Dealing with inequality constraints in large-scale semidefinite relaxations for graph coloring and maximum clique problems

Federico Battista* Marianna De Santis†

August 2, 2024

Abstract

Semidefinite programs (SDPs) can be solved in polynomial time by interior point methods. However, when the dimension of the problem gets large, interior point methods become impractical in terms of both computational time and memory requirements. Certain first-order methods, such as Alternating Direction Methods of Multipliers (ADMMs), established as suitable algorithms to deal with large-scale SDPs and gained growing attention over the past decade. In this paper, we focus on an ADMM designed for SDPs in standard form and extend it to deal with inequalities when solving SDPs in general form. Beside numerical results on randomly generated instances, where we show that our method compares favorably with respect to the state-of-the-art solver `SDPNAL+` [29], we present results on instances from SDP relaxations of classical combinatorial problems such as the graph coloring problem and the maximum clique problem. Through extensive numerical experiments, we show that even an inaccurate dual solution, obtained at a generic iteration of our proposed ADMM, can represent an efficiently recovered valid bound on the optimal solution of the combinatorial problems considered, as long as an appropriate post-processing procedure is applied.

Keywords: Semidefinite programming, Graph coloring problem, Maximum clique problem
MSC: 90C22, 90C27, 90C06

1 Introduction

Interest on semidefinite programming has considerably grown during the last two decades and is partly due to the fact that many practical problems in operations research and combinatorial optimization can be modeled or approximated by semidefinite programs [18]. The purpose of this paper is to focus on the use of augmented Lagrangian methods for dealing with semidefinite programming relaxations of two well-known combinatorial problems: the graph coloring problem and the maximum clique problem. Augmented Lagrangian methods are known to be an alternative to interior point methods and currently represent the most popular first-order algorithms used to handle large-scale semidefinite programs [2, 3, 24, 22]. As a variant of augmented Lagrangian methods, Alternating Direction Methods of Multipliers

*Department of Industrial and Systems Engineering, Lehigh University, Bethlehem PA 18015, USA. E-mail: `feb223@lehigh.edu`

†Dipartimento di Ingegneria Informatica Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto, 25 00185, Roma, Italy. E-mail: `mdesantis@diag.uniroma1.it`

(ADMMs) have gained increasing attention in recent years [26, 29, 27, 6, 1]. Falling under the class of first-order methods, their success can be attributed to the avoidance of computation, storage, and factorization of large Hessian matrices. This in turn enables a significant increase in scalability compared to interior point methods. On the other hand, this comes at some cost to accuracy, which should be properly addressed in the scenario where the semidefinite problem is a relaxation of some combinatorial problem with the goal of obtaining a valid bound on its optimal solution. In order to overcome this issue, safe-bounding procedures have been recently developed (see e.g. [1, 16, 4, 28]). These methods have mostly been employed to recover a posteriori the inaccuracies that are produced by solvers and provide a valid bound on the optimum.

In this paper, we begin by extending both an existing ADMM and a safe-bounding procedure to deal with SDPs with both equality and inequality constraints. Subsequently, we compare their performance with SDPNAL+ [29], an established state-of-the-art solver for large-scale SDPs that was awarded the Beale-Orchard-Hays Prize in 2018. In particular, we present numerical experiments on randomly generated SDPs and on instances of well-known SDP relaxations from the literature for the maximum clique problem and the graph coloring problem. The goal is twofold. First, we demonstrate the robustness of our extended ADMM equipped with a safe-bounding procedure compared to a state-of-the-art solver. Second, we show that even a low-precision dual solution obtained during a generic iteration of our ADMM can serve as a valid and efficiently recovered bound on the optimal solution of the combinatorial problems considered. As a byproduct, an extensive collection of noteworthy SDP bounds for two fundamental combinatorial optimization problems are presented.

1.1 Notation and outline

Let $\mathbb{S}_n$ be the set of n-by-n symmetric matrices. Further, let $\mathbb{S}_n^+ \subset \mathbb{S}_n$ ($\mathbb{S}_n^{++} \subset \mathbb{S}_n$) be the set of positive semidefinite (positive definite) matrices and $\mathbb{S}_n^- \subset \mathbb{S}_n$ be the set of negative semidefinite matrices. In the following, we denote by $\langle X, Y \rangle = \text{trace}(XY)$ the standard inner product in $\mathbb{S}_n$. Whenever a norm is used, we consider the Frobenius norm in the case of matrices and the Euclidean norm in the case of vectors. Letting $M \in \mathbb{R}^{m \times n}$, we denote by $\text{vec}(M)$ the mn-dimensional vector formed by stacking the columns of M on top of each other (vec^{-1} is the inverse operation). Letting $v \in \mathbb{R}^n$, we denote by $\text{Diag}(v)$ the diagonal matrix having the elements of v on the main diagonal. We denote by e_i the i-th vector of the standard basis in $\mathbb{R}^n$. Given $S \in \mathbb{S}_n$, we denote by $(S)_+$ and $(S)_-$ the projections of S onto the positive semidefinite and negative semidefinite cones, respectively. Moreover we denote by $\lambda(S)$ the vector of the eigenvalues of S and by $\lambda_{\min}(S)$ and $\lambda_{\max}(S)$ the smallest and largest eigenvalue of S, respectively. At last, we denote by $\mathbf{0}_n$ and $\mathbf{0}_{n \times \ell}$ the all-zero column vector of size n and the all-zero $n \times \ell$ matrix, respectively.

The paper is organized as follows. In Section 2, we succinctly review the ADMM algorithm `ADAL` for solving SDPs in standard form, originally presented in [24, 22, 27]. In Section 3, we discuss SDPs with both equality and inequality constraints and we extend the aforementioned method to handle such problems. Starting from a low-precision dual solution, we then apply the procedure outlined in [4] to our context to recover a valid dual bound on the optimal primal value. In Section 4, drawing from existing literature, we review well-known SDP formulations for relaxations of two fundamental combinatorial optimization problems: the maximum clique and the graph coloring problems. These formulations, in addition to randomly generated SDPs, constitute the test set for the numerical experiments we report in Section 5. At last,

some conclusions are drawn in Section 6.

2 ADAL: an ADMM for SDPs in Standard Form

In this section we review the basic concepts of `ADAL` [24, 22, 27], an Alternating Direction Method of Multipliers (ADMM) to address SDPs in standard form:

$$\begin{array}{ll} \min & \langle C, X\rangle \\ \text{s.t.} & \langle A^j, X\rangle = b_j, \quad \forall j = 1,\dots,m \\ & X \in \mathcal{S}_n^+, \end{array} \tag{1}$$

where $C \in \mathcal{S}_n$, $A^j \in \mathcal{S}_n$, for all $j = 1,\dots,m$, and $b := (b_1,\dots,b_m) \in \mathbb{R}^m$. By defining the linear operator $\mathcal{A} : \mathcal{S}_n \to \mathbb{R}^m$, with $(\mathcal{A}X)_j = \langle A^j, X\rangle$ for $A^j \in \mathcal{S}_n$, Problem (1) can be rewritten as

$$\begin{array}{ll} \min & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}X = b \\ & X \in \mathcal{S}_n^+. \end{array} \tag{PSDP-ST}$$

The dual of (PSDP-ST) is defined as

$$\begin{array}{ll} \max & b^T y \\ \text{s.t.} & \mathcal{A}^\top y + Z = C \\ & Z \in \mathcal{S}_n^+, \end{array} \tag{DSDP-ST}$$

where $\mathcal{A}^\top : \mathbb{R}^m \to \mathcal{S}_n$ is the adjoint operator of $\mathcal{A}$, namely $\mathcal{A}^\top y = \sum_{i=1}^m y_i A^i$ for $y \in \mathbb{R}^m$.

Assumption 1. *Slater condition holds for* (PSDP-ST) *and* (DSDP-ST)*; that is, there exist matrices* $\tilde{X}, \tilde{Z} \in \mathcal{S}_n^{++}$*, and* $\tilde{y} \in \mathbb{R}^m$ *satisfying* $\mathcal{A}\tilde{X} = b$ *and* $\mathcal{A}^\top \tilde{y} + \tilde{Z} = C$*.*

Under this assumption, it is well known that strong duality holds. Hence, the following KKT conditions are necessary and sufficient for the optimality of a triplet (X, y, Z) (see, e.g., [23, Sec. 4.2]):

$$\mathcal{A}X = b, \qquad \mathcal{A}^\top y + Z = C, \qquad ZX = 0, \qquad X \in \mathcal{S}_n^+, \qquad Z \in \mathcal{S}_n^+.$$

The method we consider is based on the maximization of the augmented Lagrangian built over the dual problem. Let $X \in \mathcal{S}_n$ be the Lagrange multiplier for the dual equation $\mathcal{A}^\top y + Z - C = 0$ and $\sigma > 0$ be fixed. The augmented Lagrangian of (DSDP-ST) is defined as

$$L_\sigma(y, Z; X) = b^T y - \langle \mathcal{A}^\top y + Z - C, X\rangle - \frac{\sigma}{2}\|\mathcal{A}^\top y + Z - C\|^2.$$

In augmented Lagrangian methods applied to the dual (DSDP-ST) the problem

$$\begin{array}{ll} \max & L_\sigma(y, Z; X) \\ \text{s.t.} & y \in \mathbb{R}^m, \quad Z \in \mathcal{S}_n^+, \end{array} \tag{2}$$

is addressed at every iteration, where X is fixed and $\sigma > 0$ is a penalty parameter. When the maximization of the augmented Lagrangian $L_\sigma(y, Z; X)$ is performed by iteratively optimizing with respect to y at first, and then with respect to Z, we are considering the ADMM originally proposed in [24, 22] and then extended in [27].

We present this method following [27], in which it is referred as `ADAL` (Alternating Direction Augmented Lagrangian). At each iteration of `ADAL`, the new point $(y^{k+1}, Z^{k+1}, X^{k+1})$ is computed by the following steps:

$$y^{k+1} = \operatorname*{argmax}_{y \in \mathbb{R}^m} L_{\sigma^k}(y, Z^k; X^k), \tag{3}$$

$$Z^{k+1} = \operatorname*{argmax}_{Z \in \mathcal{S}_n^+} L_{\sigma^k}(y^{k+1}, Z; X^k), \tag{4}$$

$$X^{k+1} = X^k + \sigma^k(\mathcal{A}^\top y^{k+1} + Z^{k+1} - C). \tag{5}$$

The update of y in (3) can be performed in closed form, as it derives from the first-order optimality conditions of the problem on the right-hand side of (3), i.e., y^{k+1} is the unique solution of

$$\nabla_y L_{\sigma^k}(y, Z^k; X^k) = b - \mathcal{A}(X^k + \sigma^k(\mathcal{A}^\top y + Z^k - C)) = 0.$$

That is,

$$y^{k+1} = (\mathcal{A}\mathcal{A}^\top)^{-1}\Big(\frac{1}{\sigma^k} b - \mathcal{A}(\frac{1}{\sigma^k} X^k + Z^k - C)\Big).$$

The update of Z in (4) is conducted by considering the equivalent problem

$$\min_{Z \in \mathcal{S}_n^+} \|Z + W^{k+1}\|^2, \tag{6}$$

where

$$W^{k+1} = \frac{X^k}{\sigma^k} - C + \mathcal{A}^\top y^{k+1}.$$

Solving problem (6), is equivalent to projecting $W^{k+1} \in \mathcal{S}_n$ onto the (closed convex) cone $\mathcal{S}_n^-$ and computing its additive inverse (see Algorithm 1). Such a projection is computed via the spectral decomposition of the matrix W^{k+1}. Finally, it is clear to see that the update of X in (5) can be obtained as follows:

$$\begin{aligned} X^{k+1} &= X^k + \sigma^k(\mathcal{A}^\top y^{k+1} + Z^{k+1} - C) = \\ &= \sigma^k(X^k/\sigma^k - C + \mathcal{A}^\top y^{k+1} - (X^k/\sigma^k - C + \mathcal{A}^\top y^{k+1})_-) = \\ &= \sigma^k(X^k/\sigma^k - C + \mathcal{A}^\top y^{k+1})_+ = \sigma^k(W^{k+1})_+. \end{aligned}$$

We report in Algorithm 1 the scheme of `ADAL`. The method stops as soon as the following errors related to primal feasibility ($\mathcal{A}X = b$, $X \geq 0$) and dual feasibility ($\mathcal{A}^\top y + Z + S = C$) are below a certain threshold defined, respectively, as the following:

$$r_P = \frac{\|\mathcal{A}X - b\|}{1 + \|b\|}, \qquad r_D = \frac{\|\mathcal{A}^\top y + Z - C\|}{1 + \|C\|}.$$

More precisely, the algorithm stops as soon as the quantity $\delta = \max\{r_P, r_D\}$ is less than a fixed precision $\varepsilon > 0$. It should be noted that the optimality conditions $X \in \mathcal{S}_n^+$, $Z \in \mathcal{S}_n^+$ and $ZX = 0$ are satisfied up to machine accuracy throughout the algorithm thanks to the projections

Algorithm 1 Scheme of ADAL from [27]

1: Choose $\sigma > 0$, $\varepsilon > 0$, $X \in \mathcal{S}_n^+$, $Z \in \mathcal{S}_n^+$
2: $\delta = \max\{r_P, r_D\}$
3: **while** $\delta > \varepsilon$ **do**
4: $\quad y = (\mathcal{A}\mathcal{A}^\top)^{-1}\Big(\frac{1}{\sigma}b - \mathcal{A}(\frac{1}{\sigma}X - C + Z)\Big)$
5: $\quad W = X/\sigma - C + \mathcal{A}^\top y$
6: $\quad Z = -W_-$ and $X = \sigma W_+$
7: $\quad \delta = \max\{r_P, r_D\}$
8: $\quad$ Update σ
9: **end while**

employed in ADAL. In the convergence analysis proposed in [27], Algorithm 1 is interpreted as a fixed point method, i.e., at each iteration of ADAL, the update of the primal and dual variables (X, Z) is the result of the combination of two operators, both of which are proven to be non-expansive. Indeed, we have that $(X^{k+1}, Z^{k+1}) = \mathcal{P}(W(X^k, Z^k))$, where $\mathcal{P}$ denotes the projection performed at Step 6 in Algorithm 1 and $W(X^k, Z^k) = X^k/\sigma^k - C + \mathcal{A}^\top y(Z^k, X^k)$, being the update performed at Step 4 in Algorithm 1, i.e.,

$$y(Z^k, X^k) = (\mathcal{A}\mathcal{A}^\top)^{-1}\Big(\frac{1}{\sigma^k}b - \mathcal{A}(\frac{1}{\sigma}^k X^k - C + Z^k)\Big).$$

Using the non-expansivity of $\mathcal{P}$ and W (see Lemma 3 and Lemma 4 in [27], respectively), it is possible to prove the following result (see Theorem 2 in [27]):

Theorem 1. *The sequence $\{(X^k, y^k, Z^k)\}$ generated by Algorithm 1 applied to Problem (PSDP-ST) from any starting point (X^0, y^0, Z^0) converges to a solution $(X^*, y^*, Z^*) \in \Omega^*$, where Ω^* is the set of primal and dual solutions of (PSDP-ST) and (DSDP-ST).*

3 ADAL-ineq: applying ADAL to SDPs in general form

The aim of our work is to address SDPs that include linear inequality constraints. In order to do so, we can still use ADAL, namely Algorithm 1, by applying it to the reformulation with only equality constraints obtained through the introduction of slack variables. However, when dealing with large-scale SDPs, the memory required by ADAL to solve such a reformulation would grow substantially. In this section, we show how to rewrite the steps of ADAL in terms of the matrices defining the original problem in a more efficient way that will reduce the memory requirements. For clarity, we will refer to this new version of ADAL as to ADAL-ineq. Recall that an SDP in general form can be formulated as follows:

$$\begin{array}{lll} \min & \langle C, X\rangle & \\ \text{s.t.} & \langle A^i, X\rangle \le b_i, & \forall i = 1, \dots, \ell \\ & \langle A^j, X\rangle = b_j, & \forall j = \ell+1, \dots, m \\ & X \in \mathcal{S}_n^+, & \end{array} \tag{7}$$

where $C \in \mathcal{S}_n$, $A^i \in \mathcal{S}_n$, for all $i = 1, \ldots, m$, and $b \in \mathbb{R}^m$. A standard way to deal with inequalities in problem (7) is to add slack variables $s_i \geq 0$, for all $i = 1, \ldots, \ell$, and expand the matrix variable X to $\bar{X} \in \mathcal{S}_{n+\ell}$:

$$\bar{X} := \begin{pmatrix} X & \mathbf{0}_{n\times\ell} \\ \mathbf{0}_{\ell\times n} & \mathrm{Diag}(s) \end{pmatrix}.$$

Recall that if B is a diagonal matrix, the constraint $B \succeq 0$ boils down to $B \geq 0$. In particular, imposing $\bar{X} \succeq 0$ is equivalent to considering $X \succeq 0$ and $s \geq 0$. By expanding the matrices A^i, A^j, and C, to $\bar{A}^i$, $\bar{A}^j$ and $\bar{C}$, respectively, for all $i = 1, \ldots, \ell$ and $j = \ell + 1, \ldots, m$, as

$$\bar{A}^i := \begin{pmatrix} A^i & \mathbf{0}_{n\times\ell} \\ \mathbf{0}_{\ell\times n} & e_i^T e_i \end{pmatrix}, \qquad \bar{A}^j := \begin{pmatrix} A^j & \mathbf{0}_{n\times\ell} \\ \mathbf{0}_{\ell\times n} & \mathbf{0}_{\ell\times\ell} \end{pmatrix}, \qquad \bar{C} := \begin{pmatrix} C & \mathbf{0}_{n\times\ell} \\ \mathbf{0}_{\ell\times n} & \mathbf{0}_{\ell\times\ell} \end{pmatrix},$$

problem (7) can be rewritten as an SDP in standard form as follows:

$$\begin{array}{ll} \min & \langle \bar{C}, \bar{X} \rangle \\ \text{s.t.} & \bar{\mathcal{A}}\bar{X} = b \\ & \bar{X} \in \mathcal{S}^+_{n+\ell}, \end{array} \tag{8}$$

where $b := (b_1, \ldots, b_m) \in \mathbb{R}^m$ and $\bar{\mathcal{A}} : \mathcal{S}_{n+\ell} \to \mathbb{R}^m$ is the linear operator $(\bar{\mathcal{A}}X)_i = \langle \bar{A}^i, X \rangle$ with $\bar{A}^i \in \mathcal{S}_{n+\ell}$, for all $i = 1, \ldots, m$. The dual problem of (8) is defined as

$$\begin{array}{ll} \min & b^T y \\ \text{s.t.} & \bar{\mathcal{A}}^\top y + \bar{Z} = \bar{C} \\ & \bar{Z} \in \mathcal{S}^+_{n+\ell}, \end{array} \tag{9}$$

where $\bar{\mathcal{A}}^\top : \mathbb{R}^m \to \mathcal{S}_{n+\ell}$ is the adjoint operator of $\bar{\mathcal{A}}$, namely $\bar{\mathcal{A}}^\top y = \sum_{i=1}^m y_i \bar{A}^i$ for $y \in \mathbb{R}^m$. Note that the matrix $\bar{Z} \in \mathcal{S}_{n+\ell}$ is a "surplus" matrix variable that can be written as

$$\bar{Z} := \begin{pmatrix} Z & \mathbf{0}_{n\times\ell} \\ \mathbf{0}_{\ell\times n} & \mathrm{Diag}(p) \end{pmatrix},$$

with $p \in \mathbb{R}^\ell$. In particular, the equality constraint in (9) can be rewritten as

$$\bar{C} - \bar{\mathcal{A}}^T(y) - \bar{Z} = \begin{pmatrix} C - \mathcal{A}^T y - Z & \mathbf{0}_{n\times\ell} \\ \mathbf{0}_{\ell\times n} & \mathrm{Diag}(-y - p) \end{pmatrix} = 0.$$

As for the SDPs in standard form, if we assume that both the primal (8) and the dual (9) problems have strictly feasible points (i.e. Slater's condition is satisfied) strong duality holds and $(y, \bar{Z}, \bar{X})$ is optimal for (8) and (9) if and only if the following KKT conditions hold (see, e.g., [23, Sec. 4.2]):

$$\bar{\mathcal{A}}\bar{X} = b, \qquad \bar{\mathcal{A}}^\top y + \bar{Z} = \bar{C}, \qquad \bar{Z}\bar{X} = 0, \qquad \bar{X} \in \mathcal{S}^+_{n+\ell}, \qquad \bar{Z} \in \mathcal{S}^+_{n+\ell}. \tag{10}$$

In the following, we assume that the constraints formed through the operator $\bar{\mathcal{A}}$ are linearly independent.

The memory required to store the augmented matrices $\bar{C}, \bar{A}_i, \bar{A}_j, \bar{Z}$ and $\bar{X}$ substantially increases with the number ℓ of inequalities and even using efficient sparse matrix implementations may be insufficient to computationally deal with large-scale problems. Thus, we propose rewriting the steps of `ADAL` applied to Problem (8) in terms of the original matrices C, A^i, and X, so that one is only required to store the matrices that are actually defining the problem.

Proposition 1. *Step 4 in Algorithm 1 applied to Problem (8), reading as*

$$y = (\bar{\mathcal{A}}\bar{\mathcal{A}}^\top)^{-1}\Big(\frac{1}{\sigma}\bar{b} - \bar{\mathcal{A}}(\frac{1}{\sigma}\bar{X} - \bar{C} + \bar{Z})\Big)$$

can be performed in terms of the matrices of Problem (7) only, namely in terms of the matrices C, A^i, for all $i = 1, \dots, m$, X and Z.

Proof. Let $1 \leq i \leq \ell$ be a generic index of an inequality constraint and let $\ell + 1 \leq j \leq m$ be a generic index of an equality constraint, then the following holds:

$$\begin{aligned}\langle \bar{A}^i, \bar{X}\rangle &= \langle A^i, X\rangle + s_i,\\ \langle \bar{A}^j, \bar{X}\rangle &= \langle A^j, X\rangle,\\ \langle \bar{C}, \bar{X}\rangle &= \langle C, X\rangle.\end{aligned}$$

The linear map applied to $\bar{X}$ becomes:

$$\bar{\mathcal{A}}(\bar{X}) = \begin{pmatrix} \langle A^1, X\rangle \\ \vdots \\ \langle A^\ell, X\rangle \\ \langle A^{\ell+1}, X\rangle \\ \vdots \\ \langle A^m, X\rangle \end{pmatrix} + \begin{pmatrix} s^T \\ \mathbf{0}_{m-\ell} \end{pmatrix} = \mathcal{A}(X) + \begin{pmatrix} s^T \\ \mathbf{0}_{m-\ell} \end{pmatrix}.$$

Similarly, the adjoint operator $\bar{\mathcal{A}}^T : \mathbb{R}^m \to \mathcal{S}_{n+\ell}$ of $\bar{\mathcal{A}}$ is defined as

$$\bar{\mathcal{A}}^T y := \sum_{i=1}^m y_i \bar{A}^i = \begin{pmatrix} \sum_{i=1}^m y_i A^i & \mathbf{0}_{n\times\ell} \\ \mathbf{0}_{\ell\times n} & \mathrm{Diag}(y) \end{pmatrix} = \begin{pmatrix} \mathcal{A}^T y & \mathbf{0}_{n\times\ell} \\ \mathbf{0}_{\ell\times n} & \mathrm{Diag}(y) \end{pmatrix}.$$

Using the operator vec, we can write $\bar{\mathcal{A}}(\bar{X}) = b$ as $\bar{A}\ \mathrm{vec}(\bar{X}) = b$, where

$$\bar{A} := \big(\mathrm{vec}(\bar{A}^1), \dots, \mathrm{vec}(\bar{A}^m)\big)^T \in \mathbb{R}^{m\times(n+\ell)^2}.$$

Note that the matrix $\bar{A}^i$, for all $i = 1, \dots, \ell$, corresponding to the i-th inequality constraint, is the unique matrix having 1 in position $(n+i, n+i)$. Then, $\bar{A}\bar{A}^T$ can be expressed in terms of AA^T as follows:

$$\bar{A}\bar{A}^T = AA^T + \mathrm{Diag}\begin{pmatrix} \mathbf{1}_\ell \\ \mathbf{0}_{m-\ell} \end{pmatrix},$$

where $A := \big(\mathrm{vec}(A^1), \dots, \mathrm{vec}(A^m)\big)^T \in \mathbb{R}^{m\times n^2}$. Indeed, the zero entries of $\bar{A}^i$ do not contribute in the row-by-column product and the 1 in position $(n+i, n+i)$ contributes only to the entry where $\mathrm{vec}(\bar{A}^i)$ is multiplied by itself, i.e., in position (i, i) of $\bar{A}\bar{A}^T$. According to the notation introduced, the update of the y variable can be rewritten as follows:

$$y^{k+1} = \left(AA^T + \mathrm{Diag}\begin{pmatrix} \mathbf{1}_\ell \\ \mathbf{0}_{m-\ell} \end{pmatrix}\right)^{-1} \left(\frac{1}{\sigma^k} b - A\ \mathrm{vec}\left(\frac{1}{\sigma^k} X^k - C + Z^k\right) + \begin{pmatrix} \frac{1}{\sigma^k} s^{k^T} + p^{k^T} \\ \mathbf{0}_{m-\ell} \end{pmatrix}\right).$$

□

As a consequence of Proposition 1, the spectral decomposition of the matrix W, performed in Step 6 and needed for updating the variables $\bar{X}$ and $\bar{Z}$, can also be computed without storing the augmented matrices. Indeed W^{k+1} can be written in a "block-wise" fashion as follows:

$$W^{k+1} = \begin{pmatrix} \frac{1}{\sigma^k} X^k - C + \mathcal{A}^\top y^{k+1} & \mathbf{0}_{n\times\ell} \\ \mathbf{0}_{\ell\times n} & \text{Diag}\left(\frac{1}{\sigma^k} s^k + y^{k+1}\right) \end{pmatrix}.$$

To compute the eigenvalues and eigenvectors of W^{k+1}, we begin by performing the spectral decomposition of the matrix $\frac{X^k}{\sigma^k} - C + \mathcal{A}^T y^{k+1}$. Then, we straightforwardly obtain the eigenvalues and eigenvectors corresponding to the diagonal part of W^{k+1}. Finally, we adjust the dimension of the computed eigenvectors to ensure they belong to $\mathbb{R}^{n+\ell}$.

Proposition 1 and the reasoning reported above demonstrate that it is possible to apply Algorithm 1 to Problem (8) without storing the augmented matrices by rewriting its steps in terms of the matrices that are defining the original inequality constrained problem (7). As already mentioned, we denote this version of `ADAL` as `ADAL-ineq` and its scheme is reported in Algorithm 2.

Algorithm 2 Scheme of `ADAL-ineq`

Given Problem (8), choose $\sigma > 0$, $\varepsilon > 0$, $X \in \mathcal{S}_n^+$, $Z \in \mathcal{S}_n^+$, $s \in \mathbb{R}^\ell$, $p \in \mathbb{R}^\ell$
$\delta = \max\{r_P, r_D\}$
while $\delta > \varepsilon$ **do**
  $y = \left(AA^T + \text{Diag}\begin{pmatrix}\mathbf{1}_\ell \\ \mathbf{0}_{m-\ell}\end{pmatrix}\right)^{-1}\left(\frac{1}{\sigma}b - A\ \text{vec}\left(\frac{1}{\sigma}X - C + Z\right) + \begin{pmatrix}\frac{1}{\sigma}s^T + p^T \\ \mathbf{0}_{m-\ell}\end{pmatrix}\right)$
  $W = \begin{pmatrix} \frac{1}{\sigma}X - C + \mathcal{A}^\top y & \mathbf{0}_{n\times\ell} \\ \mathbf{0}_{\ell\times n} & \text{Diag}\left(\frac{1}{\sigma}s + y\right)\end{pmatrix}$
  $Z = -W_-$ and $X = \sigma W_+$
  $\delta = \max\{r_P, r_D\}$
  Update σ
end while

From the convergence of `ADAL` [27] we can state the following:

Theorem 2. *The sequence* $\{(\bar{X}^k, y^k, \bar{Z}^k)\}$ *generated by* `ADAL-ineq` *from any starting point* $(\bar{X}^0, y^0, \bar{Z}^0)$ *converges to a solution* $(\bar{X}^*, y^*, \bar{Z}^*) \in \Omega^*$*, where* Ω^* *is the set of primal and dual solutions of* (8) *and* (9)*.*

3.1 Obtaining dual bounds

The approximation of combinatorial problems is one of the most relevant applications of semidefinite programming. This is because the optimal solution of a semidefinite relaxation can be computed in polynomial time and generally gives a better bound than that obtained solving a linear relaxation (see e.g. [25]). Given a pair of primal-dual SDPs, weak and strong duality hold under the assumption that both problems are strictly feasible. Duality results imply that the objective function value of every feasible solution of the dual SDP is a valid bound on the optimal objective function value of the primal. Therefore, every dual feasible solution, and in particular the optimal dual solution of an SDP relaxation, gives a valid bound on the

solution of the related combinatorial optimization problem. Hence, being able to compute dual feasible solutions - even of moderate quality - can be extremely useful when considering branch-and-bound frameworks to define exact solution methods for specific combinatorial optimization problems. Following ideas developed in [4], we define a post-processing procedure for `ADAL-ineq` on general SDPs, that allows one to obtain a feasible dual solution starting from a positive semidefinite matrix $\tilde{Z} \in \mathcal{S}_n^+$. Let $\mathcal{A}_{ineq}$ and $\mathcal{A}_{eq}$ be the linear operators defining the inequality and equality constraints in problem (7), respectively; thus $\mathcal{A}_{ineq} = \langle A^i, X \rangle$ with $A^i \in \mathcal{S}_n$ and $\mathcal{A}_{eq} = \langle A^j, X \rangle$ with $A^j \in \mathcal{S}_n$, for all $i = 1, \ldots, \ell$ and $j = \ell + 1, \ldots, m$. Let b_{ineq} and b_{eq} be the right-hand-side vectors accordingly defined. Introducing the adjoint operators of $\mathcal{A}_{ineq}$ and $\mathcal{A}_{eq}$, the dual problem (7) can be equivalently written as

$$\begin{array}{ll} \max & -b_{ineq}^T \lambda + b_{eq}^T \mu \\ \text{s.t.} & C + \mathcal{A}_{ineq}^\top \lambda - \mathcal{A}_{eq}^\top \mu = Z \\ & Z \in \mathcal{S}_n^+, \ \lambda \geq 0, \end{array} \tag{11}$$

with $\lambda \in \mathbb{R}^\ell$ and $\mu \in \mathbb{R}^{m-\ell}$. We can then extend the results proposed in [4] and define a procedure to obtain feasible solutions of problem (11) along with, by weak duality, valid bounds on the optimal objective function value of the primal (7). Let $\tilde{Z} \in \mathcal{S}_n^+$. If the linear programming problem

$$\begin{array}{ll} \max & -b_{ineq}^T \lambda + b_{eq}^T \mu \\ \text{s.t.} & C + \mathcal{A}_{ineq}^\top \lambda - \mathcal{A}_{eq}^\top \mu = \tilde{Z} \\ & \lambda \geq 0 \end{array} \tag{12}$$

has an optimal solution $(\tilde{\lambda}, \tilde{\mu}) \in \mathbb{R}^m$, then $(\tilde{\lambda}, \tilde{\mu}, \tilde{Z})$ is a feasible solution for (11) and the value $-b_{ineq}^T \tilde{\lambda} + b_{eq}^T \tilde{\mu}$ yields a dual bound. If (12) is unbounded, then is (11) also unbounded, implying that the primal (7) is not feasible. If (12) is infeasible, then the initial $\tilde{Z} \in \mathcal{S}_n^+$ permits neither a feasible dual solution nor a dual bound. From a practical viewpoint, once problem (7) is approximately solved by `ADAL-ineq`, one can try to obtain a feasible solution to problem (11) by addressing problem (12).

4 Bounding the clique number and the chromatic number of a graph

Given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, a set $W \subseteq V$ is a *clique* if every two vertices in W are adjacent, while it is called *stable* if no two vertices in W are adjacent. The *clique number* $\omega(G)$ and the *stability number* $\alpha(G)$ are the maximum cardinalities of a clique and a stable set in G, respectively. A k-coloring is a partition of V into k stable sets. The *chromatic number* $\chi(G)$ is the smallest integer k for which G has a k-coloring. Denoting with $\bar{G} = (V, \bar{E})$ the complementary graph of G, it holds that

$$\omega(\bar{G}) = \alpha(G) \leq \chi(\bar{G}).$$

Lovász [21] introduced the so called *theta number* $\vartheta(G)$ that is an upper bound on the clique number $\omega(\bar{G})$, the stability number $\alpha(G)$, and is also a lower bound for the chromatic number $\chi(\bar{G})$. The important property of $\vartheta(G)$ is that it can be computed with an arbitrary precision

in polynomial time, as it is the optimal value of the following SDP [13]:

$$\begin{aligned} \vartheta(G) = \quad & \max \quad && \langle J, X\rangle \\ & \text{s.t.} \quad && \operatorname{trace}(X) = 1 \\ & && X_{ij} = 0 \qquad && \{i,j\} \in E \\ & && X \in \mathcal{S}_n^+, \end{aligned}$$

where J is the n-by-n matrix of all ones, with $n = |V|$. Starting from this relaxation, several attempts for sharpening $\vartheta(G)$ as a bound for $\omega(\bar{G})$, $\alpha(G)$ and $\chi(\bar{G})$ have been made (see, e.g., [8, 14, 10, 12, 11, 19, 9]). As a first way to improve $\vartheta(G)$, we consider the numbers $\vartheta_+(G)$ and $\bar{\vartheta}_+(G)$ obtained as solutions of the following SDPs, where bounds on the entries of the matrix variables are introduced:

$$\begin{aligned} \vartheta_+(G) = \quad & \max \quad && \langle J, X\rangle \\ & \text{s.t.} \quad && \operatorname{trace}(X) = 1 \\ & && X_{ij} = 0 \quad \{i,j\} \in E \\ & && X \geq 0 \\ & && X \in \mathcal{S}_n^+, \end{aligned}$$

$$\begin{aligned} \bar{\vartheta}_+(G) = \quad & \max \quad && \langle J, X\rangle \\ & \text{s.t.} \quad && \operatorname{trace}(X) = 1 \\ & && X_{ij} \leq 0 \quad \{i,j\} \in E \\ & && X \in \mathcal{S}_n^+. \end{aligned}$$

The values $\vartheta_+(G)$ and $\bar{\vartheta}_+(G)$ are related to $\omega(\bar{G})$, $\alpha(G)$ and $\chi(\bar{G})$ as follows

$$\omega(\bar{G}) = \alpha(G) \leq \vartheta_+(G) \leq \vartheta(G) \leq \bar{\vartheta}_+(G) \leq \chi(\bar{G}).$$

In the literature, equivalent formulations for both $\vartheta_+(G)$ and $\bar{\vartheta}_+(G)$ have been proposed [18] and for our computational experiments, we consider the following formulation for $\bar{\vartheta}_+(G)$:

$$\begin{aligned} \bar{\vartheta}_+(G) = \quad & \min \quad && t \\ & \text{s.t.} \quad && X_{ii} = t - 1 \qquad && i \in V \\ & && X_{ij} = -1 \qquad && \{i,j\} \in \bar{E} \\ & && X_{ij} \geq -1 \qquad && \{i,j\} \in E \\ & && X \in \mathcal{S}_n^+, \ t \in \mathbb{R}_+. \end{aligned} \tag{13}$$

where t is an additional auxiliary variable. Problem (13) can be reformulated as (7) by including t in the matrix variable as an additional element on the diagonal. Note that, in both the formulations of $\vartheta_+(G)$ and $\bar{\vartheta}_+(G)$, the entries of the matrix X are bounded from below. In the context of ADMMs defined over the dual problem, bounds on the matrix variable can be handled by introducing a further step, where a projection onto the nonnegative orthant is performed (see e.g. [27, 4, 28]). Although these 3-block ADMMs may not theoretically converge [5], they perform well in practice.

5 Numerical results

In this section, we present the results of our computational study, where we evaluate the performance of both `ADAL-ineq` and `SDPNAL+` [29]. The comparison is conducted on randomly

generated instance, as well as on SDP relaxations of both the stable set and the graph coloring problems. The software `SDPNAL+`, accessible at `https://blog.nus.edu.sg/mattohkc/softwares/sdpnalplus/`, integrates an ADMM with a semismooth Newton-Conjugate Gradient method. It is implemented in MATLAB, using a refined management of the matrices exploiting their symmetry. This approach allows the optimization of a significant portion of its C subroutines, which are provided through Mex files. `ADAL-ineq` is developed in MATLAB utilizing its built-in functions, and is available at `https://github.com/batt95/ADAL-ineq`, along with the instances used in our numerical experiments and an alternative Python implementation.

The numerical performance of ADMMs, including `ADAL-ineq`, strongly depend on the update rule used for the penalty parameter σ. As in [4, 28], we follow the strategy by Lorenz and Tran-Dinh [20], considering at every iteration k the ratio between the norm of the primal variable X^k and norm of the dual variable Z^k. In the implementation of our post-processing procedure, described in Section 3.1, we used Gurobi 9.1.1 [15] as the solver for problem (12). The experiments were carried out on an Intel(R) Xeon(R) CPU E5-2698 v4 running at 2.20GHz, with 256GB of RAM, under Linux (Ubuntu 16.04.7).

We compare the performance of the algorithms using performance profiles proposed by Dolan and Moré [7]. Given a set of solvers $\mathcal{S}$ and a set of problems $\mathcal{P}$, the performance of a solver $s \in \mathcal{S}$ on problem $p \in \mathcal{P}$ is compared against the best performance obtained by any solver in $\mathcal{S}$ on the same problem. The performance ratio is defined as $r_{p,s} = t_{p,s} / \min\{t_{p,s'} \mid s' \in \mathcal{S}\}$, where $t_{p,s}$ is the measure we want to compare, and we consider a cumulative distribution function $\rho_s(\tau) = |\{p \in \mathcal{P} \mid r_{p,s} \leq \tau\}|/|\mathcal{P}|$. The performance profile for $s \in S$ is the plot of the function ρ_s.

5.1 Comparison on randomly generated instances

The random SDPs considered in the first experiment are created from an adaptation of the instance generator used in [22]. Given a triplet $(n, m, p) \in \mathbb{N} \times \mathbb{N} \times [0, 1]$, the resulting SDP consists of a matrix variable $X \in \mathbb{S}^+_n$ and includes m linear constraints, of which $round(pm)$ are inequalities. Instances are generated with values from $n \in \{200, 250, 500, 1000\}$, $m \in \{5000, 10000, 25000, 50000, 100000\}$ and $p \in \{0.25, 0.5, 0.75\}$. For each parameter combination, we generate 5 different instances, excluding values of n and m that result in a constraint matrix $A \in \mathbb{R}^{m \times n^2}$ with linearly dependent rows. The final test set counts 150 random SDPs. A time limit of 1800 seconds of CPU time is set.

In Table 5.1, we report the comparison between `ADAL-ineq` and `SDPNAL+` in terms of number of iterations and CPU time needed in order to reach an accuracy of 10^{-5}. For each solver and each combination of n, m and p, we report the number of instances solved within the time limit along with the average running time. We notice that for $n = 250$ and $m = 25000$, `SDPNAL+` is not able to solve any instance within the time limit, while `ADAL-ineq` is able to solve all of them with a precision of 10^{-5}. For $n = 500$ and $m = 100000$, both algorithms are not able to solve any instance within the time limit. `SDPNAL+` performs better on instances with $n = 1000$ and $m = 10000$, while for the other instances either the two solvers show similar performances or `ADAL-ineq` outperforms `SDPNAL+`. The performance profiles of `ADAL-ineq` and `SDPNAL+` on random instances are reported in Figure 5.1, highlighting the superior performance of `ADAL-ineq` with respect to `SDPNAL+`; on close to 60% of the instances `ADAL-ineq` is the fastest algorithm and is also able to solve 90% of the instances whereas `SDPNAL+` is only able to solve 80% of the instances within the time limit.

			ADAL-ineq		SDPNAL+	
n	m	$p(\%)$	#sol	CPU time	#sol	CPU time
200	10000	25	5	39.24	5	33.05
		50	5	58.24	5	109.14
		75	5	67.14	5	713.82
250	5000	25	5	7.99	5	11.05
		50	5	9.87	5	15.51
		75	5	11.28	5	16.93
	25000	25	5	838.04	0	-
		50	5	1166.45	0	-
		75	5	1114.52	0	-
500	10000	25	5	15.52	5	15.54
		50	5	16.49	5	22.45
		75	5	28.87	5	23.94
	25000	25	5	18.11	5	31.33
		50	5	30.20	5	50.78
		75	5	45.53	5	52.57
	50000	25	5	217.61	5	106.28
		50	5	260.43	5	221.66
		75	5	325.71	5	250.97
	100000	25	0	-	0	-
		50	0	-	0	-
		75	0	-	0	-
1000	10000	25	5	136.63	5	49.52
		50	5	157.21	5	58.22
		75	5	242.63	5	71.38
	50000	25	5	57.19	5	60.96
		50	5	94.09	5	109.48
		75	5	110.00	5	111.29
	100000	25	5	83.15	5	136.53
		50	5	127.37	5	181.13
		75	5	155.05	5	184.21

Table 5.1: Results on 150 random instances

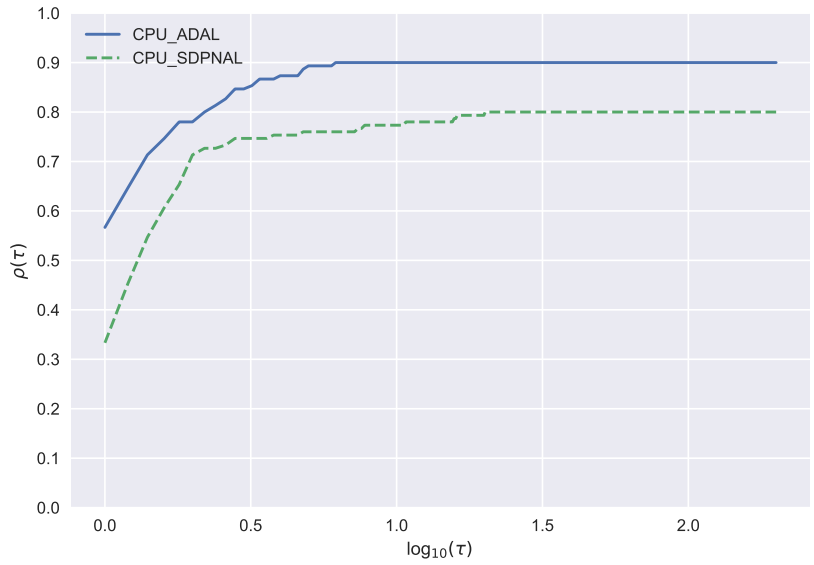

Figure 5.1: Performance profiles on CPU time. Comparison between `ADAL-ineq` and `SDPNAL+` on random instances.

5.2 Comparison on instances from SDP relaxations of the maximum clique problem

We now report the results on the SDP relaxation $\vartheta_+(G)$ for bounding the clique number (or the stability number) of a graph. We considered graphs from the second DIMACS implementation challenge [17], available at `ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/clique`. These graphs form the standard benchmark for the maximum clique problem. Hence, we considered the complement of these graphs to convert the maximum clique instances into stable set problem instances. For this experiment, a time limit of 3600 seconds was set.

In addition to employing `ADAL-ineq` and `SDPNAL+` to determine $\vartheta_+(G)$ and stopping the algorithms when the termination criteria were satisfied with a precision of 10^{-6}, we equipped `ADAL-ineq` with the post-processing procedure detailed in Section 3.1, applied every 200 iterations and after termination. Every time the post-processing procedure is called, we give as input the matrix Z^k obtained by `ADAL-ineq` at the corresponding iteration k and solve the linear programming problem (11) using Gurobi [15]. Along with the iterations of `ADAL-ineq`, we also store in memory the best dual bound found by the post-processing procedure, together with the CPU time needed to detect it. Note that every dual bound computed by the post-processing procedure and, in particular, the best dual bound are valid upper bounds on the stability number.

Table 5.2 is organized as follows: for each instance we report the name (Graph), values of $\vartheta_+(G)$ arising from, respectively, the dual objective function values returned by `ADAL-ineq`, `SDPNAL+` and the best dual bound found (BestBound) throughout the iterations of `ADAL-ineq`, along with the CPU times needed by `ADAL-ineq` and `SDPNAL+` to reach the stopping criterion, to identify the BestBound and the total time spent by the post-processing procedure. Note that the BestBound reported in the table is, in general, obtained in one of the post-processing calls when running `ADAL-ineq` and is not necessarily the one arising from the last call of the

post-processing. We highlight in bold the values of the bounds found on the instances in which they differ.

It should be noted that the post-processing procedure applied within `ADAL-ineq` is able to compute valid dual bounds on every instance but on `keller6`, where `ADAL-ineq` shows a failure. On `p_hat1500-2`, even if both `ADAL-ineq` and `SDPNAL+` did not converge within 3600 seconds, the post-processing procedure was able to compute a valid dual bound. In order to understand the quality of this bound, we ran `SDPNAL+` on the `p_hat1500-2` instance until convergence. Interestingly, it turns out that the BestBound and the optimal value match. By providing a valid dual bound, our procedure overcomes the impossibility of the solvers to reach termination criteria although the solution they achieve is close to an optimal one. Consider that for huge graphs the time needed to find the best dual bound may be greater than the time needed by `ADAL-ineq` to converge. This comes from the fact that the best dual bound can be recovered at the last iteration computed. We also wish to highlight that the overall time needed to apply the post-processing procedure in `ADAL-ineq` is small with respect to the overall time needed by the algorithm, and clearly it may be lowered by seldom applying the procedure.

In Figure 5.2, we report the performance profiles obtained with respect to the CPU time needed by `SDPNAL+` and the CPU time to identify the value of the BestBound. It is clear that `SDPNAL+` outperforms `ADAL-ineq` on these instances. However, we wish to highlight the superior performances of `ADAL-ineq` on the `p-hat` graphs, where we are often able to get the same bound as the optimal dual objective of `SDPNAL+` in a much lower CPU time.

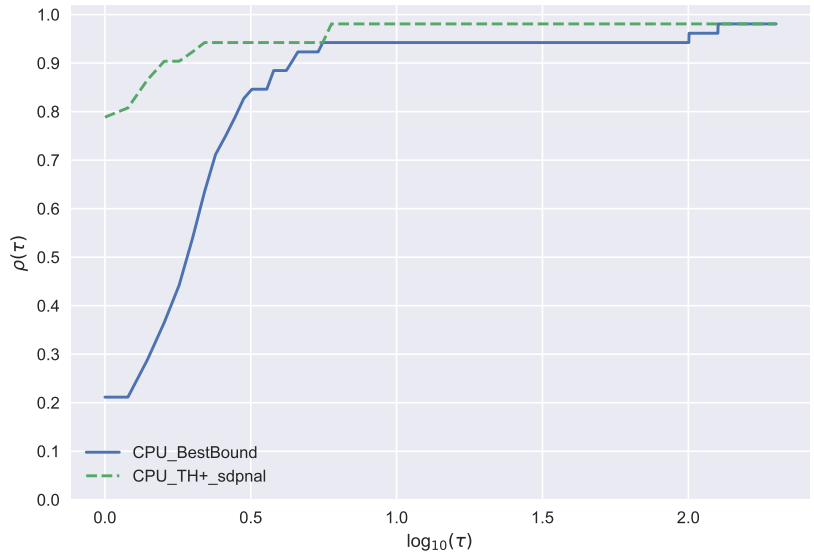

Figure 5.2: Performance profiles on CPU time. Comparison between BestBound and `SDPNAL+` on the computation of $\vartheta_+(G)$.

5.3 Comparison on instances from SDP relaxations of the coloring problem

We now report the results on the SDP relaxation $\bar{\vartheta}_+(G)$ for bounding the chromatic number of a graph. As before, we considered graphs from the second DIMACS implementation challenge [17], available at `https://sites.google.com/site/graphcoloring/downloads`. We

	$\vartheta_+(G)$			CPU times			
Graph	ADAL-ineq	SDPNAL+	BestBound	ADAL-ineq	SDPNAL+	BestBound	post-proc
DSJC125.1	38.04	38.04	38.04	2.89	3.52	2.56	0.12
DSJC125.5	11.40	11.40	11.40	1.93	0.77	1.70	0.09
DSJC125.9	4.00	4.00	4.00	2.41	1.17	2.44	0.09
DSJC500-5	22.57	22.57	22.57	6.83	4.85	7.30	0.48
DSJC1000-5	31.67	31.67	31.67	41.60	34.32	43.69	3.59
C125-9	37.55	37.55	37.55	2.88	0.99	2.73	0.11
C250-9	55.82	55.82	55.82	7.82	3.12	7.15	0.43
C500-9	83.58	83.58	83.58	30.55	8.32	31.03	1.48
C1000-9	122.60	122.60	122.60	159.79	34.93	147.00	9.74
C2000-5	44.56	44.56	44.56	389.59	534.67	398.86	22.42
C2000-9	177.73	177.73	177.73	1238.53	278.60	1247.62	68.30
brock200_1	27.20	27.20	27.20	3.45	1.06	3.12	0.24
brock200_2	14.13	14.13	14.13	1.91	1.08	1.98	0.15
brock200_3	18.67	18.67	18.67	2.24	1.07	2.31	0.15
brock200_4	21.12	21.12	21.12	2.83	0.96	2.92	0.18
brock400_1	39.33	39.33	39.33	7.81	3.84	8.09	0.53
brock400_2	39.20	39.20	39.20	8.30	4.02	8.58	0.50
brock400_3	39.16	39.16	39.16	8.64	3.59	8.92	0.48
brock400_4	39.23	39.23	39.23	7.99	3.53	8.27	0.49
brock800_1	41.87	41.87	41.87	24.31	11.67	20.96	2.66
brock800_2	42.10	42.10	42.10	23.94	12.88	20.49	2.59
brock800_3	41.88	41.88	41.88	24.52	13.02	25.87	2.57
brock800_4	42.00	42.00	42.00	23.91	12.80	20.28	2.50
p_hat300-1	10.02	10.02	10.02	18.45	16.72	8.85	0.95
p_hat300-2	26.71	26.71	26.71	211.40	161.90	28.17	11.20
p_hat300-3	40.70	40.70	40.70	35.69	36.28	16.91	1.78
p_hat500-1	13.01	13.01	13.01	34.11	14.71	22.55	2.04
p_hat500-2	38.56	38.56	38.56	580.86	537.38	92.52	32.79
p_hat500-3	57.81	57.81	57.81	99.65	33.72	61.56	5.03
p_hat700-1	15.05	15.05	15.05	59.86	33.94	43.15	4.20
p_hat700-2	48.44	48.44	48.44	1161.67	295.99	218.26	71.55
p_hat700-3	71.76	71.76	71.76	293.90	93.48	162.79	15.91
p_hat1000-1	17.52	17.52	17.52	144.76	119.26	84.75	10.38
p_hat1000-2	54.84	54.84	54.84	1815.65	697.11	487.41	121.86
p_hat1000-3	83.53	83.53	83.53	473.77	243.21	293.35	28.77
p_hat1500-1	21.89	21.89	21.89	606.67	479.28	471.58	41.41
p_hat1500-2	-	-	**76.46**	-	-	1826.66	233.69
p_hat1500-3	113.65	113.65	113.65	3014.42	879.45	1886.51	202.90
keller4	13.47	13.47	13.47	3.35	1.46	2.69	0.17
keller5	31.00	31.00	31.00	503.36	53.31	289.31	30.06
keller6	-	**63.00**	-	-	1524.62	-	146.42
sanr200_0.7	23.63	23.63	23.63	3.44	1.26	3.21	0.25
sanr200_0.9	48.90	48.90	48.90	6.04	1.73	4.80	0.34
sanr400_0.5	20.18	20.18	20.18	6.60	3.80	6.19	0.57
sanr400_0.7	33.97	33.97	33.97	7.21	4.05	7.49	0.51
MANN_a9	**17.48**	**17.48**	**17.47**	0.48	0.28	0.46	0.05
MANN_a27	132.76	132.76	132.76	561.87	5.48	550.52	30.80
hamming6-2	32.00	32.00	32.00	1.49	0.33	1.25	0.10
hamming6-4	4.00	4.00	4.00	0.10	0.08	0.11	0.01
hamming8-2	128.00	128.00	128.00	532.92	3.96	500.17	30.03
hamming8-4	16.00	16.00	16.00	2.62	1.13	2.60	0.24
hamming10-4	42.67	42.67	42.67	97.36	31.77	93.90	7.46

Table 5.2: Results on $\vartheta_+(G)$, graphs from the second DIMACS implementation challenge.

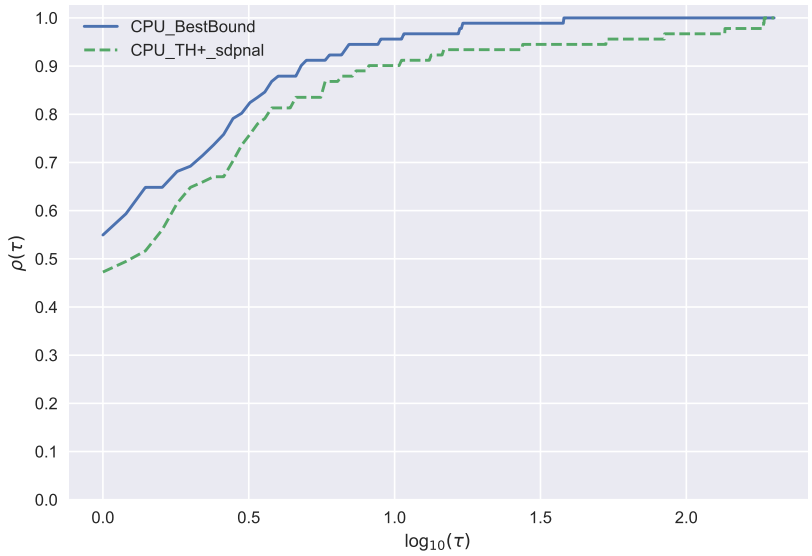

Figure 5.3: Performance profiles on CPU time. Comparison between BestBound and `SDPNAL+` on the computation of $\bar{\vartheta}_+(G)$.

ran `ADAL-ineq` and `SDPNAL+`, halting the algorithms either upon satisfaction of the termination criteria with a precision of 10^{-6} or after a time limit of 3600 seconds. As for bounding $\vartheta_+(G)$, we applied the post-processing procedure detailed in Section 3.1 every 200 iterations and after termination of `ADAL-ineq`. Every dual bound computed by the post-processing procedure and, in particular, the best dual bound is a valid lower bound on the chromatic number.

In Table 5.3 and Table 5.4, we report the same data as in Section 5.2. As previously noted, the BestBound presented in the tables may not necessarily correspond to the result of the last post-processing call.

We notice that the post-processing procedure fails in finding bounds on the chromatic number for several graphs for 19 out of 113 graphs. This is due to the precision of the dual matrix given as input as it may be too low to detect a dual feasible solution. We also notice that, on some graphs, the bound obtained is slightly larger than the dual objective function value obtained by `ADAL-ineq`. This behaviour is a consequence of the precision required by Gurobi to solve the LP in the post-processing, where we set a feasibility precision of 10^{-5}. Note that requesting a higher precision may lead to failure in the post-processing procedure. The results shown are then obtained with what, in our opinion, is a good trade off between feasibility precision and quality of the bound. The CPU time needed to compute the BestBound is often much lower with respect to the time needed by `SDPNAL+` to converge; this is confirmed by the performance profiles shown in Figure 5.3. In these profiles, we excluded the instances for which the difference in absolute value of the BestBound found by `ADAL-ineq` and the dual objective of `SDPNAL+` is less than 0.5. In particular, we excluded all the instances where the post-processing procedure was not able to compute a bound.

As a further comparison between `ADAL-ineq` and `SDPNAL+` on SDP relaxations of the chromatic number, we built instances by adding 1000, 2500 and 5000 inequalities to $\bar{\vartheta}(G)$. The inequalities were chosen randomly from those proposed by Dukanovic and Rendl [8] to

	$\bar{\vartheta}_+(G)$			CPU times			
Graph	ADAL-ineq	SDPNAL+	BestBound	ADAL-ineq	SDPNAL+	BestBound	post-proc
DSJC125.1	4.14	4.14	4.14	69.01	22.88	1.72	0.91
DSJC125.5	11.87	11.87	11.87	1.02	1.36	0.74	0.06
DSJC125.9	37.80	37.80	37.80	1.87	2.03	1.31	0.09
DSJC250.1	4.94	4.94	4.94	6.65	6.70	2.19	0.16
DSJC250.5	16.35	16.35	16.35	1.90	2.84	2.00	0.10
DSJC250.9	55.22	55.22	55.22	4.76	3.76	3.76	0.37
DSJC500.1	6.25	6.25	6.25	8.66	16.57	5.91	0.16
DSJC500.5	22.90	22.90	22.90	5.41	9.64	5.71	0.30
DSJC500.9	84.14	84.14	84.14	17.04	16.36	17.53	1.21
DSJR500.1	12.00	12.00	12.00	35.18	10.00	23.59	0.34
DSJR500.1c	83.75	83.75	83.75	-	1231.74	190.31	294.26
DSJR500.5	**122.01**	**122.00**	**122.00**	198.08	16.95	181.80	6.48
DSJC1000.1	8.36	8.36	8.36	31.65	59.76	22.83	0.57
DSJC1000.5	32.11	32.11	32.11	18.30	38.11	19.46	1.17
DSJC1000.9	122.80	122.80	122.80	72.30	63.88	70.85	6.89
fpsol2.i.1	65.00	65.00	65.00	200.57	11.71	199.60	2.67
fpsol2.i.2	30.00	30.00	30.00	28.11	9.75	25.71	0.43
fpsol2.i.3	30.00	30.00	30.00	27.36	7.82	27.43	0.42
inithx.i.1	54.00	54.00	54.00	604.51	32.25	537.71	4.96
inithx.i.2	**31.00**	**31.00**	**30.22**	387.80	12.39	35.60	3.70
inithx.i.3	**31.00**	**31.00**	**30.23**	341.12	13.64	32.57	3.45
latin_square_10	**90.00**	**89.99**	-	48.40	41.12	-	2.91
le450_15a	**15.00**	**15.00**	-	6.37	5.18	-	0.12
le450_15b	15.00	15.00	15.00	7.06	5.61	7.13	0.14
le450_15c	15.00	15.00	15.00	3.92	4.70	4.02	0.10
le450_15d	15.00	15.00	15.00	3.86	4.71	3.96	0.10
le450_25a	25.00	25.00	25.00	19.73	7.54	19.53	0.29
le450_25b	**25.00**	**25.00**	-	18.44	7.27	-	0.23
le450_25c	25.00	25.00	25.00	9.67	7.03	9.78	0.20
le450_25d	25.00	25.00	25.00	9.15	6.82	9.25	0.19
mulsol.i.1	**49.00**	**49.00**	-	18.89	3.48	-	0.66
mulsol.i.2	31.00	31.00	31.00	9.02	2.92	9.04	0.28
mulsol.i.3	31.00	31.00	31.00	8.13	3.12	7.47	0.19
mulsol.i.4	31.00	31.00	31.00	7.97	2.40	6.31	0.22
mulsol.i.5	31.00	31.00	31.00	9.88	3.34	9.01	0.19
school1	14.00	14.00	14.00	14.74	65.03	8.08	0.40
school1_nsh	14.00	14.00	14.00	12.12	75.97	7.29	0.30
zeroin.i.1	49.00	49.00	49.00	24.91	2.47	21.91	0.80
zeroin.i.2	30.00	30.00	30.00	14.63	2.43	14.13	0.48
zeroin.i.3	30.00	30.00	30.00	14.62	2.80	13.53	0.46
anna	**11.00**	**11.00**	-	9.97	1.16	-	0.13
david	**11.00**	**11.00**	-	2.46	0.59	-	0.08
huck	**11.00**	**11.00**	-	1.60	0.43	-	0.04
jean	**10.00**	**10.00**	-	1.35	0.53	-	0.03
games120	**9.00**	**9.00**	-	3.23	0.86	-	0.07
miles250	8.00	8.00	8.00	7.17	0.94	6.30	0.11
miles500	20.00	20.00	20.00	6.78	1.69	6.26	0.11
miles750	31.00	31.00	31.00	4.75	2.73	4.77	0.09
miles1000	42.00	42.00	42.00	7.81	1.64	7.61	0.16
miles1500	73.00	73.00	73.00	10.36	1.48	10.28	0.27

Table 5.3: Results on $\bar{\vartheta}_+(G)$, graphs from the second DIMACS implementation challenge.

	$\bar{\vartheta}_+(G)$			CPU times			
Graph	ADAL-ineq	SDPNAL+	BestBound	ADAL-ineq	SDPNAL+	BestBound	post-proc
queen5_5	5.00	5.00	5.00	0.01	0.11	0.04	0.03
queen6_6	6.04	6.04	6.04	0.77	0.69	0.19	0.04
queen7_7	7.00	7.00	7.00	0.08	0.29	0.08	0.01
queen8_8	8.00	8.00	8.00	0.10	0.19	0.11	0.01
queen8_12	**12.00**	**12.00**	**-**	0.55	0.62	-	0.02
queen9_9	9.00	9.00	9.00	0.15	0.23	0.16	0.01
queen10_10	10.00	10.00	10.00	0.23	0.44	0.24	0.01
queen11_11	11.00	11.00	11.00	0.46	0.47	0.47	0.01
queen12_12	12.00	12.00	12.00	0.67	0.68	0.71	0.04
queen13_13	13.00	13.00	13.00	0.76	0.64	0.80	0.04
queen14_14	14.00	14.00	14.00	1.27	0.82	1.32	0.04
queen15_15	**15.00**	**15.00**	**-**	1.38	1.26	-	0.04
queen16_16	16.00	16.00	16.00	1.84	1.46	1.90	0.06
myciel3	2.40	2.40	2.40	0.01	0.13	0.04	0.03
myciel4	2.53	2.53	2.53	0.04	0.18	0.04	0.01
myciel5	2.64	2.64	2.64	0.41	0.41	0.23	0.02
myciel6	2.73	2.73	2.73	1.73	1.16	0.51	0.04
myciel7	2.82	2.82	2.82	7.35	7.60	1.34	0.24
mug88_1	3.00	3.00	3.00	11.78	29.45	0.35	0.24
mug88_25	3.00	3.00	3.00	20.81	47.43	0.35	0.43
mug100_1	3.00	3.00	3.00	19.59	84.51	0.46	0.31
mug100_25	3.00	3.00	3.00	26.20	84.97	0.46	0.39
abb313GPIA	8.00	8.00	8.01	615.04	2949.22	55.61	4.48
ash331GPIA	3.38	3.38	3.38	125.34	17.85	38.24	1.01
ash608GPIA	3.33	3.33	3.31	265.72	41.34	129.25	1.34
ash958GPIA	**3.33**	**3.33**	**-**	529.68	124.35	0.00	2.51
will199GPIA	6.10	6.10	6.10	156.39	32.11	124.61	1.22
1-Insertions_4	2.23	2.23	2.23	1.93	1.01	0.34	0.07
1-Insertions_5	2.28	2.28	2.28	19.71	15.04	2.64	0.56
1-Insertions_6	2.31	2.31	2.31	337.22	100.65	22.80	3.29
2-Insertions_3	2.10	2.10	2.10	0.38	0.57	0.18	0.04
2-Insertions_4	2.13	2.13	2.13	25.27	9.06	1.59	0.36
2-Insertions_5	2.16	2.16	2.16	544.91	109.90	52.67	4.99
3-Insertions_3	2.07	2.07	2.07	1.22	1.00	0.31	0.06
3-Insertions_4	2.09	2.09	2.09	125.48	29.79	8.39	2.37
3-Insertions_5	**-**	**2.10**	**2.11**	-	3568.47	130.38	17.94
4-Insertions_3	2.05	2.05	2.05	2.39	2.75	0.38	0.08
4-Insertions_4	2.06	2.06	2.06	563.58	130.23	8.89	6.64
1-FullIns_3	3.06	3.06	3.06	0.32	0.35	0.13	0.05
1-FullIns_4	3.12	3.12	3.12	4.37	2.45	1.39	0.10
1-FullIns_5	3.18	3.18	3.18	71.27	17.55	18.65	1.52
2-FullIns_3	4.03	4.03	4.03	1.34	0.39	0.74	0.08
2-FullIns_4	4.06	4.06	4.06	57.51	9.21	26.76	1.64
2-FullIns_5	4.08	4.08	4.08	2670.31	184.26	381.59	19.29
3-FullIns_3	5.02	5.02	5.02	6.12	1.18	4.47	0.15
3-FullIns_4	5.03	5.03	5.03	329.51	24.03	58.31	4.94
3-FullIns_5	**-**	**5.05**	**5.04**	-	1769.01	2965.54	18.40
4-FullIns_3	6.01	6.01	6.01	21.19	2.30	1.65	0.32
4-FullIns_4	6.02	6.02	6.02	1979.86	88.40	283.54	16.15
4-FullIns_5	-	-	-	-	-	-	14.11
5-FullIns_3	7.01	7.01	7.00	61.22	2.72	12.48	0.71
5-FullIns_4	**-**	**7.01**	**7.01**	-	207.83	137.49	21.11
wap01a	**-**	**41.00**	**40.38**	-	309.86	3575.61	20.34
wap02a	**40.00**	**40.00**	**-**	538.62	473.42	-	3.29
wap03a	40.00	40.00	40.00	1594.69	2668.31	1507.80	10.12
wap04a	40.00	40.00	40.00	2175.13	2658.54	2179.94	13.84
wap05a	50.00	50.00	50.00	1099.11	24.19	918.93	11.72
wap06a	**40.00**	**40.00**	**-**	63.35	69.47	-	0.74
wap07a	40.00	40.00	40.00	309.93	426.97	145.89	3.00
wap08a	**40.00**	**40.00**	**-**	278.67	224.35	-	2.26
qg.order30	**30.00**	**30.00**	**-**	32.22	21.30	-	0.30
qg.order40	**40.00**	**40.00**	**-**	153.25	82.68	-	1.08
qg.order60	**60.00**	**60.00**	**-**	1684.60	496.86	-	9.74

Table 5.4: Results on $\bar{\vartheta}_+(G)$, graphs from the second DIMACS implementation challenge.

strengthen $\bar{\vartheta}(G)_+$ by including to (13) the following:

$$X_{ij} + X_{ik} - X_{jk} \leq t - 1, \quad \forall\, i, j, k \in V.$$

Note that the main goal of this experiment is to measure the behavior of the solvers on SDPs with an increasing number of inequalities, rather than to evaluate the improvement that these valid inequalities yield over $\bar{\vartheta}(G)_+$. In Table 5.5, we report the results on some classes of graphs where the CPU time needed to compute the BestBound is often lower with respect to the time needed by `SDPNAL+` to converge with a precision of 10^{-6}.

6 Conclusions

In this paper, we propose a numerical comparison between `ADAL-ineq`, an enhanced version of `ADAL` where the presence of linear inequality constraints is smartly handled, and `SDPNAL+`, the state-of-the-art method for solving large-scale SDPs that has been awarded the Beale-Orchard-Hays Prize in 2018. We consider random instances as well as instances from the SDP relaxations of the graph coloring problem and the maximum clique problem. The post-processing procedure used is developed to obtain a dual feasible solution which in turn gives a bound on the optimal primal value. From a practical standpoint, as long as we use SDPs to address combinatorial optimization problems, the post-processing procedure allows to stop the execution of the ADMM as soon as a "good" bound is obtained, even if the convergence criterion is far from being met. Furthermore, the fact that a dual feasible solution is detected, allows to use re-optimization techniques within branch-and-bound frameworks and is what we plan to focus on in the near future.

Acknowledgements

The authors acknowledge support within the project RM120172A2970290 which has received funding from Sapienza, University of Rome. They are also indebted to Fabrizio Rossi and Stefano Smriglio for their useful suggestions and to Griffin D. Kent for his precious help in proof reading this manuscript. Last but not least, they are grateful to two anonymous referees for the careful reading of the manuscript and the valuable comments that helped to improve the paper.

Declarations

Conflict of interest The authors declare no conflict of interest

	bounds on $\chi(G)$			CPU times			
Graph	ADAL-ineq	SDPNAL+	BestBound	ADAL-ineq	SDPNAL+	BestBound	post-proc
$\bar{\vartheta}(G)$ + 1000 inequalities from [8]							
DSJC500.5	22.90	22.90	22.90	15.97	38.73	13.77	0.54
DSJC1000.1	8.36	8.36	8.36	34.43	154.79	25.85	0.56
DSJC1000.5	32.11	32.11	32.11	33.33	154.82	34.50	1.18
myciel7	2.85	2.85	2.85	255.84	13.77	13.39	6.77
mug88_25	**3.00**	**3.00**	**-**	17.81	33.92	-	0.25
mug100_25	**3.00**	**3.00**	**-**	22.33	97.57	-	0.28
abb313GPIA	8.00	8.00	8.00	661.22	3466.57	178.77	4.49
1-Insertions_6	2.33	2.33	2.33	1001.29	233.15	65.62	8.98
2-Insertions_5	2.18	2.18	2.18	850.44	373.95	100.30	7.21
3-Insertions_5	**-**	**-**	**2.11**	-	-	415.32	18.32
4-Insertions_4	2.07	2.07	2.07	632.60	363.73	87.32	6.53
1-FullIns_5	3.19	3.19	3.19	1955.97	141.02	59.14	35.67
5-FullIns_4	**-**	**7.01**	**7.01**	-	257.83	132.12	21.35
wap03a	**40.00**	**-**	**40.00**	1629.48	-	1496.25	11.16
wap04a	**40.00**	**-**	**40.00**	2297.75	-	2302.30	13.66
$\bar{\vartheta}(G)$ + 2500 inequalities from [8]							
DSJC500.5	22.90	22.90	22.90	79.62	43.81	79.92	0.60
DSJC1000.1	8.36	8.36	8.36	38.74	167.61	29.44	0.57
DSJC1000.5	32.11	32.11	32.11	100.35	157.33	91.06	2.39
myciel7	2.87	2.87	2.87	341.20	13.80	63.14	3.51
mug88_25	3.00	3.00	3.00	83.45	81.22	36.93	1.22
mug100_25	3.00	3.00	3.00	91.18	97.93	85.68	1.43
abb313GPIA	8.00	8.00	8.00	685.60	2611.71	250.42	4.60
1-Insertions_6	2.34	2.34	2.34	827.90	553.82	111.23	6.91
2-Insertions_5	2.19	2.19	2.19	790.06	576.61	157.16	5.79
3-Insertions_5	**-**	**2.13**	**2.12**	-	-	1133.88	18.54
4-Insertions_4	2.08	2.08	2.08	625.78	537.75	143.96	4.82
1-FullIns_5	**-**	**3.19**	**3.19**	-	145.95	270.64	34.75
5-FullIns_4	**-**	**7.01**	**7.01**	-	308.79	149.04	20.10
wap03a	**40.00**	**-**	**40.00**	1932.11	-	1935.99	10.83
wap04a	**40.00**	**-**	**40.00**	2629.88	-	2011.30	14.42
$\bar{\vartheta}(G)$ + 5000 inequalities from [8]							
DSJC500.5	22.90	22.90	22.90	464.82	46.24	456.20	1.14
DSJC1000.1	8.36	8.36	8.36	63.50	163.05	48.59	0.70
DSJC1000.5	32.11	32.11	32.11	589.69	168.79	590.91	2.12
myciel7	2.89	2.89	2.89	583.55	15.44	132.16	3.90
mug88_25	3.00	3.00	3.00	199.32	116.73	78.35	1.56
mug100_25	3.00	3.00	3.00	216.21	154.84	190.52	1.39
abb313GPIA	**8.00**	**-**	**8.00**	744.60	-	356.07	4.36
1-Insertions_6	2.36	2.36	2.36	1351.80	258.02	218.89	8.25
2-Insertions_5	2.19	2.19	2.19	904.85	572.94	274.81	5.39
3-Insertions_5	**-**	**-**	**2.12**	-	-	1737.21	15.49
4-Insertions_4	2.09	2.09	2.09	736.69	831.07	315.23	4.01
1-FullIns_5	**-**	**3.19**	**3.19**	-	179.80	554.77	22.29
5-FullIns_4	**-**	**7.01**	**7.01**	-	344.28	293.47	17.54
wap03a	**40.00**	**-**	**40.00**	2591.76	-	2383.76	15.00
wap04a	**-**	**-**	**40.00**	-	-	3129.61	18.72

Table 5.5: Results on bounds on $\chi(\bar{G})$.

References

[1] Federico Battista. *On semidefinite lift-and-project of combinatorial optimization problems*. PhD thesis, Università di Roma Sapienza, 2023.

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

[3] Samuel Burer and Renato D. C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3, Ser. A):427–444, 2005.

[4] Martina Cerulli, Marianna De Santis, Elisabeth Gaar, and Angelika Wiegele. Improving ADMMs for solving doubly nonnegative programs through dual factorization. *4OR*, 19:415–448, 2021.

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

[6] Marianna De Santis, Franz Rendl, and Angelika Wiegele. Using a factored dual in augmented Lagrangian methods for semidefinite programming. *Operations Research Letters*, 46(5):523 – 528, 2018.

[7] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91:201–213, 2002.

[8] Igor Dukanovic and Franz Rendl. A semidefinite programming-based heuristic for graph coloring. *Discrete Applied Mathematics*, 156(2):180–189, 2008.

[9] Elisabeth Gaar and Franz Rendl. A bundle approach for SDPs with exact subgraph constraints. In Andrea Lodi and Viswanath Nagarajan, editors, *Integer Programming and Combinatorial Optimization*, pages 205–218. Springer International Publishing, 2019.

[10] Monia Giandomenico, Adam N. Letchford, Fabrizio Rossi, and Stefano Smriglio. An application of the Lovász–Schrijver m (k, k) operator to the stable set problem. *Mathematical programming*, 120(2):381–401, 2009.

[11] Monia Giandomenico, Adam N. Letchford, Fabrizio Rossi, and Stefano Smriglio. Ellipsoidal relaxations of the stable set problem: Theory and algorithms. *SIAM Journal on Optimization*, 25(3):1944–1963, 2015.

[12] Monia Giandomenico, Fabrizio Rossi, and Stefano Smriglio. Strong lift-and-project cutting planes for the stable set problem. *Mathematical Programming*, 141(1):165–192, 2013.

[13] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2. Springer Science & Business Media, 2012.

[14] Gerald Gruber and Franz Rendl. Computational experience with stable set relaxations. *SIAM Journal on Optimization*, 13(4):1014–1028, 2003.

[15] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2022.

[16] Christian Jansson, Denis Chaykin, and Christian Keil. Rigorous error bounds for the optimal value in semidefinite programming. *SIAM Journal on Numerical Analysis*, 46(1):180–200, 2007/08.

[17] David J. Johnson and Michael A. Trick, editors. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993*. American Mathematical Society, 1996.

[18] Monique Laurent and Franz Rendl. Semidefinite programming and integer programming. In K. Aardal, G.L. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, chapter 8, pages 393–514. Elsevier, Amsterdam, The Netherlands, 2005.

[19] Marco Locatelli. Improving upper bounds for the clique number by non-valid inequalities. *Mathematical Programming*, 150(2):511–525, 2015.

[20] Dirk A. Lorenz and Quoc Tran-Dinh. Non-stationary Douglas–Rachford and alternating direction method of multipliers: adaptive step-sizes and convergence. *Computational Optimization and Applications*, 74(1):67–92, Sep 2019.

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

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

[23] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13 of *SIAM Studies in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994.

[24] Janez Povh, Franz Rendl, and Angelika Wiegele. A Boundary Point Method to solve Semidefinite Programs. *Computing*, 78:277–286, 2006.

[25] Franz Rendl. Matrix relaxations in combinatorial optimization. In Jon Lee and Sven Leyffer, editors, *Mixed Integer Nonlinear Programming*, pages 483–511. Springer New York, 2012.

[26] Defeng Sun, Kim-Chuan Toh, and Liuqin Yang. A convergent 3-block semiproximal alternating direction method of multipliers for conic programming with 4-type constraints. *SIAM Journal on Optimization*, 25:882–915, 2015.

[27] Zaiwen Wen, Donald Goldfarb, and Wotao Yin. Alternating direction augmented Lagrangian methods for semidefinite programming. *Mathematical Programming Computation*, 2(3):203–230, 2010.

[28] Angelika Wiegele and Shudian Zhao. SDP-based bounds for graph partition via extended ADMM. *Computational Optimization and Applications*, 82(1):251–291, 2022.

[29] Liuqin Yang, Defeng Sun, and Kim-Chuan Toh. SDPNAL+: a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 7(3):331–366, 2015.