150.3 | 2742 | **166.21** | 19827 | **23827** | 2900 | **70.88** | 7139 | **23827** | 2900 | **70.88** |
| w05_150.4 | 2815 | **23.15** | **(154)** | 20115 | 2900 | **70.88** | 7139 | **23827** | 2900 | **70.88** |
| w05_150.5 | 2609 | **223.49** | 23323 | **23827** | 2900 | **70.88** | 7139 | **23827** | 2900 | **70.88** |
| w05_150.6 | 2608 | **202.49** | 37435 | **23827** | 2900 | **70.88** | 7139 | **23827** | 2900 | **70.88** |
| w05_150.7 | 2608 | **202.49** | 37435 | **23827** | 2900 | **70.88** | 7139 | **23827** | 2900 | **70.88** |
| w05_150.8 | 2781 | **108.79** | 6229 | **23827** | 2900 | **70.88** | 7139 | **23827** | 2900 | **70.88** |
| w05_150.9 | 2725 | **143.82** | 20115 | **23827** | 2900 | **70.88** | 7139 | **23827** | 2900 | **70.88** |

#### Problem | MADAM - parallel | Problem | MADAM - parallel | Problem | MADAM - parallel
---|---|---|---|---|---
| Solution | Time | Nodes | Solution | Time | Nodes | Solution | Time | Nodes |
| pw01_150.0 | 4402 | **18.07** | **205 (102)** | pw05_150.0 | 17852 | **576.37** | **66941** |
| pw01_150.1 | 4376 | **25.64** | **2623** | pw05_150.1 | 17780 | **219.07** | **37601** |
| pw01_150.2 | 4272 | **18.29** | **183 (91)** | pw05_150.2 | 17774 | **122.62** | **15415** |
| pw01_150.3 | 4285 | **12.96** | **57 (29)** | pw05_150.3 | 17742 | **261.71** | **66123** |
| pw01_150.4 | 4297 | **28.76** | **2265** | pw05_150.4 | 17735 | **148.75** | **28501** |
| pw01_150.5 | 4489 | **32.84** | **25 (10)** | pw05_150.5 | 17867 | **1132.11** | **208637** |
| pw01_150.6 | 4446 | **17.36** | **485** | pw05_150.6 | 18123 | **256.60** | **46913** |
| pw01_150.7 | 4425 | **15.91** | **453 (212)** | pw05_150.7 | 17828 | **832.00** | **194037** |
| pw01_150.8 | 4242 | **21.29** | **171 (85)** | pw05_150.8 | 17758 | **59.55** | **2237** |
| pw01_150.9 | 4287 | **39.48** | **2129** | pw05_150.9 | 17735 | **76.39** | **4015** |

### 8 Conclusions and future work

In this paper we present an efficient exact solver MADAM for Max-Cut problem that applies alternating direction method of multipliers to efficiently compute quality SDP based upper bounds. This is due to the small computational complexity per iteration of the proposed method, since it essentially consists of solving one sparse system of linear equations and projection onto non-negative orthant and positive semidefinite cone. Furthermore, tight upper bounds are obtained due to inclusion of a subset of hypermetric inequalities. Numerical results show that MADAM solver outperforms current state-of-the approaches. By combining the elements of the serial algorithm and the new strategy for faster exploration of B&B tree we develop parallel solver based on MPI that utilizes load coordinator-worker paradigm. We show that MADAM solver scales
very well and that we can greatly reduce the time needed to solve the Max-Cut problem to optimality and increase the size of instances that can be solved in a routine way.

In the future we intent to increase the performance of the algorithm by using one-sided MPI communication, thus enabling to completely remove the master process from the algorithm and more efficient exchange of the messages.

Acknowledgements

This work was supported by the Slovenian Research Agency: project N1-0057. The first author also acknowledges the financial support from the Slovenian Research Agency under the program for young researchers (MR+).

References

[1] Edward Anderson, Zhaojun Bai, Christian Bischof, L Susan Blackford, James Demmel, Jack Donggarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney, et al. LAPACK Users’ guide. SIAM, 1999.

[2] Brian Borchers. Csdpr, ac library for semidefinite programming. Optimization methods and Software, 11(1-4):613–623, 1999.

[3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning, 3(1):1–122, 2011.

[4] Samuel Burer and Renato DC Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Mathematical Programming, 95(2):329–357, 2003.

[5] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. SIAM Journal on scientific computing, 16(5):1190–1208, 1995.

[6] Caihua Chen, Bingsheng He, Yinyu Ye, and Xiaoming Yuan. The direct extension of admmm for multi-block convex minimization problems is not necessarily convergent. Mathematical Programming, 155(1-2):57–79, 2016.

[7] Yanqing Chen, Timothy A Davis, William W Hager, and Sivasankaran Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. ACM Transactions on Mathematical Software (TOMS), 35(3):1–14, 2008.

[8] Leonardo Dagum and Ramesh Menon. Openmp: An industry-standard api for shared-memory programming. Computing in Science & Engineering, (1):46–55, 1998.

[9] Timothy A Davis and Yifan Hu. The university of florida sparse matrix collection. ACM Transactions on Mathematical Software (TOMS), 38(1):1–25, 2011.

[10] Caterina De Simone and Giovanni Rinaldi. A cutting plane algorithm for the max-cut problem. Optimization Methods and Software, 3(1-3):195–214, 1994.

[11] Michel Deza and Monique Laurent. Applications of cut polyhedra. I, II. J. Comput. Appl. Math., 55(2):191–216, 217–247, 1994.

[12] Iain Dunning, Swati Gupta, and John Silberholz. What works best when? a systematic evaluation of heuristics for max-cut and qubo. INFORMS Journal on Computing, 30(3):608–624, 2018.

[13] Ilse Fischer, Gerald Gruber, Franz Rendl, and Renata Sotirov. Computational experience with a bundle approach for semidefinite cutting plane relaxations of max-cut and equipartition. Mathematical Programming, 105(2-3):451–469, 2006.

[14] Message P Forum. Mpi: A message-passing interface standard. Technical report, Knoxville, TN, USA, 1994.

[15] Michael R Garey and David S Johnson. Computers and intractability, volume 29. wh freem an New York, 2002.
[16] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[17] Nicolo Gusmeroli, Timotej Hrga, Borut Lužar, Janez Povh, Melanie Siebenhofer, and Angelika Wiegele. BiqBin: a parallel branch-and-bound solver for binary quadratic problems with linear constraints. *arXiv preprint arXiv:2004.00799*, 2020.

[18] Nicolo Gusmeroli and Angelika Wiegele. Expedis: An exact penalty method over discrete sets. *arXiv preprint arXiv:1912.09739*, 2019.

[19] Christoph Helmberg and Franz Rendl. Solving quadratic (0, 1)-problems by semidefinite programs and cutting planes. *Mathematical programming*, 82(3):291–315, 1998.

[20] Christoph Helmberg, Franz Rendl, Robert J Vanderbei, and Henry Wolkowicz. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, 6(2):342–361, 1996.

[21] Nicholas J Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear algebra and its applications*, 103:103–118, 1988.

[22] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[23] Krzysztof C Kiwiel. A survey of bundle methods for nondifferentiable optimization. *Math. Appl. Jan. Ser.*, 6(6):263–282, 1989.

[24] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. Improved semidefinite bounding procedure for solving max-cut problems to optimality. *Mathematical Programming*, 143(1-2):61–86, 2014.

[25] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. BiqCrunch: A semidefinite branch-and-bound method for solving binary quadratic problems. *ACM Transactions on Mathematical Software (TOMS)*, 43(4):1–23, 2017.

[26] Jean B Lasserre. A max-cut formulation of 0/1 programs. *Operations Research Letters*, 44(2):158–164, 2016.

[27] Frauke Liers, Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. Computing exact ground states of hard ising spin glass problems by branch-and-cut. *New optimization algorithms in physics*, 50(47-68):6, 2004.

[28] László Lovász. On the shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1):1–7, 1979.

[29] Jérôme Malick, Janez Povh, Franz Rendl, and Angelika Wiegele. Regularization methods for semidefinite programming. *SIAM Journal on Optimization*, 20(1):336–356, 2009.

[30] ApS Mosek. The MOSEK optimization toolbox for MATLAB manual, 2015.

[31] Laura Palagi, Veronica Piccialli, Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Computational approaches to max-cut. In *Handbook on semidefinite, conic and polynomial optimization*, pages 821–847. Springer, 2012.

[32] Janez Povh, Franz Rendl, and Angelika Wiegele. A boundary point method to solve semidefinite programs. *Computing*, 78(3):277–286, 2006.

[33] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2):307, 2010.

[34] Giovanni Rinaldi. Rudy, 1998. *URL* http://www-user.tu-chemnitz.de/~helmberg/rudy.tar.gz.

[35] Jos F Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999.

[36] Kim-Chuan Toh, Michael J Todd, and Reha H Tütüncü. SDPT3—a MATLAB software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11(1-4):413–444, 1999.

[37] Zaiwen Wen, Donald Goldfarb, and Wotao Yin. Alternating direction augmented lagrangian methods for semidefinite programming. *Mathematical Programming Computation*, 2(3-4):203–230, 2010.
[38] Angelika Wiegele. *Nonlinear optimization techniques applied to combinatorial optimization problems*. na, 2006.

[39] Angelika Wiegele. Biq mac library – a collection of max-cut and quadratic 0-1 programming instances of medium size. *Preprint*, 2007.

[40] Henry Wolkowicz, Romesh Saigal, and Lieven Vandenberghe. *Handbook of semidefinite programming: theory, algorithms, and applications*, volume 27. Springer Science & Business Media, 2012.