

Box-Cone Representations of Bound-Factor RLT Constraints in Semidefinite Relaxations

Katherine Charbonneau¹ Hao Hu¹ Renata Sotirov²

¹School of Mathematical and Statistical Sciences, Clemson University, Clemson, South Carolina, USA

²Department of Econometrics and Operations Research, Tilburg University, Tilburg, The Netherlands

kcharbo@clemson.edu, hhu2@clemson.edu, r.sotirov@uvt.nl

October 10, 2026

Abstract

Linear inequalities can generate large constraint systems in semidefinite relaxations of binary quadratic programs, making the quadratic subproblems of splitting methods expensive to solve. We develop a box-cone representation of bound-factor reformulation-linearization technique (RLT) constraints in semidefinite relaxations of binary quadratic programs. These constraints arise by multiplying each original linear inequality by every binary variable and its complement and linearizing the resulting products. Our key observation is that all products generated by one inequality share a common slack and admit an exact representation in the standard cone over the unit box. This box-cone representation processes the inequalities through independent explicit projections, without introducing a separate nonnegative slack for each scalar RLT inequality. We illustrate the computational use of the representation within standard two-block alternating direction method of multipliers (ADMM). The resulting subproblems require linear solves with a reusable factorization and explicit cone projections, without enlarging the semidefinite matrix. Experiments on 257 binary quadratic programs compare this implementation with SDPNAL+. A second application groups Max-Cut triangle inequalities into box cones and uses the resulting bounds within an existing branch-and-bound framework. Tests on 220 graphs show competitive performance with MADAM using only triangle cuts. Together, these applications illustrate the representation's usefulness as a building block for first-order methods.

Keywords: semidefinite programming; reformulation-linearization technique; box cone; alternating direction method of multipliers; binary quadratic programming; Max-Cut.

1 Introduction

We develop a box-cone model of the bound-factor reformulation-linearization technique (RLT) constraints in semidefinite programming (SDP) relaxations of binary quadratic programs. The model represents all bound-factor products generated by one inequality through a single box cone, preserving the relaxation while exposing a compact coupling and an explicit projection. Consider the linearly constrained binary quadratic program

$$\min\{x^\top Qx + q^\top x : Ax \leq b, Gx = h, x \in \{0, 1\}^n\}, \quad (1)$$

where $Q \in \mathbb{S}^n$, $q \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $G \in \mathbb{R}^{p \times n}$, and $h \in \mathbb{R}^p$; the data are unrestricted in sign, and Q need not be positive semidefinite. Either m or p may be zero. Applications of (1) include quadratic assignment [27], graph partitioning [9], and quadratic knapsack problems [28].

SDP relaxations provide bounds for these problems and can support exact solution methods such as branch-and-bound. They represent the variables and their pairwise products in a lifted matrix of order $n + 1$, impose positive semidefiniteness and the binary identities, and may impose entrywise nonnegativity. Bound-factor RLT [29] strengthens an SDP or linear programming relaxation by multiplying each linear constraint by the binary variables and their complements and linearizing the resulting products. This can substantially strengthen the bound, but also creates a large system of linear constraints: each original inequality generates $2n$ bound-factor RLT inequalities in addition to the original inequality. Processing these constraints can become a major cost of solving the relaxation.

In the alternating direction method of multipliers (ADMM), the treatment of these inequalities determines the cost of the subproblems. Existing approaches include iterative polyhedral projection and scalar slack-variable reformulations. Retaining coupled inequalities in a single polyhedral block gives a constrained quadratic projection subproblem; for example, de Meijer et al. [9, 8] use Dykstra’s algorithm for projections involving additional cuts. Cutting-plane schemes can also limit the inequalities imposed at each stage [8], while related methods exploit disjoint constraint supports to organize independent projections [7, 30]. Slack-variable approaches can expose simple bound projections [35, 1], but retain auxiliary coordinates for the individual scalar inequalities. Such formulations can exploit additional structure: Battista and De Santis [1] avoid explicitly forming the enlarged slack-based SDP.

The box-cone model exploits the common slack shared by the bound-factor products generated by each inequality. For a row $a_r^\top x \leq b_r$, let $t = b_r - a_r^\top x$ and let u_j denote the linearized product tx_j . The product $t(1 - x_j)$ becomes $t - u_j$, so the original inequality and its bound-factor RLT constraints are exactly $t \geq 0$ and $0 \leq u_j \leq t$ for $j = 1, \dots, n$. These conditions define the standard box cone [25]. The inequalities in $Ax \leq b$, together with their bound-factor RLT constraints, admit a Cartesian-product representation with one box cone and an independent projection per row.

This representation combines the two product families in one matrix image and exposes a separable projection computed by sorting followed by coordinatewise projection onto an interval. The image has $m(n + 1)$ coordinates, compared with $m(2n + 1)$ scalar slack coordinates in a formulation that assigns one to every original and product inequality, reducing storage for the auxiliary variables and their associated multipliers.

The RLT families and the box cone are established objects [29, 25]. Matrix formulations of the bound-factor inequalities are also established [10]. The box-cone projection is available in SCS [26], as documented in [25, Section 6.2]. Our contribution is to identify and exploit their shared-slack structure in a formulation that combines the compact cone image with an explicit affine update. The representation exposes independent box-cone projections and a structured linear coupling, allowing the bound-factor inequalities to be processed together without an iterative polyhedral projection. These properties make it a useful building block for designing efficient first-order methods for SDP–RLT relaxations.

We demonstrate this construction within standard two-block ADMM [3, 34]. The splitting separates the box-cone image from the normalization and arrow equations and incorporates the usual facial reduction for equality-generated RLT. Each iteration then uses explicit cone projections and the reusable affine solve, without an iterative polyhedral projection or a separate slack coordinate for every scalar RLT inequality. The explicit inequality projections act in the cone-image space, rather than on the full RLT feasible set in the lifted matrix space. The role of formulation is analogous to that of facial reduction in the QAP method of Oliveira, Wolkowicz, and Xu [27], where it exposes inexpensive semidefinite and polyhedral updates. Related ADMM and Peaceman–Rachford applications also exploit equality-induced facial reduction [19, 7, 20, 36, 15, 23, 6]. Numerical experiments compare our ADMM implementation with SDPNAL+ [37] on the same SDP–RLT

formulation on each instance. A second application uses the shared-bound structure of Max-Cut triangle inequalities to construct box-cone bounds within the branch-and-bound framework of MADAM [18]. This tests the modeling idea with dynamically selected cuts as well as with a fixed RLT system.

Organization. Section 2 defines the relaxation precisely. Section 3 develops the box-cone representation, its projection, and its storage implications. Section 4 applies standard two-block ADMM, including the affine projection and the usual facial reduction for equality-generated RLT. Section 5 discusses stronger RLT systems. Implementation details for the fixed SDP–RLT relaxation are given in Section S1 of the supplement. Section 6 presents the SDPNAL+ comparison and the Max-Cut application within MADAM. Section 7 discusses the scope of the contribution and future work on low-rank methods. The appendix gives the box-cone representation used in the Max-Cut application.

2 Model and scope

We construct an SDP–RLT relaxation of problem (1) that combines positive semidefiniteness, entrywise nonnegativity, and RLT constraints to obtain strong lower bounds. Bound-factor RLT constraints can strengthen these SDP bounds, but their large number can make the relaxation expensive to solve. Our key result enables their efficient treatment through an exact rowwise representation with an explicit projection. We now formulate the relaxation in the notation used for this representation.

For a positive integer d , $\mathbb{S}^d$ denotes the real symmetric matrices of order d ; $\mathbb{S}_+^d$ and $\mathbb{S}_{++}^d$ denote their positive semidefinite and positive definite subsets, respectively. Introduce the Shor matrix $Y \in \mathbb{S}^{n+1}$ and objective matrix $C \in \mathbb{S}^{n+1}$ by

$$Y = \begin{bmatrix} 1 & x^\top \\ x & X \end{bmatrix}, \quad C = \begin{bmatrix} 0 & q^\top/2 \\ q/2 & Q \end{bmatrix}.$$

Here $X \in \mathbb{S}^n$ represents the pairwise products $x_i x_j$, so the lifted objective is $\langle C, Y \rangle$. We index the rows and columns of Y by $0, \dots, n$. In the exact binary lifting, $Y = [1; x][1; x]^\top$ has rank one. The SDP relaxation drops this rank-one requirement and imposes $Y \succeq 0$, entrywise nonnegativity $Y \geq 0$, the normalization $Y_{00} = 1$, and the arrow equations $Y_{jj} = Y_{0j}$ encoding the binary identities $x_j^2 = x_j$ for $j = 1, \dots, n$.

The RLT constraints are obtained by forming valid products of each linear constraint with x_j and $1 - x_j$ and then replacing every product $x_i x_j$ by X_{ij} . Let $a_r^\top$ be row r of A and $g_\ell^\top$ row ℓ of G . Introduce the two augmented row operators

$$H = [b, -A] \in \mathbb{R}^{m \times (n+1)}, \quad F = [h, -G] \in \mathbb{R}^{p \times (n+1)}.$$

Left multiplication of Y by H collects all inequality slacks and their lifted products in one matrix:

$$HY = \begin{bmatrix} b - Ax & bx^\top - AX \end{bmatrix}.$$

The zeroth column is the slack vector, and column j contains the lifted products with x_j . For row r , denote the slack and lifted products by

$$\sigma_r(Y) := (HY)_{r0} = b_r - a_r^\top x, \quad \pi_{rj}(Y) := (HY)_{rj} = b_r x_j - a_r^\top X_{\cdot j}.$$

For the inequality $a_r^\top x \leq b_r$, the valid products are $(b_r - a_r^\top x)x_j \geq 0$ and $(b_r - a_r^\top x)(1 - x_j) \geq 0$. Linearization gives $\pi_{rj}(Y) \geq 0$ and $\sigma_r(Y) - \pi_{rj}(Y) \geq 0$, respectively. Thus the original inequality and its bound-factor RLT constraints are exactly

$$\sigma_r(Y) \geq 0, \quad 0 \leq \pi_{rj}(Y) \leq \sigma_r(Y) \quad (r = 1, \dots, m, j = 1, \dots, n). \quad (2)$$

For an equality $g_\ell^\top x = h_\ell$, linearizing $(h_\ell - g_\ell^\top x)x_j = 0$ gives $h_\ell x_j - g_\ell^\top X_{\cdot j} = 0$. Together with the original equalities, these constraints are represented by

$$FY = 0. \quad (3)$$

The zeroth column gives $h - Gx = 0$, and column j gives the linearized products with x_j . The products with $1 - x_j$ are redundant: each is the difference between the zeroth-column equation and the corresponding column- j equation.

Combining these constraints gives the doubly nonnegative (DNN)–RLT relaxation:

$$\begin{aligned} \min_Y \quad & \langle C, Y \rangle \\ \text{s.t.} \quad & Y_{00} = 1, \quad Y_{jj} = Y_{0j} \quad (j = 1, \dots, n), \\ & Y \succeq 0, \quad Y \succeq 0, \\ & FY = 0, \\ & \sigma_r(Y) \geq 0, \quad 0 \leq \pi_{rj}(Y) \leq \sigma_r(Y) \quad (r = 1, \dots, m, j = 1, \dots, n). \end{aligned} \quad (4)$$

Here $Y \succeq 0$ is entrywise. The DNN–RLT relaxation (4) contains the original linear constraints, equality-generated RLT, and all bound-factor RLT constraints in (2). It does not contain off-diagonal McCormick inequalities or products between pairs of inequality slacks. Subsection 5.1 describes an optional full-McCormick extension, which is not used in the SDPNAL+ comparison. The separate Max-Cut application in Subsection 6.2 uses selected triangle inequalities in correlation-matrix coordinates. Subsection 5.2 describes the corresponding representation of products between inequality slacks and explains why it is less attractive computationally.

3 Box-cone representation and projection

3.1 The box-cone representation and box-cone projection

Define the polyhedral cone

$$\mathcal{S} := \text{cone}(\{1\} \times [0, 1]^n) = \{(t, u) \in \mathbb{R} \times \mathbb{R}^n : t \geq 0, 0 \leq u \leq t\mathbf{1}\}.$$

This is the standard box cone [25], or equivalently the homogenization of the unit cube [5]. Here t is a common nonnegative upper bound on the coordinates of u . If the model has m inequalities, define the product of their box cones by

$$\mathcal{S}_m := \underbrace{\mathcal{S} \times \dots \times \mathcal{S}}_{m \text{ factors}} \subseteq \mathbb{R}^{m \times (n+1)},$$

with one matrix row identified with each factor.

Theorem 1 (Box-cone identity). *For any lifted matrix Y , the bound-factor RLT constraints (2) are equivalent to the single matrix-cone constraint*

$$HY \in \mathcal{S}_m.$$

Proof. By definition, $(HY)_{r0} = \sigma_r(Y)$ and $(HY)_{rj} = \pi_{rj}(Y)$. Membership of row r in $\mathcal{S}$ is therefore precisely $\sigma_r(Y) \geq 0$ and $0 \leq \pi_{rj}(Y) \leq \sigma_r(Y)$ for every j . $\square$

The shared upper bound is the essential structure. Keeping only the lower-bound-factor products would give the orthant constraint $HY \geq 0$. Including the upper-bound-factor products adds $(HY)_{rj} \leq (HY)_{r0}$, so all $2n + 1$ scalar inequalities generated by one original inequality become one cone constraint with an explicit Euclidean projection.

Proposition 1 (Closed-form projection onto the box cone). *Given $c = (c_0, c_1, \dots, c_n) \in \mathbb{R}^{n+1}$, its Euclidean projection $(t^*, u^*) = \text{proj}_{\mathcal{S}}(c)$ solves*

$$\min_{\substack{t \geq 0, \\ 0 \leq u \leq t \mathbf{1}}} \frac{1}{2}(t - c_0)^2 + \frac{1}{2} \sum_{j=1}^n (u_j - c_j)^2.$$

Define

$$\phi(t) = t - c_0 - \sum_{j=1}^n \max\{c_j - t, 0\}, \quad t \geq 0.$$

The projection is obtained by first setting

$$t^* = \begin{cases} 0, & \phi(0) \geq 0, \\ \text{the unique positive root of } \phi(t) = 0, & \phi(0) < 0, \end{cases}$$

and the remaining coordinates are

$$u_j^* = \min\{\max\{c_j, 0\}, t^*\}, \quad j = 1, \dots, n.$$

Proof. For fixed t , scalar projection gives $u_j(t) = \min\{\max\{c_j, 0\}, t\}$. Because $t \geq 0$, this implies

$$(u_j(t) - c_j)^2 = \max\{-c_j, 0\}^2 + \max\{c_j - t, 0\}^2.$$

Substituting these coordinates reduces the problem to

$$\min_{t \geq 0} \psi(t), \quad \psi(t) = \frac{1}{2}(t - c_0)^2 + \frac{1}{2} \sum_{j=1}^n \max\{c_j - t, 0\}^2 + \frac{1}{2} \sum_{j=1}^n \max\{-c_j, 0\}^2.$$

The last sum is constant, and ψ is a strictly convex piecewise-quadratic function. For $t > 0$, its derivative is $\psi'(t) = \phi(t)$, while its right derivative at zero is $\phi(0)$. Moreover, ϕ is continuous, strictly increasing, and tends to infinity. Therefore the minimizer is zero when $\phi(0) \geq 0$ and otherwise equals the unique positive root of ϕ . Substituting this value into the interval projection formula gives the stated u^* . $\square$

The scalar t^* can be computed by sorting the positive c_j and testing the associated prefix averages. Because $\mathcal{S}_m$ is a Cartesian product of m copies of $\mathcal{S}$, projection onto $\mathcal{S}_m$ applies the single-row formula independently to each row. The resulting cost is $O(mn \log n)$ comparisons and $O(mn)$ arithmetic operations, and the row projections can be evaluated together.

3.2 Storage and operator structure

The cone image HY stores each original slack once and each lower-bound product once. The upper-bound products are obtained by subtracting the latter from the former. Thus one image or associated multiplier has $m(n + 1)$ entries, whereas a separate scalar coordinate for every inequality would require $m(2n + 1)$ entries. Both counts are $O(mn)$; the reduction concerns these arrays and does not imply the same ratio for total memory.

4 Application to standard two-block ADMM

We apply standard two-block ADMM [3] to an equivalent reformulation of the DNN–RLT relaxation (4) using auxiliary matrix and cone-image variables. The reformulation separates the box-cone image from an affine matrix copy containing the normalization and arrow equations. We first derive the affine update, then recall the equality-induced face projection, and finally give the two-block formulation and updates.

4.1 The affine set and its metric projection

The affine copy used below need not be constrained to be symmetric because agreement with the positive semidefinite copy will enforce symmetry. Its normalization and arrow equations define

$$\mathcal{A} := \{W \in \mathbb{R}^{(n+1) \times (n+1)} : W_{00} = 1, \quad W_{jj} = W_{0j} \ (j = 1, \dots, n)\}.$$

The remaining subproblem is quadratic minimization over this set.

Lemma 1 (Projection onto the affine set in the K -metric). *Let $K \in \mathbb{S}_{++}^{n+1}$ and $B \in \mathbb{R}^{(n+1) \times (n+1)}$. Let $e_0, \dots, e_n$ be the coordinate vectors of $\mathbb{R}^{n+1}$, denote the j th column of W by w_j , and denote the j th column of $K^{-1}B$ by z_j . Define the constraint coefficient vectors by*

$$d_0 = e_0, \quad d_j = e_j - e_0 \quad (j = 1, \dots, n),$$

and their right-hand sides by

$$\delta_0 = 1, \quad \delta_j = 0 \quad (j = 1, \dots, n).$$

Then the unique minimizer of

$$\min_{W \in \mathcal{A}} \quad \frac{1}{2} \langle W, KW \rangle - \langle B, W \rangle$$

is obtained independently by columns as

$$w_j = z_j - K^{-1}d_j \frac{d_j^\top z_j - \delta_j}{d_j^\top K^{-1}d_j}, \quad j = 0, \dots, n. \quad (5)$$

Proof. The affine set $\mathcal{A}$ imposes the single equation $d_j^\top w_j = \delta_j$ on column j . Completing the square separates the problem into

$$\min_{d_j^\top w_j = \delta_j} \quad \frac{1}{2} (w_j - z_j)^\top K (w_j - z_j).$$

Thus w_j is the projection of z_j onto this hyperplane in the norm $\|v\|_K^2 = v^\top K v$. The Karush–Kuhn–Tucker equations are $K(w_j - z_j) + \lambda d_j = 0$ and $d_j^\top w_j = \delta_j$. Solving them gives (5); its denominator is positive because $K \succ 0$. $\square$

Consequently, this projection first solves $KZ = B$, where the columns of B are the right-hand sides. Each column z_j of Z is then corrected along $K^{-1}d_j$ to satisfy $d_j^\top w_j = \delta_j$. This is the affine-block update used in the ADMM formulation below.

4.2 Standard treatment of equality-generated RLT

We handle equality-generated RLT by the standard one-step facial-reduction construction [33, 27, 20, 16]. Let $\rho = \text{rank } F$, and let $V \in \mathbb{R}^{(n+1) \times (n+1-\rho)}$ have orthonormal columns spanning $\ker F$. The positive semidefinite matrices satisfying the equality-generated RLT equations $FY = 0$ in (3) form the exposed face

$$\mathcal{F} := \{Y \succeq 0 : FY = 0\} = V\mathbb{S}_+^{n+1-\rho}V^\top.$$

Indeed, $Y \succeq 0$ and $FY = 0$ imply $\text{range } Y \subseteq \ker F$, so $Y = VRV^\top$ with $R = V^\top YV \succeq 0$; the converse is immediate. For any matrix Z , write $Z_s = (Z + Z^\top)/2$ and $Z_a = (Z - Z^\top)/2$. Symmetric and skew-symmetric matrices are orthogonal in the Frobenius inner product. Since $V^\top V = I$, for any symmetric R we have

$$\|Z - VRV^\top\|_F^2 = \|V^\top Z_s V - R\|_F^2 + \|Z_s\|_F^2 - \|V^\top Z_s V\|_F^2 + \|Z_a\|_F^2.$$

Only the first term depends on R . Minimizing it over $R \succeq 0$ gives

$$\text{proj}_{\mathcal{F}}(Z) = V \text{proj}_{\mathbb{S}_+} \left(V^\top \frac{Z + Z^\top}{2} V \right) V^\top. \quad (6)$$

This standard reduction does not assert that $\mathcal{F}$ is the minimal face after the remaining constraints are imposed. Only the positive semidefinite copy is reduced; the nonnegative and box-cone copies remain in the original coordinates. If $p = 0$, then $V = I$ and the face projection is the ordinary positive semidefinite projection.

4.3 Two-block formulation and updates

The box-cone representation and the equality-induced face allow us to separate the relaxation into two blocks with explicit updates. We keep the arrow equations in the first block, $W \in \mathcal{A}$, and introduce N for the nonnegative matrix copy, R for the reduced positive semidefinite variable, and S for the box-cone image. Their cone constraints define the product cone

$$\mathcal{K} = \{(N, R, S) : N \geq 0, R \succeq 0, S \in \mathcal{S}_m\}.$$

Taking (N, R, S) as the second block, we write relaxation (4) equivalently as

$$\begin{aligned} \min_{W, (N, R, S)} \quad & \langle C, W \rangle \\ \text{s.t.} \quad & W \in \mathcal{A}, \quad (N, R, S) \in \mathcal{K}, \\ & \begin{bmatrix} W - N \\ W - VRV^\top \\ HW - S \end{bmatrix} = 0. \end{aligned} \quad (7)$$

The stacked equation couples the two blocks linearly: $W = N$ and $W = VRV^\top$ identify the matrix copies, while $HW = S$ links W to its box-cone image. At feasibility, $W = VRV^\top$ makes W symmetric and gives $FW = 0$; hence the columnwise equations defining $\mathcal{A}$ are the usual arrow equations. Conversely, a feasible Y in (4) gives a feasible tuple by taking $W = N = Y$, $R = V^\top YV$, and $S = HY$. Thus (7) is an equivalent formulation in standard two-block ADMM form.

We suppress iteration indices and use a superscript $+$ to denote an updated value; symbols without this superscript denote the current values. Assign a common penalty $\mu > 0$ to all three

coupling equations. Let λ_N , λ_P , and $\lambda_\star$ be the Lagrange multipliers for $W - N = 0$, $W - VRV^\top = 0$, and $HW - S = 0$, respectively. We use the scaled multipliers

$$D_N = \lambda_N/\mu, \quad D_P = \lambda_P/\mu, \quad D_\star = \lambda_\star/\mu.$$

For any coupling residual E with multiplier λ and $D = \lambda/\mu$, completing the square gives

$$\langle \lambda, E \rangle + \frac{\mu}{2} \|E\|_F^2 = \frac{\mu}{2} \|E + D\|_F^2 - \frac{\mu}{2} \|D\|_F^2.$$

Applying this identity to the three coupling equations and dropping the terms independent of the primal variables yields the expression below. Keeping the constraints $W \in \mathcal{A}$ and $(N, R, S) \in \mathcal{K}$ explicit, the scaled augmented Lagrangian for the three coupling equations, up to terms independent of the primal variables, is

$$\mathcal{L}_\mu = \langle C, W \rangle + \frac{\mu}{2} (\|W - N + D_N\|_F^2 + \|W - VRV^\top + D_P\|_F^2 + \|HW - S + D_\star\|_F^2).$$

Each iteration consists of the following three steps.

1. First block update.

With the second block and scaled multipliers fixed, W^+ minimizes

$$\min_{W \in \mathcal{A}} \quad \frac{1}{2} \langle W, KW \rangle - \langle B, W \rangle, \quad (8)$$

where $\langle W, KW \rangle = \text{tr}(W^\top KW)$. The common coefficient is

$$K = 2I + H^\top H \quad (9)$$

and the matrix of right-hand sides is

$$B = N - D_N + VRV^\top - D_P + H^\top (S - D_\star) - C/\mu. \quad (10)$$

The image coupling is left multiplication by H , whose Frobenius adjoint is left multiplication by $H^\top$. Its quadratic penalty therefore contributes $H^\top H$ to K , acting independently on each column. Other scalar formulations may also apply their operators implicitly, so comparisons with explicit constraint matrices concern a specified implementation choice. The matrix in (9) is positive definite, so Lemma 1 applies to (8). A factorization of K is computed once and reused to obtain all z_j in the matrix equation $KZ = B$ at each iteration. The vectors $K^{-1}d_j$ and denominators $d_j^\top K^{-1}d_j$ are also precomputed.

2. Second block update.

With W^+ fixed, minimization of $\mathcal{L}_\mu$ over (N, R, S) separates into three projections. Writing $\text{sym}(Z) = (Z + Z^\top)/2$, the updates are

$$\begin{aligned} N^+ &= \max\{W^+ + D_N, 0\}, \\ R^+ &= \text{proj}_{\mathbb{S}_+^{n+1-\rho}}(V^\top \text{sym}(W^+ + D_P)V), \\ S^+ &= \text{proj}_{\mathcal{S}_m}(HW^+ + D_\star). \end{aligned}$$

By (6), the R -update gives $VR^+V^\top = \text{proj}_{\mathcal{F}}(W^+ + D_P)$, implementing the face projection in reduced coordinates. The box-cone projection is evaluated row by row using Proposition 1.

3. Multiplier update. Update the scaled multipliers using the coupling residuals:

$$D_N^+ = D_N + W^+ - N^+, \quad D_P^+ = D_P + W^+ - VR^+V^\top, \quad D_\star^+ = D_\star + HW^+ - S^+.$$

Computational cost. After forming and factorizing K , each iteration costs $O(n^3)$ for the affine solves, $O(mn^2)$ for multiplication by H and $H^\top$, $O(n^2)$ for the nonnegative projection, $O(n^3)$ for the positive semidefinite projection, and $O(mn \log n)$ for the box-cone projections. Thus both ADMM subproblems use only linear solves and explicit projections; no constrained QP solver is needed.

Convergence. Assuming that (7) admits a primal–dual saddle point, standard two-block ADMM with exact subproblem solutions and a fixed positive penalty converges. The over-relaxed variant with parameter $\alpha \in (0, 2)$ is covered by [11, Theorem 8].

Valid lower bounds. The ADMM implementation separately computes residual-corrected dual bounds using standard weak-duality postprocessing. The SDP–RLT comparison tables report returned primal iterate objective values for both methods, rather than these bounds or SDPNAL+’s returned dual objectives. These tabulated values should therefore not be interpreted as certified bounds.

5 Extensions and related constraint systems

The shared-bound structure also occurs in related constraint systems. Full McCormick constraints admit additional box-cone factors, whereas products between inequality slacks require a different conic image. We describe these two RLT extensions and their implications for the ADMM application.

5.1 Full McCormick constraints

The full off-diagonal McCormick system [24] can be represented by box cones generated from both bounds $0 \leq x_i \leq 1$. This representation also enforces entrywise nonnegativity, so the ADMM formulation no longer needs a separate nonnegative matrix copy. Define

$$H_{\text{box}} = \begin{bmatrix} 0 & I \\ \mathbf{1} & -I \end{bmatrix} \in \mathbb{R}^{2n \times (n+1)}.$$

The box bounds are used only as additional RLT generators in the lifted operator. Their image is

$$H_{\text{box}}Y = \begin{bmatrix} x & X \\ \mathbf{1} - x & \mathbf{1}x^\top - X \end{bmatrix}. \quad (11)$$

Proposition 2 (McCormick inequalities as box-cone constraints). *Suppose that $Y \succeq 0$ satisfies the normalization and arrow equations. Then the full off-diagonal McCormick system*

$$\max\{0, Y_{0i} + Y_{0j} - 1\} \leq Y_{ij} \leq \min\{Y_{0i}, Y_{0j}\} \quad (1 \leq i < j \leq n) \quad (12)$$

is equivalent to

$$H_{\text{box}}Y \in \mathcal{S}_{2n}. \quad (13)$$

Either condition implies $Y \geq 0$.

Proof. Write $x_i = Y_{0i}$ and $X_{ij} = Y_{ij}$. The arrow equations and positive semidefiniteness imply $0 \leq x_i \leq 1$. The box-cone constraints in (11) are therefore

$$0 \leq X_{ij} \leq x_i, \quad 0 \leq x_j - X_{ij} \leq 1 - x_i \quad (i, j = 1, \dots, n).$$

For $i \neq j$, these are precisely (12); for $i = j$, they follow from $X_{ii} = x_i$ and $0 \leq x_i \leq 1$. Nonnegativity of x and X , together with $Y_{00} = 1$, gives $Y \geq 0$. $\square$

Accordingly, remove N and the equation $W - N = 0$ from (7), and replace $HW = S$ by $H_{\text{mc}}W = S_{\text{mc}}$, where

$$H_{\text{mc}} = \begin{bmatrix} H \\ H_{\text{box}} \end{bmatrix}, \quad S_{\text{mc}} \in \mathcal{S}_{m+2n}.$$

The second block now contains only the positive semidefinite and box-cone copies. With the common penalty μ , the affine subproblem has coefficient

$$K_{\text{mc}} = I + H_{\text{mc}}^\top H_{\text{mc}} \quad (14)$$

and matrix of right-hand sides

$$B_{\text{mc}} = VRV^\top - D_P + H_{\text{mc}}^\top (S_{\text{mc}} - D_{\text{mc}}) - C/\mu, \quad (15)$$

where D_{mc} is the scaled multiplier for the augmented box-cone coupling. Lemma 1 still applies, with one factorization reused across columns and iterations. The remaining updates are the positive semidefinite and box-cone projections and their multiplier updates from Section 4, using H_{mc} in place of H .

Full McCormick constraints fit naturally into the box-cone formulation: they add quadratically many scalar inequalities without increasing the semidefinite matrix order, and the additional box-cone projections cost $O(n^2 \log n)$ per iteration, within a logarithmic factor of the orthant projection they replace. The comparison with SDPNAL+ uses the DNN-RLT relaxation (4), without the full McCormick constraints described here. Their effect on bound quality and solution time is not evaluated in this study.

5.2 Products between inequality slacks

Products between pairs of inequality slacks give another RLT family. At a binary rank-one matrix, the slack vector $s = b - Ax$ satisfies $ss^\top = HYH^\top$, so linearization gives

$$HYH^\top \geq 0 \quad (16)$$

entrywise. The diagonal inequalities are already implied by $Y \succeq 0$, leaving at most $\binom{m}{2}$ nontrivial off-diagonal inequalities. The full image has m^2 entries, and storing it together with its associated multipliers requires $\Theta(m^2)$ storage. These products couple different inequality slacks and do not have the shared-slack structure of the bound-factor products. They therefore fall outside the rowwise box-cone representation used here and are excluded from the relaxation used in the SDPNAL+ comparison.

6 Numerical experiments

We evaluate the modeling idea in two settings. The first solves the fixed SDP-RLT relaxation (4) on 257 binary quadratic programs and compares our box-cone ADMM implementation with SDPNAL+. The second uses the box-cone representation of selected triangle inequalities to compute bounds within MADAM's Max-Cut branch-and-bound framework on 220 graphs. The first study measures relaxation solution time; the second measures complete search time and incumbent quality at the time limit.

AI assistance. AI tools assisted with polishing the manuscript, preparing auxiliary scripts for generating tables, and supporting the computational comparisons. The authors reviewed the AI-assisted text, code, and resulting tables and take responsibility for the manuscript and reported results.

6.1 Fixed SDP–RLT relaxations

The implementation uses standard over-relaxation [11], scaling, and a bounded number of penalty adjustments. Elementary preprocessing removes redundant inequalities and identifies forced-zero cone-image coordinates while preserving the relaxation and the affine update. Section S1 of the supplement specifies these procedures and the stopping criteria.

6.1.1 Scope and computational protocol

Both methods solve relaxation (4). SDPNAL+ [37, 32] represents the inequality-generated RLT constraints as individual scalar inequalities.

We select inequality-rich instances with $n \leq 1000$ satisfying

$$m \geq \frac{n+1}{2}. \quad (17)$$

These rules are applied before either solver is run.

Each method has a 3600-second iterative-solve limit and a 300-second setup cap. We report setup time, solve time, and their sum; Section S1 of the supplement specifies the timing components. Both methods use one CPU thread, 64 GiB of memory, MATLAB R2025b, and the same AMD EPYC 9654 node class.

Both solvers receive the stopping tolerance 10^{-5} and terminate according to their own stopping systems. We classify a run as solved when the solver reports convergence; for SDPNAL+, this means termination code zero. Because their splittings and residual definitions differ, the reported aggregate residuals are method-specific diagnostics, not a common accuracy certificate. Setup limits, solve limits, memory limits, unsuccessful solver termination, and software errors are reported separately. Timing comparisons use total time on instances with convergence reported by both solvers. This comparison evaluates the two implementations, including their different algorithms and stopping systems. Consequently, the timing advantage over SDPNAL+ should be interpreted under this protocol and cannot be attributed solely to the box-cone representation.

6.1.2 Benchmark formulations and data

We summarize the constraint structures, benchmark sources, and dimensions. Section S3 of the supplement gives the complete binary formulations, objectives, variable indexing, and source-to-model conversion conventions. All objectives and coefficients are taken from the cited collections.

The collections cover several inequality structures relevant to the box-cone representation: capacity and conflict constraints in quadratic knapsack [38]; graph constraints in DIMACS maximum clique and fixed-color feasibility [21]; covering constraints in OR-Library set covering [2] and minimum-weight dominating set [22]; three-cycle constraints in bipartite crossing minimization and single-row facility layout [39]; triangle constraints in clique partitioning [31]; and assignment and opening constraints in single-allocation hub median [12] and uncapacitated facility location [2]. QPLIB QBL contributes general binary quadratic models with their original linear row bounds [13].

Table 1 summarizes the benchmark set. The minimum-weight dominating-set archive contains many more admissible instances; to prevent one collection from dominating the experiment, we select source seed zero for both weight types, every supplied density, and sizes 500, 800, and 1000. This gives 30 instances. For the other collections, we include all instances covered by the source-data protocols in Supplement S3 that satisfy the stated selection rules.

Table 1: Benchmark set. Here n , p , and m are the numbers of binary variables, equality constraints, and inequality constraints, respectively; the lifted matrix has order $n + 1$.

collection	instances	n	p	m
quadratic knapsack with conflict graphs	45	100–200	0	199–3,185
QPLIB QBL	17	90–595	0	480–13,090
DIMACS maximum clique	58	28–1,000	0	72–377,247
OR-Library set covering	6	192–715	0	240–4,095
fixed- k DIMACS coloring	12	44–957	11–95	80–10,560
bipartite crossing minimization	37	90–552	0	480–8,096
single-row facility layout	6	190–595	0	2,280–13,090
single-allocation p -hub median	11	25–625	6–26	20–600
minimum-weight dominating set	30	500–1,000	0	500–1,000
clique partitioning	31	435–990	0	12,180–42,570
uncapacitated facility location	4	816	50	800
total	257	25–1,000	0–95	20–377,247

6.1.3 Results and discussion

Tables 2 and 3 report all 514 method–instance outcomes for the 257-instance benchmark set, including resource limits. A solved case denotes solver-reported convergence. The supplement provides the per-instance objectives, residuals, termination statuses, iteration counts, and setup and solve times underlying the aggregate results.

Table 2: Results for the fixed 257-instance benchmark set. Counts A/S denote box-cone ADMM and SDPNAL+, respectively. The last three columns use only jointly solved instances, with total method times t_A and t_S including setup and solve. A ratio above one favors ADMM; a dash means no jointly solved instance.

collection	solved A/S	both solved	faster A/S	median t_S/t_A
bipartite crossing minimization	37/23	23	23/0	15.90
fixed- k DIMACS coloring	5/5	5	3/2	1.48
DIMACS maximum clique	32/15	15	15/0	7.43
OR-Library set covering	6/2	2	2/0	34.51
quadratic knapsack with conflict graphs	45/28	28	28/0	13.90
QPLIB QBL	17/9	9	9/0	20.81
single-allocation p -hub median	9/11	9	0/9	0.42
single-row facility layout	6/1	1	1/0	34.70
minimum-weight dominating set	20/2	2	2/0	12.47
clique partitioning	3/0	0	–	–
uncapacitated facility location	0/1	0	–	–
total	180/97	94	83/11	13.23

On the 94 jointly solved instances, ADMM has the smaller total time on 83 and SDPNAL+ on 11. The median ratio t_S/t_A is 13.23. This is a conditional comparison on the jointly solved subset, not a speedup over the whole benchmark set. Performance varies by collection: ADMM is faster on most jointly solved instances, while SDPNAL+ is faster on all jointly solved hub-median instances.

Clique partitioning and uncapacitated facility location remain difficult for both methods. Some collection-level timing ratios use only one or two jointly solved instances.

Table 3: Complete status accounting for all 257 instances per method, using solver-reported termination.

method	solved	solver timeout	out of memory
box-cone ADMM	180	77	0
SDPNAL+	97	150	10

SDPNAL+ reaches the memory limit on ten large maximum-clique models; ADMM reaches the solve limit on those same models. All other unsolved runs reach the solve limit.

6.2 Max-Cut within MADAM’s branch-and-bound framework

The second study tests whether box-cone bounds remain useful when cuts are selected during a branch-and-bound search. We use MADAM [18] as the host framework and compare its native cut-strengthened ADMM bounding step with a box-cone replacement. MADAM supplies the branching rules, primal heuristics, and outer cut-management routines in both implementations. Its initial interior-point solve of the basic SDP relaxation is also retained.

6.2.1 Bounding implementation and comparison

The baseline retains MADAM’s triangle, pentagonal, and heptagonal cuts. The box-cone variant uses only triangle cuts, grouping the selected inequalities into box-cone constraints. The resulting splitting admits independent box-cone projections and an affine update with a diagonal coefficient matrix. Appendix A gives the representation and its connection with bound-factor RLT. Cuts are generated iteratively using MADAM’s triangle separation routine. Whenever a triple is selected, the box variant completes its four triangle inequalities and groups triples sharing a base pair. Thus it does not impose all triangle inequalities at the start of the search. Scalar cut multipliers recovered from the cone multipliers are passed back to MADAM for its cut-management steps.

The configured triangle capacities are 50,000 for the baseline and 200,000 for the box variant, allowing four-inequality completion of selected triples. These are capacity limits, not the numbers of cuts retained at every node; the larger capacity can also affect cut retention. The routine uses the same upper-bound correction as MADAM [18], after recovering nonnegative scalar cut multipliers from the box-cone multipliers; Section S5 of the supplement summarizes the bound computation and gives implementation details.

The comparison retains the search framework but changes the bounding scheme, cut families, and cut management; the resulting search trees need not coincide.

6.2.2 Instances and protocol

We use all 220 Rudy instances with at most 150 vertices distributed in the official MADAM repository, at commit [a051ba3690aa1f549fae8c5ed68348ab02b0a4e0](https://github.com/HrgaT/MADAM/commit/a051ba3690aa1f549fae8c5ed68348ab02b0a4e0).¹ There are 10, 30, 90, and 90 instances with 60, 80, 100, and 150 vertices, respectively. This is a repository-based selection, rather than a reproduction of the instance selection in the published MADAM experiment. The panel includes 24 instances used during development, which we also separate in the analysis below.

Each method is run three times per instance, giving 1320 searches. Each search has a six-hour wall-time limit, including process startup and incumbent output. Both implementations use native C code, the same compiler and numerical libraries, three MPI processes (one coordinator and two

¹Source and instances: <https://github.com/HrgaT/MADAM>.

workers), one BLAS thread per process, and 64 GiB on the same AMD EPYC 9654 node class as the first study.

A completed search requires branch-and-bound termination reported by the solver and independent verification of the returned cut using the original integer edge weights. Saved incumbent partitions are verified in the same way for interrupted searches. These checks validate cut values; optimality is determined by the solver’s numerical bounding and termination rules. Timing comparisons use only instances completed in all three repeats by both methods. For each such instance, let t_M and t_A be the median whole-search times for MADAM and box-cone ADMM, respectively; we report the median of t_M/t_A across instances. All limits are retained in the completion counts. For mutually interrupted searches we compare verified incumbent lower bounds, without treating root bounds as final search-tree bounds.

The comparison evaluates the complete implementations; it does not isolate the effect of the representation from the changes in cut selection and bounding.

6.2.3 Results

Table 4 summarizes the full panel. Per-instance Max-Cut results are provided in Section S6 of the supplement. The box variant completes all three repeats on 203 instances and MADAM on 202. Both complete every instance with at most 100 vertices. Among the 199 jointly completed instances, the box variant is faster on 128 and MADAM on 71. The median ratio t_M/t_A is 1.069, corresponding to a median time reduction of approximately 6.5%. Across individual searches, the box variant completes 612 and reaches the limit on 48; MADAM completes 608 and reaches the limit on 52. There are no software or scheduler failures. Completed objective values agree across methods and repeats wherever available.

Table 4: Max-Cut comparison on all 220 Rudy instances. Completion requires all three repeats. Counts M/A denote MADAM and box-cone ADMM. Timing ratios use median whole-search times on jointly completed instances; values above one favor the box variant.

vertices	instances	completed M/A	both completed	faster M/A	median t_M/t_A
60	10	10/10	10	4/6	1.019
80	30	30/30	30	11/19	1.040
100	90	90/90	90	28/62	1.123
150	90	72/73	69	28/41	1.055
all	220	202/203	199	71/128	1.069

The advantage varies across graph families. The median timing ratio is 1.329 on the 17 jointly completed **pw05** instances, but is slightly below one on **g05** and **w09** (0.980 and 0.985). On the 196 instances not used during development, both methods complete all repeats on 180 instances; among their 177 timing pairs, the box variant is faster on 114, with median ratio 1.067. The modest timing advantage therefore also appears outside the development subset.

Both methods reach the limit in every repeat on 13 instances. Median verified incumbent values agree on ten of these, favor MADAM on two, and favor the box variant on one. On the same 199 jointly completed instances, the box variant has a larger median search-node count on 180 instances, a smaller count on one, and the same count on 18. Taking the median node count over three repeats for each method, the median paired box/MADAM node-count ratio is 2.333. The modest whole-search timing advantage therefore accompanies larger search trees on most instances. These counts help interpret the complete implementations, but do not isolate the cost of the bounding

routine from branching, heuristics, cut management, and other search overhead. These results illustrate the practical value of the box-cone representation within an established branch-and-bound solver: using only triangle cuts, the modified bounding routine achieves comparable completion rates and a modest median timing advantage over MADAM’s baseline implementation, which also uses pentagonal and heptagonal cuts.

7 Conclusions and scope

The bound-factor RLT constraints generated by each original inequality admit an exact representation in the standard box cone. Their shared slack gives a compact matrix image and independent projections, preserving the relaxation and the order of the semidefinite matrix. The formulation applies to arbitrary signed linear rows.

Within standard two-block ADMM, the representation yields explicit cone projections and an affine solve with a reusable factorization. The experiments illustrate its use for both fixed SDP–RLT relaxations and dynamically selected Max-Cut triangles within MADAM’s search framework. Together, these applications support its usefulness as a building block for first-order methods. Further directions include exploiting the representation in other first-order splittings and in low-rank methods, including RiNNAL+ [17] and the augmented mixing method [4].

Funding

The work of Hao Hu was supported in part by the Air Force Office of Scientific Research under Award No. FA9550-23-1-0508.

Competing interests

The authors have no competing interests to declare that are relevant to the content of this article.

References

- [1] F. Battista and M. De Santis, “Dealing with inequality constraints in large-scale semidefinite relaxations for graph coloring and maximum clique problems,” *4OR*, 23:65–95, 2025, <https://doi.org/10.1007/s10288-024-00569-5>.
- [2] J. E. Beasley, “OR-Library: distributing test problems by electronic mail,” *Journal of the Operational Research Society*, 41(11):1069–1072, 1990, <https://doi.org/10.1057/jors.1990.166>.
- [3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010, <https://doi.org/10.1561/22000000016>.
- [4] D. Brosch, J. Schwiddessen, and A. Wiegele, “The augmented mixing method: computing high-accuracy primal-dual solutions to large-scale SDPs via column updates,” *Optimization Methods and Software*, 41(3):754–796, 2026, <https://doi.org/10.1080/10556788.2026.2683647>.
- [5] S. Burer and H. Dong, “Separation and relaxation for cones of quadratic forms,” *Mathematical Programming*, 137:343–370, 2013, <https://doi.org/10.1007/s10107-011-0495-6>.
- [6] F. Burkowski, H. Im, and H. Wolkowicz, “A Peaceman–Rachford splitting method for the protein side-chain positioning problem,” *INFORMS Journal on Computing*, 37(4):962–976, 2025, <https://doi.org/10.1287/ijoc.2023.0094>.

- [7] F. de Meijer and R. Sotirov, “SDP-based bounds for the quadratic cycle cover problem via cutting-plane augmented Lagrangian methods and reinforcement learning,” *INFORMS Journal on Computing*, 33(4):1262–1276, 2021, <https://doi.org/10.1287/ijoc.2021.1075>.
- [8] F. de Meijer, M. Siebenhofer, R. Sotirov, and A. Wiegele, “Spanning and splitting: Integer semidefinite programming for the quadratic minimum spanning tree problem,” *European Journal of Operational Research*, 331(2):381–395, 2026, <https://doi.org/10.1016/j.ejor.2025.10.051>.
- [9] F. de Meijer, R. Sotirov, A. Wiegele, and S. Zhao, “Partitioning through projections: strong SDP bounds for large graph partition problems,” *Computers & Operations Research*, 151, article 106088, 2023, <https://doi.org/10.1016/j.cor.2022.106088>.
- [10] S. S. Dey and B. Kocuk, “Convexification of a separable function over a polyhedral ground set,” arXiv preprint arXiv:2510.16595, 2025.
- [11] J. Eckstein and D. P. Bertsekas, “On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators,” *Mathematical Programming*, 55:293–318, 1992, <https://doi.org/10.1007/BF01581204>.
- [12] A. T. Ernst and M. Krishnamoorthy, “Efficient algorithms for the uncapacitated single allocation p -hub median problem,” *Location Science*, 4(3):139–154, 1996, [https://doi.org/10.1016/S0966-8349\(96\)00011-3](https://doi.org/10.1016/S0966-8349(96)00011-3).
- [13] F. Furini, E. Traversi, P. Belotti, A. Frangioni, A. Gleixner, N. Gould, L. Liberti, A. Lodi, R. Misener, H. Mittelmann, N. V. Sahinidis, S. Vigerske, and A. Wiegele, “QPLIB: a library of quadratic programming instances,” *Mathematical Programming Computation*, 11(2):237–265, 2019, <https://doi.org/10.1007/s12532-018-0147-4>.
- [14] M. X. Goemans and D. P. Williamson, “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming,” *Journal of the ACM*, 42(6):1115–1145, 1995, <https://doi.org/10.1145/227683.227684>.
- [15] N. Graham, H. Hu, J. Im, X. Li, and H. Wolkowicz, “A restricted dual Peaceman–Rachford splitting method for a strengthened DNN relaxation for QAP,” *INFORMS Journal on Computing*, 34(4):2125–2143, 2022, <https://doi.org/10.1287/ijoc.2022.1161>.
- [16] D. Hou, T. Tang, and K.-C. Toh, “A low-rank augmented Lagrangian method for polyhedral-SDP and moment-SOS relaxations of polynomial optimization,” *Mathematical Programming*, published online, 2026, <https://doi.org/10.1007/s10107-026-02395-5>.
- [17] D. Hou, T. Tang, and K.-C. Toh, “RiNNAL+: a Riemannian ALM solver for SDP–RLT relaxations of mixed-binary quadratic programs,” *Mathematical Programming Computation*, 18:679–735, 2026, <https://doi.org/10.1007/s12532-026-00303-8>.
- [18] T. Hrga and J. Povh, “MADAM: a parallel exact solver for max-cut based on semidefinite programming and ADMM,” *Computational Optimization and Applications*, 80:347–375, 2021, <https://doi.org/10.1007/s10589-021-00310-6>.
- [19] H. Hu and R. Sotirov, “On solving the quadratic shortest path problem,” *INFORMS Journal on Computing*, 32(2):219–233, 2020, <https://doi.org/10.1287/ijoc.2018.0861>.
- [20] H. Hu, R. Sotirov, and H. Wolkowicz, “Facial reduction for symmetry reduced semidefinite and doubly nonnegative programs,” *Mathematical Programming*, 200:475–529, 2023, <https://doi.org/10.1007/s10107-022-01890-9>.
- [21] D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, volume 26, American Mathematical Society, 1996, <https://doi.org/10.1090/dimacs/026>.
- [22] R. Jovanovic, M. Tuba, and D. Simian, “Ant colony optimization applied to minimum weight dominating set problem,” in *Proceedings of the 12th WSEAS International Conference on Automatic Control, Modelling and Simulation*, pages 322–326, 2010.
- [23] X. Li, T. K. Pong, H. Sun, and H. Wolkowicz, “A strictly contractive Peaceman–Rachford splitting method for the doubly nonnegative relaxation of the minimum cut problem,” *Computational Optimization and Applications*, 78(3):853–891, 2021, <https://doi.org/10.1007/s10589-020-00261-4>.

- [24] G. P. McCormick, “Computability of global solutions to factorable nonconvex programs: Part I—convex underestimating problems,” *Mathematical Programming*, 10:147–175, 1976, <https://doi.org/10.1007/BF01580665>.
- [25] B. O’Donoghue, “Operator splitting for a homogeneous embedding of the linear complementarity problem,” *SIAM Journal on Optimization*, 31(3):1999–2023, 2021, <https://doi.org/10.1137/20M1366307>.
- [26] B. O’Donoghue, E. Chu, N. Parikh, and S. Boyd, “Conic optimization via operator splitting and homogeneous self-dual embedding,” *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016, <https://doi.org/10.1007/s10957-016-0892-3>.
- [27] D. E. Oliveira, H. Wolkowicz, and Y. Xu, “ADMM for the SDP relaxation of the QAP,” *Mathematical Programming Computation*, 10(4):631–658, 2018, <https://doi.org/10.1007/s12532-018-0148-3>.
- [28] D. Pisinger, “The quadratic knapsack problem—a survey,” *Discrete Applied Mathematics*, 155(5):623–648, 2007, <https://doi.org/10.1016/j.dam.2006.08.007>.
- [29] H. D. Sherali and W. P. Adams, “A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems,” *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990, <https://doi.org/10.1137/0403036>.
- [30] L. Sinjorgo, R. Sotirov, and J. C. Vera, “SDP bounds on the stability number via ADMM and intermediate levels of the Lasserre hierarchy,” arXiv preprint arXiv:2506.08648, 2025.
- [31] M. M. Sørensen and A. N. Letchford, “CP-Lib: benchmark instances of the clique partitioning problem,” *Mathematical Programming Computation*, 16:93–111, 2024, <https://doi.org/10.1007/s12532-023-00249-1>.
- [32] D. F. Sun, K. C. Toh, Y. C. Yuan, and X. Y. Zhao, “SDPNAL+: A Matlab software for semidefinite programming with bound constraints (version 1.0),” *Optimization Methods and Software*, 35(1):87–115, 2020, <https://doi.org/10.1080/10556788.2019.1576176>.
- [33] L. Tunçel, “On the Slater condition for the SDP relaxations of nonconvex sets,” *Operations Research Letters*, 29(4):181–186, 2001, [https://doi.org/10.1016/S0167-6377\(01\)00093-1](https://doi.org/10.1016/S0167-6377(01)00093-1).
- [34] Z. Wen, D. Goldfarb, and W. Yin, “Alternating direction augmented Lagrangian methods for semidefinite programming,” *Mathematical Programming Computation*, 2:203–230, 2010, <https://doi.org/10.1007/s12532-010-0017-1>.
- [35] A. Wiegeler and S. Zhao, “SDP-based bounds for graph partition via extended ADMM,” *Computational Optimization and Applications*, 82:251–291, 2022, <https://doi.org/10.1007/s10589-022-00355-1>.
- [36] M. Xu and H. Hu, “SDP approach to quadratic vertex-disjoint paths problem,” arXiv preprint arXiv:2604.03452, 2026, <https://doi.org/10.48550/arXiv.2604.03452>.
- [37] L. Yang, D. Sun, and K.-C. Toh, “SDPNAL+: A majorized semismooth Newton–CG augmented Lagrangian method for semidefinite programming with nonnegative constraints,” *Mathematical Programming Computation*, 7(3):331–366, 2015, <https://doi.org/10.1007/s12532-015-0082-6>.
- [38] Q. Zhou, J.-K. Hao, Z.-Z. Jiang, and Q. Wu, “An effective hybrid search method for the quadratic knapsack problem with conflict graphs,” *Journal of the Operational Research Society*, 75(5):1000–1010, 2024, <https://doi.org/10.1080/01605682.2023.2228339>.
- [39] Zuse Institute Berlin, “POLIP: library for polynomially constrained mixed-integer programming,” quadratic linear-ordering collection, https://polip.zib.de/quadratic_linear_ordering/. Accessed 7 October 2026.

A Box-cone representation of Max-Cut triangle inequalities

Let an undirected graph have N vertices and edge weights w_{ij} . Max-Cut seeks a partition maximizing the total weight of edges between its two parts. Represent a cut by signs $z \in \{-1, 1\}^N$. The correlation matrix $Z = zz^\top$ gives cut value $\frac{1}{2} \sum_{i < j} w_{ij}(1 - Z_{ij})$, where $w_{ij} = 0$ for absent edges.

The basic SDP relaxation imposes $Z \succeq 0$ and $\text{diag}(Z) = \mathbf{1}$ [14]. For distinct vertices i, j, k , its four triangle inequalities are

$$\begin{aligned} Z_{ij} + Z_{ik} + Z_{jk} &\geq -1, & Z_{ij} - Z_{ik} - Z_{jk} &\geq -1, \\ -Z_{ij} + Z_{ik} - Z_{jk} &\geq -1, & -Z_{ij} - Z_{ik} + Z_{jk} &\geq -1. \end{aligned} \tag{18}$$

These are equivalent to

$$|Z_{ik} + Z_{jk}| \leq 1 + Z_{ij}, \quad |Z_{ik} - Z_{jk}| \leq 1 - Z_{ij}.$$

Thus, if $\mathcal{T}_{ij}$ is a set of selected third vertices for the base pair (i, j) , all corresponding triangles can be represented by

$$(1 + Z_{ij}, (Z_{ik} + Z_{jk})_{k \in \mathcal{T}_{ij}}) \in \mathcal{C}_{|\mathcal{T}_{ij}|}, \quad (1 - Z_{ij}, (Z_{ik} - Z_{jk})_{k \in \mathcal{T}_{ij}}) \in \mathcal{C}_{|\mathcal{T}_{ij}|}, \tag{19}$$

where $\mathcal{C}_d = \{(t, v) \in \mathbb{R} \times \mathbb{R}^d : t \geq 0, -t\mathbf{1} \leq v \leq t\mathbf{1}\}$ is the standard box cone over $[-1, 1]^d$. Here, $d = |\mathcal{T}_{ij}|$. Each factor shares one bound across its selected third vertices. For input $c = (c_0, c_1, \dots, c_d)$, compute t^* by the scalar-root calculation and zero-boundary test of Proposition 1, with $|c_j|$ in place of c_j for $j = 1, \dots, d$. Then project each coordinate c_j onto $[-t^*, t^*]$. The same minimization over the coordinates for fixed t gives this formula.

The connection with bound-factor RLT can be seen by fixing vertex i as an anchor: set $x_j = (1 - Z_{ij})/2$, $x_k = (1 - Z_{ik})/2$, and let $X_{jk} = (1 - Z_{ij} - Z_{ik} + Z_{jk})/4$ represent $x_j x_k$. Then (18) is precisely the McCormick system for (x_j, x_k, X_{jk}) . Different triples may use different anchors; the implementation retains a single correlation matrix and groups selected triples by a base pair.

Splitting the images in (19) from the positive semidefinite copy gives independent box-cone projections. In symmetric off-diagonal coordinates the affine update is especially simple: the identity $(a + b)^2 + (a - b)^2 = 2a^2 + 2b^2$ cancels the cross terms from the two images, leaving a diagonal quadratic coefficient for a fixed triangle set. Its entries can be reused until that set changes. This provides a second application of the shared-bound modeling principle, adapted to the selected cuts rather than to the complete RLT system in Section 2.

Supplementary numerical results

Box-Cone Representations of Bound-Factor RLT Constraints in Semidefinite Relaxations

Katherine Charbonneau¹ Hao Hu¹ Renata Sotirov²

¹School of Mathematical and Statistical Sciences, Clemson University, Clemson, South Carolina, USA

²Department of Econometrics and Operations Research, Tilburg University, Tilburg, The Netherlands

`kcharbo@clemson.edu`, `hhu2@clemson.edu`, `r.sotirov@uvt.nl`

October 7, 2026

This supplement accompanies the article named above and reports all 257 benchmark instances, with one outcome for the box-cone alternating direction method of multipliers (ADMM) and one for SDPNAL+ on each instance. The tables include all outcomes, including resource limits and unsuccessful termination. The main article gives the formulations, data sources, and the interpretation of the overall numerical results. Sections S1–S4 concern the SDP–RLT comparison. Section S5 describes the Max-Cut upper-bound computation, and Section S6 reports the 220-instance comparison with MADAM.

S1 Protocol and interpretation

Both methods solve the same doubly nonnegative Shor relaxation of a binary quadratic program, with equality-generated reformulation-linearization technique (RLT) constraints and the products of every original inequality slack with each binary variable and its complement. Off-diagonal McCormick inequalities and products between inequality slacks are excluded. For each model, n denotes the number of binary variables, p the number of equality rows, and m the number of inequality rows; the lifted matrix has order $n + 1$. The fixed selection requires $n \leq 1000$ and $m \geq (n + 1)/2$. The main article describes the selection and benchmark sources.

All executions in the SDP–RLT comparison used MATLAB R2025b Update 3 on AMD EPYC 9654 nodes of Clemson University’s Palmetto cluster, with one CPU thread and 64 GiB of memory per task. Compiled C++ MEX routines perform box-cone preprocessing and projection and construct the explicit SDPNAL+ operator. Equality and inequality rows are normalized to unit Euclidean norm in augmented form before either method is called.

Each method receives a nominal setup allowance of 300 seconds and, if setup finishes, a solve allowance of 3600 seconds. Total method time is setup plus solve. Setup includes structural preprocessing, facial-basis construction, factorization, and initialization for ADMM; for SDPNAL+ it includes explicit RLT construction, conversion to solver data, and initialization. MATLAB startup, warm-up, common loading and normalization, independent verification, and saving results are excluded. Iterative time checks can allow an overrun by the final iteration; reported times retain the measured values. Slurm’s 75-minute wall-time limit is an outer safety guard.

Both solvers receive tolerance 10^{-5} . Convergence is determined by the solver’s native stopping criteria; for SDPNAL+, it means termination code zero. Solver-reported aggregate residuals have method-specific definitions and are not a common accuracy certificate. Iteration counts are retained for reference, not for comparing computational work across methods.

The implementation below uses the notation and two-block formulation of Section 4 of the main article: W is the affine copy, N the nonnegative copy, $VRV^\top$ the positive semidefinite copy, and S the box-cone image. The scaled multipliers are D_N , D_P , and $D_\star$; a superscript $+$ denotes an updated value.

S1.1 Over-relaxation

Over-relaxation is an optional modification intended to improve numerical progress; it is not needed to define the two-block method. The choice $\alpha = 1$ recovers the ordinary updates in the main article. We use the standard over-relaxation described by Boyd et al. (Section 3.4.3 of the ADMM survey cited in the main article; <https://doi.org/10.1561/22000000016>), with $\alpha = 1.7$. This choice can improve practical convergence, but does not guarantee fewer iterations on every instance. For $\alpha \in (0, 2)$, define the over-relaxed projection arguments

$$\begin{aligned}
U_N &= \alpha W^+ + (1 - \alpha)N + D_N, \\
U_P &= \alpha W^+ + (1 - \alpha)VRV^\top + D_P, \\
U_\star &= \alpha HW^+ + (1 - \alpha)S + D_\star.
\end{aligned}$$

The product-space update is then

$$N^+ = \max\{U_N, 0\}, \quad R^+ = \text{proj}_{\mathbb{S}_+^{n+1-\rho}}(V^\top \text{sym}(U_P)V), \quad S^+ = \text{proj}_{\mathcal{S}_m}(U_\star).$$

The scaled-multiplier updates are

$$\begin{aligned}
D_N^+ &= U_N - N^+, \\
D_P^+ &= U_P - VR^+V^\top, \\
D_\star^+ &= U_\star - S^+.
\end{aligned}$$

S1.2 Scaling and penalty adaptation

The following scaling choices are numerical preconditioners; they do not change the feasible set or the minimizers. Each nonzero row of H is divided by a positive scalar. The implementation assigns the box-cone residual the penalty $\mu\beta$, where $\beta > 0$. It uses $D_\star = \lambda_\star/(\mu\beta)$, and replaces $H^\top H$ and $H^\top(S - D_\star)$ in the affine update by $\beta H^\top H$ and $\beta H^\top(S - D_\star)$, respectively. When $C \neq 0$, the implementation also divides C by $\|C\|_F$; reported objective values are restored to the original scale.

The experiments use $\alpha = 1.7$, initial penalty $\mu_0 = 0.1$, and $\beta = \min\{2, \max\{0.5, 2(n+1)/\max\{1, m\}\}\}$. Every 25 iterations, let r_{con} be the maximum of the three relative consensus residuals defined in the stopping criteria below. If $r_{\text{con}} > 5r_{\text{stat}}$, we double μ ; if $r_{\text{stat}} > 5r_{\text{con}}$, we halve it; otherwise we leave it unchanged. At most eight penalty changes are allowed. Each change from μ_{old} to μ_{new} multiplies all scaled multipliers by $\mu_{\text{old}}/\mu_{\text{new}}$, preserving the unscaled multipliers.

Because only finitely many penalty changes are allowed, an indefinitely continued iteration eventually has a fixed positive penalty. After its last change, the convergence result stated in Section 4 of the main article applies under the same saddle-point and exact-subproblem assumptions. The fixed weight $\beta > 0$ can be absorbed by scaling the third coupling equation by $\sqrt{\beta}$, as in Boyd et al., Section 3.4.2.

We initialize $W = e_0 e_0^\top$, $N = W$, and all scaled multipliers at zero. The positive semidefinite copy is initialized by projection onto $\mathcal{F}$, and the box-cone image by the projection used after the preprocessing described below. The iteration caps are 2×10^6 for ADMM and 2×10^5 for SDPNAL+, in addition to the time limits stated in the main article.

The two solvers use different iteration definitions and perform different work per iteration, so an equal iteration cap would not impose an equal computational budget. The requested tolerance and setup/solve time allowances are shared; the caps are additional termination limits.

S1.3 Elementary preprocessing for structured rows

Before running ADMM, we use the arrow equations and entrywise nonnegativity to remove implied RLT inequalities, identify matrix entries forced to zero, and replace pairs of opposite RLT inequalities by equalities. These preprocessing steps preserve the relaxation exactly and require no sign or sparsity assumption on A and b . They are optional and preserve the two-block ADMM structure, but modify the box-cone projections by omitting redundant lower bounds and fixing selected coordinates to zero. Using the arrow equation in column j gives

$$\pi_{rj}(Y) = (b_r - a_{rj})Y_{jj} - \sum_{i \neq j} a_{ri}Y_{ij}. \quad (1)$$

The coefficients in this expression are inspected using exact sign and coefficient comparisons on the supplied data. In the experiments, these data have already been normalized before the solver is called. The three steps are as follows.

1. **Remove implied inequalities.** If $b_r - a_{rj} \geq 0$ and $-a_{ri} \geq 0$ for every $i \neq j$, then $\pi_{rj}(Y) \geq 0$ is implied by $Y \geq 0$ and may be omitted.
2. **Fix entries to zero.** If $b_r - a_{rj} \leq 0$ and $-a_{ri} \leq 0$ for every $i \neq j$, the constraint $\pi_{rj}(Y) \geq 0$ forces $\pi_{rj}(Y) = 0$ and every matrix entry with a strictly negative coefficient to zero.
3. **Replace opposite inequalities by an equality.** If $\pi_{rj}(Y) \geq 0$ and $\pi_{r'j'}(Y) \geq 0$ are two RLT constraints and $\pi_{rj}(Y) = -\pi_{r'j'}(Y)$ holds identically under the arrow equations, then both expressions must equal zero.

S1.3.1 Preprocessed row projections

For row r , let J_0 index image coordinates fixed to zero and let J_f index coordinates whose lower bounds are redundant, with $J_0 \cap J_f = \emptyset$. The remaining coordinates retain both bounds. For a projection input $c = (c_0, c_1, \dots, c_n)$ and a fixed cap $t \geq 0$, the coordinate minimizers are

$$u_j(t) = \begin{cases} 0, & j \in J_0, \\ \min\{c_j, t\}, & j \in J_f, \\ \min\{\max\{c_j, 0\}, t\}, & j \notin J_0 \cup J_f. \end{cases} \quad \phi_r(t) = t - c_0 - \sum_{j \notin J_0} \max\{c_j - t, 0\}.$$

The optimal cap is zero if $\phi_r(0) \geq 0$ and otherwise is the unique positive zero of ϕ_r . Fixed coordinates contribute only a constant to the projection objective, so they do not enter this equation. Removing a lower bound changes the coordinate minimizer but not its contribution to the cap equation. These formulas follow by the same scalar minimization as the box-cone projection in the main paper.

The implementation applies these masks to the cone-image update S . Matrix entries forced to zero by the preprocessing remain consequences of the model; they are not separately fixed in the nonnegative-copy update. The operator H , the normalization and arrow equations, and the affine coefficient $2I + \beta H^\top H$ are unchanged. Opposite product inequalities are enforced by fixing their image coordinates to zero.

S1.4 Stopping criteria

We stop when both primal feasibility and stationarity are satisfied to the prescribed tolerance. The diagnostics are evaluated after each multiplier update, using the scaled objective and row operators described in Subsection S1.2. Convert the scaled multipliers to $\lambda_N = \mu_k D_N$, $\lambda_P = \mu_k D_P$, and $\lambda_\star = \mu_k \beta D_\star$, where $\beta = 1$ in the unweighted method. Let $T_{\mathcal{A}}$ be the tangent space of $\mathcal{A}$, consisting of matrices Z with $Z_{00} = 0$ and $Z_{jj} = Z_{0j}$ for $j = 1, \dots, n$. The relative stationarity residual is

$$r_{\text{stat}} = \frac{\|\text{proj}_{T_{\mathcal{A}}}(C + \lambda_N + \lambda_P + H^\top \lambda_\star)\|_F}{\max\{1, \|C\|_F, \|\lambda_N + \lambda_P + H^\top \lambda_\star\|_F\}}.$$

Stationarity is evaluated directly to account for over-relaxation.

For consensus, the Frobenius norm of each difference $W - N$, $W - VRV^\top$, and $HW - S$ is divided by the maximum of one and the Frobenius norms of its two terms. We also check relative nonsymmetry, $\|W - W^\top\|_F / \max\{1, \|W\|_F\}$, and the largest absolute equality violation or positive inequality violation among the normalization and arrow equations, entrywise nonnegativity, $FW = 0$, and all original inequality and bound-factor RLT constraints. These model checks use the row data supplied to the solver, before any additional internal row scaling; in the experiments, the supplied augmented rows already have unit Euclidean norm. ADMM reports convergence when every one of these residuals, including r_{stat} , is at most the requested tolerance.

Let $P = VRV^\top \succeq 0$ be the semidefinite copy. Then

$$\text{dist}_F(W, \mathbb{S}_+^{n+1}) \leq \|W - P\|_F.$$

Thus the normalized consensus residual for $W - P$ controls proximity to the PSD cone at each stopping check. A separate eigenvalue check on the returned matrix is performed only after termination. The stopping test measures approximate satisfaction of the optimality conditions; it does not provide a guaranteed objective gap.

Residual-corrected dual bounds. Computing dual bounds by weak-duality postprocessing is standard. The present SDP–RLT comparison tables report returned primal iterate objective values for both methods and do not compare corrected dual bounds.

S2 Status accounting and table conventions

The following labels occur in the detailed tables:

converged The solver reports convergence.

solve limit The iterative solve exhausted its time allowance.

memory limit The scheduler terminated the task for exceeding its memory allocation.

Table S1 accounts for all 514 outcomes: 504 returned solver results and 10 scheduler-confirmed memory limits.

Table S1: Complete outcome counts; each method has 257 tasks.

Outcome	ADMM	SDPNAL+
Converged	180	97
Setup limit	0	0
Solve limit	77	150
Memory limit	0	10
Total	257	257

Tables S2–S12 give the per-instance results. In each detailed table, “eq.” and “ineq.” give p and m . Each caption states whether the original problem is a minimization, maximization, or feasibility problem. “Obj.” is the objective value at the returned iterate, in the original problem’s objective sense. These values are not certified bounds. Coloring rows have no objective; feasibility of the SDP relaxation does not establish an integral coloring. “Solver res.” denotes the native aggregate residual. Objectives and residuals are rounded to six and two significant digits, respectively; times are rounded to seconds, so zero may denote a subsecond time. Dashes denote unavailable values after a memory limit. Statuses follow native termination, not rounded residuals.

S3 Benchmark formulations and conversion conventions

This section specifies the binary programs used to construct the relaxations. All decision variables in the converted models are binary. The instance identifiers in the per-instance tables identify the retained source models; the main article provides the bibliographic references and selection rules. Each model is converted to $\min\{x^\top Qx + q^\top x + \kappa : Ax \leq b, Gx = h, x \in \{0,1\}^n\}$, where Q is symmetric and κ is the source objective constant. A coefficient a on a single off-diagonal monomial $x_i x_j$ is stored as $Q_{ij} = Q_{ji} = a/2$; a diagonal monomial ax_i^2 is stored as $Q_{ii} = a$. Repeated monomials are accumulated. Maximization objectives, including their constants, are negated internally and restored for reporting. The constant does not enter the ADMM updates.

The relaxation and row normalization are specified in Section S1.

S3.1 Quadratic knapsack with conflicts

The quadratic knapsack problem with conflict graphs (QKPCG) files supply item profits a_i , pair profits a_{ij} for $i < j$, weights w_i , capacity B , and an unordered conflict edge set E_c . The formulation is

$$\max \sum_{i=1}^n a_i x_i + \sum_{1 \leq i < j \leq n} a_{ij} x_i x_j \quad \text{subject to} \quad \sum_{i=1}^n w_i x_i \leq B, \quad x_i + x_j \leq 1 \quad (ij \in E_c).$$

Each pair profit is counted once. There are $1 + |E_c|$ inequality constraints and no equalities. We use the 45 medium-sized instances from groups 1–9 of the published collection.

S3.2 QPLIB QBL

For each QPLIB file of type QBL (quadratic objective, binary variables, and linear constraints), let a_{ij} denote its listed quadratic entries with $i \geq j$, c_i its linear entries, c_0 its constant, and Lx its linear row expressions. The source objective used in the conversion is

$$\frac{1}{2} \sum_i a_{ii} x_i^2 + \frac{1}{2} \sum_{i > j} a_{ij} x_i x_j + \sum_i c_i x_i + c_0, \quad \ell_r \leq (Lx)_r \leq u_r.$$

Thus a listed diagonal entry contributes $a_{ii}/2$ to Q_{ii} and a listed off-diagonal entry contributes $a_{ij}/4$ to each symmetric entry of Q , before any objective sign reversal. A row with finite equal bounds becomes an equality. Otherwise each finite upper bound gives $(Lx)_r \leq u_r$ and each finite lower bound gives $-(Lx)_r \leq -\ell_r$. Infinite sides are omitted. The QBL type specifies binary variables, a quadratic objective, and linear constraints; no additional application-specific constraints are introduced.

S3.3 Maximum clique and fixed-color feasibility

Let a DIMACS graph have vertices $1, \dots, v$ and unordered edge set E . The maximum-clique model is

$$\max \sum_{i=1}^v x_i \quad \text{subject to} \quad x_i + x_j \leq 1 \quad (1 \leq i < j \leq v, ij \notin E).$$

The DIMACS instances are interpreted as undirected simple graphs. Exactly one inequality is included per nonedge, so $n = v$ and $m = \binom{v}{2} - |E|$. Selection uses this nonedge count.

For fixed-color feasibility, x_{it} indicates that vertex $i \in \{1, \dots, v\}$ receives color $t \in \{1, \dots, k\}$. Here k is the color count supplied with the DIMACS instance and included in its name; the counts used below are not assumed to be optimal. The model asks whether the following constraints admit a binary solution:

$$\sum_{t=1}^k x_{it} = 1 \quad (i = 1, \dots, v), \quad x_{it} + x_{jt} \leq 1 \quad (ij \in E, t = 1, \dots, k).$$

Variables are ordered vertex first, then color. Here $n = vk$, $p = v$, and $m = k|E|$. The 12 retained models use **david**, **huck** ($k = 11$), **jean** ($k = 10$), **myciel3**–**myciel6** ($k = 4, 5, 6, 7$), and **queen5.5**–**queen9_9** ($k = 5, 7, 7, 9, 10$). No symmetry-breaking constraints are added. A source-graph coloring witness is checked for each model.

S3.4 Set covering and minimum-weight dominating set

In the set-covering problem, let $a_{ij} = 1$ when source row i lists column j , and zero otherwise, and let c_j be the supplied column cost. The formulation is

$$\min \sum_{j=1}^n c_j x_j \quad \text{subject to} \quad \sum_{j=1}^n a_{ij} x_j \geq 1 \quad (i = 1, \dots, m).$$

The OR-Library costs and row lists are used directly; conversion negates each row to obtain a less-than inequality. The retained instances are `scplr10–scplr13`, `scpcyc6`, and `scpcyc7`.

The minimum-weight dominating-set problem has the same objective and constraints as set covering, with $m = n$, c_j equal to the supplied vertex weight, and $a_{ij} = 1$ precisely when $i = j$ or vertices i and j are adjacent. The source connection matrix already includes its unit diagonal. Both supplied weight types are retained at source seed zero, every supplied density, and vertex counts 500, 800, and 1000, giving 30 models. Neither covering family has equality rows.

S3.5 Bipartite crossing minimization and single-row facility layout

For an ordering of v objects, use variables x_{ij} for $1 \leq i < j \leq v$, with $x_{ij} = 1$ indicating that i precedes j . The complete constraint system is

$$0 \leq x_{ij} + x_{jk} - x_{ik} \leq 1 \quad (1 \leq i < j < k \leq v).$$

Each triple contributes two inequality rows. Single-row facility layout uses one ordering, with $n = \binom{v}{2}$ and $m = 2\binom{v}{3}$. Bipartite crossing minimization uses separate orderings of the two layers, of sizes v_1, v_2 , and applies this system separately in each layer. Thus $n = \binom{v_1}{2} + \binom{v_2}{2}$ and $m = 2\binom{v_1}{3} + 2\binom{v_2}{3}$. Neither model has equality rows.

The selected ordering models from POLIP, the library for polynomially constrained mixed-integer programming, minimize a free continuous variable z with one coupling row. Let $x = (x_1, \dots, x_n)$ list the binary ordering variables in the order supplied by the source file. Denote the linear, quadratic, and constant coefficients in its coupling row by d_a, d_{ab}, d_0 , respectively.

The source coupling row is

$$z + \sum_a d_a x_a + \sum_{a \leq b} d_{ab} x_a x_b + d_0 \geq 0.$$

Let $p(x)$ denote the quadratic polynomial following z in this row. The constraint is therefore $z + p(x) \geq 0$, or equivalently $z \geq -p(x)$. Here z has no separate bounds and appears in no other constraint. For any fixed feasible ordering x , minimizing z therefore sets $z = -p(x)$. We eliminate z and minimize $-p(x)$, retaining all ordering constraints. Thus, the equivalent quadratic objective in the binary ordering variables is $-\sum_a d_a x_a - \sum_{a \leq b} d_{ab} x_a x_b - d_0$. Here x_a denotes the a th component of this vector; a, b index those components rather than the objects being ordered. The conversion preserves the supplied variable order and all cycle constraints, and eliminates only z . The quadratic coefficients are symmetrized by the convention at the start of this section and the constant $-d_0$ is restored in reported objectives. Source models containing nonbinary ordering variables are excluded.

For example, POLIP's `bip.cross.min.16.16.25.1` has 240 binary ordering variables and the continuous variable z ; eliminating z yields the 240-variable model used here.

S3.6 Single-allocation hub median

Let v be the number of nodes, f_{ij} the supplied directed flows, and d_{ij} the Euclidean distance between the supplied node coordinates. Let a, b, c be the collection, transfer, and distribution cost multipliers, and let k be the prescribed number of hubs. The variable x_{ih} assigns node i to hub h , and x_{hh} opens hub h . The formulation minimizes

$$\frac{1}{1000} \left[a \sum_{i,h=1}^v \left(\sum_{j=1}^v f_{ij} \right) d_{ih} x_{ih} + c \sum_{i,h=1}^v \left(\sum_{j=1}^v f_{ji} \right) d_{hi} x_{ih} + b \sum_{i,j,h,l=1}^v f_{ij} d_{hl} x_{ih} x_{jl} \right]$$

subject to

$$\sum_{h=1}^v x_{ih} = 1 \quad (i = 1, \dots, v), \quad \sum_{h=1}^v x_{hh} = k, \quad x_{ih} \leq x_{hh} \quad (i \neq h).$$

Variables are ordered by assigned node i , then hub h . There are v^2 variables, $v + 1$ equality rows, and $v(v - 1)$ inequality rows; the tautological links with $i = h$ are omitted. The factor 1/1000 follows the source-data cost convention. The retained AP instances are 5.3, 10.2–10.5, 20.3, and 25.3; the CAB instances are 10.3.4, 15.3.4, 20.3.4, and 25.3.4. Their supplied hub counts and cost multipliers are used without generating new combinations.

S3.7 Clique partitioning

For v vertices, CP-Lib supplies a weight w_{ij} for each undirected edge $\{i, j\}$ of the complete graph. The variable x_{ij} indicates that i and j lie in the same cluster. The formulation is

$$\max \sum_{1 \leq i < j \leq v} w_{ij} x_{ij} \quad \text{subject to} \quad \begin{cases} x_{ij} + x_{ik} - x_{jk} \leq 1, \\ x_{ij} + x_{jk} - x_{ik} \leq 1, \\ x_{ik} + x_{jk} - x_{ij} \leq 1 \end{cases} \quad (1 \leq i < j < k \leq v).$$

There is no prescribed number of clusters. Pair variables and weights use the strict upper triangle in row order. The objective is linear in these variables: internally $Q = 0$ and $q_{ij} = -w_{ij}$. There are $\binom{v}{2}$ variables, $3\binom{v}{3}$ inequalities, and no equality rows.

S3.8 Uncapacitated facility location

Let f and c denote the numbers of facilities and customers, respectively. With supplied opening costs o_i and assignment costs a_{ji} , the formulation is

$$\min \sum_{i=1}^f o_i y_i + \sum_{j=1}^c \sum_{i=1}^f a_{ji} z_{ij} \quad \text{subject to} \quad \sum_{i=1}^f z_{ij} = 1 \quad (j = 1, \dots, c), \quad z_{ij} \leq y_i \quad (i = 1, \dots, f, j = 1, \dots, c).$$

The OR-Library files **cap71**–**cap74** are interpreted as uncapacitated instances: the capacity and demand fields are ignored, and the supplied assignment costs are not multiplied by demand. The variable vector lists all opening variables first, followed by assignments in order of customer index. Thus $n = f(c + 1)$, $p = c$, and $m = fc$.

S4 Per-instance results

Bipartite crossing minimization

Table S2: Results for Bipartite crossing minimization. Minimization problems.

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
bcm_10_10_10_1	90	0	480	ADMM	converged	0.400941	1.0e−5	726	0	2	2
				SDPNAL+	converged	0.401931	7.8e−6	1049	1	41	42
bcm_10_10_20_1	90	0	480	ADMM	converged	8.07616	9.9e−6	18680	0	57	57
				SDPNAL+	converged	8.07877	1.1e−4	2690	2	123	125
bcm_10_10_30_1	90	0	480	ADMM	converged	43.9015	9.9e−6	11992	0	39	39
				SDPNAL+	converged	43.9000	1.4e−5	11129	7	261	267
bcm_10_10_40_1	90	0	480	ADMM	converged	128.837	1.0e−5	13518	0	35	35
				SDPNAL+	converged	128.834	8.7e−5	7300	1	226	227
bcm_10_10_50_1	90	0	480	ADMM	converged	241.862	9.6e−6	1753	0	5	6
				SDPNAL+	converged	241.844	9.6e−6	2405	1	72	73
bcm_10_10_70_1	90	0	480	ADMM	converged	667.394	1.0e−5	1074	0	3	3
				SDPNAL+	converged	667.364	9.9e−6	2224	1	49	50
bcm_10_10_90_1	90	0	480	ADMM	converged	1371.27	1.0e−5	4848	0	13	13
				SDPNAL+	converged	1371.33	3.9e−5	18245	1	473	474
bcm_12_12_115_1	132	0	880	ADMM	converged	2137.92	9.7e−6	1305	0	10	10
				SDPNAL+	converged	2137.93	9.9e−6	3651	1	237	238
bcm_12_12_129_1	132	0	880	ADMM	converged	3007.67	9.4e−6	1994	0	13	14
				SDPNAL+	converged	3007.69	4.5e−5	4201	1	372	374
bcm_12_12_14_1	132	0	880	ADMM	converged	0.811761	9.9e−6	2013	0	13	13
				SDPNAL+	converged	0.812099	5.2e−5	1454	2	153	155
bcm_12_12_28_1	132	0	880	ADMM	converged	27.9288	1.0e−5	10203	0	65	65
				SDPNAL+	converged	27.9282	1.0e−4	4685	1	512	513
bcm_12_12_43_1	132	0	880	ADMM	converged	134.651	1.0e−5	4504	0	29	29
				SDPNAL+	solve limit	132.975	9.5e−3	57629	1	3600	3601
bcm_12_12_57_1	132	0	880	ADMM	converged	270.805	9.9e−6	10848	0	60	60
				SDPNAL+	converged	270.807	3.4e−4	5572	1	713	714
bcm_12_12_72_1	132	0	880	ADMM	converged	594.818	1.0e−5	2384	0	14	14
				SDPNAL+	converged	594.856	2.0e−5	3790	1	314	315
bcm_12_12_86_1	132	0	880	ADMM	converged	960.675	9.1e−6	2489	0	15	15
				SDPNAL+	converged	960.728	3.0e−5	4428	1	287	288
bcm_14_14_117_1	182	0	1,456	ADMM	converged	1828.99	8.7e−6	4498	0	55	55
				SDPNAL+	converged	1829.06	3.4e−4	3531	1	1016	1017
bcm_14_14_137_1	182	0	1,456	ADMM	converged	2801.43	1.0e−5	5032	0	61	61
				SDPNAL+	solve limit	2801.24	3.6e−3	15771	1	3600	3601

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
bcm_14_14_176_1	182	0	1,456	ADMM	converged	5770.16	7.5e−6	4883	0	79	79
				SDPNAL+	solve limit	5770.36	2.9e−4	15625	1	3600	3601
bcm_14_14_19_1	182	0	1,456	ADMM	converged	2.24377	9.8e−6	4475	0	65	65
				SDPNAL+	converged	2.24514	8.6e−4	2034	1	442	443
bcm_14_14_58_1	182	0	1,456	ADMM	converged	290.174	9.9e−6	1181	0	23	23
				SDPNAL+	converged	290.078	4.0e−5	1878	1	460	461
bcm_14_14_98_1	182	0	1,456	ADMM	converged	1201.08	1.0e−5	2644	0	46	46
				SDPNAL+	converged	1201.20	5.1e−4	3205	1	925	927
bcm_16_16_102_1	240	0	2,240	ADMM	converged	1118.94	7.2e−6	3864	0	100	100
				SDPNAL+	converged	1119.20	1.1e−3	2407	1	1269	1270
bcm_16_16_128_1	240	0	2,240	ADMM	converged	2114.16	9.5e−6	4145	0	106	106
				SDPNAL+	solve limit	2114.25	4.5e−4	8246	1	3600	3601
bcm_16_16_153_1	240	0	2,240	ADMM	converged	3510.34	9.6e−6	1619	0	51	51
				SDPNAL+	converged	3510.32	1.0e−4	3285	1	1600	1602
bcm_16_16_179_1	240	0	2,240	ADMM	converged	5195.04	1.0e−5	2986	0	78	78
				SDPNAL+	converged	5195.38	5.5e−4	3403	1	1763	1765
bcm_16_16_230_1	240	0	2,240	ADMM	converged	10205.0	9.8e−6	4374	0	119	119
				SDPNAL+	solve limit	10206.0	2.7e−3	3980	1	3602	3603
bcm_16_16_25_1	240	0	2,240	ADMM	converged	10.0527	1.0e−5	8656	0	222	222
				SDPNAL+	converged	10.0525	6.7e−5	4271	1	2760	2761
bcm_16_16_51_1	240	0	2,240	ADMM	converged	131.789	1.0e−5	20337	0	521	521
				SDPNAL+	solve limit	131.782	8.0e−4	6710	1	3600	3601
bcm_16_16_76_1	240	0	2,240	ADMM	converged	479.831	9.9e−6	3377	0	89	89
				SDPNAL+	converged	479.989	2.3e−4	2204	1	1413	1414
bcm_18_18_32_1	306	0	3,264	ADMM	converged	9.28632	1.0e−5	25276	0	1066	1067
				SDPNAL+	solve limit	9.24545	3.8e−4	3022	1	3601	3602
bcm_18_18_64_1	306	0	3,264	ADMM	converged	241.390	1.0e−5	18978	0	1013	1014
				SDPNAL+	solve limit	241.375	5.4e−3	3241	1	3601	3602
bcm_20_20_40_1	380	0	4,560	ADMM	converged	32.6535	1.0e−5	11200	0	1060	1060
				SDPNAL+	solve limit	32.6717	7.4e−3	1569	2	3602	3604
bcm_20_20_80_1	380	0	4,560	ADMM	converged	411.478	9.8e−6	9345	0	794	794
				SDPNAL+	solve limit	411.251	1.0e−2	1750	2	3602	3603
bcm_22_22_48_1	462	0	6,160	ADMM	converged	52.4673	1.0e−5	24981	1	3488	3489
				SDPNAL+	solve limit	53.5242	5.3e−2	1384	1	3602	3603
bcm_22_22_96_1	462	0	6,160	ADMM	converged	709.348	9.9e−6	3401	1	577	578
				SDPNAL+	solve limit	728.618	2.4e−1	770	2	3602	3603
bcm_24_24_115_1	552	0	8,096	ADMM	converged	1074.86	1.0e−5	3119	1	923	924
				SDPNAL+	solve limit	2025.75	5.1e−1	548	2	3601	3603
bcm_24_24_57_1	552	0	8,096	ADMM	converged	101.291	8.4e−6	7009	1	1571	1572
				SDPNAL+	solve limit	402.505	8.3e−1	544	2	3601	3602

Clique partitioning

Table S3: Results for Clique partitioning. Maximization problems.

instance	<i>n</i>	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
cpp_abr_cars	528	0	16,368	ADMM	solve limit	1500.91	7.4e−4	7778	2	3600	3602
				SDPNAL+	solve limit	483.131	8.2e−1	530	2	3602	3604
cpp_abr_cetacea	630	0	21,420	ADMM	solve limit	975.681	4.0e−3	5161	3	3600	3603
				SDPNAL+	solve limit	−148.015	5.5e−1	337	2	3617	3619
cpp_abr_lung_cancer	496	0	14,880	ADMM	solve limit	3468.83	1.6e−4	8770	2	3600	3602
				SDPNAL+	solve limit	899.480	3.4e−1	431	2	3602	3604
cpp_abr_micro	780	0	29,640	ADMM	solve limit	960.404	1.1e−3	3156	5	3601	3606
				SDPNAL+	solve limit	5.60376	9.3e−1	149	4	3636	3640
cpp_abr_wildcats	435	0	12,180	ADMM	solve limit	1303.85	3.7e−5	13945	1	3600	3601
				SDPNAL+	solve limit	1227.97	1.5e−1	721	2	3908	3910
cpp_abr_workers	561	0	17,952	ADMM	solve limit	965.566	1.6e−3	6840	2	3600	3602
				SDPNAL+	solve limit	223.754	3.5e−1	439	2	3603	3605
cpp_correlation_corr40_1	780	0	29,640	ADMM	solve limit	2180.24	2.4e−4	2484	6	3602	3608
				SDPNAL+	solve limit	434.769	1.0e0	130	4	3645	3649
cpp_correlation_corr40_10	780	0	29,640	ADMM	solve limit	2294.11	2.8e−4	2125	6	3601	3608
				SDPNAL+	solve limit	503.225	1.6e0	204	3	3630	3633
cpp_correlation_corr40_2	780	0	29,640	ADMM	solve limit	1842.62	2.7e−4	2683	6	3600	3606
				SDPNAL+	solve limit	366.168	9.3e−1	152	5	3635	3640
cpp_correlation_corr40_3	780	0	29,640	ADMM	solve limit	2296.65	4.2e−4	2108	6	3601	3607
				SDPNAL+	solve limit	496.393	1.4e0	184	5	3648	3653
cpp_correlation_corr40_4	780	0	29,640	ADMM	solve limit	2079.10	3.1e−4	2352	6	3601	3607
				SDPNAL+	solve limit	462.976	4.9e0	241	3	3621	3624
cpp_correlation_corr40_5	780	0	29,640	ADMM	solve limit	2237.25	3.4e−4	2420	7	3602	3608
				SDPNAL+	solve limit	491.440	1.1e0	162	4	3637	3641
cpp_correlation_corr40_6	780	0	29,640	ADMM	solve limit	2513.21	2.3e−4	2245	5	3601	3606
				SDPNAL+	solve limit	658.715	4.6e0	241	3	3632	3635
cpp_correlation_corr40_7	780	0	29,640	ADMM	solve limit	2284.90	6.3e−4	2064	6	3601	3607
				SDPNAL+	solve limit	312.984	1.1e0	101	6	3635	3641
cpp_correlation_corr40_8	780	0	29,640	ADMM	solve limit	2176.68	2.1e−4	2099	6	3601	3607
				SDPNAL+	solve limit	448.315	1.6e0	219	3	3637	3640
cpp_correlation_corr40_9	780	0	29,640	ADMM	solve limit	2130.55	5.2e−4	2151	6	3600	3606
				SDPNAL+	solve limit	360.934	1.2e0	168	4	3633	3637
cpp_mcf_gro_80	903	0	37,023	ADMM	solve limit	52.8472	1.3e−2	2142	10	3601	3611
				SDPNAL+	solve limit	−42.7701	1.3e0	188	9	3636	3645
cpp_mcf_ira_95	465	0	13,485	ADMM	converged	38.0000	1.0e−5	3983	1	1692	1693
				SDPNAL+	solve limit	2.09966	5.5e−1	557	2	3605	3607
cpp_mcf_kin_80	703	0	25,308	ADMM	solve limit	41.0102	6.2e−5	3959	5	3601	3605
				SDPNAL+	solve limit	−17.5636	8.5e−1	275	3	3627	3630

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
cpp_mcf_mas_97	595	0	19,635	ADMM	solve limit	41.0134	1.1e−4	4062	3	3601	3604
				SDPNAL+	solve limit	−8.27485	4.3e−1	388	2	3615	3617
cpp_mcf_mcc_72	780	0	29,640	ADMM	solve limit	43.0571	3.1e−3	2069	9	3600	3609
				SDPNAL+	solve limit	−42.1882	5.0e−1	129	4	3656	3660
cpp_mcf_sei_88	528	0	16,368	ADMM	converged	53.9956	9.8e−6	6901	2	2975	2977
				SDPNAL+	solve limit	−10.6764	2.7e−1	485	2	3604	3606
cpp_mcf_sul_91	465	0	13,485	ADMM	converged	45.9968	9.8e−6	6447	1	1999	2000
				SDPNAL+	solve limit	8.13932	6.7e−1	512	1	3608	3609
cpp_random_cpn35_1	595	0	19,635	ADMM	solve limit	7931.02	1.5e−4	7061	3	3600	3603
				SDPNAL+	solve limit	−102507	6.0e−1	303	2	3623	3625
cpp_random_cpn35_2	595	0	19,635	ADMM	solve limit	7257.19	6.5e−5	7094	3	3600	3603
				SDPNAL+	solve limit	−120539	7.2e−1	286	2	3613	3616
cpp_random_cpn35_3	595	0	19,635	ADMM	solve limit	7500.01	1.3e−4	7109	3	3600	3603
				SDPNAL+	solve limit	−91337.8	4.6e−1	415	2	3605	3607
cpp_random_cpn35_4	595	0	19,635	ADMM	solve limit	7629.81	7.4e−5	7156	3	3600	3603
				SDPNAL+	solve limit	−82842.4	4.6e−1	389	2	3613	3615
cpp_random_cpn45_1	990	0	42,570	ADMM	solve limit	12675.8	4.2e−3	1907	12	3601	3614
				SDPNAL+	solve limit	−259326	2.9e0	64	9	3689	3698
cpp_random_cpn45_2	990	0	42,570	ADMM	solve limit	13624.1	7.8e−3	1709	13	3601	3614
				SDPNAL+	solve limit	−427796	2.0e0	130	4	3634	3639
cpp_random_cpn45_3	990	0	42,570	ADMM	solve limit	13359.3	8.8e−3	1709	15	3601	3616
				SDPNAL+	solve limit	−194402	1.1e0	46	10	3687	3697
cpp_random_cpn45_4	990	0	42,570	ADMM	solve limit	11722.6	1.5e−2	1712	13	3601	3614
				SDPNAL+	solve limit	−567885	1.9e0	154	4	3627	3631

DIMACS coloring

Table S4: Results for DIMACS coloring. Fixed-color feasibility problems; objectives are not applicable.

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
coloring_david_k11	957	87	4,466	ADMM	solve limit	−	2.7e−1	6391	1	3600	3602
				SDPNAL+	solve limit	−	6.4e−1	420	34	3606	3640
coloring_huck_k11	814	74	3,311	ADMM	solve limit	−	2.4e−1	13447	1	3600	3601
				SDPNAL+	solve limit	−	1.3e0	557	32	3611	3643
coloring_jean_k10	800	80	2,540	ADMM	solve limit	−	2.0e−1	14146	1	3600	3601
				SDPNAL+	solve limit	−	7.8e−2	564	27	3617	3644
coloring_myciel3_k4	44	11	80	ADMM	converged	−	9.8e−6	3483	0	3	3
				SDPNAL+	converged	−	9.6e−6	564	1	3	4

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
coloring_myciel4_k5	115	23	355	ADMM	converged	–	9.8e–6	10724	0	28	28
				SDPNAL+	converged	–	7.1e–6	723	1	42	43
coloring_myciel5_k6	282	47	1,416	ADMM	converged	–	9.9e–6	47208	0	1082	1082
				SDPNAL+	converged	–	8.3e–6	577	3	646	649
coloring_myciel6_k7	665	95	5,285	ADMM	solve limit	–	1.6e–1	17276	1	3600	3601
				SDPNAL+	solve limit	–	4.2e–1	260	52	3610	3662
coloring_queen5.5_k5	125	25	800	ADMM	converged	–	9.9e–6	16572	0	98	98
				SDPNAL+	converged	–	9.3e–6	1160	2	258	259
coloring_queen6.6_k7	252	36	2,030	ADMM	converged	–	1.0e–5	48278	0	1631	1631
				SDPNAL+	converged	–	6.8e–6	724	2	895	897
coloring_queen7.7_k7	343	49	3,332	ADMM	solve limit	–	4.1e–3	42717	0	3600	3600
				SDPNAL+	solve limit	–	2.0e–3	1076	3	3601	3604
coloring_queen8.8_k9	576	64	6,552	ADMM	solve limit	–	1.1e–1	11488	1	3600	3601
				SDPNAL+	solve limit	–	7.3e–1	455	6	3601	3607
coloring_queen9.9_k10	810	81	10,560	ADMM	solve limit	–	1.7e–1	4783	2	3600	3602
				SDPNAL+	solve limit	–	1.0e0	260	18	3621	3639

DIMACS maximum clique

Table S5: Results for DIMACS maximum clique. Maximization problems.

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
clique_MANN_a27	378	0	702	ADMM	converged	131.050	9.9e–6	19179	0	708	708
				SDPNAL+	converged	131.117	2.6e–5	14741	2	3354	3356
clique_MANN_a9	45	0	72	ADMM	converged	17.0882	9.8e–6	1177	0	1	1
				SDPNAL+	converged	17.0899	9.9e–6	1388	1	7	8
clique_brock200_1	200	0	5,066	ADMM	converged	27.1987	9.8e–6	2215	0	109	109
				SDPNAL+	converged	27.1958	2.6e–5	3021	2	2568	2569
clique_brock200_2	200	0	10,024	ADMM	converged	14.1321	1.0e–5	1875	1	151	152
				SDPNAL+	solve limit	14.2501	1.8e–3	1389	2	3603	3605
clique_brock200_3	200	0	7,852	ADMM	converged	18.6726	9.9e–6	3997	0	260	260
				SDPNAL+	solve limit	18.6732	2.8e–5	1771	1	3601	3602
clique_brock200_4	200	0	6,811	ADMM	converged	21.1195	9.8e–6	4385	0	251	251
				SDPNAL+	solve limit	21.1511	6.4e–4	1808	1	3601	3603
clique_brock400_1	400	0	20,077	ADMM	converged	39.3269	9.9e–6	7095	2	3081	3083
				SDPNAL+	solve limit	40.5107	3.7e–2	456	2	3606	3607
clique_brock400_2	400	0	20,014	ADMM	converged	39.1925	1.0e–5	7092	3	3090	3092
				SDPNAL+	solve limit	39.9914	2.6e–2	471	1	3603	3605
clique_brock400_3	400	0	20,119	ADMM	converged	39.1564	9.9e–6	7121	2	2044	2046
				SDPNAL+	solve limit	36.9968	2.6e–2	281	2	3609	3611

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
clique_brock400_4	400	0	20,035	ADMM	converged	39.2280	1.0e−5	6943	2	1988	1990
				SDPNAL+	solve limit	92.3017	4.3e0	135	3	3629	3632
clique_brock800_1	800	0	112,095	ADMM	solve limit	14.2240	9.7e−2	1224	31	3602	3633
				SDPNAL+	memory limit	—	—	—	—	—	—
clique_brock800_2	800	0	111,434	ADMM	solve limit	13.2593	1.0e−1	1211	31	3601	3632
				SDPNAL+	memory limit	—	—	—	—	—	—
clique_brock800_3	800	0	112,267	ADMM	solve limit	12.0522	3.9e−1	1047	47	3603	3650
				SDPNAL+	memory limit	—	—	—	—	—	—
clique_brock800_4	800	0	111,957	ADMM	solve limit	18.6850	2.5e−1	994	51	3600	3651
				SDPNAL+	memory limit	—	—	—	—	—	—
clique_c_fat200_1	200	0	18,366	ADMM	converged	11.9992	9.9e−6	20845	1	2659	2660
				SDPNAL+	solve limit	9.33916	1.1e−2	454	2	3603	3605
clique_c_fat200_2	200	0	16,665	ADMM	solve limit	23.0968	7.3e−4	28982	1	3600	3601
				SDPNAL+	solve limit	22.8289	2.1e−3	1057	2	3602	3603
clique_c_fat200_5	200	0	11,427	ADMM	converged	57.9997	9.4e−6	4622	1	391	391
				SDPNAL+	solve limit	60.3838	2.0e−2	1832	1	3601	3602
clique_c_fat500_1	500	0	120,291	ADMM	solve limit	11.1074	9.2e−3	1450	25	3602	3628
				SDPNAL+	solve limit	106.433	7.7e0	68	4	3688	3692
clique_c_fat500_10	500	0	78,123	ADMM	solve limit	165.593	5.7e−3	1765	19	3602	3621
				SDPNAL+	solve limit	122.687	5.6e0	85	3	3666	3669
clique_c_fat500_2	500	0	115,611	ADMM	solve limit	28.1534	1.7e−2	1824	21	3600	3622
				SDPNAL+	solve limit	96.9325	5.0e0	42	5	3706	3711
clique_c_fat500_5	500	0	101,559	ADMM	solve limit	70.2503	5.7e−2	1829	21	3601	3622
				SDPNAL+	solve limit	92.0141	2.6e0	30	9	3737	3747
clique_hamming6_2	64	0	192	ADMM	converged	32.0001	9.7e−6	4115	0	5	5
				SDPNAL+	converged	32.0000	5.9e−6	736	1	4	5
clique_hamming6_4	64	0	1,312	ADMM	converged	4.00175	9.9e−6	1142	0	4	4
				SDPNAL+	converged	4.00271	6.2e−5	2631	1	150	151
clique_hamming8_2	256	0	1,024	ADMM	converged	128.000	1.0e−5	12350	0	207	207
				SDPNAL+	converged	128.000	9.3e−6	1745	1	430	431
clique_hamming8_4	256	0	11,776	ADMM	converged	15.9990	1.0e−5	5061	1	564	565
				SDPNAL+	solve limit	15.9633	1.1e−2	1411	1	3601	3602
clique_johnson16_2.4	120	0	1,680	ADMM	converged	8.00330	9.9e−6	980	0	9	9
				SDPNAL+	converged	8.01046	8.9e−6	2405	1	334	335
clique_johnson32_2.4	496	0	14,880	ADMM	converged	16.0099	1.0e−5	2783	2	896	898
				SDPNAL+	solve limit	22.7746	6.2e−2	574	1	3601	3603
clique_johnson8_2.4	28	0	168	ADMM	converged	4.00098	9.8e−6	458	0	0	0
				SDPNAL+	converged	3.99980	9.7e−6	904	1	4	5
clique_johnson8_4.4	70	0	560	ADMM	converged	13.9990	9.9e−6	978	0	2	2
				SDPNAL+	converged	14.0001	9.3e−6	1099	1	34	35
clique_keller4	171	0	5,100	ADMM	converged	13.4651	1.0e−5	3031	0	98	98
				SDPNAL+	solve limit	13.5157	1.0e−3	2690	1	3601	3602

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
clique_keller5	776	0	74,710	ADMM	solve limit	35.8835	2.1e-2	1510	27	3600	3627
				SDPNAL+	solve limit	116.101	2.0e0	29	6	3752	3757
clique_p_hat1000_1	1,000	0	377,247	ADMM	solve limit	4.27871	1.4e-1	238	246	3613	3860
				SDPNAL+	memory limit	—	—	—	—	—	—
clique_p_hat1000_2	1,000	0	254,701	ADMM	solve limit	12.6448	1.4e-1	451	87	3601	3689
				SDPNAL+	memory limit	—	—	—	—	—	—
clique_p_hat1000_3	1,000	0	127,754	ADMM	solve limit	259.015	1.0e-1	560	89	3601	3690
				SDPNAL+	memory limit	—	—	—	—	—	—
clique_p_hat300_1	300	0	33,917	ADMM	solve limit	10.0239	1.9e-5	7147	3	3600	3603
				SDPNAL+	solve limit	40.0793	2.2e0	172	2	3629	3632
clique_p_hat300_2	300	0	22,922	ADMM	solve limit	26.5839	3.7e-5	10246	2	3600	3602
				SDPNAL+	solve limit	32.6480	4.7e-1	231	2	3620	3622
clique_p_hat300_3	300	0	11,460	ADMM	converged	40.6169	1.0e-5	5919	1	906	907
				SDPNAL+	solve limit	38.8012	2.5e-2	473	2	3606	3607
clique_p_hat500_1	500	0	93,181	ADMM	solve limit	13.9497	2.6e-3	1621	27	3600	3627
				SDPNAL+	solve limit	111.880	5.9e0	59	4	3673	3676
clique_p_hat500_2	500	0	61,804	ADMM	solve limit	38.5691	8.2e-4	2873	13	3601	3614
				SDPNAL+	solve limit	121.845	6.9e0	102	3	3659	3662
clique_p_hat500_3	500	0	30,950	ADMM	solve limit	57.7672	2.5e-5	6318	4	3600	3604
				SDPNAL+	solve limit	141.026	5.0e0	94	3	3649	3652
clique_p_hat700_1	700	0	183,651	ADMM	solve limit	22.1073	1.0e-1	547	48	3604	3653
				SDPNAL+	memory limit	—	—	—	—	—	—
clique_p_hat700_2	700	0	122,922	ADMM	solve limit	53.8544	3.6e-2	820	29	3600	3629
				SDPNAL+	memory limit	—	—	—	—	—	—
clique_p_hat700_3	700	0	61,640	ADMM	solve limit	52.8020	1.6e-2	1072	19	3601	3619
				SDPNAL+	solve limit	127.542	2.9e0	40	5	3776	3781
clique_san1000	1,000	0	249,000	ADMM	solve limit	2.95964	1.3e-1	161	142	3612	3753
				SDPNAL+	memory limit	—	—	—	—	—	—
clique_san200_0.7_1	200	0	5,970	ADMM	converged	30.0005	9.9e-6	7001	0	406	407
				SDPNAL+	converged	29.9999	5.0e-6	719	2	1985	1987
clique_san200_0.7_2	200	0	5,970	ADMM	converged	17.9997	9.5e-6	14146	0	881	881
				SDPNAL+	solve limit	19.1638	2.0e-2	2253	1	3600	3601
clique_san200_0.9_1	200	0	1,990	ADMM	converged	69.9976	9.5e-6	4495	0	113	113
				SDPNAL+	converged	70.0000	8.6e-7	723	1	451	452
clique_san200_0.9_2	200	0	1,990	ADMM	converged	59.9988	9.3e-6	3719	0	79	79
				SDPNAL+	converged	60.0000	2.9e-6	896	1	536	537
clique_san200_0.9_3	200	0	1,990	ADMM	converged	43.9998	9.8e-6	7799	0	136	137
				SDPNAL+	converged	44.0003	8.9e-6	2499	1	1014	1015
clique_san400_0.5_1	400	0	39,900	ADMM	solve limit	11.6885	2.8e-4	6651	4	3600	3604
				SDPNAL+	solve limit	95.7296	4.3e0	150	3	3619	3622
clique_san400_0.7_1	400	0	23,940	ADMM	solve limit	36.9274	2.4e-3	7355	3	3600	3603
				SDPNAL+	solve limit	31.7552	8.6e-1	192	2	3616	3618

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
clique_san400_0.7_2	400	0	23,940	ADMM	solve limit	29.0426	6.2e−4	7357	3	3600	3604
				SDPNAL+	solve limit	3.16741	7.0e−1	224	3	3622	3624
clique_san400_0.7_3	400	0	23,940	ADMM	converged	22.0141	1.0e−5	5434	3	2576	2578
				SDPNAL+	solve limit	119.115	3.5e0	111	3	3637	3640
clique_san400_0.9_1	400	0	7,980	ADMM	converged	99.9992	9.6e−6	5594	1	954	955
				SDPNAL+	solve limit	100.225	4.7e−3	886	1	3612	3613
clique_sanr200_0.7	200	0	6,032	ADMM	converged	23.6341	9.8e−6	4484	0	228	228
				SDPNAL+	converged	23.6378	1.2e−4	1401	1	2091	2092
clique_sanr200_0.9	200	0	2,037	ADMM	converged	48.6297	1.0e−5	2032	0	40	40
				SDPNAL+	converged	48.6297	2.1e−5	1913	1	1089	1090
clique_sanr400_0.5	400	0	39,816	ADMM	solve limit	20.1424	1.6e−3	5149	9	3600	3610
				SDPNAL+	solve limit	56.7566	2.3e0	174	2	3635	3638
clique_sanr400_0.7	400	0	23,931	ADMM	converged	33.9683	1.0e−5	6026	2	2026	2028
				SDPNAL+	solve limit	36.0318	2.0e−1	226	2	3624	3626

Minimum-weight dominating set

Table S6: Results for Minimum-weight dominating set. Minimization problems.

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
mwds_t1_v1000e10000_0	1,000	0	1,000	ADMM	converged	1583.69	9.9e−6	3831	1	1456	1456
				SDPNAL+	solve limit	1585.65	1.3e−2	747	2	3604	3606
mwds_t1_v1000e1000_0	1,000	0	1,000	ADMM	solve limit	10480.5	1.0e−4	11981	0	3600	3600
				SDPNAL+	solve limit	12542.1	4.0e−2	1455	2	3601	3603
mwds_t1_v1000e15000_0	1,000	0	1,000	ADMM	converged	1027.45	9.6e−6	4355	1	3010	3011
				SDPNAL+	solve limit	1024.92	2.1e−2	527	2	3606	3609
mwds_t1_v1000e20000_0	1,000	0	1,000	ADMM	converged	770.205	1.0e−5	4485	1	2878	2879
				SDPNAL+	solve limit	767.671	2.3e−2	451	2	3607	3610
mwds_t1_v1000e5000_0	1,000	0	1,000	ADMM	converged	3190.41	1.0e−5	2415	1	1741	1741
				SDPNAL+	solve limit	3189.67	1.1e−2	943	2	3603	3605
mwds_t1_v500e10000_0	500	0	500	ADMM	converged	389.981	1.0e−5	4012	0	368	368
				SDPNAL+	converged	389.983	1.0e−5	1441	1	3010	3012
mwds_t1_v500e1000_0	500	0	500	ADMM	converged	3616.15	1.0e−5	5333	0	440	440
				SDPNAL+	solve limit	3616.28	2.8e−4	6488	2	3600	3602
mwds_t1_v500e2000_0	500	0	500	ADMM	converged	2004.94	9.5e−6	2417	0	185	185
				SDPNAL+	converged	2004.95	1.2e−5	4541	1	3100	3101
mwds_t1_v500e5000_0	500	0	500	ADMM	converged	812.938	1.0e−5	3047	0	240	240
				SDPNAL+	solve limit	812.948	7.1e−5	2098	1	3601	3602

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
mwds_t1_v500e500_0	500	0	500	ADMM	converged	5476.85	1.0e−5	28763	0	1491	1491
				SDPNAL+	solve limit	5219.26	2.7e−2	6837	2	3600	3603
mwds_t1_v800e0_0	800	0	800	ADMM	solve limit	8580.89	3.7e−5	21459	0	3600	3600
				SDPNAL+	solve limit	8687.26	2.8e−2	4096	1	3600	3602
mwds_t1_v800e10000_0	800	0	800	ADMM	converged	1008.00	9.8e−6	3714	0	834	834
				SDPNAL+	solve limit	1008.02	1.1e−4	1439	2	3604	3607
mwds_t1_v800e1000_0	800	0	800	ADMM	solve limit	7748.64	2.1e−5	15483	0	3600	3600
				SDPNAL+	solve limit	11865.6	2.2e−1	2324	2	3600	3602
mwds_t1_v800e2000_0	800	0	800	ADMM	converged	4731.24	1.0e−5	7460	0	1831	1831
				SDPNAL+	solve limit	5062.57	6.1e−2	2931	1	3601	3602
mwds_t1_v800e5000_0	800	0	800	ADMM	converged	2047.07	1.0e−5	2651	0	603	603
				SDPNAL+	solve limit	2047.12	2.1e−4	2127	2	3601	3602
mwds_t2_v1000e10000_0	1,000	0	1,000	ADMM	solve limit	4326.49	4.1e−5	9365	1	3600	3601
				SDPNAL+	solve limit	4461.90	4.2e−3	770	2	3602	3604
mwds_t2_v1000e1000_0	1,000	0	1,000	ADMM	solve limit	1200.55	7.3e−5	11025	0	3600	3600
				SDPNAL+	solve limit	2244.16	8.5e−2	1025	2	3602	3604
mwds_t2_v1000e15000_0	1,000	0	1,000	ADMM	solve limit	4977.91	1.2e−5	8754	1	3600	3601
				SDPNAL+	solve limit	5091.05	6.2e−3	384	2	3607	3609
mwds_t2_v1000e20000_0	1,000	0	1,000	ADMM	solve limit	5646.87	1.3e−5	8650	1	3600	3601
				SDPNAL+	solve limit	5743.56	2.5e−2	349	3	3612	3614
mwds_t2_v1000e5000_0	1,000	0	1,000	ADMM	solve limit	2983.54	4.8e−5	8901	1	3600	3601
				SDPNAL+	solve limit	3063.82	2.8e−2	751	2	3601	3603
mwds_t2_v500e10000_0	500	0	500	ADMM	converged	3284.55	1.0e−5	13282	0	980	981
				SDPNAL+	solve limit	3284.64	3.1e−4	2414	8	3601	3610
mwds_t2_v500e1000_0	500	0	500	ADMM	converged	975.001	9.5e−6	19832	0	1280	1280
				SDPNAL+	solve limit	975.458	4.4e−4	10328	1	3600	3601
mwds_t2_v500e2000_0	500	0	500	ADMM	converged	1484.02	9.9e−6	48261	0	3416	3416
				SDPNAL+	solve limit	1485.58	6.1e−4	4959	2	3600	3602
mwds_t2_v500e5000_0	500	0	500	ADMM	converged	2465.38	1.0e−5	8745	0	639	639
				SDPNAL+	solve limit	2465.17	6.0e−4	2637	2	3601	3603
mwds_t2_v500e500_0	500	0	500	ADMM	converged	682.994	9.2e−6	13430	0	684	684
				SDPNAL+	solve limit	683.393	7.7e−3	8859	1	3600	3602
mwds_t2_v800e0_0	800	0	800	ADMM	converged	1023.99	9.2e−6	14604	0	2915	2915
				SDPNAL+	solve limit	1077.42	2.4e−2	3498	1	3601	3602
mwds_t2_v800e10000_0	800	0	800	ADMM	converged	3977.23	1.0e−5	10616	0	2360	2361
				SDPNAL+	solve limit	3996.04	1.4e−3	1129	2	3602	3604
mwds_t2_v800e1000_0	800	0	800	ADMM	converged	1120.03	9.9e−6	14526	0	2561	2561
				SDPNAL+	solve limit	1300.10	5.5e−2	2807	1	3600	3602
mwds_t2_v800e2000_0	800	0	800	ADMM	solve limit	1677.46	3.8e−5	17429	0	3600	3600
				SDPNAL+	solve limit	1705.49	4.8e−3	2652	2	3600	3602
mwds_t2_v800e5000_0	800	0	800	ADMM	solve limit	2803.35	1.7e−5	15786	0	3600	3600
				SDPNAL+	solve limit	2804.95	1.2e−3	1715	2	3605	3606

OR-Library set covering

Table S7: Results for OR-Library set covering. Minimization problems.

instance	<i>n</i>	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
set_covering_scpc1r10	210	0	511	ADMM	converged	21.5387	9.8e−6	6629	0	93	94
				SDPNAL+	converged	21.5385	1.0e−5	15985	1	3513	3514
set_covering_scpc1r11	330	0	1,023	ADMM	converged	16.8283	1.0e−5	4188	0	220	220
				SDPNAL+	solve limit	17.0518	6.2e−3	2953	2	3600	3602
set_covering_scpc1r12	495	0	2,047	ADMM	converged	16.6365	1.0e−5	2952	1	566	567
				SDPNAL+	solve limit	16.6710	4.7e−3	696	2	3602	3604
set_covering_scpc1r13	715	0	4,095	ADMM	converged	14.4014	9.8e−6	3681	5	2823	2828
				SDPNAL+	solve limit	17.0907	5.2e−1	151	12	3623	3634
set_covering_scpcyc6	192	0	240	ADMM	converged	51.2009	9.8e−6	3358	0	28	28
				SDPNAL+	converged	51.2011	4.0e−5	15435	6	885	892
set_covering_scpcyc7	448	0	672	ADMM	converged	121.078	9.8e−6	5492	0	330	330
				SDPNAL+	solve limit	121.257	3.0e−3	10368	6	3600	3607

QKPCG

Table S8: Results for QKPCG. Maximization problems.

instance	<i>n</i>	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
1qkpcg1	100	0	199	ADMM	converged	17203.7	9.7e−6	10126	0	27	27
				SDPNAL+	converged	17203.7	1.7e−5	11124	1	290	292
1qkpcg2	100	0	199	ADMM	converged	13560.0	1.0e−5	7067	0	20	20
				SDPNAL+	converged	13560.0	1.0e−5	7638	2	173	175
1qkpcg3	100	0	199	ADMM	converged	16360.0	1.0e−5	32108	0	104	104
				SDPNAL+	converged	16360.3	9.9e−6	165894	1	1957	1958
1qkpcg4	100	0	199	ADMM	converged	20121.8	1.0e−5	19087	0	64	64
				SDPNAL+	converged	20121.6	1.2e−5	16929	2	516	519
1qkpcg5	100	0	199	ADMM	converged	17224.6	1.0e−5	6726	0	19	20
				SDPNAL+	converged	17224.3	8.1e−6	10107	1	134	135
2qkpcg1	100	0	397	ADMM	converged	11315.8	1.0e−5	23033	0	67	67
				SDPNAL+	converged	11315.7	1.1e−5	18133	1	435	435
2qkpcg2	100	0	397	ADMM	converged	11023.7	1.0e−5	41523	0	131	131
				SDPNAL+	converged	11023.7	1.0e−5	26326	1	1366	1368
2qkpcg3	100	0	397	ADMM	converged	11492.8	1.0e−5	24375	0	88	88
				SDPNAL+	converged	11492.8	1.0e−5	16749	2	525	526
2qkpcg4	100	0	397	ADMM	converged	15719.2	1.0e−5	47348	0	210	210
				SDPNAL+	solve limit	15719.4	3.7e−5	70583	1	3600	3602

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
2qkpcg5	100	0	397	ADMM	converged	12646.9	1.0e−5	40639	0	179	179
				SDPNAL+	converged	12646.8	1.4e−5	31436	2	2299	2301
3qkpcg1	100	0	793	ADMM	converged	8075.81	1.0e−5	4899	0	22	22
				SDPNAL+	converged	8075.93	1.9e−5	3780	2	425	427
3qkpcg2	100	0	793	ADMM	converged	7916.73	1.0e−5	4024	0	17	17
				SDPNAL+	converged	7916.90	8.6e−6	3526	1	320	321
3qkpcg3	100	0	793	ADMM	converged	8231.14	1.0e−5	2413	0	11	11
				SDPNAL+	converged	8231.29	9.7e−6	2698	1	167	168
3qkpcg4	100	0	793	ADMM	converged	8926.56	1.0e−5	2035	0	9	9
				SDPNAL+	converged	8926.66	1.4e−5	3386	2	501	502
3qkpcg5	100	0	793	ADMM	converged	8413.11	1.0e−5	3118	0	12	12
				SDPNAL+	converged	8413.25	2.0e−5	3156	2	482	484
4qkpcg1	150	0	448	ADMM	converged	20742.9	1.0e−5	27684	0	224	224
				SDPNAL+	solve limit	20742.1	9.7e−5	50174	1	3600	3601
4qkpcg2	150	0	448	ADMM	converged	21759.7	1.0e−5	5622	0	44	44
				SDPNAL+	converged	21759.5	9.9e−6	15334	2	582	585
4qkpcg3	150	0	448	ADMM	converged	22101.0	9.9e−6	79819	0	493	493
				SDPNAL+	solve limit	21859.4	5.6e−3	96967	1	3600	3601
4qkpcg4	150	0	448	ADMM	converged	31341.2	1.0e−5	13051	0	88	88
				SDPNAL+	converged	31341.3	1.8e−5	37125	1	1895	1896
4qkpcg5	150	0	448	ADMM	converged	21025.5	1.0e−5	19918	0	131	131
				SDPNAL+	converged	21025.6	1.0e−5	19031	1	1551	1552
5qkpcg1	150	0	895	ADMM	converged	15526.6	1.0e−5	32526	0	266	266
				SDPNAL+	solve limit	15526.2	1.3e−4	35367	1	3600	3601
5qkpcg2	150	0	895	ADMM	converged	15896.6	1.0e−5	28504	0	246	246
				SDPNAL+	converged	15896.5	1.1e−5	21733	1	3338	3339
5qkpcg3	150	0	895	ADMM	converged	15634.5	1.0e−5	12699	0	105	105
				SDPNAL+	converged	15634.5	1.1e−5	13132	2	1438	1439
5qkpcg4	150	0	895	ADMM	converged	19359.0	1.0e−5	14897	0	108	108
				SDPNAL+	converged	19358.9	1.1e−5	17280	1	1680	1681
5qkpcg5	150	0	895	ADMM	converged	16040.8	1.0e−5	19343	0	132	132
				SDPNAL+	converged	16040.9	2.0e−5	16470	1	1559	1560
6qkpcg1	150	0	1,789	ADMM	converged	10621.0	1.0e−5	5225	0	59	59
				SDPNAL+	converged	10621.4	9.8e−6	3534	2	2299	2301
6qkpcg2	150	0	1,789	ADMM	converged	10870.9	1.0e−5	9414	0	112	112
				SDPNAL+	converged	10871.0	1.5e−5	4296	1	2592	2593
6qkpcg3	150	0	1,789	ADMM	converged	10119.1	1.0e−5	4788	0	58	58
				SDPNAL+	converged	10119.5	9.9e−6	3046	1	2103	2105
6qkpcg4	150	0	1,789	ADMM	converged	12104.9	1.0e−5	6240	0	73	73
				SDPNAL+	converged	12105.3	6.1e−6	4237	2	2527	2529
6qkpcg5	150	0	1,789	ADMM	converged	10635.7	1.0e−5	5042	0	74	74
				SDPNAL+	converged	10635.8	9.7e−6	4719	2	2650	2651

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
7qkpcg1	200	0	797	ADMM	converged	27200.9	1.0e−5	62011	0	1059	1059
				SDPNAL+	solve limit	26849.0	1.5e−2	19218	1	3600	3601
7qkpcg2	200	0	797	ADMM	converged	30575.6	1.0e−5	18841	0	301	301
				SDPNAL+	solve limit	30576.5	4.6e−4	28033	1	3600	3601
7qkpcg3	200	0	797	ADMM	converged	26961.2	1.0e−5	27159	0	526	526
				SDPNAL+	solve limit	27109.7	1.1e−2	22085	1	3600	3601
7qkpcg4	200	0	797	ADMM	converged	34193.6	1.0e−5	19363	0	247	247
				SDPNAL+	solve limit	34337.6	7.7e−3	21151	1	3600	3601
7qkpcg5	200	0	797	ADMM	converged	26137.1	1.0e−5	26175	0	382	382
				SDPNAL+	solve limit	26179.5	1.1e−2	23026	1	3600	3601
8qkpcg1	200	0	1,593	ADMM	converged	18863.2	1.0e−5	27304	0	418	418
				SDPNAL+	solve limit	18895.1	2.0e−3	9922	1	3600	3601
8qkpcg2	200	0	1,593	ADMM	converged	20288.6	1.0e−5	16824	0	248	248
				SDPNAL+	solve limit	20293.8	3.4e−4	9790	1	3600	3601
8qkpcg3	200	0	1,593	ADMM	converged	18288.1	1.0e−5	15497	0	247	247
				SDPNAL+	converged	18288.1	1.1e−5	14235	1	3471	3472
8qkpcg4	200	0	1,593	ADMM	converged	21714.9	1.0e−5	9588	0	185	185
				SDPNAL+	solve limit	21718.6	4.6e−4	8604	1	3600	3602
8qkpcg5	200	0	1,593	ADMM	converged	19103.5	1.0e−5	30669	0	645	645
				SDPNAL+	solve limit	19105.5	2.8e−4	13031	2	3600	3602
9qkpcg1	200	0	3,185	ADMM	converged	12755.4	1.0e−5	8051	0	237	238
				SDPNAL+	solve limit	12759.1	3.7e−4	3556	2	3600	3602
9qkpcg2	200	0	3,185	ADMM	converged	13070.1	1.0e−5	9568	0	353	353
				SDPNAL+	solve limit	13071.9	2.1e−4	3628	2	3600	3603
9qkpcg3	200	0	3,185	ADMM	converged	13032.0	1.0e−5	11269	0	400	401
				SDPNAL+	solve limit	13035.1	2.5e−4	3664	2	3600	3602
9qkpcg4	200	0	3,185	ADMM	converged	14424.6	1.0e−5	9262	0	282	282
				SDPNAL+	solve limit	14425.2	5.3e−5	4582	1	3600	3602
9qkpcg5	200	0	3,185	ADMM	converged	12543.5	1.0e−5	9278	0	273	274
				SDPNAL+	converged	12543.8	8.8e−6	4358	1	3247	3249

QPLIB QBL

Table S9: Results for QPLIB QBL. Instances 5935, 5944, 5962, 5971, and 5980 are maximization problems; the remaining instances are minimization problems.

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
qplib_2315	595	0	13,090	ADMM	converged	−44088.0	1.0e−5	6221	3	2773	2776
				SDPNAL+	solve limit	−18869.1	7.5e−1	435	2	3606	3608

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
qplib_2357	240	0	2,240	ADMM	converged	-732.169	9.9e-6	3377	0	88	88
				SDPNAL+	converged	-732.014	2.1e-4	2204	1	868	869
qplib_2359	306	0	3,264	ADMM	converged	-717.610	1.0e-5	18978	0	1014	1014
				SDPNAL+	solve limit	-717.605	3.9e-4	5232	1	3600	3601
qplib_3584	528	0	10,912	ADMM	converged	-37749.5	1.0e-5	4958	1	1397	1398
				SDPNAL+	solve limit	-4223.86	5.4e-1	559	2	3628	3630
qplib_3752	462	0	6,160	ADMM	converged	-1511.65	9.9e-6	3401	1	506	506
				SDPNAL+	solve limit	-1409.00	3.9e-1	1047	2	3601	3603
qplib_3757	552	0	8,096	ADMM	converged	-639.709	8.4e-6	7009	1	2079	2081
				SDPNAL+	solve limit	-606.008	1.9e-1	809	2	3603	3604
qplib_3762	90	0	480	ADMM	converged	-345.606	1.0e-5	1074	0	4	4
				SDPNAL+	converged	-345.639	8.2e-6	2338	1	76	77
qplib_3772	380	0	4,560	ADMM	converged	-1045.52	9.8e-6	9345	0	691	692
				SDPNAL+	converged	-1045.43	1.1e-3	2423	1	3460	3461
qplib_3803	190	0	2,280	ADMM	converged	-9762.09	1.0e-5	1381	0	31	31
				SDPNAL+	converged	-9762.23	9.0e-6	2174	1	903	904
qplib_3841	300	0	4,600	ADMM	converged	-2699.16	9.7e-6	3289	0	195	196
				SDPNAL+	solve limit	-2702.82	2.4e-2	4155	1	3601	3602
qplib_3860	435	0	8,120	ADMM	converged	-29162.1	1.0e-5	4073	1	747	748
				SDPNAL+	solve limit	-28914.1	7.2e-2	1005	1	3603	3604
qplib_3883	182	0	1,456	ADMM	converged	-941.920	1.0e-5	2644	0	39	39
				SDPNAL+	converged	-941.805	8.6e-4	3107	1	594	595
qplib_5935	100	0	1,237	ADMM	converged	6113.36	1.0e-5	1255	0	8	8
				SDPNAL+	converged	6113.38	9.9e-6	2262	1	251	252
qplib_5944	100	0	2,475	ADMM	converged	2052.23	1.0e-5	3729	0	39	39
				SDPNAL+	converged	2052.31	1.0e-5	3811	1	738	739
qplib_5962	150	0	2,793	ADMM	converged	9361.30	9.8e-6	1310	0	26	26
				SDPNAL+	converged	9362.08	3.0e-6	2371	1	1113	1115
qplib_5971	150	0	5,587	ADMM	converged	3023.59	1.0e-5	1456	0	45	45
				SDPNAL+	converged	3024.05	9.8e-6	2706	1	2151	2153
qplib_5980	150	0	8,381	ADMM	converged	979.823	1.0e-5	7297	0	294	295
				SDPNAL+	solve limit	980.733	2.8e-4	4232	1	3601	3602

Single-allocation p-hub median

Table S10: Results for Single-allocation p-hub median. Minimization problems.

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
phub_ap_10_2	100	11	90	ADMM	converged	167491	1.0e−5	10375	0	20	20
				SDPNAL+	converged	167491	9.7e−6	1221	1	12	13
phub_ap_10_3	100	11	90	ADMM	converged	136007	9.9e−6	14036	0	32	32
				SDPNAL+	converged	136008	5.5e−6	1662	1	20	21
phub_ap_10_4	100	11	90	ADMM	converged	112395	9.9e−6	15266	0	26	26
				SDPNAL+	converged	112395	4.5e−6	1970	1	15	16
phub_ap_10_5	100	11	90	ADMM	converged	91106.3	1.0e−5	33747	0	57	57
				SDPNAL+	converged	91105.3	8.5e−6	2324	1	17	18
phub_ap_20_3	400	21	380	ADMM	converged	151534	1.0e−5	33889	0	1148	1148
				SDPNAL+	converged	151539	4.7e−6	2460	2	332	334
phub_ap_25_3	625	26	600	ADMM	solve limit	155354	3.3e−4	30499	0	3600	3600
				SDPNAL+	converged	155257	4.4e−6	4729	3	1632	1635
phub_ap_5_3	25	6	20	ADMM	converged	59366.2	9.8e−6	4705	0	3	3
				SDPNAL+	converged	59365.1	9.2e−6	722	1	2	3
phub_cab_10_3_4	100	11	90	ADMM	converged	567367	9.8e−6	13876	0	29	29
				SDPNAL+	converged	567360	8.7e−6	1037	1	10	11
phub_cab_15_3_4	225	16	210	ADMM	converged	2.14047e6	9.9e−6	21581	0	231	231
				SDPNAL+	converged	2.14048e6	8.7e−6	2143	1	96	97
phub_cab_20_3_4	400	21	380	ADMM	converged	4.87849e6	1.0e−5	34098	0	1739	1740
				SDPNAL+	converged	4.87853e6	3.5e−6	2825	2	464	466
phub_cab_25_3_4	625	26	600	ADMM	solve limit	7.70180e6	9.0e−5	33725	0	3600	3600
				SDPNAL+	converged	7.70045e6	9.6e−6	4726	4	3549	3554

Single-row facility layout

Table S11: Results for Single-row facility layout. Minimization problems.

instance	n	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
srflp_am_2009_33dept_set1	528	0	10,912	ADMM	converged	48340.5	1.0e−5	4958	1	1777	1778
				SDPNAL+	solve limit	72491.9	4.3e0	444	2	3603	3605
srflp_am_2009_35dept_set1	595	0	13,090	ADMM	converged	54783.0	1.0e−5	6221	2	2930	2932
				SDPNAL+	solve limit	97806.3	4.2e−1	560	2	3607	3610
srflp_anva_2008_25dept_set1	300	0	4,600	ADMM	converged	3735.84	9.7e−6	3289	0	220	221
				SDPNAL+	solve limit	3735.20	5.4e−3	4100	1	3601	3602

instance	<i>n</i>	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
srflp_anva_2008.30dept_set1	435	0	8,120	ADMM	converged	6655.18	1.0e−5	3068	1	680	681
				SDPNAL+	solve limit	7718.69	6.7e−1	1017	2	3601	3603
srflp_heku_1991.20dept_set1	190	0	2,280	ADMM	converged	13146.9	1.0e−5	1381	0	27	28
				SDPNAL+	converged	13146.8	8.8e−6	2178	2	956	958
srflp_heku_1991.30dept_set1	435	0	8,120	ADMM	converged	35963.9	1.0e−5	4073	1	670	671
				SDPNAL+	solve limit	36215.6	7.2e−2	1003	2	3602	3604

Uncapacitated facility location

Table S12: Results for Uncapacitated facility location. Minimization problems.

instance	<i>n</i>	eq.	ineq.	method	status	obj.	solver res.	iter.	setup	solve	total
uf1_cap71	816	50	800	ADMM	solve limit	935560	6.0e−4	11980	0	3600	3600
				SDPNAL+	solve limit	946180	4.4e−4	1614	18	3600	3619
uf1_cap72	816	50	800	ADMM	solve limit	973264	1.3e−3	10256	0	3600	3600
				SDPNAL+	solve limit	978402	4.1e−4	3842	10	3605	3616
uf1_cap73	816	50	800	ADMM	solve limit	1.01182e6	1.3e−4	15992	0	3600	3600
				SDPNAL+	converged	1.01085e6	1.0e−5	4397	10	3601	3611
uf1_cap74	816	50	800	ADMM	solve limit	1.03287e6	7.3e−4	10315	0	3600	3601
				SDPNAL+	solve limit	1.03656e6	2.6e−4	2447	21	3600	3621

S5 Max-Cut implementation and bounds

S5.1 Upper-bound computation

We adapt MADAM’s upper-bound postprocessing to the dual variables produced by our box-cone formulation. We recover nonnegative scalar cut multipliers and apply the eigenvalue correction of Hrga and Povh (Section 4.3; *Computational Optimization and Applications*, 80:347–375, 2021, <https://doi.org/10.1007/s10589-021-00310-6>). This gives an upper bound in exact arithmetic without requiring ADMM convergence. The floating-point implementation adds a small rounding cushion but does not provide an interval-certified bound. MADAM’s outer routine adds the objective contribution of fixed variables before applying its pruning test.

S5.2 Implementation details

The box-cone subproblem uses standard over-relaxed ADMM with parameter 1.7, explicit symmetric-box projections, and full semidefinite projections. The penalty starts at one at each search node and is carried between cut rounds. Every 25 iterations it is doubled or halved when the relative consensus and stationarity residuals differ by a factor greater than ten, within $[10^{-4}, 10^4]$, with at most eight changes for each fixed cut system. Scaled multipliers are rescaled to preserve their unscaled values. Both variants retain MADAM’s outer tolerance schedule, starting at 5×10^{-2} and decreasing by a factor of 0.6 to 10^{-3} , and its ADMM iteration-limit parameter of 2000.

Both codes are separate copies of the official MADAM source. Shared changes correct the datatype of a transmitted upper bound, make parameter parsing robust to tabs, and save native output and incumbent partitions. The baseline retains the original capacity of 50,000 triangle inequalities; the box variant permits 200,000 to accommodate four-inequality completion. This can also affect cut retention.

The shared build uses GCC 12.3.0, OpenMPI 4.1.6, and OpenBLAS 0.3.25.

S6 Max-Cut per-instance results

We report all 220 Rudy graphs with at most 150 vertices distributed in the official MADAM repository; N denotes the number of vertices. M denotes MADAM and A denotes box-cone ADMM. Each method has three repeats with a six-hour whole-search limit per repeat. Completed counts are out of three. Median time is shown only when all three repeats complete, rounded to one decimal place; “limit” means none completes and “mixed” means one or two complete. Best cut is the largest independently verified cut value across all three repeats, including interrupted searches.

For the 199 instances completed in all repeats by both methods, search-node comparisons use the median node count over three repeats for each method. The median paired box/MADAM node-count ratio is 2.333; the box variant uses more nodes on 180 instances, fewer on one, and the same number on 18. Interrupted-search node totals were not recorded and are excluded from this comparison. Whole-search times and node counts do not isolate the cost of the bounding routine. For the 13 instances on which both methods reach the limit in every repeat, median verified incumbent values agree on ten instances. The unequal pairs are 17,852 versus 17,832 on `pw05_150.0`, 715 versus 713 on `pm1d_150.6`, and 3141 versus 3142 on `g05_150.8`, with MADAM listed first. These comparisons use the median across repeats; the table reports the largest cut value across repeats.

Table S13: Per-instance Max-Cut results. M: MADAM; A: box-cone ADMM.

Instance	N	Completed M/A	Median seconds M/A	Best cut M/A
<code>g05_60.0</code>	60	3/3	1.5/1.5	536/536
<code>g05_60.1</code>	60	3/3	1.2/1.4	532/532
<code>g05_60.2</code>	60	3/3	1.9/1.8	529/529
<code>g05_60.3</code>	60	3/3	1.5/1.1	538/538
<code>g05_60.4</code>	60	3/3	2.1/2.1	527/527
<code>g05_60.5</code>	60	3/3	1.3/1.3	533/533
<code>g05_60.6</code>	60	3/3	1.8/1.6	531/531
<code>g05_60.7</code>	60	3/3	1.9/1.9	535/535
<code>g05_60.8</code>	60	3/3	1.9/1.6	530/530
<code>g05_60.9</code>	60	3/3	2.2/2.7	533/533
<code>g05_80.0</code>	80	3/3	4.4/4.8	929/929
<code>g05_80.1</code>	80	3/3	1.8/2.3	941/941
<code>g05_80.2</code>	80	3/3	2.7/2.4	934/934
<code>g05_80.3</code>	80	3/3	11.8/11.7	923/923
<code>g05_80.4</code>	80	3/3	4.8/8.3	932/932
<code>g05_80.5</code>	80	3/3	3.4/3.6	926/926
<code>g05_80.6</code>	80	3/3	3.0/3.0	929/929
<code>g05_80.7</code>	80	3/3	2.5/2.0	929/929
<code>g05_80.8</code>	80	3/3	4.0/4.2	925/925
<code>g05_80.9</code>	80	3/3	4.8/5.6	923/923
<code>pm1d_80.0</code>	80	3/3	3.4/3.1	227/227
<code>pm1d_80.1</code>	80	3/3	7.4/9.5	245/245
<code>pm1d_80.2</code>	80	3/3	6.2/4.9	284/284
<code>pm1d_80.3</code>	80	3/3	2.9/2.5	291/291
<code>pm1d_80.4</code>	80	3/3	3.4/3.3	251/251
<code>pm1d_80.5</code>	80	3/3	10.0/10.0	242/242
<code>pm1d_80.6</code>	80	3/3	5.7/5.8	205/205
<code>pm1d_80.7</code>	80	3/3	8.1/7.6	249/249
<code>pm1d_80.8</code>	80	3/3	4.8/3.2	293/293
<code>pm1d_80.9</code>	80	3/3	2.7/2.4	258/258
<code>pm1s_80.0</code>	80	3/3	1.4/1.2	79/79
<code>pm1s_80.1</code>	80	3/3	1.6/1.4	85/85
<code>pm1s_80.2</code>	80	3/3	3.1/2.7	82/82
<code>pm1s_80.3</code>	80	3/3	2.2/2.0	81/81
<code>pm1s_80.4</code>	80	3/3	1.6/1.4	70/70
<code>pm1s_80.5</code>	80	3/3	1.7/1.7	87/87
<code>pm1s_80.6</code>	80	3/3	2.0/2.0	73/73
<code>pm1s_80.7</code>	80	3/3	1.8/1.5	83/83
<code>pm1s_80.8</code>	80	3/3	2.0/2.0	81/81
<code>pm1s_80.9</code>	80	3/3	1.4/1.4	70/70
<code>g05_100.0</code>	100	3/3	31.5/41.5	1430/1430

Continued on next page

Table S13 (continued)

Instance	N	Completed M/A	Median seconds M/A	Best cut M/A
g05_100.1	100	3/3	183.8/210.9	1425/1425
g05_100.2	100	3/3	21.2/26.2	1432/1432
g05_100.3	100	3/3	61.8/47.7	1424/1424
g05_100.4	100	3/3	7.2/10.2	1440/1440
g05_100.5	100	3/3	12.8/18.2	1436/1436
g05_100.6	100	3/3	23.2/21.9	1434/1434
g05_100.7	100	3/3	31.8/49.8	1431/1431
g05_100.8	100	3/3	23.1/30.2	1432/1432
g05_100.9	100	3/3	44.6/21.4	1430/1430
pm1d_100.0	100	3/3	43.7/45.5	340/340
pm1d_100.1	100	3/3	80.3/76.2	324/324
pm1d_100.2	100	3/3	51.0/45.2	389/389
pm1d_100.3	100	3/3	17.5/21.0	400/400
pm1d_100.4	100	3/3	42.8/42.4	363/363
pm1d_100.5	100	3/3	17.9/20.6	441/441
pm1d_100.6	100	3/3	13.2/13.7	367/367
pm1d_100.7	100	3/3	10.3/9.8	361/361
pm1d_100.8	100	3/3	7.3/7.0	385/385
pm1d_100.9	100	3/3	21.2/19.0	405/405
pm1s_100.0	100	3/3	3.0/3.1	127/127
pm1s_100.1	100	3/3	7.7/7.4	126/126
pm1s_100.2	100	3/3	2.3/2.7	125/125
pm1s_100.3	100	3/3	12.4/8.4	111/111
pm1s_100.4	100	3/3	6.5/4.7	128/128
pm1s_100.5	100	3/3	4.2/2.5	128/128
pm1s_100.6	100	3/3	4.4/5.2	122/122
pm1s_100.7	100	3/3	1.4/1.3	112/112
pm1s_100.8	100	3/3	4.1/3.2	120/120
pm1s_100.9	100	3/3	2.7/2.9	127/127
pw01_100.0	100	3/3	4.0/2.7	2019/2019
pw01_100.1	100	3/3	6.1/4.4	2060/2060
pw01_100.2	100	3/3	5.6/3.1	2032/2032
pw01_100.3	100	3/3	3.8/5.0	2067/2067
pw01_100.4	100	3/3	2.1/1.6	2039/2039
pw01_100.5	100	3/3	1.7/1.9	2108/2108
pw01_100.6	100	3/3	5.9/2.7	2032/2032
pw01_100.7	100	3/3	3.9/4.1	2074/2074
pw01_100.8	100	3/3	1.8/1.8	2022/2022
pw01_100.9	100	3/3	9.4/7.6	2005/2005
pw05_100.0	100	3/3	83.5/62.9	8190/8190
pw05_100.1	100	3/3	27.1/21.7	8045/8045
pw05_100.2	100	3/3	41.6/29.1	8039/8039
pw05_100.3	100	3/3	11.0/8.1	8139/8139
pw05_100.4	100	3/3	43.0/33.0	8125/8125
pw05_100.5	100	3/3	11.6/9.6	8169/8169
pw05_100.6	100	3/3	65.0/53.8	8217/8217
pw05_100.7	100	3/3	24.1/15.4	8249/8249
pw05_100.8	100	3/3	21.0/8.2	8199/8199
pw05_100.9	100	3/3	22.0/18.1	8099/8099
pw09_100.0	100	3/3	34.8/24.8	13585/13585
pw09_100.1	100	3/3	50.2/45.5	13417/13417
pw09_100.2	100	3/3	17.0/13.9	13461/13461
pw09_100.3	100	3/3	13.9/11.1	13656/13656
pw09_100.4	100	3/3	21.0/18.9	13514/13514
pw09_100.5	100	3/3	61.7/42.1	13574/13574
pw09_100.6	100	3/3	26.0/18.2	13640/13640
pw09_100.7	100	3/3	51.9/43.6	13501/13501
pw09_100.8	100	3/3	27.9/20.1	13593/13593
pw09_100.9	100	3/3	19.2/16.2	13658/13658
w01_100.0	100	3/3	3.5/2.0	651/651

Continued on next page

Table S13 (continued)

Instance	N	Completed M/A	Median seconds M/A	Best cut M/A
w01_100.1	100	3/3	4.5/2.1	719/719
w01_100.2	100	3/3	11.6/4.4	676/676
w01_100.3	100	3/3	3.7/2.5	813/813
w01_100.4	100	3/3	1.7/1.5	668/668
w01_100.5	100	3/3	5.0/2.7	643/643
w01_100.6	100	3/3	5.6/10.5	654/654
w01_100.7	100	3/3	7.0/3.2	725/725
w01_100.8	100	3/3	1.6/2.3	721/721
w01_100.9	100	3/3	1.9/1.9	729/729
w05_100.0	100	3/3	42.9/37.3	1646/1646
w05_100.1	100	3/3	19.6/14.9	1606/1606
w05_100.2	100	3/3	13.4/14.2	1902/1902
w05_100.3	100	3/3	33.5/38.0	1627/1627
w05_100.4	100	3/3	50.2/56.1	1546/1546
w05_100.5	100	3/3	66.0/47.1	1581/1581
w05_100.6	100	3/3	14.3/11.6	1479/1479
w05_100.7	100	3/3	13.0/12.3	1987/1987
w05_100.8	100	3/3	45.0/44.3	1311/1311
w05_100.9	100	3/3	10.9/8.8	1752/1752
w09_100.0	100	3/3	40.5/33.5	2121/2121
w09_100.1	100	3/3	208.5/220.6	2096/2096
w09_100.2	100	3/3	58.2/73.7	2738/2738
w09_100.3	100	3/3	87.5/79.7	1990/1990
w09_100.4	100	3/3	50.4/53.6	2033/2033
w09_100.5	100	3/3	5.9/4.6	2433/2433
w09_100.6	100	3/3	14.7/10.9	2220/2220
w09_100.7	100	3/3	25.4/23.8	2252/2252
w09_100.8	100	3/3	24.3/21.9	1843/1843
w09_100.9	100	3/3	31.6/33.9	2043/2043
g05_150.0	150	0/0	limit/limit	3148/3148
g05_150.1	150	3/3	12215.3/12377.6	3149/3149
g05_150.2	150	3/0	17831.5/limit	3144/3141
g05_150.3	150	0/3	limit/19860.8	3146/3146
g05_150.4	150	3/3	19536.1/11678.4	3148/3148
g05_150.5	150	3/3	7897.4/5536.7	3153/3153
g05_150.6	150	0/0	limit/limit	3146/3146
g05_150.7	150	0/0	limit/limit	3144/3143
g05_150.8	150	0/0	limit/limit	3141/3142
g05_150.9	150	0/0	limit/limit	3145/3145
pm1d_150.0	150	0/3	limit/17778.4	717/717
pm1d_150.1	150	3/3	10734.5/9502.6	687/687
pm1d_150.2	150	3/3	1298.0/1491.4	656/656
pm1d_150.3	150	3/3	178.6/209.5	762/762
pm1d_150.4	150	3/3	18909.7/15652.8	671/671
pm1d_150.5	150	3/3	17111.8/15866.2	606/606
pm1d_150.6	150	0/0	limit/limit	715/713
pm1d_150.7	150	3/3	17265.7/16632.6	708/708
pm1d_150.8	150	3/3	12478.8/11555.2	739/739
pm1d_150.9	150	3/3	1615.7/1590.8	640/640
pm1s_150.0	150	3/3	600.1/998.9	207/207
pm1s_150.1	150	3/3	42.5/73.9	225/225
pm1s_150.2	150	3/3	1014.4/2486.4	211/211
pm1s_150.3	150	3/3	462.2/370.4	228/228
pm1s_150.4	150	3/3	168.8/146.8	195/195
pm1s_150.5	150	3/3	28.1/43.2	217/217
pm1s_150.6	150	3/3	2720.5/2866.7	231/231
pm1s_150.7	150	3/3	881.6/649.1	223/223
pm1s_150.8	150	3/3	1052.2/1328.0	175/175
pm1s_150.9	150	3/3	1267.0/810.6	174/174
pw01_150.0	150	3/3	35.9/45.0	4402/4402

Continued on next page

Table S13 (continued)

Instance	N	Completed M/A	Median seconds M/A	Best cut M/A
pw01_150.1	150	3/3	226.0/255.2	4376/4376
pw01_150.2	150	3/3	112.0/91.1	4272/4272
pw01_150.3	150	3/3	28.1/21.7	4285/4285
pw01_150.4	150	3/3	295.2/302.3	4297/4297
pw01_150.5	150	3/3	226.0/34.1	4489/4489
pw01_150.6	150	3/3	259.7/161.6	4446/4446
pw01_150.7	150	3/3	142.2/134.7	4425/4425
pw01_150.8	150	3/3	79.7/43.9	4242/4242
pw01_150.9	150	3/3	432.5/779.5	4287/4287
pw05_150.0	150	0/0	limit/limit	17852/17852
pw05_150.1	150	3/3	13281.8/8455.6	17780/17780
pw05_150.2	150	3/3	4680.4/3521.1	17774/17774
pw05_150.3	150	3/3	14572.5/11109.1	17742/17742
pw05_150.4	150	3/3	9354.3/6808.0	17735/17735
pw05_150.5	150	0/0	limit/limit	17867/17867
pw05_150.6	150	3/3	12813.6/7267.4	18123/18123
pw05_150.7	150	0/0	limit/limit	17828/17828
pw05_150.8	150	3/3	618.9/602.4	17758/17758
pw05_150.9	150	3/3	2824.2/2034.1	17735/17735
pw09_150.0	150	3/3	3118.2/2550.8	30015/30015
pw09_150.1	150	3/3	6559.7/7803.3	30003/30003
pw09_150.2	150	0/3	limit/17996.5	29914/29914
pw09_150.3	150	3/3	1386.0/1353.5	30059/30059
pw09_150.4	150	3/3	12910.9/12169.8	29975/29975
pw09_150.5	150	3/3	664.3/878.7	30301/30301
pw09_150.6	150	3/3	5538.5/5913.7	30280/30280
pw09_150.7	150	0/1	limit/mixed	30211/30211
pw09_150.8	150	3/2	19582.0/mixed	29954/29954
pw09_150.9	150	3/3	8072.0/6493.4	29878/29878
w01_150.0	150	3/3	408.0/377.9	1394/1394
w01_150.1	150	3/3	199.6/201.9	1422/1422
w01_150.2	150	3/3	18.6/33.1	1315/1315
w01_150.3	150	3/3	111.5/110.6	1264/1264
w01_150.4	150	3/3	64.6/72.5	1194/1194
w01_150.5	150	3/3	40.4/26.1	1303/1303
w01_150.6	150	3/3	264.3/288.1	1283/1283
w01_150.7	150	3/3	205.6/233.5	1260/1260
w01_150.8	150	3/3	141.5/81.7	1167/1167
w01_150.9	150	3/3	30.2/28.4	1217/1217
w05_150.0	150	3/3	14955.7/13564.2	3008/3008
w05_150.1	150	3/3	6018.8/5389.4	3032/3032
w05_150.2	150	3/3	2450.7/3284.9	2900/2900
w05_150.3	150	2/3	mixed/12217.6	2742/2742
w05_150.4	150	3/3	164.8/233.1	2815/2815
w05_150.5	150	3/3	11721.9/10362.6	2609/2609
w05_150.6	150	3/3	6143.6/5774.8	2835/2835
w05_150.7	150	3/0	11040.2/limit	2608/2608
w05_150.8	150	3/3	2120.1/2562.4	2781/2781
w05_150.9	150	3/3	6495.1/5935.4	2725/2725
w09_150.0	150	3/3	10884.0/10628.9	4278/4278
w09_150.1	150	0/0	limit/limit	4177/4177
w09_150.2	150	3/3	1656.1/1758.0	4019/4019
w09_150.3	150	0/0	limit/limit	3451/3451
w09_150.4	150	3/3	14991.7/16082.2	3831/3831
w09_150.5	150	3/3	1730.5/2326.9	3579/3579
w09_150.6	150	3/3	2853.2/3605.6	3889/3889
w09_150.7	150	0/0	limit/limit	3304/3304
w09_150.8	150	0/0	limit/limit	3976/3976
w09_150.9	150	3/3	11603.3/8543.3	3593/3593